Non-Relational Databases

Contents

Articles

Introduction **1**

Data management 1
Database 4
Database system 21
Database management system 38
Types of DBMS 55
Data store 56
Technical Information Project 57

Introduction to Data Modeling **58**

Data modeling 58
Data model 65
Database model 79
Database design 86
Conceptual schema 90
Data structure diagram 92
Hierarchical database model 94
Network model 96
Navigational database 97

ERD **100**

Entity-relationship model 100
Has-a 108
Many-to-many (data model) 110
Enhanced Entity-Relationship Model 111
Weak entity 112
Associative Entities 114
Structured-Entity-Relationship-Model 115
Barker's Notation 117
Peter Chen 118

Other Data Modeling Techniques **122**

Unified Modeling Language 122
The Third Manifesto 131

Three schema approach 132
White pages schema 135
Anchor Modeling 136
Bachman diagram 139
Bitemporal Modeling 141
Bitemporal data 142
IDEF1X 142
Universal Data Element Framework 150
Terminology model 153
Georelational data model 154
Semantic data model 154
Relational Model/Tasmania 157

Relational Schema **161**
Relational model 161
Relational database 171
Relational database management system 175
Life cycle of a relational database 176
Logical data model 178
Logical schema 180
Relation (database) 180
Table (database) 182
Tuple 183
Row (database) 187
Attribute domain 188
Candidate key 188
Unique key 191
Natural key 195
Key field 196
Compound key 199
Foreign key 200
Persistent Object Identifier 204
Cardinality (data modeling) 205
Recordset 206
Superkey 207
Integrity constraints 208
Check Constraint 209
Propagation constraint 211

Transition constraint 212
Wide and narrow data 212
Universal relation assumption 213
Reference table 214
Junction table 214
Nested set model 216
Information schema 219
Codd's 12 rules 220
Edgar F. Codd 223

Relational algebra **226**

Relational algebra 226
Projection (relational algebra) 238
Rename (relational algebra) 239
Selection (relational algebra) 240
Generalized selection 241
Range query 242
Monotonic query 242
Recursive join 243
Relvar 244
Relational calculus 245
Tuple relational calculus 245

Overview **250**

Query language 250
Data Definition Language 251
Varchar 253
Data Manipulation Language 254
Create, read, update and delete 255
SQL 256
SQL-92 270
SQL:1999 271
SQL:2003 273
SQL:2008 274
SQL:2011 276
Data 277
Metadata 279

Database objects **290**
Table 290
Column 291
Field 292
Row 293
Data type 294
Statements **299**
Select 299
Result set 304
Synonym (database) 304
Alias (SQL) 306
Insert 307
Update 310
Merge 312
Delete 314
Join 317
Set operations 330
Commit 334
Rollback 335
Truncate 336
Views **337**
View (database) 337
Materialized view 340
Advanced Issues **342**
Hierarchical query 342
Hint (SQL) 345
Extensions **346**
SQL/CLI 346
SQL/JRT 346
SQL/MED 347
SQL/OLB 348
SQL/PSM 348
SQL/Schemata 349
StreamSQL 350

dBase and derived languages **351**
DBase 351
Clip (compiler) 358
Clipper (programming language) 359
Flagship compiler 362
FoxPro 2 364
Harbour (software) 366
Visual FoxPro 376
Visual Objects 382
XBase 383
XBase++ 386
XHarbour 389
Other Query Languages **398**
QUEL query languages 398
Query by Example 400
SQR 402
.QL 404
Yahoo! query language 405
YQL (programming language) 406
YANG 406
WQL 407
Versa (query language) 408
SPARQL 410
RDF query language 414
Access query language 415
Nonprocedural language 416
Facebook Query Language 417
Databases Programming **418**
Active database 418
Database trigger 418
Stored procedure 424
PL/SQL 427
SQL PL 437
SQL programming tool 437
User-defined function 439

Cursor (databases) 442
SQL Problems Requiring Cursors 447
WxSQLite3 451

Database connectivity **452**
Application programming interface 452
Structured Query Language Interface 459
Database transaction 459
Prepared statement 462
Open Database Connectivity 465
Java Database Connectivity 472
ODBC 478
OLE DB provider 485
OLE DB 486
UnixODBC 487
IODBC 489
Pool (computer science) 490
Connection pool 491
Remote Database Access 492
SQLJ 493
Native Queries 496
Meta-SQL 496
ADO.NET 498

Software and Tools **500**
List of relational database management systems 500
Comparison of relational database management systems 503
Comparison of database tools 526
Comparison of object-relational database management systems 533
Transactions per second 535

SQL Server **536**
Microsoft SQL Server 536
Microsoft SQL Server Master Data Services 549
Transact-SQL 551
WCF Data Services 553
Windows Internal Database 555
SQL Server Agent 556

SQL Server Compact 557
SQL CLR 559
Microsoft Transaction Server 560

Oracle **562**
Database schema 562
Oracle Database 565
Oracle Exadata 581
Oracle Coherence 585
NVL 586
Pro*C 588

Tools **589**
Toad Data Modeler 589
SQuirreL SQL Client 591
SQuirreL SQL Client Plugin API 593
SQLPro SQL Client 597
Navicat 599
ModelRight 604

Other DBMS **605**
MySQL 605
SQL Anywhere 614
SQLite 616
SESAM (database) 620
LAMP 621
Watcom SQL 624
Microsoft Access 625
VMDS 636
Superbase database 639
Rocket U2 642
PointBase 645
R:BASE System 646
REAL Server 650
MaxDB 652
Adaptive Server Enterprise 654
Advantage Database Server 656
Ingres (database) 658

Normal Form **669**
Data redundancy 669
Database normalization 669
Functional dependency 677
Armstrong's axioms 681
Transitive dependency 683
Superkey 684
First normal form 685
Second normal form 688
Third normal form 691
Boyce–Codd normal form 694
Lossless-Join Decomposition 698
Join dependency 699
Multivalued dependency 700
Fourth normal form 702
Fifth normal form 705
Sixth normal form 707
Denormalization 709
Domain/key normal form 710
Single Source of Truth 712
Single version of the truth 714
Principle of Orthogonal Design 715
Transaction Management **716**
Database transaction 716
Transaction processing 719
Concurrency control 722
Transaction Control Language 730
ACID 730
Atomicity (database systems) 734
Isolation (database systems) 735
Durability (database systems) 741
Atomic commit 741
Schedule (computer science) 744
Serializability 750
Precedence graph 757
Serializability theory 758

Read–write conflict 765
Write–read conflict 766
Write–write conflict 767
Lock (database) 767
Record locking 768
Multiple granularity locking 770
Two-phase locking 771
Readers–writer lock 778
Blind write 779
Conservative two-phase locking 779
Strong strict two-phase locking 780
Index locking 786
Snapshot isolation 787
Non-lock concurrency control 789
Commitment ordering 790
Long-running transaction 812
Timestamp-based concurrency control 813
Pseudoconversational transaction 815
Thomas write rule 816
Global concurrency control 817
Global serializability 818
Modular concurrency control 825
Multiversion concurrency control 826
Optimistic concurrency control 830
Autocommit 832
Transaction log 832
Savepoint 834
No-force 834
Non-blocking algorithm 835
Data recovery 837
Point-in-time recovery 841
Redo log 841
Extreme transaction processing 843
In-database processing 844
Locks with ordered sharing 846
Nested transaction 846
Transaction processing system 847
Transaction processing systems 853

Transaction server 859
Priority inversion 864
Priority ceiling protocol 867
Priority inheritance 868

Query Optimization and Indexing **869**

Query optimization 869
Query optimizer 871
Query plan 874
Index (database) 876
Partial index 880
Expression index 881
Reverse index 882
Bitmap index 883
Inverted index 886
Sargable 889
V-optimal histograms 890
Cardinality (SQL statements) 893
Online aggregation 894

Distributed DB **895**

Very large database 895
Big data 895
XLDB 903
Secondary database server 905
Centralized database 906
Distributed database 906
Distributed database management system 910
Distributed file system 913
Distributed data store 917
Heterogeneous Database System 918
Simple Sloppy Semantic Database 919
Distributed transaction 920
Network transparency 921
Long-lived transaction 922
Distributed concurrency control 922
Consistency model 923
Distributed Transaction Coordinator 925

Two-phase commit protocol 926
Three-phase commit protocol 930
Xeround 932
Vector-field consistency 935
Storage area network 936
Partition (database) 940
Shared nothing architecture 941
Shard (database architecture) 942
Quorum (distributed computing) 945

Physical Design **947**
Physical data model 947
Physical schema 948
Storage model 948
Storage block 949
Tablespace 949
Database tuning 950
Database dump 951
Spindling 952

Data Mapping and Integration Tasks **953**
Data mapping 953
Semantic integration 955
Semantic translation 956
Record linkage 957
Metadata discovery 964
Schema matching 966
Schema crosswalk 968
Schema evolution 971
Data integration 972
Ontology-based data integration 977
Ontology merging 978

Parallel **979**
MapReduce 979
Apache Hadoop 988
Pig (programming language) 997
H-Store 999

Information Security **1000**

Data control language 1000

SQL injection 1000

Data Warehouseing **1008**

Business intelligence 1008

Reactive business intelligence 1017

Business analytics 1017

Sales intelligence 1020

Performance intelligence 1021

Data warehouse 1021

Data warehouse architectures 1029

Data mart 1030

The Kimball Lifecycle 1032

Time variance 1033

Federated database system 1034

Single Source of Truth 1038

XLeratorDB 1040

Design **1044**

Dimension (data warehouse) 1044

Fact (data warehouse) 1047

Measure (data warehouse) 1055

Fact table 1055

Degenerate dimension 1057

Slowly changing dimension 1058

Star schema 1064

Dimension table 1067

Dimensional Fact Model 1068

Snowflake schema 1070

Denormalization 1073

Single version of the truth 1074

Transaction data 1075

Enterprise bus matrix 1076

OLAP **1078**

Statistical database 1078

Online transaction processing 1079

Operational system 1081
Operational data store 1081
Operational database 1082
Online analytical processing 1083
OLAP cube 1087
Aggregate (data warehouse) 1090
MOLAP 1091
ROLAP 1093
HOLAP 1095
Thomsen Diagrams 1096
Spreadmart 1097
MultiDimensional eXpressions 1099

OLAP Servers and Tools **1102**
Comparison of OLAP Servers 1102
Applix 1107
BusinessObjects OLAP Intelligence 1108
Crystal Analysis 1108
CubePort 1110
Essbase 1112
Microsoft Analysis Services 1119
Mondrian OLAP server 1123
OLE DB for OLAP 1124
Oracle OLAP 1125
Palo (OLAP database) 1126
Panorama Software 1128
ProClarity 1129
SAP BI Accelerator 1130
SAP NetWeaver Business Intelligence 1131
NEVOD DMB 1134

Data Transforamtion **1136**
Extract, transform, load 1136
Staging (data) 1142
Vocabulary-based transformation 1144
Surrogate key 1145
Variable data publishing 1150
Semantic warehousing 1151

Scriptella 1153

Data Quality **1155**

Bit rot 1155
Cleansing and Conforming Data 1156
Data auditing 1158
Data cleansing 1158
Data corruption 1162
Data integrity 1164
Data profiling 1167
Data quality 1169
Data quality assessment 1174
Data quality assurance 1174
Data Quality Firewall 1179
Data truncation 1180
Data validation 1180
Data verification 1183
Database integrity 1183
Database preservation 1184
Declarative Referential Integrity 1184
Digital continuity 1185
Digital preservation 1186
User:TheAmazing0and1/draftdigipres 1199
Dirty data 1206
Entity integrity 1206
Information quality 1207
Link rot 1210
One-for-one checking 1214
Referential integrity 1214
Soft error 1215
Two pass verification 1221
Validation rule 1222

XML **1224**

Semi-structured data 1224
Semi-structured model 1226
Standard Generalized Markup Language 1227
XML 1237

XML database 1249
XML Schema Language comparison 1252
XML schema 1257
XML validation 1259
Xpath data model 1260
Path expression 1261
XQuery 1262
XSA 1268
XSIL 1269
SQL/XML 1269
Soma File 1271
Regular Language description for XML 1272
PureXML 1272
List of XML schemas 1275

Object database **1278**
Object database 1278
Object Definition Language 1282
Object Query Language 1283
Object-oriented SQL 1284
Object Exchange Model 1284
Object-relational database 1285
Object-relational impedance mismatch 1288
Object-relational mapping 1294
Polymorphic association 1296
Polyinstantiation 1296
Single Table Inheritance 1297
Versant Object Database 1298
Terminology-oriented database 1306
Odaba 1307
Object Data Management Group 1308
List of object database management systems 1309
Comparison of object database management systems 1311
PostgreSQL 1313
PL/pgSQL 1327
PL/Perl 1328
JADE (programming language) 1329
ObjectDB 1334

Versant Object Database 1336
Zope Object Database 1345

No-SQL Databases **1349**
Strozzi NoSQL (RDBMS) 1349
NoSQL 1351
Graph database 1360
DEX (Graph database) 1370
Neo4j 1372
Sones GraphDB 1373
Apache Cassandra 1376
Triplestore 1381
Keyspace (distributed data store) 1386
Super column 1387
BigTable 1388
Flat file database 1390
Terminfo 1393
Termcap 1395
MultiValue 1397
OpenInsight 1399
Document-oriented database 1400
MongoDB 1404

Spatiotemporal Databases **1409**
Temporal database 1409
RRDtool 1415
Spatial database 1417
Spatial query 1420
Spatiotemporal database 1421
Object-based spatial database 1422
Tuple-versioning 1424
Valid time 1425
Transaction time 1426
Geospatial metadata 1426
Geographical database 1430
Time series database 1430
Operational historian 1432

Special Databases **1435**
Real-time database 1435
In-memory database 1440
Probabilistic database 1442
Deductive database 1445
Deductive language 1446
Mobile database 1446
Well Known Databases **1448**
Standard data model 1448
Suppliers and Parts database 1449
Internet Movie Database 1451
YAGO (ontology) 1457
World Wide Molecular Matrix 1458
Voter database 1459
VIOLIN vaccine database 1462
Census of Governments 1463
Management information base 1463
Planetary Data System 1467
Parameter Value Language 1469
National Data Repository 1469
Data Centers **1487**
Data center 1487
Virtual data room 1502
Virtual facility 1503
Network-neutral data center 1505
Related Technologies **1506**
Virtual directory 1506
Virtuoso Universal Server 1509
Workflow engine 1514
Metadata Models **1515**
Metadata 1515
Meta-data management 1524
Metadatabase 1527
Metadirectory 1528

Metadata controller 1529
Data element 1529
Metadata publishing 1530
Metadata registry 1532
Metadata facility for Java 1536
Object Management Group 1537
Semantics of Business Vocabulary and Business Rules 1542
Business Motivation Model 1547
Business Process Definition Metamodel 1548
Knowledge Discovery Metamodel 1550
Resources, events, agents (accounting model) 1553
Learning object metadata 1555
Learning object 1561
OGML 1564

References

Article Sources and Contributors 1565
Image Sources, Licenses and Contributors 1601

Article Licenses

License 1608

Introduction

Data management

Data management comprises all the disciplines related to managing data as a valuable resource.

Overview

The official definition provided by DAMA International, the professional organization for those in the data management profession, is: "Data Resource Management is the development and execution of architectures, policies, practices and procedures that properly manage the full data lifecycle needs of an enterprise." {{DAMA International}} This definition is fairly broad and encompasses a number of professions which may not have direct technical contact with lower-level aspects of data management, such as relational database management.

Alternatively, the definition provided in the DAMA Data Management Body of Knowledge (DAMA-DMBOK) is: "Data management is the development, execution and supervision of plans, policies, programs and practices that control, protect, deliver and enhance the value of data and information assets."[1]

The concept of "Data Management" arose in the 1980s as technology moved from sequential processing (first cards, then tape) to random access processing. Since it was now technically possible to store a single fact in a single place and access that using random access disk, those suggesting that "Data Management" was more important than "Process Management" used arguments such as "a customer's home address is stored in 75 (or some other large number) places in our computer systems." During this period, random access processing was not competitively fast, so those suggesting "Process Management" was more important than "Data Management" used batch processing time as their primary argument. As applications moved more and more into real-time, interactive applications, it became obvious to most practitioners that both management processes were important. If the data was not well defined, the data would be mis-used in applications. If the process wasn't well defined, it was impossible to meet user needs.

Data Management shows manageable of data through specific data source

Corporate Data Quality Management

Corporate Data Quality Management (CDQM) is, according to the European Foundation for Quality Management and the Competence Center Corporate Data Quality (CC CDQ, University of St. Gallen), the whole set of activities intended to improve corporate data quality (both reactive and preventive). Main premise of CDQM is the business relevance of high-quality corporate data. CDQM comprises with following activity areas:[2]

- **Strategy for Corporate Data Quality**: As CDQM is affected by various business drivers and requires involvement of multiple divisions in an organization; it must be considered a company-wide endeavor.
- **Corporate Data Quality Controlling**: Effective CDQM requires compliance with standards, policies, and procedures. Compliance is monitored according to previously defined metrics and performance indicators and reported to stakeholders.
- **Corporate Data Quality Organization**: CDQM requires clear roles and responsibilities for the use of corporate data. The CDQM organization defines tasks and privileges for decision making for CDQM.
- **Corporate Data Quality Processes and Methods**: In order to handle corporate data properly and in a standardized way across the entire organization and to ensure corporate data quality, standard procedures and guidelines must be embedded in company’s daily processes.

- **Data Architecture for Corporate Data Quality**: The data architecture consists of the data object model - which comprises the unambiguous definition and the conceptual model of corporate data - and the data storage and distribution architecture.
- **Applications for Corporate Data Quality**: Software applications support the activities of Corporate Data Quality Management. Their use must be planned, monitored, managed and continuously improved.

Topics in Data Management

Topics in Data Management, grouped by the DAMA DMBOK Framework,[3] include:

1. Data governance • Data asset • Data governance • Data steward 3. Data Architecture, Analysis and Design • Data analysis • Data architecture • Data modeling 5. Database Management • Data maintenance • Database administration • Database management system 7. Data Security Management • Data access • Data erasure • Data privacy • Data security 9. Data Quality Management • Data cleansing • Data integrity • Data enrichment • Data quality • Data quality assurance	1. Reference and Master Data Management • Data integration • Master data management • Reference data 3. Data Warehousing and Business Intelligence Management • Business intelligence • Data mart • Data mining • Data movement (extract, transform and load) • Data warehousing 5. Document, Record and Content Management • Document management system • Records management 7. Meta Data Management • Meta-data management • Metadata • Metadata discovery • Metadata publishing • Metadata registry 9. Contact Data Management • Business continuity planning • Marketing operations • Customer data integration • Identity management • Identity theft • Data theft • ERP software • CRM software • Address (geography) • Postal code • Email address • Telephone number

Body of Knowledge

The DAMA Guide to the Data Management Body of Knowledge" (DAMA-DMBOK Guide), under the guidance of a new DAMA-DMBOK Editorial Board. This publication is available from April 5, 2009.

Usage

In modern management usage, one can easily discern a trend away from the term 'data' in composite expressions to the term information or even knowledge when talking in non-technical context. Thus there exists not only data management, but also information management and knowledge management. This is a misleading trend as it obscures that traditional data is managed or somehow processed on second looks. The distinction between data and

derived values can be seen in the information ladder. While data can exist as such, 'information' and 'knowledge' are always in the "eye" (or rather the brain) of the beholder and can only be measured in relative units.

Integrated data management

Integrated data management (IDM) is a tools approach to facilitate data management and improve performance. IDM consists of an integrated, modular environment to manage enterprise application data, and optimize data-driven applications over its lifetime.[4][5][6][7] IDM's purpose is to

- Produce enterprise-ready applications faster
- Improve data access, speed iterative testing
- Empower collaboration between architects, developers and DBAs

Consistently achieve service level targets

- Automate and simplify operations
- Provide contextual intelligence across the solution stack

Support business growth

- Accommodate new initiatives without expanding infrastructure
- Simplify application upgrades, consolidation and retirement

Facilitate alignment, consistency and governance

- Define business policies and standards up front; share, extend, and apply throughout the lifecycle

References

[1] http://www.dama.org/files/public/DI_DAMA_DMBOK_Guide_Presentation_2007.pdf "DAMA-DMBOK Guide (Data Management Body of Knowledge) Introduction & Project Status"

[2] EFQM ; IWI-HSG: EFQM Framework for Corporate Data Quality Management. Brussels : EFQM Press, 2011. - Forthcoming.

[3] http://www.dama.org/i4a/pages/index.cfm?pageid=3364 "DAMA-DMBOK Functional Framework"

[4] Integrated Data Management: Managing data across its lifecycle (http://www.ibm.com/developerworks/data/library/techarticle/dm-0807hayes/?S_TACT=105AGX11&S_CMP=FP#ibm-content) by Holly Hayes

[5] Organizations thrive on Data (http://www.ibmsystemsmagmainframedigital.com/nxtbooks/ibmsystemsmag/mainframe_20090708/index.php#/34) by Eric Naiburg

[6] Fragmented Management Across The Data Life Cycle Increases Cost And Risk (http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/data-management/optim/reports/fragmented.pdf) - A commissioned study conducted by Forrester Consulting on behalf of IBM October 2008

[7] http://publib.boulder.ibm.com/infocenter/idm/v2r1/index.jsp ntegrated IBM Data Management information center.

External links

- Data management (http://www.dmoz.org/Computers/Software/Master_Data_Management/Articles/) on the Open Directory Project

Database

A **database** is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database management systems (**DBMS**s) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker Pro. A database is not generally portable across different DBMSs, but different DBMSs can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one database.

Terminology and overview

Formally, "database" refers to the data themselves and supporting data structures. Databases are created to operate large quantities of information by inputting, storing, retrieving, and managing that information. Databases are set up so that one set of software programs provides all users with access to all the data.

A "database management system" (DBMS) is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term "database" when used casually often refers to both a DBMS and the data it manipulates.

Outside the world of professional information technology, the term *database* is sometimes used casually to refer to any collection of data (perhaps a spreadsheet, maybe even a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.[1]

The interactions catered for by most existing DBMSs fall into four main groups:

- **Data definition –** Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- **Update –** Inserting, modifying, and deleting data.
- **Retrieval –** Obtaining information either for end-user queries and reports or for processing by applications.
- **Administration –** Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

Both a database and its DBMS conform to the principles of a particular database model.[2] "Database system" refers collectively to the database model, database management system, and database.[3]

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used for recovery of data if any of the disks fail. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications. DBMSs may be built around a custom multitasking kernel with built-in networking support, but modern DBMSs typically rely on a standard operating system to provide these functions. [*citation needed*] Since DBMSs comprise a significant economical market, computer and storage vendors often take into account DBMS requirements in their own development plans.[*citation needed*]

Databases and DBMSs can be categorized according to the database model(s) that they support (such as relational or XML), the type(s) of computer they run on (from a server cluster to a mobile phone), the query language(s) used to access the database (such as SQL or XQuery), and their internal engineering, which affects performance, scalability, resilience, and security.

Applications and roles

Most organizations in developed countries today depend on databases for their business operations. Increasingly, databases are not only used to support the internal operations of the organization, but also to underpin its online interactions with customers and suppliers (see Enterprise software). Databases are not used only to hold administrative information, but are often embedded within applications to hold more specialized data: for example engineering data or economic models. Examples of database applications include computerized library systems, flight reservation systems, and computerized parts inventory systems.

Client-server or transactional DBMSs are often complex to maintain high performance, availability and security when many users are querying and updating the database at the same time. Personal, desktop-based database systems tend to be less complex. For example, FileMaker and Microsoft Access come with built-in graphical user interfaces.

General-purpose and special-purpose DBMSs

A DBMS has evolved into a complex software system and its development typically requires thousands of person-years of development effort.[4] Some general-purpose DBMSs such as Adabas, Oracle and DB2 have been undergoing upgrades since the 1970s. General-purpose DBMSs aim to meet the needs of as many applications as possible, which adds to the complexity. However, the fact that their development cost can be spread over a large number of users means that they are often the most cost-effective approach. However, a general-purpose DBMS is not always the optimal solution: in some cases a general-purpose DBMS may introduce unnecessary overhead. Therefore, there are many examples of systems that use special-purpose databases. A common example is an email system: email systems are designed to optimize the handling of email messages, and do not need significant portions of a general-purpose DBMS functionality.

Many databases have application software that accesses the database on behalf of end-users, without exposing the DBMS interface directly. Application programmers may use a wire protocol directly, or more likely through an application programming interface. Database designers and database administrators interact with the DBMS through dedicated interfaces to build and maintain the applications' databases, and thus need some more knowledge and understanding about how DBMSs operate and the DBMSs' external interfaces and tuning parameters.

General-purpose databases are usually developed by one organization or community of programmers, while a different group builds the applications that use it. In many companies, specialized database administrators maintain databases, run reports, and may work on code that runs on the databases themselves (rather than in the client application).

History

Following the technology progress in the areas of processors, computer memory, computer storage and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational,[5] SQL/relational, and post-relational.

The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

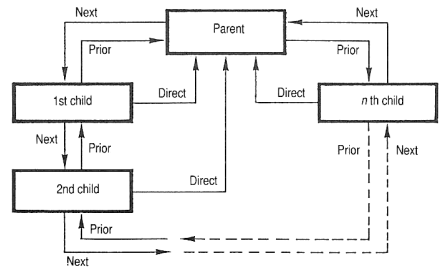
The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware became powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2014[6] they remain dominant except in niche areas. The dominant database language, standardised SQL for the relational model, has influenced database languages for other data models.[*citation needed*]

Object databases developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.

The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.

1960s, navigational DBMS

The introduction of the term *database* coincided with the availability of direct-access storage (disks and drums) from the mid-1960s onwards. The term represented a contrast with the tape-based systems of the past, allowing shared interactive use rather than daily batch processing. The Oxford English dictionary cites a 1962 report by the System Development Corporation of California as the first to use the term "data-base" in a specific technical sense.



A closed chain of records in a navigational database model (e.g. CODASYL), with ***next pointers***, ***prior pointers*** and ***direct pointers*** provided by keys in the various records.

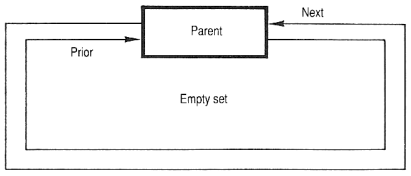


Illustration of an ***empty set***

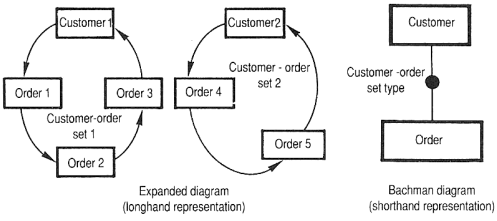


Illustration of a set type using a ***Bachman diagram***

The record set, basic structure of navigational (e.g. CODASYL) databse model. A set consists of one parent record (also called "the owner"), and n child records (also called members records)

Basic structure of navigational CODASYL database model.

As computers grew in speed and capability, a number of general-purpose database systems emerged; by the mid-1960s a number of such systems had come into commercial use. Interest in a standard began to grow, and Charles Bachman, author of one such product, the Integrated Data Store (IDS), founded the "Database Task Group" within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971 theDatabase Task Group delivered their standard, which generally became known as the "CODASYL approach", and soon a number of commercial products based on this approach entered the market.

The CODASYL approach relied on the "manual" navigation of a linked data set which was formed into a large network. Applications could find records by one of three methods:

- use of a primary key (known as a CALC key, typically implemented by hashing)
- navigating relationships (called sets) from one record to another
- scanning all the records in a sequential order.

Later systems added B-Trees to provide alternate access paths. Many CODASYL databases also added a very straightforward query language. However, in the final tally, CODASYL was very complex and required significant training and effort to produce useful applications.

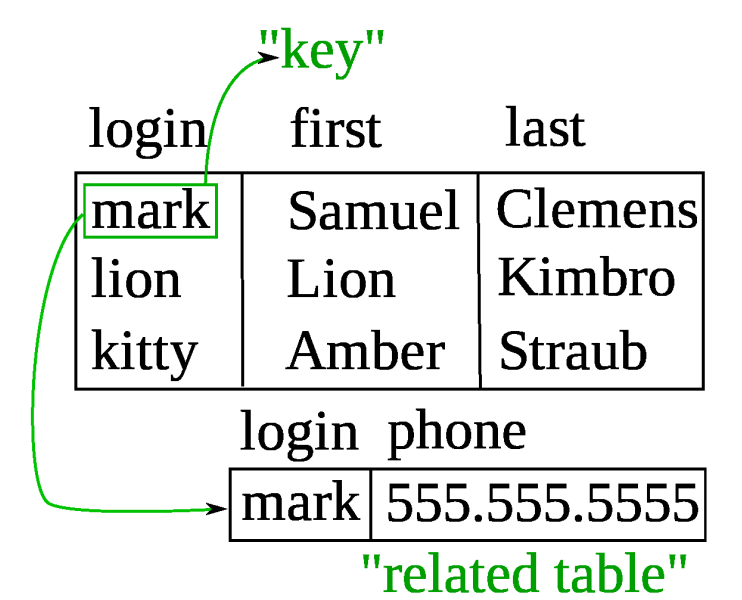
IBM also had their own DBMS system in 1968, known as *IMS*. IMS was a development of software written for the Apollo program on the System/360. IMS was generally similar in concept to CODASYL, but used a strict hierarchy for its model of data navigation instead of CODASYL's network model. Both concepts later became known as navigational databases due to the way data was accessed, and Bachman's 1973 Turing Award presentation was *The Programmer as Navigator*. IMS is classifiedWikipedia:Manual of Style/Words to watch#Unsupported attributions as a hierarchical database. IDMS and Cincom Systems' TOTAL database are classified as network databases. IMS remains in use as of 2014[6].

1970s, relational DBMS

Edgar Codd worked at IBM in San Jose, California, in one of their offshoot offices that was primarily involved in the development of hard disk systems. He was unhappy with the navigational model of the CODASYL approach, notably the lack of a "search" facility. In 1970, he wrote a number of papers that outlined a new approach to database construction that eventually culminated in the groundbreaking *A Relational Model of Data for Large Shared Data Banks*.[7]

In this paper, he described a new system for storing and working with large databases. Instead of records being stored in some sort of linked list of free-form records as in CODASYL, Codd's idea was to use a "table" of fixed-length records, with each table used for a different type of entity. A linked-list system would be very inefficient when storing "sparse" databases where some of the data for any one record could be left empty. The relational model solved this by splitting the data into a series of normalized tables (or *relations*), with optional elements being moved out of the main table to where they would take up room only if needed. Data may be freely inserted, deleted and edited in these tables, with the DBMS doing whatever maintenance needed to present a table view to the application/user.

The relational model also allowed the content of the database to evolve without constant rewriting of links and pointers. The relational part comes from entities referencing other entities in what is known as one-to-many relationship, like a traditional hierarchical model, and many-to-many relationship, like a navigational (network) model. Thus, a relational model can express both hierarchical and navigational models, as well as its native tabular model, allowing for pure or combined modeling in terms of these three models, as the application requires.



In the relational model, related records are linked together with a "key"

For instance, a common use of a database system is to track information about users, their name, login information, various addresses and phone numbers. In the navigational approach all of these data would be placed in a single record, and unused items would simply not be placed in the database. In the relational approach, the data would be *normalized* into a user table, an address table and a phone number table (for instance). Records would be created in these optional tables only if the address or phone numbers were actually provided.

Linking the information back together is the key to this system. In the relational model, some bit of information was used as a "key", uniquely defining a particular record. When information was being collected about a user, information stored in the optional tables would be found by searching for this key. For instance, if the login name of a user is unique, addresses and phone numbers for that user would be recorded with the login name as its key. This simple "re-linking" of related data back into a single collection is something that traditional computer languages are not designed for.

Just as the navigational approach would require programs to loop in order to collect records, the relational approach would require loops to collect information about any *one* record. Codd's solution to the necessary looping was a set-oriented language, a suggestion that would later spawn the ubiquitous SQL. Using a branch of mathematics known as tuple calculus, he demonstrated that such a system could support all the operations of normal databases (inserting, updating etc.) as well as providing a simple system for finding and returning *sets* of data in a single operation.

Codd's paper was picked up by two people at Berkeley, Eugene Wong and Michael Stonebraker. They started a project known as INGRES using funding that had already been allocated for a geographical database project and student programmers to produce code. Beginning in 1973, INGRES delivered its first test products which were generally ready for widespread use in 1979. INGRES was similar to System R in a number of ways, including the

use of a "language" for data access, known as QUEL. Over time, INGRES moved to the emerging SQL standard.

IBM itself did one test implementation of the relational model, PRTV, and a production one, Business System 12, both now discontinued. Honeywell wrote MRDS for Multics, and now there are two new implementations: Alphora Dataphor and Rel. Most other DBMS implementations usually called *relational* are actually SQL DBMSs.

In 1970, the University of Michigan began development of the MICRO Information Management System[8] based on D.L. Childs' Set-Theoretic Data model.[9][10][11] Micro was used to manage very large data sets by the US Department of Labor, the U.S. Environmental Protection Agency, and researchers from the University of Alberta, the University of Michigan, and Wayne State University. It ran on IBM mainframe computers using the Michigan Terminal System.[12] The system remained in production until 1998.

Integrated approach

In the 1970s and 1980s attempts were made to build database systems with integrated hardware and software. The underlying philosophy was that such integration would provide higher performance at lower cost. Examples were IBM System/38, the early offering of Teradata, and the Britton Lee, Inc. database machine.

Another approach to hardware support for database management was ICL's CAFS accelerator, a hardware disk controller with programmable search capabilities. In the long term, these efforts were generally unsuccessful because specialized database machines could not keep pace with the rapid development and progress of general-purpose computers. Thus most database systems nowadays are software systems running on general-purpose hardware, using general-purpose computer data storage. However this idea is still pursued for certain applications by some companies like Netezza and Oracle (Exadata).

Late 1970s, SQL DBMS

IBM started working on a prototype system loosely based on Codd's concepts as *System R* in the early 1970s. The first version was ready in 1974/5, and work then started on multi-table systems in which the data could be split so that all of the data for a record (some of which is optional) did not have to be stored in a single large "chunk". Subsequent multi-user versions were tested by customers in 1978 and 1979, by which time a standardized query language – SQL[*citation needed*] – had been added. Codd's ideas were establishing themselves as both workable and superior to CODASYL, pushing IBM to develop a true production version of System R, known as *SQL/DS*, and, later, *Database 2* (DB2).

Larry Ellison's Oracle started from a different chain, based on IBM's papers on System R, and beat IBM to market when the first version was released in 1978.[*citation needed*]

Stonebraker went on to apply the lessons from INGRES to develop a new database, Postgres, which is now known as PostgreSQL. PostgreSQL is often used for global mission critical applications (the .org and .info domain name registries use it as their primary data store, as do many large companies and financial institutions).

In Sweden, Codd's paper was also read and Mimer SQL was developed from the mid-1970s at Uppsala University. In 1984, this project was consolidated into an independent enterprise. In the early 1980s, Mimer introduced transaction handling for high robustness in applications, an idea that was subsequently implemented on most other DBMSs.

Another data model, the entity-relationship model, emerged in 1976 and gained popularity for database design as it emphasized a more familiar description than the earlier relational model. Later on, entity-relationship constructs were retrofitted as a data modeling construct for the relational model, and the difference between the two have become irrelevant.[*citation needed*]

1980s, on the desktop

The 1980s ushered in the age of desktop computing. The new computers empowered their users with spreadsheets like Lotus 1,2,3 and database software like dBASE. The dBASE product was lightweight and easy for any computer user to understand out of the box. C. Wayne Ratliff the creator of dBASE stated: “dBASE was different from programs like BASIC, C, FORTRAN, and COBOL in that a lot of the dirty work had already been done. The data manipulation is done by dBASE instead of by the user, so the user can concentrate on what he is doing, rather than having to mess with the dirty details of opening, reading, and closing files, and managing space allocation.“ [13] dBASE was one of the top selling software titles in the 1980s and early 1990s.

1980s, object-oriented

The 1980s, along with a rise in object oriented programming, saw a growth in how data in various databases were handled. Programmers and designers began to treat the data in their databases as objects. That is to say that if a person's data were in a database, that person's attributes, such as their address, phone number, and age, were now considered to belong to that person instead of being extraneous data. This allows for relations between data to be relations to objects and their attributes and not to individual fields.[14] The term "object-relational impedance mismatch" described the inconvenience of translating between programmed objects and database tables. Object databases and object-relational databases attempt to solve this problem by providing an object-oriented language (sometimes as extensions to SQL) that programmers can use as alternative to purely relational SQL. On the programming side, libraries known as object-relational mappings (ORMs) attempt to solve the same problem.

2000s, NoSQL and NewSQL

The next generation of post-relational databases in the 2000s became known as NoSQL databases, including fast key-value stores and document-oriented databases. XML databases are a type of structured document-oriented database that allows querying based on XML document attributes.

NoSQL databases are often very fast, do not require fixed table schemas, avoid join operations by storing denormalized data, and are designed to scale horizontally.

In recent years there was a high demand for massively distributed databases with high partition tolerance but according to the CAP theorem it is impossible for a distributed system to simultaneously provide consistency, availability and partition tolerance guarantees. A distributed system can satisfy any two of these guarantees at the same time, but not all three. For that reason many NoSQL databases are using what is called eventual consistency to provide both availability and partition tolerance guarantees with a maximum level of data consistency.

The most popular NoSQL systems include: MongoDB, Couchbase, Riak, memcached, Redis, CouchDB, Hazelcast, Apache Cassandra and HBase. Note that all are open-source software products.

A number of new relational databases continuing use of SQL but aiming for performance comparable to NoSQL are known as NewSQL.

Research

Database technology has been an active research topic since the 1960s, both in academia and in the research and development groups of companies (for example IBM Research). Research activity includes theory and development of prototypes. Notable research topics have included models, the atomic transaction concept and related concurrency control techniques, query languages and query optimization methods, RAID, and more.

The database research area has several dedicated academic journals (for example, ACM Transactions on Database Systems-TODS, Data and Knowledge Engineering-DKE) and annual conferences (e.g., ACM SIGMOD, ACM PODS, VLDB, IEEE ICDE).

Examples

One way to classify databases involves the type of their contents, for example: bibliographic, document-text, statistical, or multimedia objects. Another way is by their application area, for example: accounting, music compositions, movies, banking, manufacturing, or insurance. A third way is by some technical aspect, such as the database structure or interface type. This section lists a few of the adjectives used to characterize different kinds of databases.

- An in-memory database is a database that primarily resides in main memory, but is typically backed-up by non-volatile computer data storage. Main memory databases are faster than disk databases, and so are often used where response time is critical, such as in telecommunications network equipment.SAP HANA platform is a very hot topic for in-memory database. By May 2012, HANA was able to run on servers with 100TB main memory powered by IBM. The co founder of the company claimed that the system was big enough to run the 8 largest SAP customers.
- An active database includes an event-driven architecture which can respond to conditions both inside and outside the database. Possible uses include security monitoring, alerting, statistics gathering and authorization. Many databases provide active database features in the form of database triggers.
- A cloud database relies on cloud technology. Both the database and most of its DBMS reside remotely, "in the cloud", while its applications are both developed by programmers and later maintained and utilized by (application's) end-users through a web browser and Open APIs.
- Data warehouses archive data from operational databases and often from external sources such as market research firms. The warehouse becomes the central source of data for use by managers and other end-users who may not have access to operational data. For example, sales data might be aggregated to weekly totals and converted from internal product codes to use UPCs so that they can be compared with ACNielsen data. Some basic and essential components of data warehousing include retrieving, analyzing, and mining data, transforming, loading and managing data so as to make them available for further use.
- A deductive database combines logic programming with a relational database, for example by using the Datalog language.
- A distributed database is one in which both the data and the DBMS span multiple computers.
- A document-oriented database is designed for storing, retrieving, and managing document-oriented, or semi structured data, information. Document-oriented databases are one of the main categories of NoSQL databases.
- An embedded database system is a DBMS which is tightly integrated with an application software that requires access to stored data in such a way that the DBMS is hidden from the application's end-users and requires little or no ongoing maintenance.[15]
- **End-user databases** consist of data developed by individual end-users. Examples of these are collections of documents, spreadsheets, presentations, multimedia, and other files. Several products exist to support such databases. Some of them are much simpler than full fledged DBMSs, with more elementary DBMS functionality.

- A federated database system comprises several distinct databases, each with its own DBMS. It is handled as a single database by a federated database management system (FDBMS), which transparently integrates multiple autonomous DBMSs, possibly of different types (in which case it would also be a heterogeneous database system), and provides them with an integrated conceptual view.
- Sometimes the term *multi-database* is used as a synonym to federated database, though it may refer to a less integrated (e.g., without an FDBMS and a managed integrated schema) group of databases that cooperate in a single application. In this case typically middleware is used for distribution, which typically includes an atomic commit protocol (ACP), e.g., the two-phase commit protocol, to allow distributed (global) transactions across the participating databases.
- A graph database is a kind of NoSQL database that uses graph structures with nodes, edges, and properties to represent and store information. General graph databases that can store any graph are distinct from specialized graph databases such as triplestores and network databases.
- In a hypertext or hypermedia database, any word or a piece of text representing an object, e.g., another piece of text, an article, a picture, or a film, can be hyperlinked to that object. Hypertext databases are particularly useful for organizing large amounts of disparate information. For example, they are useful for organizing online encyclopedias, where users can conveniently jump around the text. The World Wide Web is thus a large distributed hypertext database.
- A knowledge base (abbreviated **KB**, **kb** or Δ[16]) is a special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. Also a collection of data representing problems with their solutions and related experiences.
- A mobile database can be carried on or synchronized from a mobile computing device.
- Operational databases store detailed data about the operations of an organization. They typically process relatively high volumes of updates using transactions. Examples include customer databases that record contact, credit, and demographic information about a business' customers, personnel databases that hold information such as salary, benefits, skills data about employees, enterprise resource planning systems that record details about product components, parts inventory, and financial databases that keep track of the organization's money, accounting and financial dealings.
- A parallel database seeks to improve performance through parallelization for tasks such as loading data, building indexes and evaluating queries.

  The major parallel DBMS architectures which are induced by the underlying hardware architecture are:

  - **Shared memory architecture**, where multiple processors share the main memory space, as well as other data storage.
  - **Shared disk architecture**, where each processing unit (typically consisting of multiple processors) has its own main memory, but all units share the other storage.
  - **Shared nothing architecture**, where each processing unit has its own main memory and other storage.
- Probabilistic databases employ fuzzy logic to draw inferences from imprecise data.
- Real-time databases process transactions fast enough for the result to come back and be acted on right away.
- A spatial database can store the data with multidimensional features. The queries on such data include location based queries, like "Where is the closest hotel in my area?".
- A temporal database has built-in time aspects, for example a temporal data model and a temporal version of SQL. More specifically the temporal aspects usually include valid-time and transaction-time.
- A terminology-oriented database builds upon an object-oriented database, often customized for a specific field.
- An unstructured data database is intended to store in a manageable and protected way diverse objects that do not fit naturally and conveniently in common databases. It may include email messages, documents, journals,

multimedia objects, etc. The name may be misleading since some objects can be highly structured. However, the entire possible object collection does not fit into a predefined structured framework. Most established DBMSs now support unstructured data in various ways, and new dedicated DBMSs are emerging.

Design and modeling

The first task of a database designer is to produce a conceptual data model that reflects the structure of the information to be held in the database. A common approach to this is to develop an entity-relationship model, often with the aid of drawing tools. Another popular approach is the Unified Modeling Language. A successful data model will accurately reflect the possible state of the external world being modeled: for example, if people can have more than one phone number, it will allow this information to be captured. Designing a good conceptual data model requires a good understanding of the application domain; it typically involves asking deep questions about the things of interest to an organisation, like "can a customer also be a supplier?", or "if a product is sold with two different forms of packaging, are those the same product or different products?", or "if a plane flies from New York to Dubai via Frankfurt, is that one flight or two (or maybe even three)?". The answers to these questions establish definitions of the terminology used for entities (customers, products, flights, flight segments) and their relationships and attributes.

Producing the conceptual data model sometimes involves input from business processes, or the analysis of workflow in the organization. This can help to establish what information is needed in the database, and what can be left out. For example, it can help when deciding whether the database needs to hold historic data as well as current data.

Having produced a conceptual data model that users are happy with, the next stage is to translate this into a schema that implements the relevant data structures within the database. This process is often called logical database design, and the output is a logical data model expressed in the form of a schema. Whereas the conceptual data model is (in theory at least) independent of the choice of database technology, the logical data model will be expressed in terms of a particular database model supported by the chosen DBMS. (The terms *data model* and *database model* are often used interchangeably, but in this article we use *data model* for the design of a specific database, and *database model* for the modelling notation used to express that design.)

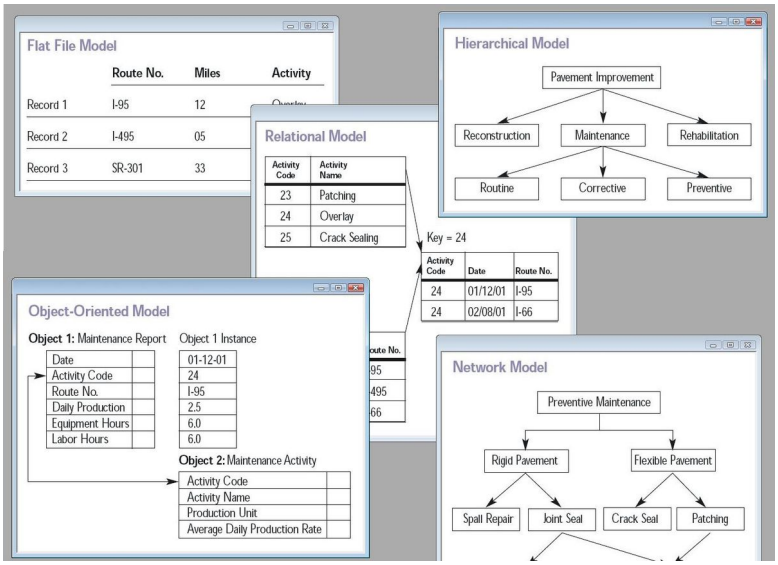
The most popular database model for general-purpose databases is the relational model, or more precisely, the relational model as represented by the SQL language. The process of creating a logical database design using this model uses a methodical approach known as normalization. The goal of normalization is to ensure that each elementary "fact" is only recorded in one place, so that insertions, updates, and deletions automatically maintain consistency.

The final stage of database design is to make the decisions that affect performance, scalability, recovery, security, and the like. This is often called *physical database design*. A key goal during this stage is data independence, meaning that the decisions made for performance optimization purposes should be invisible to end-users and applications. Physical design is driven mainly by performance requirements, and requires a good knowledge of the expected workload and access patterns, and a deep understanding of the features offered by the chosen DBMS.

Another aspect of physical database design is security. It involves both defining access control to database objects as well as defining security levels and methods for the data itself.

Models

A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model (or the SQL approximation of relational), which uses a table-based format.



Collage of five types of database models.

Common logical data models for databases include:

- Hierarchical database model
- Network model
- Relational model
- Entity–relationship model
  - Enhanced entity–relationship model
- Object model
- Document model
- Entity–attribute–value model
- Star schema

An object-relational database combines the two related structures.

Physical data models include:

- Inverted index
- Flat file

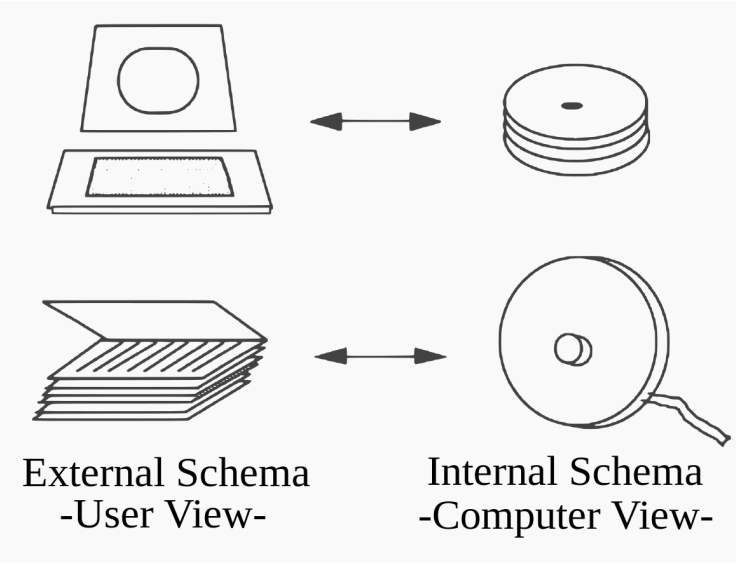
Other models include:

- Associative model
- Multidimensional model
- Multivalue model
- Semantic model
- XML database
- Named graph

External, conceptual, and internal views

A database management system provides three views of the database data:

- The **external level** defines how each group of end-users sees the organization of data in the database. A single database can have any number of views at the external level.
- The **conceptual level** unifies the various external views into a compatible global view. It provides the synthesis of all the external views. It is out of the scope of the various database end-users, and is rather of interest to database application developers and database administrators.



Traditional view of data[17]

- The **internal level** (or *physical level*) is the internal organization of data inside a DBMS (see Implementation section below). It is concerned with cost, performance, scalability and other operational matters. It deals with storage layout of the data, using storage structures such as indexes to enhance performance. Occasionally it stores data of individual views (materialized views), computed from generic data, if performance justification exists for such redundancy. It balances all the external views' performance requirements, possibly conflicting, in an attempt to optimize overall performance across all activities.

While there is typically only one conceptual (or logical) and physical (or internal) view of the data, there can be any number of different external views. This allows users to see database information in a more business-related way rather than from a technical, processing viewpoint. For example, a financial department of a company needs the payment details of all employees as part of the company's expenses, but does not need details about employees that are the interest of the human resources department. Thus different departments need different *views* of the company's database.

The three-level database architecture relates to the concept of *data independence* which was one of the major initial driving forces of the relational model. The idea is that changes made at a certain level do not affect the view at a higher level. For example, changes in the internal level do not affect application programs written using conceptual level interfaces, which reduces the impact of making physical changes to improve performance.

The conceptual view provides a level of indirection between internal and external. On one hand it provides a common view of the database, independent of different external view structures, and on the other hand it abstracts away details of how the data is stored or managed (internal level). In principle every level, and even every external view, can be presented by a different data model. In practice usually a given DBMS uses the same data model for both the external and the conceptual levels (e.g., relational model). The internal level, which is hidden inside the DBMS and depends on its implementation (see Implementation section below), requires a different level of detail and uses its own types of data structure types.

Separating the *external*, *conceptual* and *internal* levels was a major feature of the relational database model implementations that dominate 21st century databases.

Languages

Database languages are special-purpose languages, which do one or more of the following:

- Data definition language – defines data types and the relationships among them
- Data manipulation language – performs tasks such as inserting, updating, or deleting data occurrences
- Query language – allows searching for information and computing derived information

Database languages are specific to a particular data model. Notable examples include:

- SQL combines the roles of data definition, data manipulation, and query in a single language. It was one of the first commercial languages for the relational model, although it departs in some respects from the relational model as described by Codd (for example, the rows and columns of a table can be ordered). SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. The standards have been regularly enhanced since and is supported (with varying degrees of conformance) by all mainstream commercial relational DBMSs.
- OQL is an object model language standard (from the Object Data Management Group). It has influenced the design of some of the newer query languages like JDOQL and EJB QL.
- XQuery is a standard XML query language implemented by XML database systems such as MarkLogic and eXist, by relational databases with XML capability such as Oracle and DB2, and also by in-memory XML processors such as Saxon.
- SQL/XML combines XQuery with SQL.

A database language may also incorporate features like:

- DBMS-specific Configuration and storage engine management
- Computations to modify query results, like counting, summing, averaging, sorting, grouping, and cross-referencing
- Constraint enforcement (e.g. in an automotive database, only allowing one engine type per car)
- Application programming interface version of the query language, for programmer convenience

Performance, security, and availability

Because of the critical importance of database technology to the smooth running of an enterprise, database systems include complex mechanisms to deliver the required performance, security, and availability, and allow database administrators to control the use of these features.

Storage

Database storage is the container of the physical materialization of a database. It comprises the *internal* (physical) *level* in the database architecture. It also contains all the information needed (e.g., metadata, "data about the data", and internal data structures) to reconstruct the *conceptual level* and *external level* from the internal level when needed. Putting data into permanent storage is generally the responsibility of the database engine a.k.a. "storage engine". Though typically accessed by a DBMS through the underlying operating system (and often utilizing the operating systems' file systems as intermediates for storage layout), storage properties and configuration setting are extremely important for the efficient operation of the DBMS, and thus are closely maintained by database administrators. A DBMS, while in operation, always has its database residing in several types of storage (e.g., memory and external storage). The database data and the additional needed information, possibly in very large amounts, are coded into bits. Data typically reside in the storage in structures that look completely different from the way the data look in the conceptual and external levels, but in ways that attempt to optimize (the best possible) these levels' reconstruction when needed by users and programs, as well as for computing additional types of needed information from the data (e.g., when querying the database).

Some DBMSs support specifying which character encoding was used to store data, so multiple encodings can be used in the same database.

Various low-level database storage structures are used by the storage engine to serialize the data model so it can be written to the medium of choice. Techniques such as indexing may be used to improve performance. Conventional storage is row-oriented, but there are also column-oriented and correlation databases.

Materialized views

Often storage redundancy is employed to increase performance. A common example is storing *materialized views*, which consist of frequently needed *external views* or query results. Storing such views saves the expensive computing of them each time they are needed. The downsides of materialized views are the overhead incurred when updating them to keep them synchronized with their original updated database data, and the cost of storage redundancy.

Replication

Occasionally a database employs storage redundancy by database objects replication (with one or more copies) to increase data availability (both to improve performance of simultaneous multiple end-user accesses to a same database object, and to provide resiliency in a case of partial failure of a distributed database). Updates of a replicated object need to be synchronized across the object copies. In many cases the entire database is replicated.

Security

Database security deals with all various aspects of protecting the database content, its owners, and its users. It ranges from protection from intentional unauthorized database uses to unintentional database accesses by unauthorized entities (e.g., a person or a computer program).

Database access control deals with controlling who (a person or a certain computer program) is allowed to access what information in the database. The information may comprise specific database objects (e.g., record types, specific records, data structures), certain computations over certain objects (e.g., query types, or specific queries), or utilizing specific access paths to the former (e.g., using specific indexes or other data structures to access information). Database access controls are set by special authorized (by the database owner) personnel that uses dedicated protected security DBMS interfaces.

This may be managed directly on an individual basis, or by the assignment of individuals and privileges to groups, or (in the most elaborate models) through the assignment of individuals and groups to roles which are then granted entitlements. Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called "subschemas". For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data. If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases.

Data security in general deals with protecting specific chunks of data, both physically (i.e., from corruption, or destruction, or removal; e.g., see physical security), or the interpretation of them, or parts of them to meaningful information (e.g., by looking at the strings of bits that they comprise, concluding specific valid credit-card numbers; e.g., see data encryption).

Change and access logging records who accessed which attributes, what was changed, and when it was changed. Logging services allow for a forensic database audit later by keeping a record of access occurrences and changes. Sometimes application-level code is used to record changes rather than leaving this to the database. Monitoring can be set up to attempt to detect security breaches.

Transactions and concurrency

Database transactions can be used to introduce some level of fault tolerance and data integrity after recovery from a crash. A database transaction is a unit of work, typically encapsulating a number of operations over a database (e.g., reading a database object, writing, acquiring lock, etc.), an abstraction supported in database and also other systems. Each transaction has well defined boundaries in terms of which program/code executions are included in that transaction (determined by the transaction's programmer via special transaction commands).

The acronym ACID describes some ideal properties of a database transaction: Atomicity, Consistency, Isolation, and Durability.

Migration

See also section Database migration in article Data migration

A database built with one DBMS is not portable to another DBMS (i.e., the other DBMS cannot run it). However, in some situations it is desirable to move, migrate a database from one DBMS to another. The reasons are primarily economical (different DBMSs may have different total costs of ownership or TCOs), functional, and operational (different DBMSs may have different capabilities). The migration involves the database's transformation from one DBMS type to another. The transformation should maintain (if possible) the database related application (i.e., all related application programs) intact. Thus, the database's conceptual and external architectural levels should be maintained in the transformation. It may be desired that also some aspects of the architecture internal level are maintained. A complex or large database migration may be a complicated and costly (one-time) project by itself, which should be factored into the decision to migrate. This in spite of the fact that tools may exist to help migration between specific DBMSs. Typically a DBMS vendor provides tools to help importing databases from other popular DBMSs.

Building, maintaining, and tuning

After designing a database for an application, the next stage is building the database. Typically an appropriate general-purpose DBMS can be selected to be utilized for this purpose. A DBMS provides the needed user interfaces to be utilized by database administrators to define the needed application's data structures within the DBMS's respective data model. Other user interfaces are used to select needed DBMS parameters (like security related, storage allocation parameters, etc.).

When the database is ready (all its data structures and other needed components are defined) it is typically populated with initial application's data (database initialization, which is typically a distinct project; in many cases using specialized DBMS interfaces that support bulk insertion) before making it operational. In some cases the database becomes operational while empty of application data, and data is accumulated during its operation.

After the database is created, initialised and populated it needs to be maintained. Various database parameters may need changing and the database may need to be tuned (tuning) for better performance; application's data structures may be changed or added, new related application programs may be written to add to the application's functionality, etc. Databases are often confused with spreadsheets such as Microsoft Excel (Microsoft Access is a database management system, Excel is a spreadsheet program). Both can be used to store information, however a database is more efficient and flexible at storing large amounts of data. Below is a simple comparison of spreadsheets and databases.

Spreadsheet strengths	Spreadsheet Weaknesses
Very simple data storage	Data integrity problems, including inaccurate, inconsistent and out of date data and formulas.
Relatively easy to use	Difficult to validate data e.g. an incorrect formula
Require less planning	

Database strengths	Database Weaknesses
Methods for keeping data up to date and consistent	Require more planning and designing
Data is of higher quality than data stored in spreadsheets	Harder to change structure once database is built
Good for storing and organizing information.	Requires more technical knowledge to administrate

Backup and restore

Sometimes it is desired to bring a database back to a previous state (for many reasons, e.g., cases when the database is found corrupted due to a software error, or if it has been updated with erroneous data). To achieve this a **backup** operation is done occasionally or continuously, where each desired database state (i.e., the values of its data and their embedding in database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When this state is needed, i.e., when it is decided by a database administrator to bring the database back to this state (e.g., by specifying this state by a desired point in time when the database was in this state), these files are utilized to **restore** that state.

Other

Other DBMS features might include:

- Database logs
- Graphics component for producing graphs and charts, especially in a data warehouse system
- **Query optimizer –** Performs query optimization on every query to choose for it the most efficient *query plan* (a partial order (tree) of operations) to be executed to compute the query result. May be specific to a particular storage engine.
- Tools or hooks for database design, application programming, application program maintenance, database performance analysis and monitoring, database configuration monitoring, DBMS hardware configuration (a DBMS and related database may span computers, networks, and storage units) and related database mapping (especially for a distributed DBMS), storage allocation and database layout monitoring, storage migration, etc.

References

[1] Jeffrey Ullman 1997: *First course in database systems*, Prentice–Hall Inc., Simon & Schuster, Page 1, ISBN 0-13-861337-0.

[2] Tsitchizris, D. C. and F. H. Lochovsky (1982). *Data Models.* Englewood-Cliffs, Prentice–Hall.

[3] Beynon-Davies P. (2004). *Database Systems* 3rd Edition. Palgrave, Basingstoke, UK. ISBN 1-4039-1601-2

[4] . This article quotes a development time of 5 years involving 750 people for DB2 release 9 alone

[5] (Turing Award Lecture 1973)

[6] http://en.wikipedia.org/w/index.php?title=Database&action=edit

[7] Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks" (http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf). In: *Communications of the ACM* 13 (6): 377–387.

[8] William Hershey and Carol Easthope, "A set theoretic data structure and retrieval language" (https://docs.google.com/open?id=0B4t_NX-QeWDYNmVhYjAwMWMtYzc3ZS00YjI0LWJhMjgtZTYyODZmNmFkNThh), Spring Joint Computer Conference, May 1972 in *ACM SIGIR Forum*, Volume 7, Issue 4 (December 1972), pp. 45–55, DOI= 10.1145/1095495.1095500 (http://doi.acm.org/10.1145/1095495.1095500)

[9] Ken North, "Sets, Data Models and Data Independence" (http://drdobbs.com/blogs/database/228700616), *Dr. Dobb's*, 10 March 2010

[10] *Description of a set-theoretic data structure* (http://hdl.handle.net/2027.42/4163), D. L. Childs, 1968, Technical Report 3 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

[11] *Feasibility of a Set-Theoretic Data Structure : A General Structure Based on a Reconstituted Definition of Relation* (http://hdl.handle.net/2027.42/4164), D. L. Childs, 1968, Technical Report 6 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

[12] *MICRO Information Management System (Version 5.0) Reference Manual* (http://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B4t_NX-QeWDYZGMwOTRmOTItZTg2Zi00YmJkLTg4MTktN2E4MWU0YmZlMjE3), M.A. Kahn, D.L. Rumelhart, and B.L. Bronson, October 1977, Institute of Labor and Industrial Relations (ILIR), University of Michigan and Wayne State University

[13] Interview with Wayne Ratliff (http://www.foxprohistory.org/interview_wayne_ratliff.htm). The FoxPro History. Retrieved on 2013-07-12.

[14] Development of an object-oriented DBMS; Portland, Oregon, United States; Pages: 472 – 482; 1986; ISBN 0-89791-204-7

[15] Graves, Steve. "COTS Databases For Embedded Systems" (http://www.embedded-computing.com/articles/id/?2020), *Embedded Computing Design* magazine, January 2007. Retrieved on August 13, 2008.

[16] Argumentation in Artificial Intelligence by Iyad Rahwan, Guillermo R. Simari

[17] itl.nist.gov (1993) *Integration Definition for Information Modeling (IDEFIX)* (http://www.itl.nist.gov/fipspubs/idef1x.doc). 21 December 1993.

Further reading

- Ling Liu and Tamer M. Özsu (Eds.) (2009). " Encyclopedia of Database Systems (http://www.springer.com/computer/database+management+&+information+retrieval/book/978-0-387-49616-0), 4100 p. 60 illus. ISBN 978-0-387-49616-0.
- Beynon-Davies, P. (2004). Database Systems. 3rd Edition. Palgrave, Houndmills, Basingstoke.
- Connolly, Thomas and Carolyn Begg. *Database Systems.* New York: Harlow, 2002.
- Date, C. J. (2003). *An Introduction to Database Systems, Fifth Edition*. Addison Wesley. ISBN 0-201-51381-1.
- Gray, J. and Reuter, A. *Transaction Processing: Concepts and Techniques*, 1st edition, Morgan Kaufmann Publishers, 1992.
- Kroenke, David M. and David J. Auer. *Database Concepts.* 3rd ed. New York: Prentice, 2007.
- Raghu Ramakrishnan and Johannes Gehrke, *Database Management Systems* (http://pages.cs.wisc.edu/~dbbook/)
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts* (http://www.db-book.com/)
- Discussion on database systems, (http://www.bbconsult.co.uk/Documents/Database-Systems.docx)
- Lightstone, S.; Teorey, T.; Nadeau, T. (2007). *Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more*. Morgan Kaufmann Press. ISBN 0-12-369389-6.
- Teorey, T.; Lightstone, S. and Nadeau, T. *Database Modeling & Design: Logical Design*, 4th edition, Morgan Kaufmann Press, 2005. ISBN 0-12-685352-5

External links

- Database (http://www.dmoz.org/Computers/Data_Formats/Database) on the Open Directory Project

Database system

A **database** is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database management systems (**DBMS**s) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker Pro. A database is not generally portable across different DBMSs, but different DBMSs can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one database.

Terminology and overview

Formally, "database" refers to the data themselves and supporting data structures. Databases are created to operate large quantities of information by inputting, storing, retrieving, and managing that information. Databases are set up so that one set of software programs provides all users with access to all the data.

A "database management system" (DBMS) is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term "database" when used casually often refers to both a DBMS and the data it manipulates.

Outside the world of professional information technology, the term *database* is sometimes used casually to refer to any collection of data (perhaps a spreadsheet, maybe even a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.[1]

The interactions catered for by most existing DBMSs fall into four main groups:

- **Data definition –** Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- **Update –** Inserting, modifying, and deleting data.
- **Retrieval –** Obtaining information either for end-user queries and reports or for processing by applications.
- **Administration –** Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

Both a database and its DBMS conform to the principles of a particular database model.[2] "Database system" refers collectively to the database model, database management system, and database.[3]

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used for recovery of data if any of the disks fail. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications. DBMSs may be built around a custom multitasking kernel with built-in networking support, but modern DBMSs typically rely on a standard operating system to provide these functions. [*citation needed*] Since DBMSs comprise a significant economical market, computer and storage vendors often take into account DBMS requirements in their own development plans.[*citation needed*]

Databases and DBMSs can be categorized according to the database model(s) that they support (such as relational or XML), the type(s) of computer they run on (from a server cluster to a mobile phone), the query language(s) used to access the database (such as SQL or XQuery), and their internal engineering, which affects performance, scalability, resilience, and security.

Applications and roles

Most organizations in developed countries today depend on databases for their business operations. Increasingly, databases are not only used to support the internal operations of the organization, but also to underpin its online interactions with customers and suppliers (see Enterprise software). Databases are not used only to hold administrative information, but are often embedded within applications to hold more specialized data: for example engineering data or economic models. Examples of database applications include computerized library systems, flight reservation systems, and computerized parts inventory systems.

Client-server or transactional DBMSs are often complex to maintain high performance, availability and security when many users are querying and updating the database at the same time. Personal, desktop-based database systems tend to be less complex. For example, FileMaker and Microsoft Access come with built-in graphical user interfaces.

General-purpose and special-purpose DBMSs

A DBMS has evolved into a complex software system and its development typically requires thousands of person-years of development effort.[4] Some general-purpose DBMSs such as Adabas, Oracle and DB2 have been undergoing upgrades since the 1970s. General-purpose DBMSs aim to meet the needs of as many applications as possible, which adds to the complexity. However, the fact that their development cost can be spread over a large number of users means that they are often the most cost-effective approach. However, a general-purpose DBMS is not always the optimal solution: in some cases a general-purpose DBMS may introduce unnecessary overhead. Therefore, there are many examples of systems that use special-purpose databases. A common example is an email system: email systems are designed to optimize the handling of email messages, and do not need significant portions of a general-purpose DBMS functionality.

Many databases have application software that accesses the database on behalf of end-users, without exposing the DBMS interface directly. Application programmers may use a wire protocol directly, or more likely through an application programming interface. Database designers and database administrators interact with the DBMS through dedicated interfaces to build and maintain the applications' databases, and thus need some more knowledge and understanding about how DBMSs operate and the DBMSs' external interfaces and tuning parameters.

General-purpose databases are usually developed by one organization or community of programmers, while a different group builds the applications that use it. In many companies, specialized database administrators maintain databases, run reports, and may work on code that runs on the databases themselves (rather than in the client application).

History

Following the technology progress in the areas of processors, computer memory, computer storage and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational,[5] SQL/relational, and post-relational.

The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

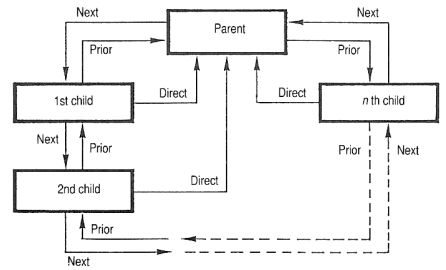
The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware became powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2014[6] they remain dominant except in niche areas. The dominant database language, standardised SQL for the relational model, has influenced database languages for other data models.[*citation needed*]

Object databases developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.

The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.

1960s, navigational DBMS

The introduction of the term *database* coincided with the availability of direct-access storage (disks and drums) from the mid-1960s onwards. The term represented a contrast with the tape-based systems of the past, allowing shared interactive use rather than daily batch processing. The Oxford English dictionary cites a 1962 report by the System Development Corporation of California as the first to use the term "data-base" in a specific technical sense.



A closed chain of records in a navigational database model (e.g. CODASYL), with ***next pointers***, ***prior pointers*** and ***direct pointers*** provided by keys in the various records.

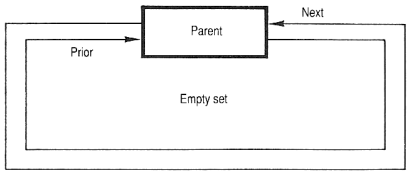


Illustration of an ***empty set***

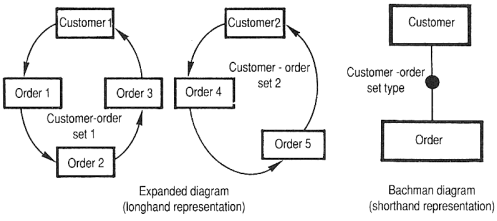


Illustration of a set type using a ***Bachman diagram***

The record set, basic structure of navigational (e.g. CODASYL) databse model. A set consists of one parent record (also called "the owner"), and n child records (also called members records)

Basic structure of navigational CODASYL database model.

As computers grew in speed and capability, a number of general-purpose database systems emerged; by the mid-1960s a number of such systems had come into commercial use. Interest in a standard began to grow, and Charles Bachman, author of one such product, the Integrated Data Store (IDS), founded the "Database Task Group" within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971 theDatabase Task Group delivered their standard, which generally became known as the "CODASYL approach", and soon a number of commercial products based on this approach entered the market.

The CODASYL approach relied on the "manual" navigation of a linked data set which was formed into a large network. Applications could find records by one of three methods:

- use of a primary key (known as a CALC key, typically implemented by hashing)
- navigating relationships (called sets) from one record to another
- scanning all the records in a sequential order.

Later systems added B-Trees to provide alternate access paths. Many CODASYL databases also added a very straightforward query language. However, in the final tally, CODASYL was very complex and required significant training and effort to produce useful applications.

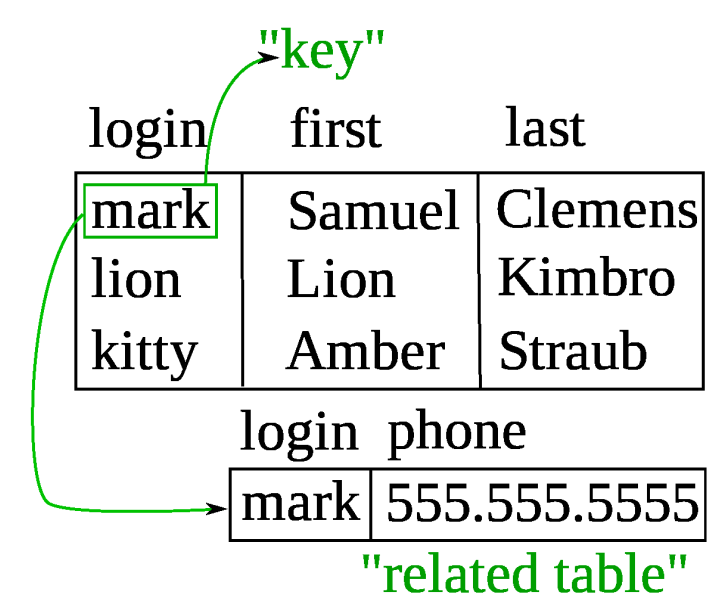
IBM also had their own DBMS system in 1968, known as *IMS*. IMS was a development of software written for the Apollo program on the System/360. IMS was generally similar in concept to CODASYL, but used a strict hierarchy for its model of data navigation instead of CODASYL's network model. Both concepts later became known as navigational databases due to the way data was accessed, and Bachman's 1973 Turing Award presentation was *The Programmer as Navigator*. IMS is classifiedWikipedia:Manual of Style/Words to watch#Unsupported attributions as a hierarchical database. IDMS and Cincom Systems' TOTAL database are classified as network databases. IMS remains in use as of 2014[6].

1970s, relational DBMS

Edgar Codd worked at IBM in San Jose, California, in one of their offshoot offices that was primarily involved in the development of hard disk systems. He was unhappy with the navigational model of the CODASYL approach, notably the lack of a "search" facility. In 1970, he wrote a number of papers that outlined a new approach to database construction that eventually culminated in the groundbreaking *A Relational Model of Data for Large Shared Data Banks*.[6]

In this paper, he described a new system for storing and working with large databases. Instead of records being stored in some sort of linked list of free-form records as in CODASYL, Codd's idea was to use a "table" of fixed-length records, with each table used for a different type of entity. A linked-list system would be very inefficient when storing "sparse" databases where some of the data for any one record could be left empty. The relational model solved this by splitting the data into a series of normalized tables (or *relations*), with optional elements being moved out of the main table to where they would take up room only if needed. Data may be freely inserted, deleted and edited in these tables, with the DBMS doing whatever maintenance needed to present a table view to the application/user.

The relational model also allowed the content of the database to evolve without constant rewriting of links and pointers. The relational part comes from entities referencing other entities in what is known as one-to-many relationship, like a traditional hierarchical model, and many-to-many relationship, like a navigational (network) model. Thus, a relational model can express both hierarchical and navigational models, as well as its native tabular model, allowing for pure or combined modeling in terms of these three models, as the application requires.



In the relational model, related records are linked together with a "key"

For instance, a common use of a database system is to track information about users, their name, login information, various addresses and phone numbers. In the navigational approach all of these data would be placed in a single record, and unused items would simply not be placed in the database. In the relational approach, the data would be *normalized* into a user table, an address table and a phone number table (for instance). Records would be created in these optional tables only if the address or phone numbers were actually provided.

Linking the information back together is the key to this system. In the relational model, some bit of information was used as a "key", uniquely defining a particular record. When information was being collected about a user, information stored in the optional tables would be found by searching for this key. For instance, if the login name of a user is unique, addresses and phone numbers for that user would be recorded with the login name as its key. This simple "re-linking" of related data back into a single collection is something that traditional computer languages are not designed for.

Just as the navigational approach would require programs to loop in order to collect records, the relational approach would require loops to collect information about any *one* record. Codd's solution to the necessary looping was a set-oriented language, a suggestion that would later spawn the ubiquitous SQL. Using a branch of mathematics known as tuple calculus, he demonstrated that such a system could support all the operations of normal databases (inserting, updating etc.) as well as providing a simple system for finding and returning *sets* of data in a single operation.

Codd's paper was picked up by two people at Berkeley, Eugene Wong and Michael Stonebraker. They started a project known as INGRES using funding that had already been allocated for a geographical database project and student programmers to produce code. Beginning in 1973, INGRES delivered its first test products which were generally ready for widespread use in 1979. INGRES was similar to System R in a number of ways, including the

use of a "language" for data access, known as QUEL. Over time, INGRES moved to the emerging SQL standard.

IBM itself did one test implementation of the relational model, PRTV, and a production one, Business System 12, both now discontinued. Honeywell wrote MRDS for Multics, and now there are two new implementations: Alphora Dataphor and Rel. Most other DBMS implementations usually called *relational* are actually SQL DBMSs.

In 1970, the University of Michigan began development of the MICRO Information Management System[7] based on D.L. Childs' Set-Theoretic Data model.[8][9][10] Micro was used to manage very large data sets by the US Department of Labor, the U.S. Environmental Protection Agency, and researchers from the University of Alberta, the University of Michigan, and Wayne State University. It ran on IBM mainframe computers using the Michigan Terminal System.[11] The system remained in production until 1998.

Integrated approach

In the 1970s and 1980s attempts were made to build database systems with integrated hardware and software. The underlying philosophy was that such integration would provide higher performance at lower cost. Examples were IBM System/38, the early offering of Teradata, and the Britton Lee, Inc. database machine.

Another approach to hardware support for database management was ICL's CAFS accelerator, a hardware disk controller with programmable search capabilities. In the long term, these efforts were generally unsuccessful because specialized database machines could not keep pace with the rapid development and progress of general-purpose computers. Thus most database systems nowadays are software systems running on general-purpose hardware, using general-purpose computer data storage. However this idea is still pursued for certain applications by some companies like Netezza and Oracle (Exadata).

Late 1970s, SQL DBMS

IBM started working on a prototype system loosely based on Codd's concepts as *System R* in the early 1970s. The first version was ready in 1974/5, and work then started on multi-table systems in which the data could be split so that all of the data for a record (some of which is optional) did not have to be stored in a single large "chunk". Subsequent multi-user versions were tested by customers in 1978 and 1979, by which time a standardized query language – SQL[*citation needed*] – had been added. Codd's ideas were establishing themselves as both workable and superior to CODASYL, pushing IBM to develop a true production version of System R, known as *SQL/DS*, and, later, *Database 2* (DB2).

Larry Ellison's Oracle started from a different chain, based on IBM's papers on System R, and beat IBM to market when the first version was released in 1978.[*citation needed*]

Stonebraker went on to apply the lessons from INGRES to develop a new database, Postgres, which is now known as PostgreSQL. PostgreSQL is often used for global mission critical applications (the .org and .info domain name registries use it as their primary data store, as do many large companies and financial institutions).

In Sweden, Codd's paper was also read and Mimer SQL was developed from the mid-1970s at Uppsala University. In 1984, this project was consolidated into an independent enterprise. In the early 1980s, Mimer introduced transaction handling for high robustness in applications, an idea that was subsequently implemented on most other DBMSs.

Another data model, the entity-relationship model, emerged in 1976 and gained popularity for database design as it emphasized a more familiar description than the earlier relational model. Later on, entity-relationship constructs were retrofitted as a data modeling construct for the relational model, and the difference between the two have become irrelevant.[*citation needed*]

1980s, on the desktop

The 1980s ushered in the age of desktop computing. The new computers empowered their users with spreadsheets like Lotus 1,2,3 and database software like dBASE. The dBASE product was lightweight and easy for any computer user to understand out of the box. C. Wayne Ratliff the creator of dBASE stated: “dBASE was different from programs like BASIC, C, FORTRAN, and COBOL in that a lot of the dirty work had already been done. The data manipulation is done by dBASE instead of by the user, so the user can concentrate on what he is doing, rather than having to mess with the dirty details of opening, reading, and closing files, and managing space allocation.“ [12] dBASE was one of the top selling software titles in the 1980s and early 1990s.

1980s, object-oriented

The 1980s, along with a rise in object oriented programming, saw a growth in how data in various databases were handled. Programmers and designers began to treat the data in their databases as objects. That is to say that if a person's data were in a database, that person's attributes, such as their address, phone number, and age, were now considered to belong to that person instead of being extraneous data. This allows for relations between data to be relations to objects and their attributes and not to individual fields.[13] The term "object-relational impedance mismatch" described the inconvenience of translating between programmed objects and database tables. Object databases and object-relational databases attempt to solve this problem by providing an object-oriented language (sometimes as extensions to SQL) that programmers can use as alternative to purely relational SQL. On the programming side, libraries known as object-relational mappings (ORMs) attempt to solve the same problem.

2000s, NoSQL and NewSQL

The next generation of post-relational databases in the 2000s became known as NoSQL databases, including fast key-value stores and document-oriented databases. XML databases are a type of structured document-oriented database that allows querying based on XML document attributes.

NoSQL databases are often very fast, do not require fixed table schemas, avoid join operations by storing denormalized data, and are designed to scale horizontally.

In recent years there was a high demand for massively distributed databases with high partition tolerance but according to the CAP theorem it is impossible for a distributed system to simultaneously provide consistency, availability and partition tolerance guarantees. A distributed system can satisfy any two of these guarantees at the same time, but not all three. For that reason many NoSQL databases are using what is called eventual consistency to provide both availability and partition tolerance guarantees with a maximum level of data consistency.

The most popular NoSQL systems include: MongoDB, Couchbase, Riak, memcached, Redis, CouchDB, Hazelcast, Apache Cassandra and HBase. Note that all are open-source software products.

A number of new relational databases continuing use of SQL but aiming for performance comparable to NoSQL are known as NewSQL.

Research

Database technology has been an active research topic since the 1960s, both in academia and in the research and development groups of companies (for example IBM Research). Research activity includes theory and development of prototypes. Notable research topics have included models, the atomic transaction concept and related concurrency control techniques, query languages and query optimization methods, RAID, and more.

The database research area has several dedicated academic journals (for example, ACM Transactions on Database Systems-TODS, Data and Knowledge Engineering-DKE) and annual conferences (e.g., ACM SIGMOD, ACM PODS, VLDB, IEEE ICDE).

Examples

One way to classify databases involves the type of their contents, for example: bibliographic, document-text, statistical, or multimedia objects. Another way is by their application area, for example: accounting, music compositions, movies, banking, manufacturing, or insurance. A third way is by some technical aspect, such as the database structure or interface type. This section lists a few of the adjectives used to characterize different kinds of databases.

- An in-memory database is a database that primarily resides in main memory, but is typically backed-up by non-volatile computer data storage. Main memory databases are faster than disk databases, and so are often used where response time is critical, such as in telecommunications network equipment.SAP HANA platform is a very hot topic for in-memory database. By May 2012, HANA was able to run on servers with 100TB main memory powered by IBM. The co founder of the company claimed that the system was big enough to run the 8 largest SAP customers.
- An active database includes an event-driven architecture which can respond to conditions both inside and outside the database. Possible uses include security monitoring, alerting, statistics gathering and authorization. Many databases provide active database features in the form of database triggers.
- A cloud database relies on cloud technology. Both the database and most of its DBMS reside remotely, "in the cloud", while its applications are both developed by programmers and later maintained and utilized by (application's) end-users through a web browser and Open APIs.
- Data warehouses archive data from operational databases and often from external sources such as market research firms. The warehouse becomes the central source of data for use by managers and other end-users who may not have access to operational data. For example, sales data might be aggregated to weekly totals and converted from internal product codes to use UPCs so that they can be compared with ACNielsen data. Some basic and essential components of data warehousing include retrieving, analyzing, and mining data, transforming, loading and managing data so as to make them available for further use.
- A deductive database combines logic programming with a relational database, for example by using the Datalog language.
- A distributed database is one in which both the data and the DBMS span multiple computers.
- A document-oriented database is designed for storing, retrieving, and managing document-oriented, or semi structured data, information. Document-oriented databases are one of the main categories of NoSQL databases.
- An embedded database system is a DBMS which is tightly integrated with an application software that requires access to stored data in such a way that the DBMS is hidden from the application's end-users and requires little or no ongoing maintenance.[14]
- **End-user databases** consist of data developed by individual end-users. Examples of these are collections of documents, spreadsheets, presentations, multimedia, and other files. Several products exist to support such databases. Some of them are much simpler than full fledged DBMSs, with more elementary DBMS functionality.

- A federated database system comprises several distinct databases, each with its own DBMS. It is handled as a single database by a federated database management system (FDBMS), which transparently integrates multiple autonomous DBMSs, possibly of different types (in which case it would also be a heterogeneous database system), and provides them with an integrated conceptual view.
- Sometimes the term *multi-database* is used as a synonym to federated database, though it may refer to a less integrated (e.g., without an FDBMS and a managed integrated schema) group of databases that cooperate in a single application. In this case typically middleware is used for distribution, which typically includes an atomic commit protocol (ACP), e.g., the two-phase commit protocol, to allow distributed (global) transactions across the participating databases.
- A graph database is a kind of NoSQL database that uses graph structures with nodes, edges, and properties to represent and store information. General graph databases that can store any graph are distinct from specialized graph databases such as triplestores and network databases.
- In a hypertext or hypermedia database, any word or a piece of text representing an object, e.g., another piece of text, an article, a picture, or a film, can be hyperlinked to that object. Hypertext databases are particularly useful for organizing large amounts of disparate information. For example, they are useful for organizing online encyclopedias, where users can conveniently jump around the text. The World Wide Web is thus a large distributed hypertext database.
- A knowledge base (abbreviated **KB**, **kb** or Δ[15]) is a special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. Also a collection of data representing problems with their solutions and related experiences.
- A mobile database can be carried on or synchronized from a mobile computing device.
- Operational databases store detailed data about the operations of an organization. They typically process relatively high volumes of updates using transactions. Examples include customer databases that record contact, credit, and demographic information about a business' customers, personnel databases that hold information such as salary, benefits, skills data about employees, enterprise resource planning systems that record details about product components, parts inventory, and financial databases that keep track of the organization's money, accounting and financial dealings.
- A parallel database seeks to improve performance through parallelization for tasks such as loading data, building indexes and evaluating queries.

  The major parallel DBMS architectures which are induced by the underlying hardware architecture are:

  - **Shared memory architecture**, where multiple processors share the main memory space, as well as other data storage.
  - **Shared disk architecture**, where each processing unit (typically consisting of multiple processors) has its own main memory, but all units share the other storage.
  - **Shared nothing architecture**, where each processing unit has its own main memory and other storage.
- Probabilistic databases employ fuzzy logic to draw inferences from imprecise data.
- Real-time databases process transactions fast enough for the result to come back and be acted on right away.
- A spatial database can store the data with multidimensional features. The queries on such data include location based queries, like "Where is the closest hotel in my area?".
- A temporal database has built-in time aspects, for example a temporal data model and a temporal version of SQL. More specifically the temporal aspects usually include valid-time and transaction-time.
- A terminology-oriented database builds upon an object-oriented database, often customized for a specific field.
- An unstructured data database is intended to store in a manageable and protected way diverse objects that do not fit naturally and conveniently in common databases. It may include email messages, documents, journals,

multimedia objects, etc. The name may be misleading since some objects can be highly structured. However, the entire possible object collection does not fit into a predefined structured framework. Most established DBMSs now support unstructured data in various ways, and new dedicated DBMSs are emerging.

Design and modeling

The first task of a database designer is to produce a conceptual data model that reflects the structure of the information to be held in the database. A common approach to this is to develop an entity-relationship model, often with the aid of drawing tools. Another popular approach is the Unified Modeling Language. A successful data model will accurately reflect the possible state of the external world being modeled: for example, if people can have more than one phone number, it will allow this information to be captured. Designing a good conceptual data model requires a good understanding of the application domain; it typically involves asking deep questions about the things of interest to an organisation, like "can a customer also be a supplier?", or "if a product is sold with two different forms of packaging, are those the same product or different products?", or "if a plane flies from New York to Dubai via Frankfurt, is that one flight or two (or maybe even three)?". The answers to these questions establish definitions of the terminology used for entities (customers, products, flights, flight segments) and their relationships and attributes.

Producing the conceptual data model sometimes involves input from business processes, or the analysis of workflow in the organization. This can help to establish what information is needed in the database, and what can be left out. For example, it can help when deciding whether the database needs to hold historic data as well as current data.

Having produced a conceptual data model that users are happy with, the next stage is to translate this into a schema that implements the relevant data structures within the database. This process is often called logical database design, and the output is a logical data model expressed in the form of a schema. Whereas the conceptual data model is (in theory at least) independent of the choice of database technology, the logical data model will be expressed in terms of a particular database model supported by the chosen DBMS. (The terms *data model* and *database model* are often used interchangeably, but in this article we use *data model* for the design of a specific database, and *database model* for the modelling notation used to express that design.)

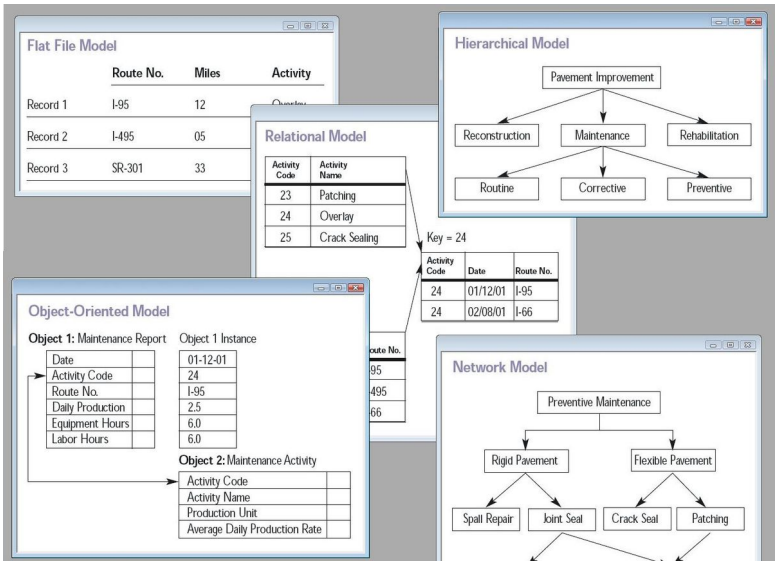
The most popular database model for general-purpose databases is the relational model, or more precisely, the relational model as represented by the SQL language. The process of creating a logical database design using this model uses a methodical approach known as normalization. The goal of normalization is to ensure that each elementary "fact" is only recorded in one place, so that insertions, updates, and deletions automatically maintain consistency.

The final stage of database design is to make the decisions that affect performance, scalability, recovery, security, and the like. This is often called *physical database design*. A key goal during this stage is data independence, meaning that the decisions made for performance optimization purposes should be invisible to end-users and applications. Physical design is driven mainly by performance requirements, and requires a good knowledge of the expected workload and access patterns, and a deep understanding of the features offered by the chosen DBMS.

Another aspect of physical database design is security. It involves both defining access control to database objects as well as defining security levels and methods for the data itself.

Models

A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model (or the SQL approximation of relational), which uses a table-based format.



Collage of five types of database models.

Common logical data models for databases include:

- Hierarchical database model
- Network model
- Relational model
- Entity–relationship model
  - Enhanced entity–relationship model
- Object model
- Document model
- Entity–attribute–value model
- Star schema

An object-relational database combines the two related structures.

Physical data models include:

- Inverted index
- Flat file

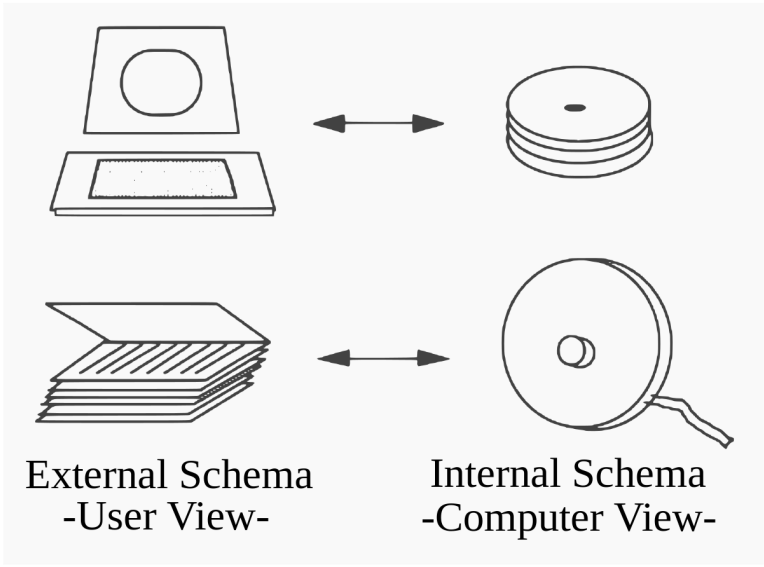
Other models include:

- Associative model
- Multidimensional model
- Multivalue model
- Semantic model
- XML database
- Named graph

External, conceptual, and internal views

A database management system provides three views of the database data:

- The **external level** defines how each group of end-users sees the organization of data in the database. A single database can have any number of views at the external level.
- The **conceptual level** unifies the various external views into a compatible global view. It provides the synthesis of all the external views. It is out of the scope of the various database end-users, and is rather of interest to database application developers and database administrators.
- The **internal level** (or *physical level*) is the internal organization of data inside a DBMS (see Implementation section below). It is concerned with cost, performance, scalability and other operational matters. It deals with storage layout of the data, using storage structures such as indexes to enhance performance. Occasionally it stores data of individual views (materialized views), computed from generic data, if performance justification exists for such redundancy. It balances all the external views' performance requirements, possibly conflicting, in an attempt to optimize overall performance across all activities.



Traditional view of data[16]

While there is typically only one conceptual (or logical) and physical (or internal) view of the data, there can be any number of different external views. This allows users to see database information in a more business-related way rather than from a technical, processing viewpoint. For example, a financial department of a company needs the payment details of all employees as part of the company's expenses, but does not need details about employees that are the interest of the human resources department. Thus different departments need different *views* of the company's database.

The three-level database architecture relates to the concept of *data independence* which was one of the major initial driving forces of the relational model. The idea is that changes made at a certain level do not affect the view at a higher level. For example, changes in the internal level do not affect application programs written using conceptual level interfaces, which reduces the impact of making physical changes to improve performance.

The conceptual view provides a level of indirection between internal and external. On one hand it provides a common view of the database, independent of different external view structures, and on the other hand it abstracts away details of how the data is stored or managed (internal level). In principle every level, and even every external view, can be presented by a different data model. In practice usually a given DBMS uses the same data model for both the external and the conceptual levels (e.g., relational model). The internal level, which is hidden inside the DBMS and depends on its implementation (see Implementation section below), requires a different level of detail and uses its own types of data structure types.

Separating the *external*, *conceptual* and *internal* levels was a major feature of the relational database model implementations that dominate 21st century databases.

Languages

Database languages are special-purpose languages, which do one or more of the following:

- Data definition language – defines data types and the relationships among them
- Data manipulation language – performs tasks such as inserting, updating, or deleting data occurrences
- Query language – allows searching for information and computing derived information

Database languages are specific to a particular data model. Notable examples include:

- SQL combines the roles of data definition, data manipulation, and query in a single language. It was one of the first commercial languages for the relational model, although it departs in some respects from the relational model as described by Codd (for example, the rows and columns of a table can be ordered). SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. The standards have been regularly enhanced since and is supported (with varying degrees of conformance) by all mainstream commercial relational DBMSs.
- OQL is an object model language standard (from the Object Data Management Group). It has influenced the design of some of the newer query languages like JDOQL and EJB QL.
- XQuery is a standard XML query language implemented by XML database systems such as MarkLogic and eXist, by relational databases with XML capability such as Oracle and DB2, and also by in-memory XML processors such as Saxon.
- SQL/XML combines XQuery with SQL.

A database language may also incorporate features like:

- DBMS-specific Configuration and storage engine management
- Computations to modify query results, like counting, summing, averaging, sorting, grouping, and cross-referencing
- Constraint enforcement (e.g. in an automotive database, only allowing one engine type per car)
- Application programming interface version of the query language, for programmer convenience

Performance, security, and availability

Because of the critical importance of database technology to the smooth running of an enterprise, database systems include complex mechanisms to deliver the required performance, security, and availability, and allow database administrators to control the use of these features.

Storage

Database storage is the container of the physical materialization of a database. It comprises the *internal* (physical) *level* in the database architecture. It also contains all the information needed (e.g., metadata, "data about the data", and internal data structures) to reconstruct the *conceptual level* and *external level* from the internal level when needed. Putting data into permanent storage is generally the responsibility of the database engine a.k.a. "storage engine". Though typically accessed by a DBMS through the underlying operating system (and often utilizing the operating systems' file systems as intermediates for storage layout), storage properties and configuration setting are extremely important for the efficient operation of the DBMS, and thus are closely maintained by database administrators. A DBMS, while in operation, always has its database residing in several types of storage (e.g., memory and external storage). The database data and the additional needed information, possibly in very large amounts, are coded into bits. Data typically reside in the storage in structures that look completely different from the way the data look in the conceptual and external levels, but in ways that attempt to optimize (the best possible) these levels' reconstruction when needed by users and programs, as well as for computing additional types of needed information from the data (e.g., when querying the database).

Some DBMSs support specifying which character encoding was used to store data, so multiple encodings can be used in the same database.

Various low-level database storage structures are used by the storage engine to serialize the data model so it can be written to the medium of choice. Techniques such as indexing may be used to improve performance. Conventional storage is row-oriented, but there are also column-oriented and correlation databases.

Materialized views

Often storage redundancy is employed to increase performance. A common example is storing *materialized views*, which consist of frequently needed *external views* or query results. Storing such views saves the expensive computing of them each time they are needed. The downsides of materialized views are the overhead incurred when updating them to keep them synchronized with their original updated database data, and the cost of storage redundancy.

Replication

Occasionally a database employs storage redundancy by database objects replication (with one or more copies) to increase data availability (both to improve performance of simultaneous multiple end-user accesses to a same database object, and to provide resiliency in a case of partial failure of a distributed database). Updates of a replicated object need to be synchronized across the object copies. In many cases the entire database is replicated.

Security

Database security deals with all various aspects of protecting the database content, its owners, and its users. It ranges from protection from intentional unauthorized database uses to unintentional database accesses by unauthorized entities (e.g., a person or a computer program).

Database access control deals with controlling who (a person or a certain computer program) is allowed to access what information in the database. The information may comprise specific database objects (e.g., record types, specific records, data structures), certain computations over certain objects (e.g., query types, or specific queries), or utilizing specific access paths to the former (e.g., using specific indexes or other data structures to access information). Database access controls are set by special authorized (by the database owner) personnel that uses dedicated protected security DBMS interfaces.

This may be managed directly on an individual basis, or by the assignment of individuals and privileges to groups, or (in the most elaborate models) through the assignment of individuals and groups to roles which are then granted entitlements. Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called "subschemas". For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data. If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases.

Data security in general deals with protecting specific chunks of data, both physically (i.e., from corruption, or destruction, or removal; e.g., see physical security), or the interpretation of them, or parts of them to meaningful information (e.g., by looking at the strings of bits that they comprise, concluding specific valid credit-card numbers; e.g., see data encryption).

Change and access logging records who accessed which attributes, what was changed, and when it was changed. Logging services allow for a forensic database audit later by keeping a record of access occurrences and changes. Sometimes application-level code is used to record changes rather than leaving this to the database. Monitoring can be set up to attempt to detect security breaches.

Transactions and concurrency

Database transactions can be used to introduce some level of fault tolerance and data integrity after recovery from a crash. A database transaction is a unit of work, typically encapsulating a number of operations over a database (e.g., reading a database object, writing, acquiring lock, etc.), an abstraction supported in database and also other systems. Each transaction has well defined boundaries in terms of which program/code executions are included in that transaction (determined by the transaction's programmer via special transaction commands).

The acronym ACID describes some ideal properties of a database transaction: Atomicity, Consistency, Isolation, and Durability.

Migration

See also section Database migration in article Data migration

A database built with one DBMS is not portable to another DBMS (i.e., the other DBMS cannot run it). However, in some situations it is desirable to move, migrate a database from one DBMS to another. The reasons are primarily economical (different DBMSs may have different total costs of ownership or TCOs), functional, and operational (different DBMSs may have different capabilities). The migration involves the database's transformation from one DBMS type to another. The transformation should maintain (if possible) the database related application (i.e., all related application programs) intact. Thus, the database's conceptual and external architectural levels should be maintained in the transformation. It may be desired that also some aspects of the architecture internal level are maintained. A complex or large database migration may be a complicated and costly (one-time) project by itself, which should be factored into the decision to migrate. This in spite of the fact that tools may exist to help migration between specific DBMSs. Typically a DBMS vendor provides tools to help importing databases from other popular DBMSs.

Building, maintaining, and tuning

After designing a database for an application, the next stage is building the database. Typically an appropriate general-purpose DBMS can be selected to be utilized for this purpose. A DBMS provides the needed user interfaces to be utilized by database administrators to define the needed application's data structures within the DBMS's respective data model. Other user interfaces are used to select needed DBMS parameters (like security related, storage allocation parameters, etc.).

When the database is ready (all its data structures and other needed components are defined) it is typically populated with initial application's data (database initialization, which is typically a distinct project; in many cases using specialized DBMS interfaces that support bulk insertion) before making it operational. In some cases the database becomes operational while empty of application data, and data is accumulated during its operation.

After the database is created, initialised and populated it needs to be maintained. Various database parameters may need changing and the database may need to be tuned (tuning) for better performance; application's data structures may be changed or added, new related application programs may be written to add to the application's functionality, etc. Databases are often confused with spreadsheets such as Microsoft Excel (Microsoft Access is a database management system, Excel is a spreadsheet program). Both can be used to store information, however a database is more efficient and flexible at storing large amounts of data. Below is a simple comparison of spreadsheets and databases.

Spreadsheet strengths	Spreadsheet Weaknesses
Very simple data storage	Data integrity problems, including inaccurate, inconsistent and out of date data and formulas.
Relatively easy to use	Difficult to validate data e.g. an incorrect formula
Require less planning	

Database strengths	Database Weaknesses
Methods for keeping data up to date and consistent	Require more planning and designing
Data is of higher quality than data stored in spreadsheets	Harder to change structure once database is built
Good for storing and organizing information.	Requires more technical knowledge to administrate

Backup and restore

Sometimes it is desired to bring a database back to a previous state (for many reasons, e.g., cases when the database is found corrupted due to a software error, or if it has been updated with erroneous data). To achieve this a **backup** operation is done occasionally or continuously, where each desired database state (i.e., the values of its data and their embedding in database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When this state is needed, i.e., when it is decided by a database administrator to bring the database back to this state (e.g., by specifying this state by a desired point in time when the database was in this state), these files are utilized to **restore** that state.

Other

Other DBMS features might include:

- Database logs
- Graphics component for producing graphs and charts, especially in a data warehouse system
- **Query optimizer –** Performs query optimization on every query to choose for it the most efficient *query plan* (a partial order (tree) of operations) to be executed to compute the query result. May be specific to a particular storage engine.
- Tools or hooks for database design, application programming, application program maintenance, database performance analysis and monitoring, database configuration monitoring, DBMS hardware configuration (a DBMS and related database may span computers, networks, and storage units) and related database mapping (especially for a distributed DBMS), storage allocation and database layout monitoring, storage migration, etc.

References

[1] Jeffrey Ullman 1997: *First course in database systems*, Prentice–Hall Inc., Simon & Schuster, Page 1, ISBN 0-13-861337-0.

[2] Tsitchizris, D. C. and F. H. Lochovsky (1982). *Data Models.* Englewood-Cliffs, Prentice–Hall.

[3] Beynon-Davies P. (2004). *Database Systems* 3rd Edition. Palgrave, Basingstoke, UK. ISBN 1-4039-1601-2

[4] . This article quotes a development time of 5 years involving 750 people for DB2 release 9 alone

[5] (Turing Award Lecture 1973)

[6] Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks" (http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf). In: *Communications of the ACM* 13 (6): 377–387.

[7] William Hershey and Carol Easthope, "A set theoretic data structure and retrieval language" (https://docs.google.com/open?id=0B4t_NX-QeWDYNmVhYjAwMWMtYzc3ZS00YjI0LWJhMjgtZTYyODZmNmFkNThh), Spring Joint Computer Conference, May 1972 in *ACM SIGIR Forum*, Volume 7, Issue 4 (December 1972), pp. 45–55, DOI= 10.1145/1095495.1095500 (http://doi.acm.org/10.1145/1095495.1095500)

[8] Ken North, "Sets, Data Models and Data Independence" (http://drdobbs.com/blogs/database/228700616), *Dr. Dobb's*, 10 March 2010

[9] *Description of a set-theoretic data structure* (http://hdl.handle.net/2027.42/4163), D. L. Childs, 1968, Technical Report 3 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

[10] *Feasibility of a Set-Theoretic Data Structure : A General Structure Based on a Reconstituted Definition of Relation* (http://hdl.handle.net/2027.42/4164), D. L. Childs, 1968, Technical Report 6 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

[11] *MICRO Information Management System (Version 5.0) Reference Manual* (http://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B4t_NX-QeWDYZGMwOTRmOTItZTg2Zi00YmJkLTg4MTktN2E4MWU0YmZlMjE3), M.A. Kahn, D.L. Rumelhart, and B.L. Bronson, October 1977, Institute of Labor and Industrial Relations (ILIR), University of Michigan and Wayne State University

[12] Interview with Wayne Ratliff (http://www.foxprohistory.org/interview_wayne_ratliff.htm). The FoxPro History. Retrieved on 2013-07-12.

[13] Development of an object-oriented DBMS; Portland, Oregon, United States; Pages: 472 – 482; 1986; ISBN 0-89791-204-7

[14] Graves, Steve. "COTS Databases For Embedded Systems" (http://www.embedded-computing.com/articles/id/?2020), *Embedded Computing Design* magazine, January 2007. Retrieved on August 13, 2008.

[15] Argumentation in Artificial Intelligence by Iyad Rahwan, Guillermo R. Simari

[16] itl.nist.gov (1993) *Integration Definition for Information Modeling (IDEFIX)* (http://www.itl.nist.gov/fipspubs/idef1x.doc). 21 December 1993.

Further reading

- Ling Liu and Tamer M. Özsu (Eds.) (2009). " Encyclopedia of Database Systems (http://www.springer.com/computer/database+management+&+information+retrieval/book/978-0-387-49616-0), 4100 p. 60 illus. ISBN 978-0-387-49616-0.
- Beynon-Davies, P. (2004). Database Systems. 3rd Edition. Palgrave, Houndmills, Basingstoke.
- Connolly, Thomas and Carolyn Begg. *Database Systems.* New York: Harlow, 2002.
- Date, C. J. (2003). *An Introduction to Database Systems, Fifth Edition*. Addison Wesley. ISBN 0-201-51381-1.
- Gray, J. and Reuter, A. *Transaction Processing: Concepts and Techniques*, 1st edition, Morgan Kaufmann Publishers, 1992.
- Kroenke, David M. and David J. Auer. *Database Concepts.* 3rd ed. New York: Prentice, 2007.
- Raghu Ramakrishnan and Johannes Gehrke, *Database Management Systems* (http://pages.cs.wisc.edu/~dbbook/)
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts* (http://www.db-book.com/)
- Discussion on database systems, (http://www.bbconsult.co.uk/Documents/Database-Systems.docx)
- Lightstone, S.; Teorey, T.; Nadeau, T. (2007). *Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more*. Morgan Kaufmann Press. ISBN 0-12-369389-6.
- Teorey, T.; Lightstone, S. and Nadeau, T. *Database Modeling & Design: Logical Design*, 4th edition, Morgan Kaufmann Press, 2005. ISBN 0-12-685352-5

External links

- Database (http://www.dmoz.org/Computers/Data_Formats/Database) on the Open Directory Project

Database management system

A **database** is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database management systems (**DBMS**s) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker Pro. A database is not generally portable across different DBMSs, but different DBMSs can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one database.

Terminology and overview

Formally, "database" refers to the data themselves and supporting data structures. Databases are created to operate large quantities of information by inputting, storing, retrieving, and managing that information. Databases are set up so that one set of software programs provides all users with access to all the data.

A "database management system" (DBMS) is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term "database" when used casually often refers to both a DBMS and the data it manipulates.

Outside the world of professional information technology, the term *database* is sometimes used casually to refer to any collection of data (perhaps a spreadsheet, maybe even a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.[1]

The interactions catered for by most existing DBMSs fall into four main groups:

- **Data definition –** Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- **Update –** Inserting, modifying, and deleting data.
- **Retrieval –** Obtaining information either for end-user queries and reports or for processing by applications.
- **Administration –** Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

Both a database and its DBMS conform to the principles of a particular database model.[2] "Database system" refers collectively to the database model, database management system, and database.[3]

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used for recovery of data if any of the disks fail. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications. DBMSs may be built around a custom multitasking kernel with built-in networking support, but modern DBMSs typically rely on a standard operating system to provide these functions. [*citation needed*] Since DBMSs comprise a significant economical market, computer and storage vendors often take into account DBMS requirements in their own development plans.[*citation needed*]

Databases and DBMSs can be categorized according to the database model(s) that they support (such as relational or XML), the type(s) of computer they run on (from a server cluster to a mobile phone), the query language(s) used to access the database (such as SQL or XQuery), and their internal engineering, which affects performance, scalability, resilience, and security.

Applications and roles

Most organizations in developed countries today depend on databases for their business operations. Increasingly, databases are not only used to support the internal operations of the organization, but also to underpin its online interactions with customers and suppliers (see Enterprise software). Databases are not used only to hold administrative information, but are often embedded within applications to hold more specialized data: for example engineering data or economic models. Examples of database applications include computerized library systems, flight reservation systems, and computerized parts inventory systems.

Client-server or transactional DBMSs are often complex to maintain high performance, availability and security when many users are querying and updating the database at the same time. Personal, desktop-based database systems tend to be less complex. For example, FileMaker and Microsoft Access come with built-in graphical user interfaces.

General-purpose and special-purpose DBMSs

A DBMS has evolved into a complex software system and its development typically requires thousands of person-years of development effort.[4] Some general-purpose DBMSs such as Adabas, Oracle and DB2 have been undergoing upgrades since the 1970s. General-purpose DBMSs aim to meet the needs of as many applications as possible, which adds to the complexity. However, the fact that their development cost can be spread over a large number of users means that they are often the most cost-effective approach. However, a general-purpose DBMS is not always the optimal solution: in some cases a general-purpose DBMS may introduce unnecessary overhead. Therefore, there are many examples of systems that use special-purpose databases. A common example is an email system: email systems are designed to optimize the handling of email messages, and do not need significant portions of a general-purpose DBMS functionality.

Many databases have application software that accesses the database on behalf of end-users, without exposing the DBMS interface directly. Application programmers may use a wire protocol directly, or more likely through an application programming interface. Database designers and database administrators interact with the DBMS through dedicated interfaces to build and maintain the applications' databases, and thus need some more knowledge and understanding about how DBMSs operate and the DBMSs' external interfaces and tuning parameters.

General-purpose databases are usually developed by one organization or community of programmers, while a different group builds the applications that use it. In many companies, specialized database administrators maintain databases, run reports, and may work on code that runs on the databases themselves (rather than in the client application).

History

Following the technology progress in the areas of processors, computer memory, computer storage and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational,[5] SQL/relational, and post-relational.

The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

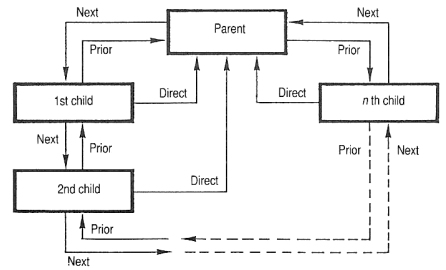
The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware became powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2014[6] they remain dominant except in niche areas. The dominant database language, standardised SQL for the relational model, has influenced database languages for other data models.[*citation needed*]

Object databases developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.

The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.

1960s, navigational DBMS

The introduction of the term *database* coincided with the availability of direct-access storage (disks and drums) from the mid-1960s onwards. The term represented a contrast with the tape-based systems of the past, allowing shared interactive use rather than daily batch processing. The Oxford English dictionary cites a 1962 report by the System Development Corporation of California as the first to use the term "data-base" in a specific technical sense.



A closed chain of records in a navigational database model (e.g. CODASYL), with ***next pointers***, ***prior pointers*** and ***direct pointers*** provided by keys in the various records.

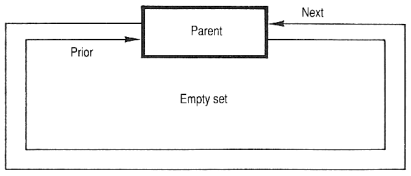


Illustration of an ***empty set***

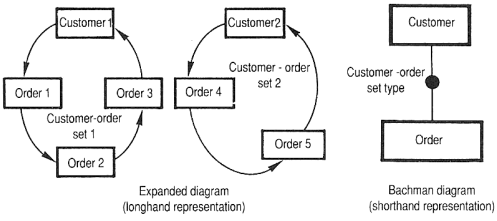


Illustration of a set type using a ***Bachman diagram***

The record set, basic structure of navigational (e.g. CODASYL) databse model. A set consists of one parent record (also called "the owner"), and n child records (also called members records)

Basic structure of navigational CODASYL database model.

As computers grew in speed and capability, a number of general-purpose database systems emerged; by the mid-1960s a number of such systems had come into commercial use. Interest in a standard began to grow, and Charles Bachman, author of one such product, the Integrated Data Store (IDS), founded the "Database Task Group" within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971 theDatabase Task Group delivered their standard, which generally became known as the "CODASYL approach", and soon a number of commercial products based on this approach entered the market.

The CODASYL approach relied on the "manual" navigation of a linked data set which was formed into a large network. Applications could find records by one of three methods:

- use of a primary key (known as a CALC key, typically implemented by hashing)
- navigating relationships (called sets) from one record to another
- scanning all the records in a sequential order.

Later systems added B-Trees to provide alternate access paths. Many CODASYL databases also added a very straightforward query language. However, in the final tally, CODASYL was very complex and required significant training and effort to produce useful applications.

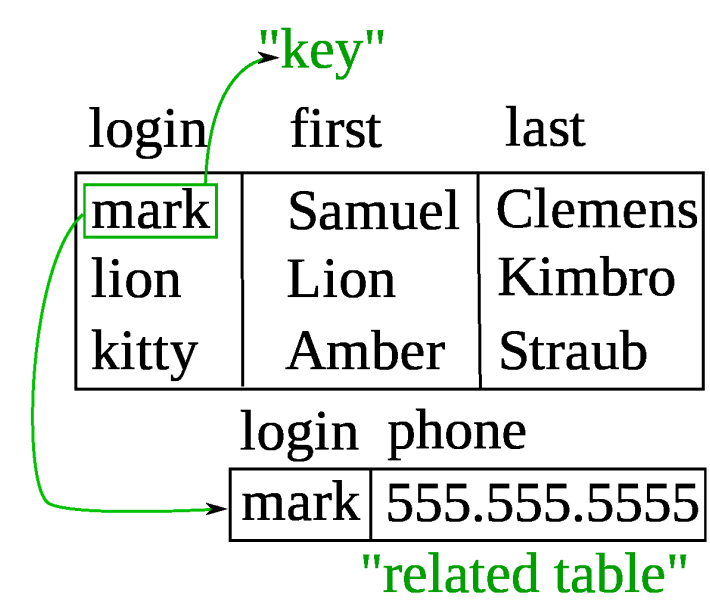
IBM also had their own DBMS system in 1968, known as *IMS*. IMS was a development of software written for the Apollo program on the System/360. IMS was generally similar in concept to CODASYL, but used a strict hierarchy for its model of data navigation instead of CODASYL's network model. Both concepts later became known as navigational databases due to the way data was accessed, and Bachman's 1973 Turing Award presentation was *The Programmer as Navigator*. IMS is classifiedWikipedia:Manual of Style/Words to watch#Unsupported attributions as a hierarchical database. IDMS and Cincom Systems' TOTAL database are classified as network databases. IMS remains in use as of 2014[6].

1970s, relational DBMS

Edgar Codd worked at IBM in San Jose, California, in one of their offshoot offices that was primarily involved in the development of hard disk systems. He was unhappy with the navigational model of the CODASYL approach, notably the lack of a "search" facility. In 1970, he wrote a number of papers that outlined a new approach to database construction that eventually culminated in the groundbreaking *A Relational Model of Data for Large Shared Data Banks*.[6]

In this paper, he described a new system for storing and working with large databases. Instead of records being stored in some sort of linked list of free-form records as in CODASYL, Codd's idea was to use a "table" of fixed-length records, with each table used for a different type of entity. A linked-list system would be very inefficient when storing "sparse" databases where some of the data for any one record could be left empty. The relational model solved this by splitting the data into a series of normalized tables (or *relations*), with optional elements being moved out of the main table to where they would take up room only if needed. Data may be freely inserted, deleted and edited in these tables, with the DBMS doing whatever maintenance needed to present a table view to the application/user.

The relational model also allowed the content of the database to evolve without constant rewriting of links and pointers. The relational part comes from entities referencing other entities in what is known as one-to-many relationship, like a traditional hierarchical model, and many-to-many relationship, like a navigational (network) model. Thus, a relational model can express both hierarchical and navigational models, as well as its native tabular model, allowing for pure or combined modeling in terms of these three models, as the application requires.



In the relational model, related records are linked together with a "key"

For instance, a common use of a database system is to track information about users, their name, login information, various addresses and phone numbers. In the navigational approach all of these data would be placed in a single record, and unused items would simply not be placed in the database. In the relational approach, the data would be *normalized* into a user table, an address table and a phone number table (for instance). Records would be created in these optional tables only if the address or phone numbers were actually provided.

Linking the information back together is the key to this system. In the relational model, some bit of information was used as a "key", uniquely defining a particular record. When information was being collected about a user, information stored in the optional tables would be found by searching for this key. For instance, if the login name of a user is unique, addresses and phone numbers for that user would be recorded with the login name as its key. This simple "re-linking" of related data back into a single collection is something that traditional computer languages are not designed for.

Just as the navigational approach would require programs to loop in order to collect records, the relational approach would require loops to collect information about any *one* record. Codd's solution to the necessary looping was a set-oriented language, a suggestion that would later spawn the ubiquitous SQL. Using a branch of mathematics known as tuple calculus, he demonstrated that such a system could support all the operations of normal databases (inserting, updating etc.) as well as providing a simple system for finding and returning *sets* of data in a single operation.

Codd's paper was picked up by two people at Berkeley, Eugene Wong and Michael Stonebraker. They started a project known as INGRES using funding that had already been allocated for a geographical database project and student programmers to produce code. Beginning in 1973, INGRES delivered its first test products which were generally ready for widespread use in 1979. INGRES was similar to System R in a number of ways, including the

use of a "language" for data access, known as QUEL. Over time, INGRES moved to the emerging SQL standard.

IBM itself did one test implementation of the relational model, PRTV, and a production one, Business System 12, both now discontinued. Honeywell wrote MRDS for Multics, and now there are two new implementations: Alphora Dataphor and Rel. Most other DBMS implementations usually called *relational* are actually SQL DBMSs.

In 1970, the University of Michigan began development of the MICRO Information Management System[7] based on D.L. Childs' Set-Theoretic Data model.[8][9][10] Micro was used to manage very large data sets by the US Department of Labor, the U.S. Environmental Protection Agency, and researchers from the University of Alberta, the University of Michigan, and Wayne State University. It ran on IBM mainframe computers using the Michigan Terminal System.[11] The system remained in production until 1998.

Integrated approach

In the 1970s and 1980s attempts were made to build database systems with integrated hardware and software. The underlying philosophy was that such integration would provide higher performance at lower cost. Examples were IBM System/38, the early offering of Teradata, and the Britton Lee, Inc. database machine.

Another approach to hardware support for database management was ICL's CAFS accelerator, a hardware disk controller with programmable search capabilities. In the long term, these efforts were generally unsuccessful because specialized database machines could not keep pace with the rapid development and progress of general-purpose computers. Thus most database systems nowadays are software systems running on general-purpose hardware, using general-purpose computer data storage. However this idea is still pursued for certain applications by some companies like Netezza and Oracle (Exadata).

Late 1970s, SQL DBMS

IBM started working on a prototype system loosely based on Codd's concepts as *System R* in the early 1970s. The first version was ready in 1974/5, and work then started on multi-table systems in which the data could be split so that all of the data for a record (some of which is optional) did not have to be stored in a single large "chunk". Subsequent multi-user versions were tested by customers in 1978 and 1979, by which time a standardized query language – SQL[*citation needed*] – had been added. Codd's ideas were establishing themselves as both workable and superior to CODASYL, pushing IBM to develop a true production version of System R, known as *SQL/DS*, and, later, *Database 2* (DB2).

Larry Ellison's Oracle started from a different chain, based on IBM's papers on System R, and beat IBM to market when the first version was released in 1978.[*citation needed*]

Stonebraker went on to apply the lessons from INGRES to develop a new database, Postgres, which is now known as PostgreSQL. PostgreSQL is often used for global mission critical applications (the .org and .info domain name registries use it as their primary data store, as do many large companies and financial institutions).

In Sweden, Codd's paper was also read and Mimer SQL was developed from the mid-1970s at Uppsala University. In 1984, this project was consolidated into an independent enterprise. In the early 1980s, Mimer introduced transaction handling for high robustness in applications, an idea that was subsequently implemented on most other DBMSs.

Another data model, the entity-relationship model, emerged in 1976 and gained popularity for database design as it emphasized a more familiar description than the earlier relational model. Later on, entity-relationship constructs were retrofitted as a data modeling construct for the relational model, and the difference between the two have become irrelevant.[*citation needed*]

1980s, on the desktop

The 1980s ushered in the age of desktop computing. The new computers empowered their users with spreadsheets like Lotus 1,2,3 and database software like dBASE. The dBASE product was lightweight and easy for any computer user to understand out of the box. C. Wayne Ratliff the creator of dBASE stated: “dBASE was different from programs like BASIC, C, FORTRAN, and COBOL in that a lot of the dirty work had already been done. The data manipulation is done by dBASE instead of by the user, so the user can concentrate on what he is doing, rather than having to mess with the dirty details of opening, reading, and closing files, and managing space allocation.“ [12] dBASE was one of the top selling software titles in the 1980s and early 1990s.

1980s, object-oriented

The 1980s, along with a rise in object oriented programming, saw a growth in how data in various databases were handled. Programmers and designers began to treat the data in their databases as objects. That is to say that if a person's data were in a database, that person's attributes, such as their address, phone number, and age, were now considered to belong to that person instead of being extraneous data. This allows for relations between data to be relations to objects and their attributes and not to individual fields.[13] The term "object-relational impedance mismatch" described the inconvenience of translating between programmed objects and database tables. Object databases and object-relational databases attempt to solve this problem by providing an object-oriented language (sometimes as extensions to SQL) that programmers can use as alternative to purely relational SQL. On the programming side, libraries known as object-relational mappings (ORMs) attempt to solve the same problem.

2000s, NoSQL and NewSQL

The next generation of post-relational databases in the 2000s became known as NoSQL databases, including fast key-value stores and document-oriented databases. XML databases are a type of structured document-oriented database that allows querying based on XML document attributes.

NoSQL databases are often very fast, do not require fixed table schemas, avoid join operations by storing denormalized data, and are designed to scale horizontally.

In recent years there was a high demand for massively distributed databases with high partition tolerance but according to the CAP theorem it is impossible for a distributed system to simultaneously provide consistency, availability and partition tolerance guarantees. A distributed system can satisfy any two of these guarantees at the same time, but not all three. For that reason many NoSQL databases are using what is called eventual consistency to provide both availability and partition tolerance guarantees with a maximum level of data consistency.

The most popular NoSQL systems include: MongoDB, Couchbase, Riak, memcached, Redis, CouchDB, Hazelcast, Apache Cassandra and HBase. Note that all are open-source software products.

A number of new relational databases continuing use of SQL but aiming for performance comparable to NoSQL are known as NewSQL.

Research

Database technology has been an active research topic since the 1960s, both in academia and in the research and development groups of companies (for example IBM Research). Research activity includes theory and development of prototypes. Notable research topics have included models, the atomic transaction concept and related concurrency control techniques, query languages and query optimization methods, RAID, and more.

The database research area has several dedicated academic journals (for example, ACM Transactions on Database Systems-TODS, Data and Knowledge Engineering-DKE) and annual conferences (e.g., ACM SIGMOD, ACM PODS, VLDB, IEEE ICDE).

Examples

One way to classify databases involves the type of their contents, for example: bibliographic, document-text, statistical, or multimedia objects. Another way is by their application area, for example: accounting, music compositions, movies, banking, manufacturing, or insurance. A third way is by some technical aspect, such as the database structure or interface type. This section lists a few of the adjectives used to characterize different kinds of databases.

- An in-memory database is a database that primarily resides in main memory, but is typically backed-up by non-volatile computer data storage. Main memory databases are faster than disk databases, and so are often used where response time is critical, such as in telecommunications network equipment.SAP HANA platform is a very hot topic for in-memory database. By May 2012, HANA was able to run on servers with 100TB main memory powered by IBM. The co founder of the company claimed that the system was big enough to run the 8 largest SAP customers.
- An active database includes an event-driven architecture which can respond to conditions both inside and outside the database. Possible uses include security monitoring, alerting, statistics gathering and authorization. Many databases provide active database features in the form of database triggers.
- A cloud database relies on cloud technology. Both the database and most of its DBMS reside remotely, "in the cloud", while its applications are both developed by programmers and later maintained and utilized by (application's) end-users through a web browser and Open APIs.
- Data warehouses archive data from operational databases and often from external sources such as market research firms. The warehouse becomes the central source of data for use by managers and other end-users who may not have access to operational data. For example, sales data might be aggregated to weekly totals and converted from internal product codes to use UPCs so that they can be compared with ACNielsen data. Some basic and essential components of data warehousing include retrieving, analyzing, and mining data, transforming, loading and managing data so as to make them available for further use.
- A deductive database combines logic programming with a relational database, for example by using the Datalog language.
- A distributed database is one in which both the data and the DBMS span multiple computers.
- A document-oriented database is designed for storing, retrieving, and managing document-oriented, or semi structured data, information. Document-oriented databases are one of the main categories of NoSQL databases.
- An embedded database system is a DBMS which is tightly integrated with an application software that requires access to stored data in such a way that the DBMS is hidden from the application's end-users and requires little or no ongoing maintenance.[14]
- **End-user databases** consist of data developed by individual end-users. Examples of these are collections of documents, spreadsheets, presentations, multimedia, and other files. Several products exist to support such databases. Some of them are much simpler than full fledged DBMSs, with more elementary DBMS functionality.

- A federated database system comprises several distinct databases, each with its own DBMS. It is handled as a single database by a federated database management system (FDBMS), which transparently integrates multiple autonomous DBMSs, possibly of different types (in which case it would also be a heterogeneous database system), and provides them with an integrated conceptual view.
- Sometimes the term *multi-database* is used as a synonym to federated database, though it may refer to a less integrated (e.g., without an FDBMS and a managed integrated schema) group of databases that cooperate in a single application. In this case typically middleware is used for distribution, which typically includes an atomic commit protocol (ACP), e.g., the two-phase commit protocol, to allow distributed (global) transactions across the participating databases.
- A graph database is a kind of NoSQL database that uses graph structures with nodes, edges, and properties to represent and store information. General graph databases that can store any graph are distinct from specialized graph databases such as triplestores and network databases.
- In a hypertext or hypermedia database, any word or a piece of text representing an object, e.g., another piece of text, an article, a picture, or a film, can be hyperlinked to that object. Hypertext databases are particularly useful for organizing large amounts of disparate information. For example, they are useful for organizing online encyclopedias, where users can conveniently jump around the text. The World Wide Web is thus a large distributed hypertext database.
- A knowledge base (abbreviated **KB**, **kb** or Δ[15]) is a special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. Also a collection of data representing problems with their solutions and related experiences.
- A mobile database can be carried on or synchronized from a mobile computing device.
- Operational databases store detailed data about the operations of an organization. They typically process relatively high volumes of updates using transactions. Examples include customer databases that record contact, credit, and demographic information about a business' customers, personnel databases that hold information such as salary, benefits, skills data about employees, enterprise resource planning systems that record details about product components, parts inventory, and financial databases that keep track of the organization's money, accounting and financial dealings.
- A parallel database seeks to improve performance through parallelization for tasks such as loading data, building indexes and evaluating queries.

  The major parallel DBMS architectures which are induced by the underlying hardware architecture are:

  - **Shared memory architecture**, where multiple processors share the main memory space, as well as other data storage.
  - **Shared disk architecture**, where each processing unit (typically consisting of multiple processors) has its own main memory, but all units share the other storage.
  - **Shared nothing architecture**, where each processing unit has its own main memory and other storage.
- Probabilistic databases employ fuzzy logic to draw inferences from imprecise data.
- Real-time databases process transactions fast enough for the result to come back and be acted on right away.
- A spatial database can store the data with multidimensional features. The queries on such data include location based queries, like "Where is the closest hotel in my area?".
- A temporal database has built-in time aspects, for example a temporal data model and a temporal version of SQL. More specifically the temporal aspects usually include valid-time and transaction-time.
- A terminology-oriented database builds upon an object-oriented database, often customized for a specific field.
- An unstructured data database is intended to store in a manageable and protected way diverse objects that do not fit naturally and conveniently in common databases. It may include email messages, documents, journals,

multimedia objects, etc. The name may be misleading since some objects can be highly structured. However, the entire possible object collection does not fit into a predefined structured framework. Most established DBMSs now support unstructured data in various ways, and new dedicated DBMSs are emerging.

Design and modeling

The first task of a database designer is to produce a conceptual data model that reflects the structure of the information to be held in the database. A common approach to this is to develop an entity-relationship model, often with the aid of drawing tools. Another popular approach is the Unified Modeling Language. A successful data model will accurately reflect the possible state of the external world being modeled: for example, if people can have more than one phone number, it will allow this information to be captured. Designing a good conceptual data model requires a good understanding of the application domain; it typically involves asking deep questions about the things of interest to an organisation, like "can a customer also be a supplier?", or "if a product is sold with two different forms of packaging, are those the same product or different products?", or "if a plane flies from New York to Dubai via Frankfurt, is that one flight or two (or maybe even three)?". The answers to these questions establish definitions of the terminology used for entities (customers, products, flights, flight segments) and their relationships and attributes.

Producing the conceptual data model sometimes involves input from business processes, or the analysis of workflow in the organization. This can help to establish what information is needed in the database, and what can be left out. For example, it can help when deciding whether the database needs to hold historic data as well as current data.

Having produced a conceptual data model that users are happy with, the next stage is to translate this into a schema that implements the relevant data structures within the database. This process is often called logical database design, and the output is a logical data model expressed in the form of a schema. Whereas the conceptual data model is (in theory at least) independent of the choice of database technology, the logical data model will be expressed in terms of a particular database model supported by the chosen DBMS. (The terms *data model* and *database model* are often used interchangeably, but in this article we use *data model* for the design of a specific database, and *database model* for the modelling notation used to express that design.)

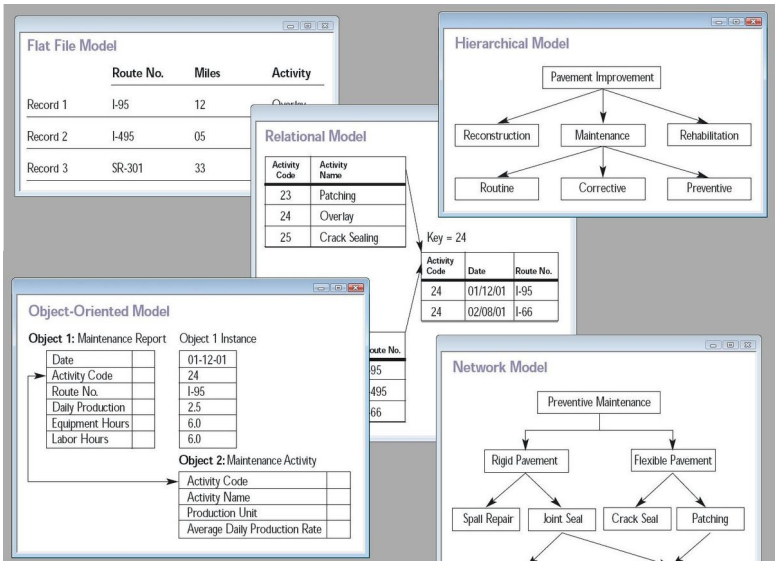
The most popular database model for general-purpose databases is the relational model, or more precisely, the relational model as represented by the SQL language. The process of creating a logical database design using this model uses a methodical approach known as normalization. The goal of normalization is to ensure that each elementary "fact" is only recorded in one place, so that insertions, updates, and deletions automatically maintain consistency.

The final stage of database design is to make the decisions that affect performance, scalability, recovery, security, and the like. This is often called *physical database design*. A key goal during this stage is data independence, meaning that the decisions made for performance optimization purposes should be invisible to end-users and applications. Physical design is driven mainly by performance requirements, and requires a good knowledge of the expected workload and access patterns, and a deep understanding of the features offered by the chosen DBMS.

Another aspect of physical database design is security. It involves both defining access control to database objects as well as defining security levels and methods for the data itself.

Models

A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model (or the SQL approximation of relational), which uses a table-based format.



Collage of five types of database models.

Common logical data models for databases include:

- Hierarchical database model
- Network model
- Relational model
- Entity–relationship model
  - Enhanced entity–relationship model
- Object model
- Document model
- Entity–attribute–value model
- Star schema

An object-relational database combines the two related structures.

Physical data models include:

- Inverted index
- Flat file

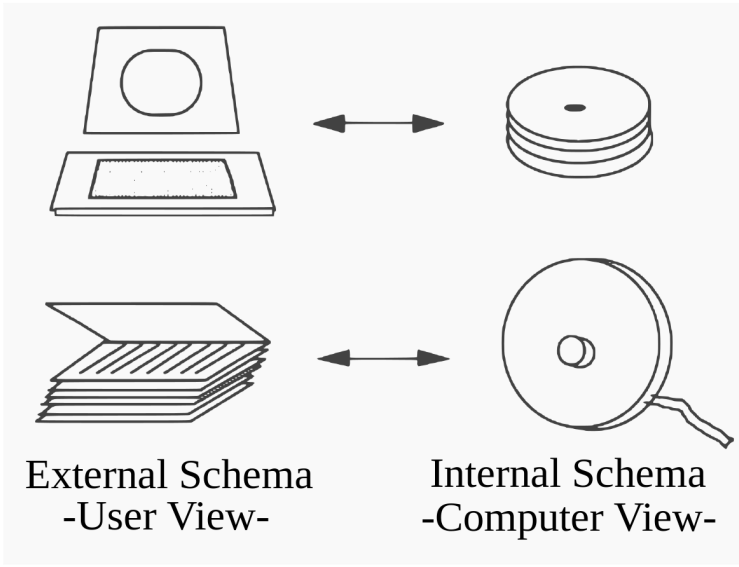
Other models include:

- Associative model
- Multidimensional model
- Multivalue model
- Semantic model
- XML database
- Named graph

External, conceptual, and internal views

A database management system provides three views of the database data:

- The **external level** defines how each group of end-users sees the organization of data in the database. A single database can have any number of views at the external level.
- The **conceptual level** unifies the various external views into a compatible global view. It provides the synthesis of all the external views. It is out of the scope of the various database end-users, and is rather of interest to database application developers and database administrators.
- The **internal level** (or *physical level*) is the internal organization of data inside a DBMS (see Implementation section below). It is concerned with cost, performance, scalability and other operational matters. It deals with storage layout of the data, using storage structures such as indexes to enhance performance. Occasionally it stores data of individual views (materialized views), computed from generic data, if performance justification exists for such redundancy. It balances all the external views' performance requirements, possibly conflicting, in an attempt to optimize overall performance across all activities.



Traditional view of data[16]

While there is typically only one conceptual (or logical) and physical (or internal) view of the data, there can be any number of different external views. This allows users to see database information in a more business-related way rather than from a technical, processing viewpoint. For example, a financial department of a company needs the payment details of all employees as part of the company's expenses, but does not need details about employees that are the interest of the human resources department. Thus different departments need different *views* of the company's database.

The three-level database architecture relates to the concept of *data independence* which was one of the major initial driving forces of the relational model. The idea is that changes made at a certain level do not affect the view at a higher level. For example, changes in the internal level do not affect application programs written using conceptual level interfaces, which reduces the impact of making physical changes to improve performance.

The conceptual view provides a level of indirection between internal and external. On one hand it provides a common view of the database, independent of different external view structures, and on the other hand it abstracts away details of how the data is stored or managed (internal level). In principle every level, and even every external view, can be presented by a different data model. In practice usually a given DBMS uses the same data model for both the external and the conceptual levels (e.g., relational model). The internal level, which is hidden inside the DBMS and depends on its implementation (see Implementation section below), requires a different level of detail and uses its own types of data structure types.

Separating the *external*, *conceptual* and *internal* levels was a major feature of the relational database model implementations that dominate 21st century databases.

Languages

Database languages are special-purpose languages, which do one or more of the following:

- Data definition language – defines data types and the relationships among them
- Data manipulation language – performs tasks such as inserting, updating, or deleting data occurrences
- Query language – allows searching for information and computing derived information

Database languages are specific to a particular data model. Notable examples include:

- SQL combines the roles of data definition, data manipulation, and query in a single language. It was one of the first commercial languages for the relational model, although it departs in some respects from the relational model as described by Codd (for example, the rows and columns of a table can be ordered). SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. The standards have been regularly enhanced since and is supported (with varying degrees of conformance) by all mainstream commercial relational DBMSs.
- OQL is an object model language standard (from the Object Data Management Group). It has influenced the design of some of the newer query languages like JDOQL and EJB QL.
- XQuery is a standard XML query language implemented by XML database systems such as MarkLogic and eXist, by relational databases with XML capability such as Oracle and DB2, and also by in-memory XML processors such as Saxon.
- SQL/XML combines XQuery with SQL.

A database language may also incorporate features like:

- DBMS-specific Configuration and storage engine management
- Computations to modify query results, like counting, summing, averaging, sorting, grouping, and cross-referencing
- Constraint enforcement (e.g. in an automotive database, only allowing one engine type per car)
- Application programming interface version of the query language, for programmer convenience

Performance, security, and availability

Because of the critical importance of database technology to the smooth running of an enterprise, database systems include complex mechanisms to deliver the required performance, security, and availability, and allow database administrators to control the use of these features.

Storage

Database storage is the container of the physical materialization of a database. It comprises the *internal* (physical) *level* in the database architecture. It also contains all the information needed (e.g., metadata, "data about the data", and internal data structures) to reconstruct the *conceptual level* and *external level* from the internal level when needed. Putting data into permanent storage is generally the responsibility of the database engine a.k.a. "storage engine". Though typically accessed by a DBMS through the underlying operating system (and often utilizing the operating systems' file systems as intermediates for storage layout), storage properties and configuration setting are extremely important for the efficient operation of the DBMS, and thus are closely maintained by database administrators. A DBMS, while in operation, always has its database residing in several types of storage (e.g., memory and external storage). The database data and the additional needed information, possibly in very large amounts, are coded into bits. Data typically reside in the storage in structures that look completely different from the way the data look in the conceptual and external levels, but in ways that attempt to optimize (the best possible) these levels' reconstruction when needed by users and programs, as well as for computing additional types of needed information from the data (e.g., when querying the database).

Some DBMSs support specifying which character encoding was used to store data, so multiple encodings can be used in the same database.

Various low-level database storage structures are used by the storage engine to serialize the data model so it can be written to the medium of choice. Techniques such as indexing may be used to improve performance. Conventional storage is row-oriented, but there are also column-oriented and correlation databases.

Materialized views

Often storage redundancy is employed to increase performance. A common example is storing *materialized views*, which consist of frequently needed *external views* or query results. Storing such views saves the expensive computing of them each time they are needed. The downsides of materialized views are the overhead incurred when updating them to keep them synchronized with their original updated database data, and the cost of storage redundancy.

Replication

Occasionally a database employs storage redundancy by database objects replication (with one or more copies) to increase data availability (both to improve performance of simultaneous multiple end-user accesses to a same database object, and to provide resiliency in a case of partial failure of a distributed database). Updates of a replicated object need to be synchronized across the object copies. In many cases the entire database is replicated.

Security

Database security deals with all various aspects of protecting the database content, its owners, and its users. It ranges from protection from intentional unauthorized database uses to unintentional database accesses by unauthorized entities (e.g., a person or a computer program).

Database access control deals with controlling who (a person or a certain computer program) is allowed to access what information in the database. The information may comprise specific database objects (e.g., record types, specific records, data structures), certain computations over certain objects (e.g., query types, or specific queries), or utilizing specific access paths to the former (e.g., using specific indexes or other data structures to access information). Database access controls are set by special authorized (by the database owner) personnel that uses dedicated protected security DBMS interfaces.

This may be managed directly on an individual basis, or by the assignment of individuals and privileges to groups, or (in the most elaborate models) through the assignment of individuals and groups to roles which are then granted entitlements. Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called "subschemas". For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data. If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases.

Data security in general deals with protecting specific chunks of data, both physically (i.e., from corruption, or destruction, or removal; e.g., see physical security), or the interpretation of them, or parts of them to meaningful information (e.g., by looking at the strings of bits that they comprise, concluding specific valid credit-card numbers; e.g., see data encryption).

Change and access logging records who accessed which attributes, what was changed, and when it was changed. Logging services allow for a forensic database audit later by keeping a record of access occurrences and changes. Sometimes application-level code is used to record changes rather than leaving this to the database. Monitoring can be set up to attempt to detect security breaches.

Transactions and concurrency

Database transactions can be used to introduce some level of fault tolerance and data integrity after recovery from a crash. A database transaction is a unit of work, typically encapsulating a number of operations over a database (e.g., reading a database object, writing, acquiring lock, etc.), an abstraction supported in database and also other systems. Each transaction has well defined boundaries in terms of which program/code executions are included in that transaction (determined by the transaction's programmer via special transaction commands).

The acronym ACID describes some ideal properties of a database transaction: Atomicity, Consistency, Isolation, and Durability.

Migration

See also section Database migration in article Data migration

A database built with one DBMS is not portable to another DBMS (i.e., the other DBMS cannot run it). However, in some situations it is desirable to move, migrate a database from one DBMS to another. The reasons are primarily economical (different DBMSs may have different total costs of ownership or TCOs), functional, and operational (different DBMSs may have different capabilities). The migration involves the database's transformation from one DBMS type to another. The transformation should maintain (if possible) the database related application (i.e., all related application programs) intact. Thus, the database's conceptual and external architectural levels should be maintained in the transformation. It may be desired that also some aspects of the architecture internal level are maintained. A complex or large database migration may be a complicated and costly (one-time) project by itself, which should be factored into the decision to migrate. This in spite of the fact that tools may exist to help migration between specific DBMSs. Typically a DBMS vendor provides tools to help importing databases from other popular DBMSs.

Building, maintaining, and tuning

After designing a database for an application, the next stage is building the database. Typically an appropriate general-purpose DBMS can be selected to be utilized for this purpose. A DBMS provides the needed user interfaces to be utilized by database administrators to define the needed application's data structures within the DBMS's respective data model. Other user interfaces are used to select needed DBMS parameters (like security related, storage allocation parameters, etc.).

When the database is ready (all its data structures and other needed components are defined) it is typically populated with initial application's data (database initialization, which is typically a distinct project; in many cases using specialized DBMS interfaces that support bulk insertion) before making it operational. In some cases the database becomes operational while empty of application data, and data is accumulated during its operation.

After the database is created, initialised and populated it needs to be maintained. Various database parameters may need changing and the database may need to be tuned (tuning) for better performance; application's data structures may be changed or added, new related application programs may be written to add to the application's functionality, etc. Databases are often confused with spreadsheets such as Microsoft Excel (Microsoft Access is a database management system, Excel is a spreadsheet program). Both can be used to store information, however a database is more efficient and flexible at storing large amounts of data. Below is a simple comparison of spreadsheets and databases.

Spreadsheet strengths	Spreadsheet Weaknesses
Very simple data storage	Data integrity problems, including inaccurate, inconsistent and out of date data and formulas.
Relatively easy to use	Difficult to validate data e.g. an incorrect formula
Require less planning	

Database strengths	Database Weaknesses
Methods for keeping data up to date and consistent	Require more planning and designing
Data is of higher quality than data stored in spreadsheets	Harder to change structure once database is built
Good for storing and organizing information.	Requires more technical knowledge to administrate

Backup and restore

Sometimes it is desired to bring a database back to a previous state (for many reasons, e.g., cases when the database is found corrupted due to a software error, or if it has been updated with erroneous data). To achieve this a **backup** operation is done occasionally or continuously, where each desired database state (i.e., the values of its data and their embedding in database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When this state is needed, i.e., when it is decided by a database administrator to bring the database back to this state (e.g., by specifying this state by a desired point in time when the database was in this state), these files are utilized to **restore** that state.

Other

Other DBMS features might include:

- Database logs
- Graphics component for producing graphs and charts, especially in a data warehouse system
- **Query optimizer –** Performs query optimization on every query to choose for it the most efficient *query plan* (a partial order (tree) of operations) to be executed to compute the query result. May be specific to a particular storage engine.
- Tools or hooks for database design, application programming, application program maintenance, database performance analysis and monitoring, database configuration monitoring, DBMS hardware configuration (a DBMS and related database may span computers, networks, and storage units) and related database mapping (especially for a distributed DBMS), storage allocation and database layout monitoring, storage migration, etc.

References

[1] Jeffrey Ullman 1997: *First course in database systems*, Prentice–Hall Inc., Simon & Schuster, Page 1, ISBN 0-13-861337-0.

[2] Tsitchizris, D. C. and F. H. Lochovsky (1982). *Data Models.* Englewood-Cliffs, Prentice–Hall.

[3] Beynon-Davies P. (2004). *Database Systems* 3rd Edition. Palgrave, Basingstoke, UK. ISBN 1-4039-1601-2

[4] . This article quotes a development time of 5 years involving 750 people for DB2 release 9 alone

[5] (Turing Award Lecture 1973)

[6] Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks" (http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf). In: *Communications of the ACM* 13 (6): 377–387.

[7] William Hershey and Carol Easthope, "A set theoretic data structure and retrieval language" (https://docs.google.com/open?id=0B4t_NX-QeWDYNmVhYjAwMWMtYzc3ZS00YjI0LWJhMjgtZTYyODZmNmFkNThh), Spring Joint Computer Conference, May 1972 in *ACM SIGIR Forum*, Volume 7, Issue 4 (December 1972), pp. 45–55, DOI= 10.1145/1095495.1095500 (http://doi.acm.org/10.1145/1095495.1095500)

[8] Ken North, "Sets, Data Models and Data Independence" (http://drdobbs.com/blogs/database/228700616), *Dr. Dobb's*, 10 March 2010

[9] *Description of a set-theoretic data structure* (http://hdl.handle.net/2027.42/4163), D. L. Childs, 1968, Technical Report 3 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

[10] *Feasibility of a Set-Theoretic Data Structure : A General Structure Based on a Reconstituted Definition of Relation* (http://hdl.handle.net/2027.42/4164), D. L. Childs, 1968, Technical Report 6 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

[11] *MICRO Information Management System (Version 5.0) Reference Manual* (http://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B4t_NX-QeWDYZGMwOTRmOTItZTg2Zi00YmJkLTg4MTktN2E4MWU0YmZlMjE3), M.A. Kahn, D.L. Rumelhart, and B.L. Bronson, October 1977, Institute of Labor and Industrial Relations (ILIR), University of Michigan and Wayne State University

[12] Interview with Wayne Ratliff (http://www.foxprohistory.org/interview_wayne_ratliff.htm). The FoxPro History. Retrieved on 2013-07-12.

[13] Development of an object-oriented DBMS; Portland, Oregon, United States; Pages: 472 – 482; 1986; ISBN 0-89791-204-7

[14] Graves, Steve. "COTS Databases For Embedded Systems" (http://www.embedded-computing.com/articles/id/?2020), *Embedded Computing Design* magazine, January 2007. Retrieved on August 13, 2008.

[15] Argumentation in Artificial Intelligence by Iyad Rahwan, Guillermo R. Simari

[16] itl.nist.gov (1993) *Integration Definition for Information Modeling (IDEFIX)* (http://www.itl.nist.gov/fipspubs/idef1x.doc). 21 December 1993.

Further reading

- Ling Liu and Tamer M. Özsu (Eds.) (2009). " Encyclopedia of Database Systems (http://www.springer.com/computer/database+management+&+information+retrieval/book/978-0-387-49616-0), 4100 p. 60 illus. ISBN 978-0-387-49616-0.
- Beynon-Davies, P. (2004). Database Systems. 3rd Edition. Palgrave, Houndmills, Basingstoke.
- Connolly, Thomas and Carolyn Begg. *Database Systems.* New York: Harlow, 2002.
- Date, C. J. (2003). *An Introduction to Database Systems, Fifth Edition*. Addison Wesley. ISBN 0-201-51381-1.
- Gray, J. and Reuter, A. *Transaction Processing: Concepts and Techniques*, 1st edition, Morgan Kaufmann Publishers, 1992.
- Kroenke, David M. and David J. Auer. *Database Concepts.* 3rd ed. New York: Prentice, 2007.
- Raghu Ramakrishnan and Johannes Gehrke, *Database Management Systems* (http://pages.cs.wisc.edu/~dbbook/)
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts* (http://www.db-book.com/)
- Discussion on database systems, (http://www.bbconsult.co.uk/Documents/Database-Systems.docx)
- Lightstone, S.; Teorey, T.; Nadeau, T. (2007). *Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more*. Morgan Kaufmann Press. ISBN 0-12-369389-6.
- Teorey, T.; Lightstone, S. and Nadeau, T. *Database Modeling & Design: Logical Design*, 4th edition, Morgan Kaufmann Press, 2005. ISBN 0-12-685352-5

External links

- Database (http://www.dmoz.org/Computers/Data_Formats/Database) on the Open Directory Project

Types of DBMS

There are four main types of database management systems (DBMS) and these are based upon their management of database structures. In other words, the **types of DBMS** are entirely dependent upon how the database is structured by that particular DBMS.

Hierarchical DBMS

A DBMS is said to be hierarchical if the relationships among data in the database are established in such a way that one data item is present as the subordinate of another one or a sub unit. Here subordinate means that items have "parent-child" relationships among them. Direct relationships exist between any two records that are stored consecutively. The data structure "tree" is followed by the DBMS to structure the database. No backward movement is possible/allowed in the hierarchical database.

The hierarchical data model was developed by IBM in 1968 and introduced in information management systems. This model is like a structure of a tree with the records forming the nodes and fields forming the branches of the tree. In the hierarchical model, records are linked in the form of an organization chart. A tree structure may establish one-to-many relationship.....

Network DBMS

A DBMS is said to be a Network DBMS if the relationships among data in the database are of type many-to-many. The relationships among many-to-many appears in the form of a network. Thus the structure of a network database is extremely complicated because of these many-to-many relationships in which one record can be used as a key of the entire database. A network database is structured in the form of a graph that is also a data structure. Though the structure of such a DBMS is highly complicated however it has two basic elements i.e. records and sets to designate many-to-many relationships. Mainly high-level languages such as Pascal, C++, COBOL and FORTRAN etc. were used to implement the records and set structures.

Relational DBMS

A DBMS is said to be a Relational DBMS or RDBMS if the database relationships are treated in the form of a table. There are three keys on relational DBMS: relation, domain and attributes. A network means it contains a fundamental constructs sets or records sets contains one to many relationship,records contains fields statical table that is composed of rows and columns is used to organize the database and its structure and is actually a two dimension array in the computer memory. A number of RDBMSs are available, some popular examples are Oracle, Sybase, Ingress, Informix, Microsoft SQL Server, and Microsoft Access.

Object-oriented DBMS

Able to handle many new data types, including graphics, photographs, audio, and video, object-oriented databases represent a significant advance over their other database cousins. Hierarchical and network databases are all designed to handle structured data; that is, data that fits nicely into fields, rows, and columns. They are useful for handling small snippets of information such as names, addresses, zip codes, product numbers, and any kind of statistic or number you can think of. On the other hand, an object-oriented database can be used to store data from a variety of media sources, such as photographs and text, and produce work, as output, in a multimedia format.[1]

- Object-oriented databases use small, reusable chunks of software called objects. The objects themselves are stored in the object-oriented database. Each object consists of two elements: 1) a piece of data (e.g., sound, video, text, or graphics), and 2) the instructions, or software programs called methods, for what to do with the data. Part two

of this definition requires a little more explanation. The instructions contained within the object are used to do something with the data in the object. For example, test scores would be within the object as would the instructions for calculating average test score.

- Object-oriented databases have two disadvantages. First, they are more costly to develop. Second, most organizations are reluctant to abandon or convert from those databases that they have already invested money in developing and implementing. However, the benefits to object-oriented databases are compelling. The ability to mix and match reusable objects provides incredible multimedia capability. Healthcare organizations, for example, can store, track, and recall CAT scans, X-rays, electrocardiograms and many other forms of crucial data.

References

[1] http://www.personal.psu.edu/glh10/ist110/topic/topic07/topic07_06.html

Data store

A **data store** is a data repository of a set of integrated objects. These objects are modeled using classes defined in database schemas. Data store includes not only data repositories like databases, it is a more general concept that includes also flat files that can store data.

Some data stores do represent data in only one schema, while other data stores use several schemas for this task. An example are RDBMS-based data stores like MySQL or PostgreSQL.

Types

Data stores can be of different types, including:

- Paper files
- Simple files like a spreadsheet
- File systems
- Databases
  - Relational databases are the most common type of database in the 2000s. Examples include MySQL, PostgreSQL, Microsoft SQL Server, and Oracle Database.
  - Object-oriented databases, like Caché or ConceptBase. They can save objects of an object-oriented design.
- Distributed data stores, like Apache Cassandra, Druid (open-source data store) or Dynamo
- Directory services
- VMware uses "datastore" to refer to a file that stores a virtual machine[1]

References

[1] http://pubs.vmware.com/vi3/sdk/ReferenceGuide/vim.Datastore.html

Technical Information Project

The **Technical Information Project** (TIP) was an early database project. TIP included over 25,000 records and was used to explore bibliographic coupling between works.

Developed by Meyer Mike Kessler at MIT around 1964, some of the innovations in TIP included the use of wild cards, and boolean searching.

References

- Chronology of Information Science [1]
- *Bourne, C.P. and Hahn, T. B. A History of Online Information Services*, 1963-1976. Cambridge, MA: MIT Press, 2003.

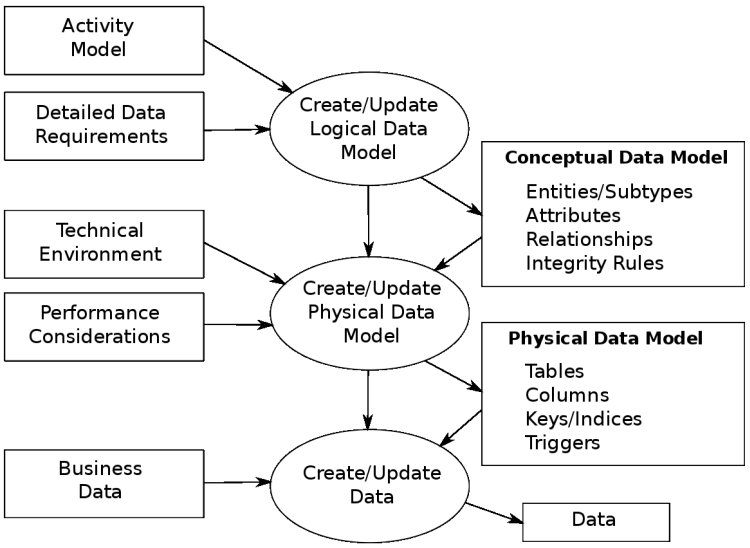
References

[1] http://www.libsci.sc.edu/bob/istchron/ISCNET/ISCHRON.HTM

Introduction to Data Modeling

Data modeling

Data modeling in software engineering is the process of creating a data model for an information system by applying formal data modeling techniques.



The data modeling process. The figure illustrates the way data models are developed and used today. A conceptual data model is developed based on the data requirements for the application that is being developed, perhaps in the context of an activity model. The data model will normally consist of entity types, attributes, relationships, integrity rules, and the definitions of those objects. This is then used as the start point for interface or database design.

Overview

Data modeling is a process used to define and analyze data requirements needed to support the business processes within the scope of corresponding information systems in organizations. Therefore, the process of data modeling involves professional data modelers working closely with business stakeholders, as well as potential users of the information system.

There are three different types of data models produced while progressing from requirements to the actual database to be used for the information system.[1] The data requirements are initially recorded as a conceptual data model which is essentially a set of technology independent specifications about the data and is used to discuss initial requirements with the business stakeholders. The conceptual model is then translated into a logical data model, which documents structures of the data that can be implemented in databases. Implementation of one conceptual data model may require multiple logical data models. The last step in data modeling is transforming the logical data model to a physical data model that organizes the data into tables, and accounts for access, performance and storage details. Data modeling defines not just data elements, but also their structures and the relationships between them.[2]

Data modeling techniques and methodologies are used to model data in a standard, consistent, predictable manner in order to manage it as a resource. The use of data modeling standards is strongly recommended for all projects requiring a standard means of defining and analyzing data within an organization, e.g., using data modeling:

- to assist business analysts, programmers, testers, manual writers, IT package selectors, engineers, managers, related organizations and clients to understand and use an agreed semi-formal model the concepts of the organization and how they relate to one another
- to manage data as a resource
- for the integration of information systems
- for designing databases/data warehouses (aka data repositories)

Data modeling may be performed during various types of projects and in multiple phases of projects. Data models are progressive; there is no such thing as the final data model for a business or application. Instead a data model should be considered a living document that will change in response to a changing business. The data models should ideally be stored in a repository so that they can be retrieved, expanded, and edited over time. Whitten et al. (2004) determined two types of data modeling:

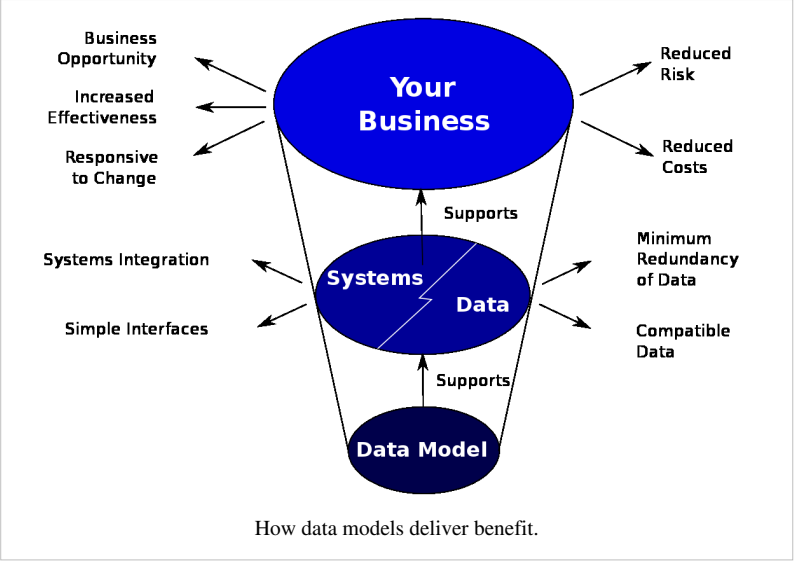
- Strategic data modeling: This is part of the creation of an information systems strategy, which defines an overall vision and architecture for information systems is defined. Information engineering is a methodology that embraces this approach.
- Data modeling during systems analysis: In systems analysis logical data models are created as part of the development of new databases.

Data modeling is also used as a technique for detailing business requirements for specific databases. It is sometimes called *database modeling* because a data model is eventually implemented in a database.[]

Data modeling topics

Data models

Data models provide a structure for data used within information systems by providing specific definition and format. If a data model is used consistently across systems then compatibility of data can be achieved. If the same data structures are used to store and access data then different applications can share data seamlessly. The results of this are indicated in the diagram. However, systems and interfaces often cost more than they should, to build, operate, and maintain. They may also constrain the business rather than support it. This may occur when the quality of the data models implemented in systems and interfaces is poor.[]



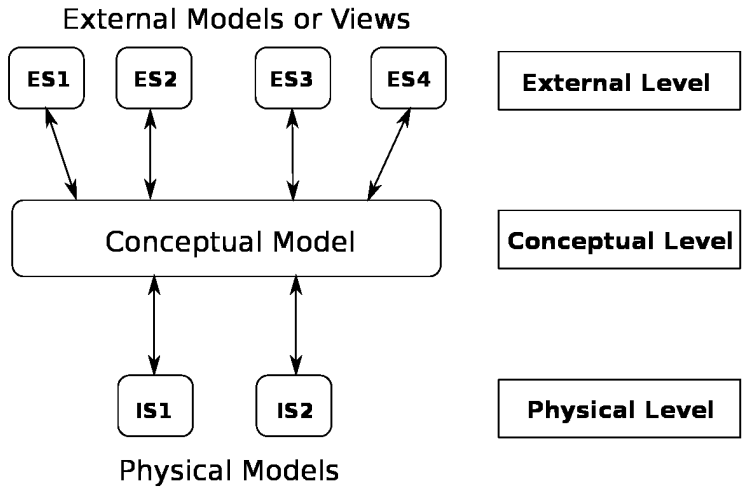
How data models deliver benefit.

- Business rules, specific to how things are done in a particular place, are often fixed in the structure of a data model. This means that small changes in the way business is conducted lead to large changes in computer systems and interfaces. So, business rules need to be implemented in a flexible way that does not result in complicated dependencies, rather the data model should be flexible enough so that changes in the business can be implemented within the data model in a relatively quick and efficient way.
- Entity types are often not identified, or are identified incorrectly. This can lead to replication of data, data structure and functionality, together with the attendant costs of that duplication in development and maintenance.Therefore, data definitions should be made as explicit and easy to understand as possible to minimize misinterpretation and duplication.
- Data models for different systems are arbitrarily different. The result of this is that complex interfaces are required between systems that share data. These interfaces can account for between 25-70% of the cost of current systems. Required interfaces should be considered inherently while designing a data model, as a data model on its own would not be usable without interfaces within different systems.

- Data cannot be shared electronically with customers and suppliers, because the structure and meaning of data has not been standardised. To obtain optimal value from an implemented data model, it is very important to define standards that will ensure that data models will both meet business needs and be consistent.

Conceptual, logical and physical schemas

In 1975 ANSI described three kinds of data-model *instance*:[3]



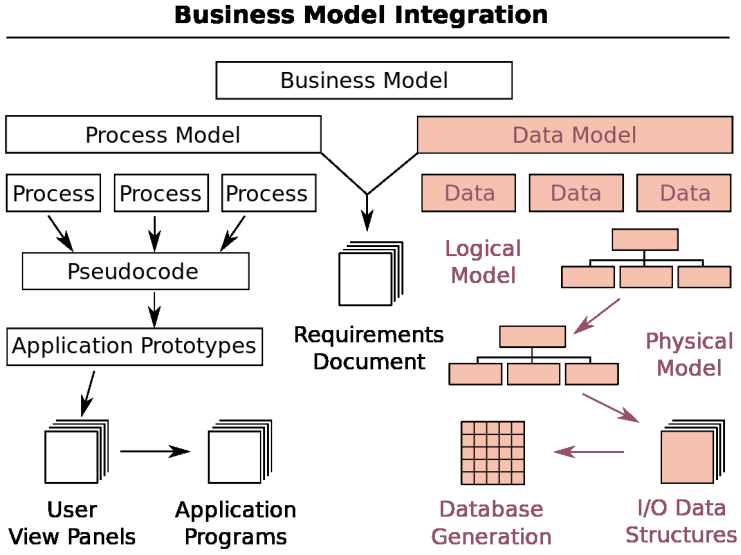
The ANSI/SPARC three level architecture. This shows that a data model can be an external model (or view), a conceptual model, or a physical model. This is not the only way to look at data models, but it is a useful way, particularly when comparing models.

- Conceptual schema: describes the semantics of a domain (the scope of the model). For example, it may be a model of the interest area of an organization or of an industry. This consists of entity classes, representing kinds of things of significance in the domain, and relationships assertions about associations between pairs of entity classes. A conceptual schema specifies the kinds of facts or propositions that can be expressed using the model. In that sense, it defines the allowed expressions in an artificial "language" with a scope that is limited by the scope of the model. Simply described, a conceptual schema is the first step in organizing the data requirements.
- Logical schema: describes the structure of some domain of information. This consists of descriptions of (for example) tables, columns, object-oriented classes, and XML tags. The logical schema and conceptual schema are sometimes implemented as one and the same.
- Physical schema: describes the physical means used to store data. This is concerned with partitions, CPUs, tablespaces, and the like.

According to ANSI, this approach allows the three perspectives to be relatively independent of each other. Storage technology can change without affecting either the logical or the conceptual schema. The table/column structure can change without (necessarily) affecting the conceptual schema. In each case, of course, the structures must remain consistent across all schemas of the same data model.

Data modeling process

In the context of business process integration (see figure), data modeling complements business process modeling, and ultimately results in database generation.



Data modeling in the context of Business Process Integration.[4]

The process of designing a database involves producing the previously described three types of schemas - conceptual, logical, and physical. The database design documented in these schemas are converted through a Data Definition Language, which can then be used to generate a database. A fully attributed data model contains detailed attributes (descriptions) for every entity within it. The term "database design" can describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term "database design" could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the Database Management System or DBMS.

In the process, system interfaces account for 25% to 70% of the development and support costs of current systems. The primary reason for this cost is that these systems do not share a common data model. If data models are developed on a system by system basis, then not only is the same analysis repeated in overlapping areas, but further analysis must be performed to create the interfaces between them. Most systems within an organization contain the same basic data, redeveloped for a specific purpose. Therefore, an efficiently designed basic data model can minimize rework with minimal modifications for the purposes of different systems within the organization

Modeling methodologies

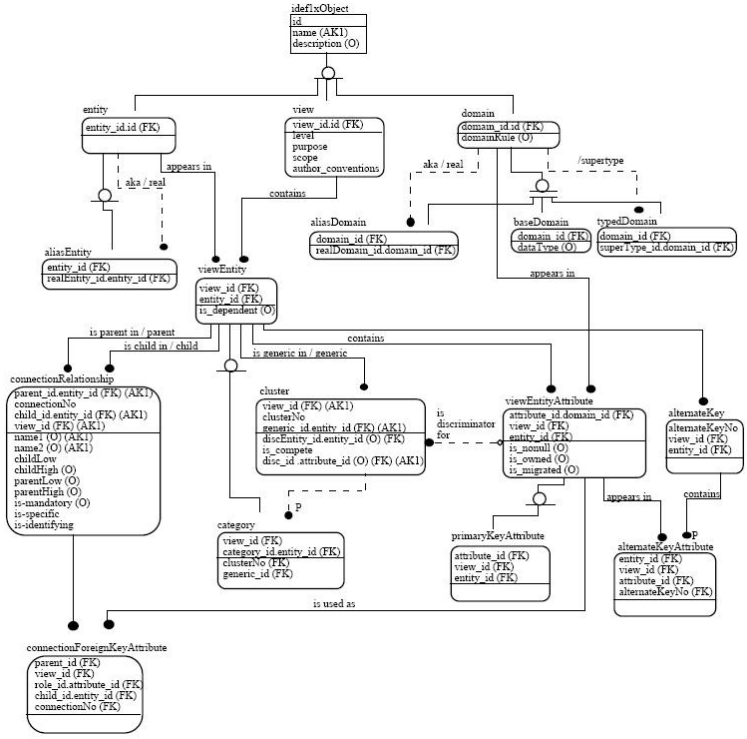
Data models represent information areas of interest. While there are many ways to create data models, according to Len Silverston (1997)[5] only two modeling methodologies stand out, top-down and bottom-up:

- Bottom-up models or View Integration models are often the result of a reengineering effort. They usually start with existing data structures forms, fields on application screens, or reports. These models are usually physical, application-specific, and incomplete from an enterprise perspective. They may not promote data sharing, especially if they are built without reference to other parts of the organization.
- Top-down logical data models, on the other hand, are created in an abstract way by getting information from people who know the subject area. A system may not implement all the entities in a logical model, but the model serves as a reference point or template.

Sometimes models are created in a mixture of the two methods: by considering the data needs and structure of an application and by consistently referencing a subject-area model. Unfortunately, in many environments the distinction between a logical data model and a physical data model is blurred. In addition, some CASE tools don't make a distinction between logical and physical data models.

Entity relationship diagrams

There are several notations for data modeling. The actual model is frequently called "Entity relationship model", because it depicts data in terms of the entities and relationships described in the data. An entity-relationship model (ERM) is an abstract conceptual representation of structured data. Entity-relationship modeling is a relational schema database modeling method, used in software engineering to produce a type of conceptual data model (or semantic data model) of a system, often a relational database, and its requirements in a top-down fashion.



Example of an IDEF1X Entity relationship diagrams used to model IDEF1X itself. The name of the view is mm. The domain hierarchy and constraints are also given. The constraints are expressed as sentences in the formal theory of the meta model.[6]

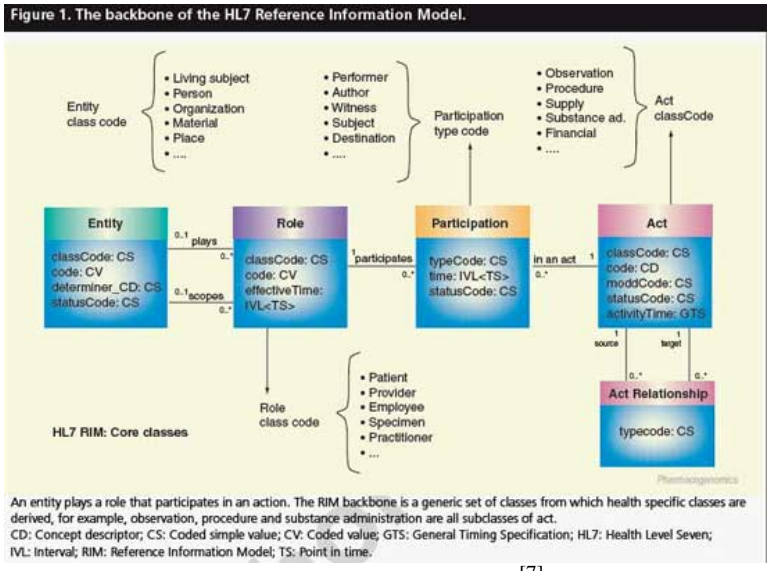
These models are being used in the first stage of information system design during the requirements analysis to describe information needs or the type of information that is to be stored in a database. The data modeling technique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain universe of discourse i.e. area of interest.

Several techniques have been developed for the design of data models. While these methodologies guide data modelers in their work, two different people using the same methodology will often come up with very different results. Most notable are:

- Bachman diagrams
- Barker's Notation
- Chen's Notation
- Data Vault Modeling
- Extended Backus–Naur form
- IDEF1X
- Object-relational mapping
- Object-Role Modeling
- Relational Model
- Relational Model/Tasmania

Generic data modeling

Generic data models are generalizations of conventional data models. They define standardized general relation types, together with the kinds of things that may be related by such a relation type. The definition of generic data model is similar to the definition of a natural language. For example, a generic data model may define relation types such as a 'classification relation', being a binary relation between an individual thing and a kind of thing (a class) and a 'part-whole relation', being a binary relation between two things, one with the role of part, the other with the role of whole, regardless the kind of things that are related.



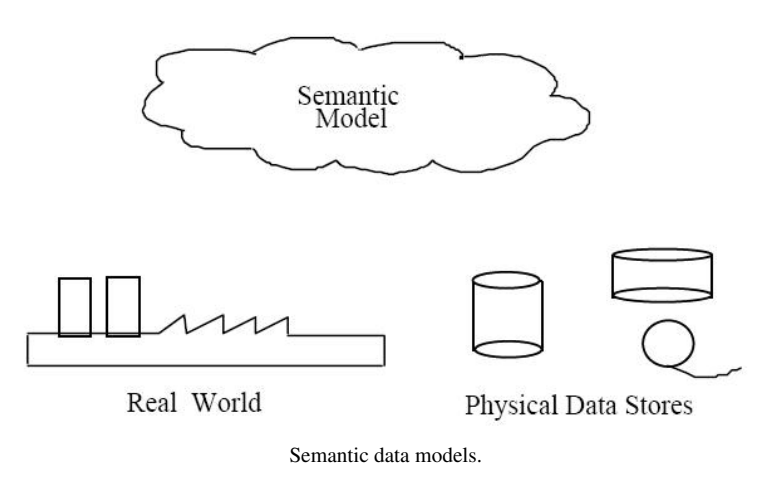
Example of a Generic data model.[7]

Given an extensible list of classes, this allows the classification of any individual thing and to specify part-whole relations for any individual object. By standardization of an extensible list of relation types, a generic data model enables the expression of an unlimited number of kinds of facts and will approach the capabilities of natural languages. Conventional data models, on the other hand, have a fixed and limited domain scope, because the instantiation (usage) of such a model only allows expressions of kinds of facts that are predefined in the model.

Semantic data modeling

The logical data structure of a DBMS, whether hierarchical, network, or relational, cannot totally satisfy the requirements for a conceptual definition of data because it is limited in scope and biased toward the implementation strategy employed by the DBMS.

Therefore, the need to define data from a conceptual view has led to the development of semantic data modeling techniques. That is, techniques to define the meaning of data within the context of its interrelationships with other data. As illustrated in the figure the real world, in terms of resources, ideas, events, etc., are symbolically defined within physical data stores. A semantic data model is an abstraction which defines how the stored symbols relate to the real world. Thus, the model must be a true representation of the real world.



Semantic data models.

A semantic data model can be used to serve many purposes, such as:

- planning of data resources
- building of shareable databases

- evaluation of vendor software
- integration of existing databases

The overall goal of semantic data models is to capture more meaning of data by integrating relational concepts with more powerful abstraction concepts known from the Artificial Intelligence field. The idea is to provide high level modeling primitives as integral part of a data model in order to facilitate the representation of real world situations.[8]

References

[1] Simison, Graeme. C. & Witt, Graham. C. (2005).*Data Modeling Essentials*.3rd Edition. Morgan Kauffman Publishers. ISBN 0-12-644551-6

[2] Data Integration Glossary (http://knowledge.fhwa.dot.gov/tam/aashto.nsf/All+Documents/4825476B2B5C687285256B1F00544258/$FILE/DIGloss.pdf), U.S. Department of Transportation, August 2001.

[3] American National Standards Institute. 1975. *ANSI/X3/SPARC Study Group on Data Base Management Systems; Interim Report*. FDT (Bulletin of ACM SIGMOD) 7:2.

[4] Paul R. Smith & Richard Sarfaty (1993). Creating a strategic plan for configuration management using Computer Aided Software Engineering (CASE) tools. (http://www.osti.gov/energycitations/purl.cover.jsp;jsessionid=6192EDBFBAB7DCED13883C55F221221A?purl=/10160331-YhIRrY/) Paper For 1993 National DOE/Contractors and Facilities CAD/CAE User's Group.

[5] Len Silverston, W.H.Inmon, Kent Graziano (2007). *The Data Model Resource Book*. Wiley, 1997. ISBN 0-471-15364-8. Reviewed by Van Scott on tdan.com (http://www.tdan.com/view-book-reviews/5593). Accessed 1 Nov 2008.

[6] FIPS Publication 184 (http://www.itl.nist.gov/fipspubs/idef1x.doc) released of IDEF1X by the Computer Systems Laboratory of the National Institute of Standards and Technology (NIST). 21 December 1993.

[7] Amnon Shabo (2006). Clinical genomics data standards for pharmacogenetics and pharmacogenomics (http://healthit.hhs.gov/portal/server.pt?open=512&objID=1263&mode=2).

[8] "Semantic data modeling" In: *Metaclasses and Their Application*. Book Series Lecture Notes in Computer Science. Publisher Springer Berlin / Heidelberg. Volume Volume 943/1995.

Further reading

- J.H. ter Bekke (1991). *Semantic Data Modeling in Relational Environments*
- John Vincent Carlis, Joseph D. Maguire (2001). *Mastering Data Modeling: A User-driven Approach*.
- Alan Chmura, J. Mark Heumann (2005). *Logical Data Modeling: What it is and how to Do it*.
- Martin E. Modell (1992). *Data Analysis, Data Modeling, and Classification*.
- M. Papazoglou, Stefano Spaccapietra, Zahir Tari (2000). *Advances in Object-oriented Data Modeling*.
- G. Lawrence Sanders (1995). *Data Modeling*
- Graeme C. Simsion, Graham C. Witt (2005). *Data Modeling Essentials*'
- Graeme Simsion (2007). *Data Modeling: Theory and Practice*.
- Chris Bradley, Donna Burbank, Steve Hoberman (2009). *Data Modeling for the Business: A Handbook for Aligning the Business with IT using High-Level Data Models*
- Matthew West (2011) *Developing High Quality Data Models*

External links

- Agile/Evolutionary Data Modeling (http://www.agiledata.org/essays/agileDataModeling.html)
- Data modeling articles (http://www.softdevarticles.com/modules/weblinks/viewcat.php?cid=21)
- Database Modelling in UML (http://www.methodsandtools.com/archive/archive.php?id=9)
- Data Modeling 101 (http://www.agiledata.org/essays/dataModeling101.html)
- Semantic data modeling (http://www.jhterbekke.net/SemanticDataModeling.html)
- System Development, Methodologies and Modeling (http://www.cems.uwe.ac.uk/~tdrewry/modeling.htm) Notes on by Tony Drewry
- Request For Proposal - Information Management Metamodel (IMM) (http://www.omg.org/cgi-bin/doc?ab/05-12-02) of the Object Management Group
- Data Modeling is NOT just for DBMS's Part 1 (http://www.ipl.com/papers/Data modelling is NOT JUST for DBMS part 1.pdf) Chris Bradley
- Data Modeling is NOT just for DBMS's Part 2 (http://www.ipl.com/papers/Data modelling is NOT JUST for DBMS part 2.pdf) Chris Bradley

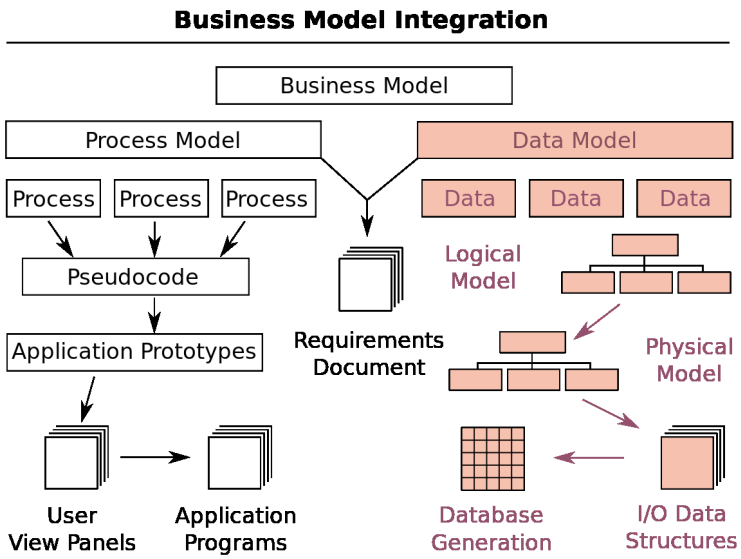
Data model

In software engineering, the term **data model** is used in two related senses. In the sense covered by this article, it is a description of the objects represented by a computer system together with their properties and relationships; these are typically "real world" objects such as products, suppliers, customers, and orders. In the second sense, covered by the article database model, it means a collection of concepts and rules used in defining data models: for example the relational model uses relations and tuples, while the network model uses records, sets, and fields.

Data models are often used as an aid to communication between the business people defining the requirements for a computer system and the technical people defining the design in response to those requirements. They are used to show the data needed and created by business processes.

According to Hoberman (2009), "A data model is a wayfinding tool for both business and IT professionals, which uses a set of symbols and text to precisely explain a subset of real information to improve communication within the organization and thereby lead to a more flexible and stable application environment."[2]

A data model explicitly determines the structure of data. Data models are specified in a data modeling notation, which is often graphical in form.[3]



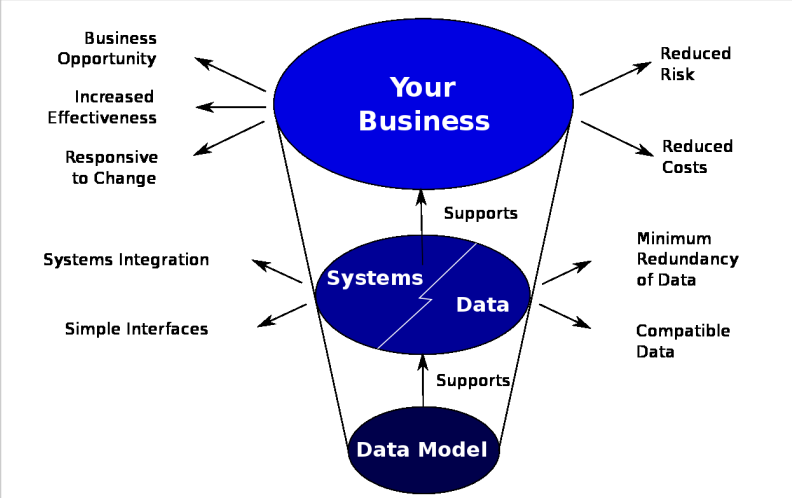
Overview of data modeling context: Data model is based on Data, Data relationship, Data semantic and Data constraint. A data model provides the details of information to be stored, and is of primary use when the final product is the generation of computer software code for an application or the preparation of a functional specification to aid a computer software make-or-buy decision. The figure is an example of the interaction between process and data models.[1]

A data model can be sometimes referred to as a data structure, especially in the context of programming languages. Data models are often complemented by function models, especially in the context of enterprise models.

Overview

Managing large quantities of structured and unstructured data is a primary function of information systems. Data models describe structured data for storage in data management systems such as relational databases. They typically do not describe unstructured data, such as word processing documents, email messages, pictures, digital audio, and video.

The role of data models



How data models deliver benefit.[4]

The main aim of data models is to support the development of information systems by providing the definition and format of data. According to West and Fowler (1999) "if this is done consistently across systems then compatibility of data can be achieved. If the same data structures are used to store and access data then different applications can share data. The results of this are indicated above. However, systems and interfaces often cost more than they should, to build, operate, and maintain. They may also constrain the business rather than support it. A major cause is that the quality of the data models implemented in systems and interfaces is poor".

- "Business rules, specific to how things are done in a particular place, are often fixed in the structure of a data model. This means that small changes in the way business is conducted lead to large changes in computer systems and interfaces".
- "Entity types are often not identified, or incorrectly identified. This can lead to replication of data, data structure, and functionality, together with the attendant costs of that duplication in development and maintenance".
- "Data models for different systems are arbitrarily different. The result of this is that complex interfaces are required between systems that share data. These interfaces can account for between 25-70% of the cost of current systems".
- "Data cannot be shared electronically with customers and suppliers, because the structure and meaning of data has not been standardised. For example, engineering design data and drawings for process plant are still sometimes exchanged on paper".

The reason for these problems is a lack of standards that will ensure that data models will both meet business needs and be consistent. According to Hoberman (2009), "A data model is a wayfinding tool for both business and IT professionals, which uses a set of symbols and text to precisely explain a subset of real information to improve communication within the organization and thereby lead to a more flexible and stable application environment."[2]

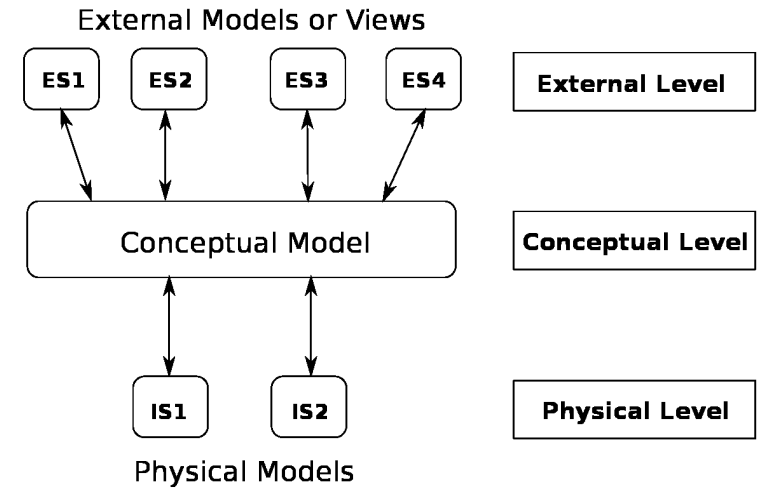
A data model explicitly determines the structure of data or structured data. Typical applications of data models include database models, design of information systems, and enabling exchange of data. Usually data models are specified in a data modeling language.[3]

Communication and precision are the two key benefits that make a data model important to applications that use and exchange data. A data model is the medium which project team members from different backgrounds and with different levels of experience can communicate with one another. Precision means that the terms and rules on a data model can be interpreted only one way and are not ambiguous.[2]

A data model can be sometimes referred to as a data structure, especially in the context of programming languages. Data models are often complemented by function models, especially in the context of enterprise models.

Three perspectives

A data model *instance* may be one of three kinds according to ANSI in 1975:[5]



The ANSI/SPARC three level architecture. This shows that a data model can be an external model (or view), a conceptual model, or a physical model. This is not the only way to look at data models, but it is a useful way, particularly when comparing models.

- Conceptual data model : describes the semantics of a domain, being the scope of the model. For example, it may be a model of the interest area of an organization or industry. This consists of entity classes, representing kinds of things of significance in the domain, and relationships assertions about associations between pairs of entity classes. A conceptual schema specifies the kinds of facts or propositions that can be expressed using the model. In that sense, it defines the allowed expressions in an artificial 'language' with a scope that is limited by the scope of the model. The use of conceptual schema has evolved to become a powerful communication tool with business users. Often called a subject area model (SAM) or high-level data model (HDM), this model is used to communicate core data concepts, rules, and definitions to a business user as part of an overall application development or enterprise initiative. The number of objects should be very small and focused on key concepts. Try to limit this model to one page, although for extremely large organizations or complex projects, the model might span two or more pages.[6]
- Logical data model : describes the semantics, as represented by a particular data manipulation technology. This consists of descriptions of tables and columns, object oriented classes, and XML tags, among other things.
- Physical data model : describes the physical means by which data are stored. This is concerned with partitions, CPUs, tablespaces, and the like.

The significance of this approach, according to ANSI, is that it allows the three perspectives to be relatively independent of each other. Storage technology can change without affecting either the logical or the conceptual model. The table/column structure can change without (necessarily) affecting the conceptual model. In each case, of course, the structures must remain consistent with the other model. The table/column structure may be different from a direct translation of the entity classes and attributes, but it must ultimately carry out the objectives of the conceptual entity class structure. Early phases of many software development projects emphasize the design of a conceptual data model. Such a design can be detailed into a logical data model. In later stages, this model may be translated into physical data model. However, it is also possible to implement a conceptual model directly.

History

One of the earliest pioneering works in modelling information systems was done by Young and Kent (1958),[7][8] who argued for "a precise and abstract way of specifying the informational and time characteristics of a data processing problem". They wanted to create "a notation that should enable the analyst to organize the problem around any piece of hardware". Their work was a first effort to create an abstract specification and invariant basis for designing different alternative implementations using different hardware components. A next step in IS modelling was taken by CODASYL, an IT industry consortium formed in 1959, who essentially aimed at the same thing as Young and Kent: the development of "a proper structure for machine independent problem definition language, at the system level of data processing". This led to the development of a specific IS information algebra.

In the 1960s data modeling gained more significance with the initiation of the management information system (MIS) concept. According to Leondes (2002), "during that time, the information system provided the data and information for management purposes. The first generation database system, called Integrated Data Store (IDS), was designed by Charles Bachman at General Electric. Two famous database models, the network data model and the hierarchical data model, were proposed during this period of time".[9] Towards the end of the 1960s Edgar F. Codd worked out his theories of data arrangement, and proposed the relational model for database management based on first-order predicate logic.[10]

In the 1970s entity relationship modeling emerged as a new type of conceptual data modeling, originally proposed in 1976 by Peter Chen. Entity relationship models were being used in the first stage of information system design during the requirements analysis to describe information needs or the type of information that is to be stored in a database. This technique can describe any ontology, i.e., an overview and classification of concepts and their relationships, for a certain area of interest.

In the 1970s G.M. Nijssen developed "Natural Language Information Analysis Method" (NIAM) method, and developed this in the 1980s in cooperation with Terry Halpin into Object-Role Modeling (ORM).

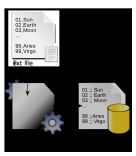
Bill Kent, in his 1978 book *Data and Reality* compared a data model to a map of a territory, emphasizing that in the real world, "highways are not painted red, rivers don't have county lines running down the middle, and you can't see contour lines on a mountain". In contrast to other researchers who tried to create models that were mathematically clean and elegant, Kent emphasized the essential messiness of the real world, and the task of the data modeller to create order out of chaos without excessively distorting the truth.

In the 1980s according to Jan L. Harrington (2000) "the development of the object-oriented paradigm brought about a fundamental change in the way we look at data and the procedures that operate on data. Traditionally, data and procedures have been stored separately: the data and their relationship in a database, the procedures in an application program. Object orientation, however, combined an entity's procedure with its data."[11]

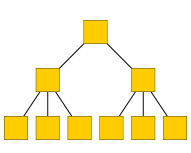
Types of data models

Database model

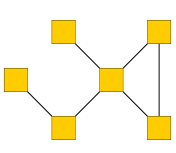
A database model is a specification describing how a database is structured and used. Several such models have been suggested. Common models include:



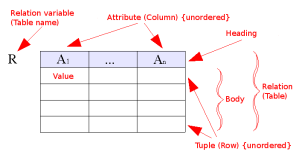
Flat model



Hierarchical model

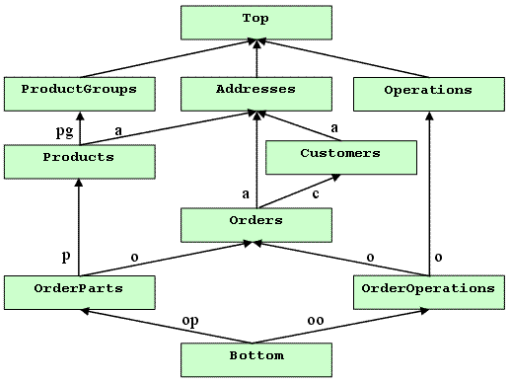


Network model

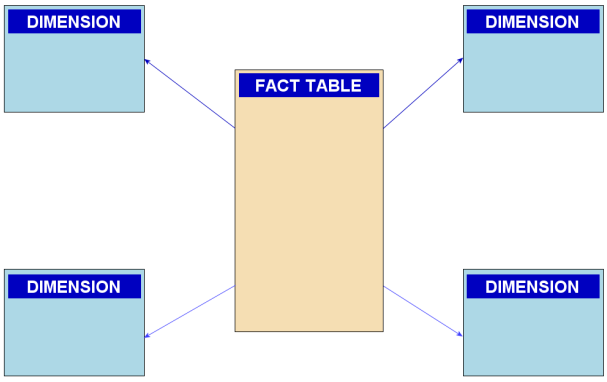


Relational model

- Flat model: This may not strictly qualify as a data model. The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another.
- Hierarchical model: In this model data is organized into a tree-like structure, implying a single upward link in each record to describe the nesting, and a sort field to keep the records in a particular order in each same-level list.
- Network model: This model organizes data using two fundamental constructs, called records and sets. Records contain fields, and sets define one-to-many relationships between records: one owner, many members.
- Relational model: is a database model based on first-order predicate logic. Its core idea is to describe a database as a collection of predicates over a finite set of predicate variables, describing constraints on the possible values and combinations of values.



Concept-oriented model

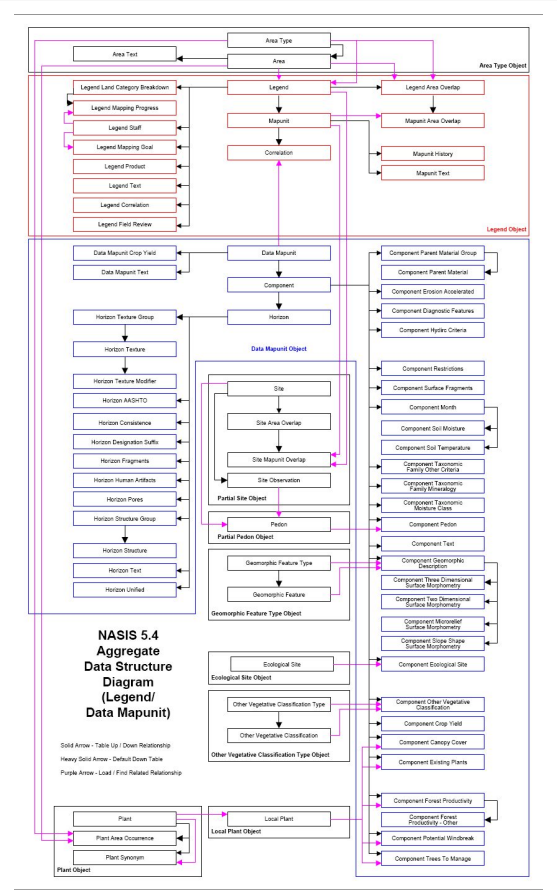


Star schema

- Object-relational model: Similar to a relational database model, but objects, classes and inheritance are directly supported in database schemas and in the query language.
- Star schema is the simplest style of data warehouse schema. The star schema consists of a few "fact tables" (possibly only one, justifying the name) referencing any number of "dimension tables". The star schema is considered an important special case of the snowflake schema.

Data Structure Diagram

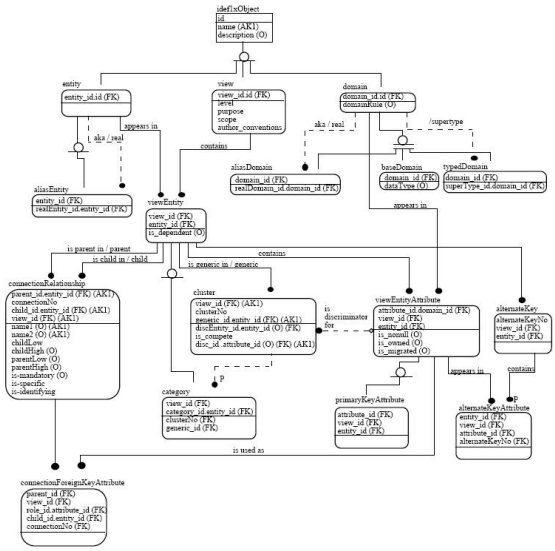
A data structure diagram (DSD) is a diagram and data model used to describe conceptual data models by providing graphical notations which document entities and their relationships, and the constraints that bind them. The basic graphic elements of DSDs are boxes, representing entities, and arrows, representing relationships. Data structure diagrams are most useful for documenting complex data entities.



Example of a Data Structure Diagram.

Data structure diagrams are an extension of the entity-relationship model (ER model). In DSDs, attributes are specified inside the entity boxes rather than outside of them, while relationships are drawn as boxes composed of attributes which specify the constraints that bind entities together. The E-R model, while robust, doesn't provide a way to specify the constraints between relationships, and becomes visually cumbersome when representing entities with several attributes. DSDs differ from the ER model in that the ER model focuses on the relationships between different entities, whereas DSDs focus on the relationships of the elements within an entity and enable users to fully see the links and relationships between each entity.

There are several styles for representing data structure diagrams, with the notable difference in the manner of defining cardinality. The choices are between arrow heads, inverted arrow heads (crow's feet), or numerical representation of the cardinality.



Example of an IDEF1X Entity relationship diagrams used to model IDEF1X itself.[12]

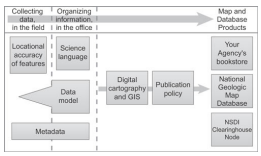
Entity-relationship model

An entity-relationship model (ERM) is an abstract conceptual data model (or semantic data model) used in software engineering to represent structured data. There are several notations used for ERMs.

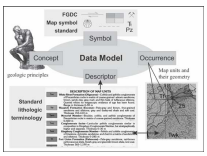
Geographic data model

A data model in Geographic information systems is a mathematical construct for representing geographic objects or surfaces as data. For example,

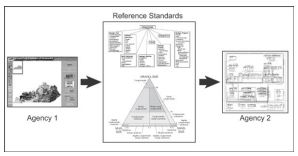
- the vector data model represents geography as collections of points, lines, and polygons;
- the raster data model represent geography as cell matrixes that store numeric values;
- and the Triangulated irregular network (TIN) data model represents geography as sets of contiguous, nonoverlapping triangles.[13]



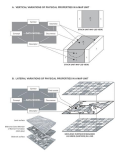
Groups relate to process of making a map[1]



NGMDB data model applications[2]



NGMDB databases linked together[3]



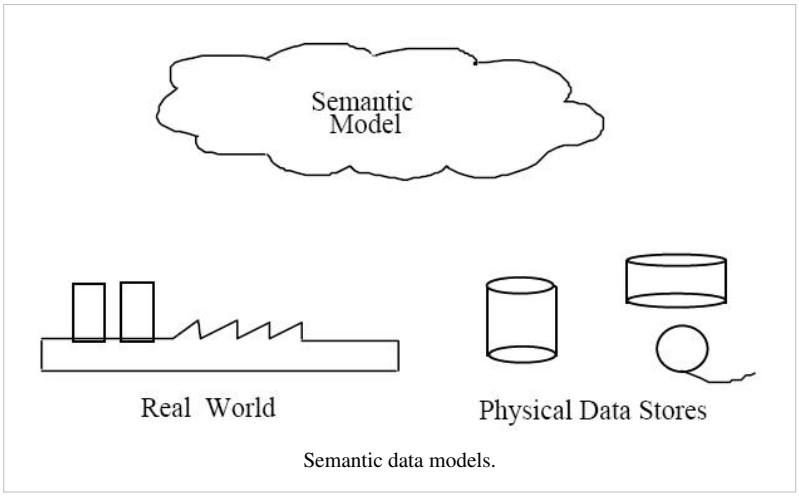
Representing 3D map information[4]

Generic data model

Generic data models are generalizations of conventional data models. They define standardised general relation types, together with the kinds of things that may be related by such a relation type. Generic data models are developed as an approach to solve some shortcomings of conventional data models. For example, different modelers usually produce different conventional data models of the same domain. This can lead to difficulty in bringing the models of different people together and is an obstacle for data exchange and data integration. Invariably, however, this difference is attributable to different levels of abstraction in the models and differences in the kinds of facts that can be instantiated (the semantic expression capabilities of the models). The modelers need to communicate and agree on certain elements which are to be rendered more concretely, in order to make the differences less significant.

Semantic data model

A semantic data model in software engineering is a technique to define the meaning of data within the context of its interrelationships with other data. A semantic data model is an abstraction which defines how the stored symbols relate to the real world. A semantic data model is sometimes called a conceptual data model.



Semantic data models.

The logical data structure of a database management system (DBMS), whether hierarchical, network, or relational, cannot totally satisfy the requirements for a conceptual definition of data because it is limited in scope and biased toward the implementation strategy employed by the DBMS. Therefore, the need to define data from a conceptual view has led to the development of semantic data modeling techniques. That is, techniques to define the meaning of data within the context of its interrelationships with other data. As illustrated in the figure. The real world, in terms of resources, ideas, events, etc., are symbolically defined within physical data stores. A semantic data model is an abstraction which defines how the stored symbols relate to the real world. Thus, the model must be a true representation of the real world.

Data model topics

Data architecture

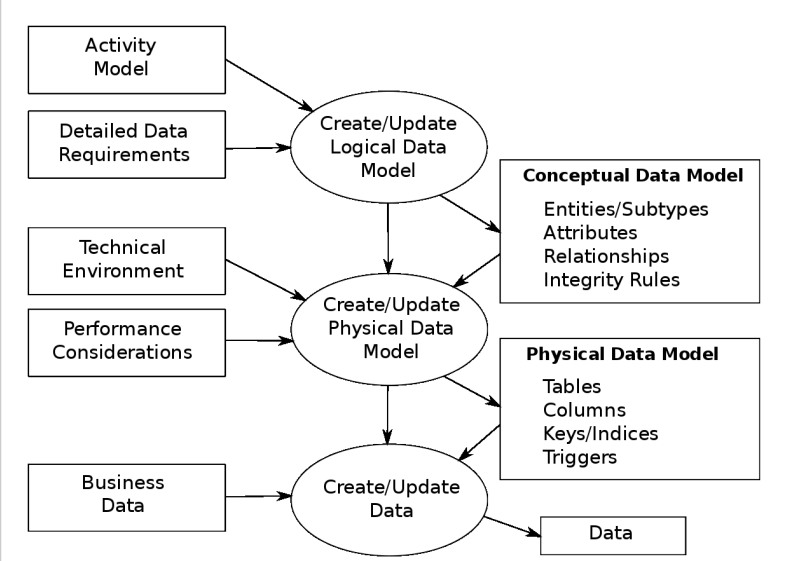
Data architecture is the design of data for use in defining the target state and the subsequent planning needed to hit the target state. It is usually one of several architecture domains that form the pillars of an enterprise architecture or solution architecture.

A data architecture describes the data structures used by a business and/or its applications. There are descriptions of data in storage and data in motion; descriptions of data stores, data groups and data items; and mappings of those data artifacts to data qualities, applications, locations etc.

Essential to realizing the target state, Data architecture describes how data is processed, stored, and utilized in a given system. It provides criteria for data processing operations that make it possible to design data flows and also control the flow of data in the system.

Data modeling

Data modeling in software engineering is the process of creating a data model by applying formal data model descriptions using data modeling techniques. Data modeling is a technique for defining business requirements for a database. It is sometimes called *database modeling* because a data model is eventually implemented in a database.[14]



The data modeling process.

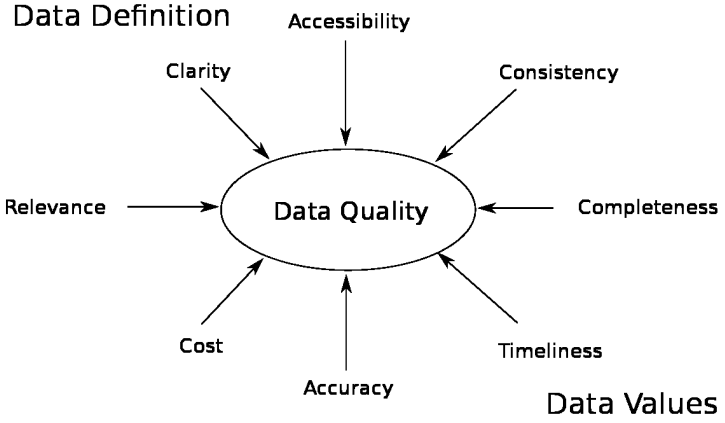
The figure illustrates the way data models are developed and used today. A conceptual data model is developed based on the data requirements for the application that is being developed, perhaps in the context of an activity model. The data model will normally consist of entity types, attributes, relationships, integrity rules, and the definitions of those objects. This is then used as the start point for interface or database design.

Data properties

Some important properties of data for which requirements need to be met are:

- definition-related properties
  - *relevance*: the usefulness of the data in the context of your business.
  - *clarity*: the availability of a clear and shared definition for the data.
  - *consistency*: the compatibility of the same type of data from different sources.

- content-related properties
  - *timeliness*: the availability of data at the time required and how up to date that data is.
  - *accuracy*: how close to the truth the data is.
- properties related to both definition and content
  - *completeness*: how much of the required data is available.
  - *accessibility*: where, how, and to whom the data is available or not available (e.g. security).



Some important properties of data.

- *cost*: the cost incurred in obtaining the data, and making it available for use.

Data organization

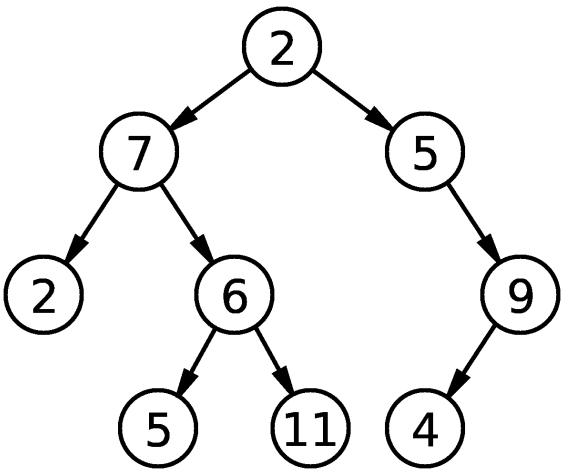
Another kind of data model describes how to organize data using a database management system or other data management technology. It describes, for example, relational tables and columns or object-oriented classes and attributes. Such a data model is sometimes referred to as the *physical data model*, but in the original ANSI three schema architecture, it is called "logical". In that architecture, the physical model describes the storage media (cylinders, tracks, and tablespaces). Ideally, this model is derived from the more conceptual data model described above. It may differ, however, to account for constraints like processing capacity and usage patterns.

While *data analysis* is a common term for data modeling, the activity actually has more in common with the ideas and methods of *synthesis* (inferring general concepts from particular instances) than it does with *analysis* (identifying component concepts from more general ones). {*Presumably we call ourselves systems analysts because no one can say systems synthesists.*} Data modeling strives to bring the data structures of interest together into a cohesive, inseparable, whole by eliminating unnecessary data redundancies and by relating data structures with relationships.

A different approach is to use adaptive systems such as artificial neural networks that can autonomously create implicit models of data.

Data structure

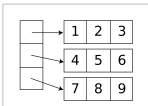
A data structure is a way of storing data in a computer so that it can be used efficiently. It is an organization of mathematical and logical concepts of data. Often a carefully chosen data structure will allow the most efficient algorithm to be used. The choice of the data structure often begins from the choice of an abstract data type.



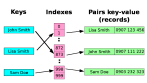
A binary tree, a simple type of branching linked data structure.

A data model describes the structure of the data within a given domain and, by implication, the underlying structure of that domain itself. This means that a data model in fact specifies a dedicated *grammar* for a dedicated artificial language for that domain. A data model represents classes of entities (kinds of things) about which a company wishes to hold information, the attributes of that information, and relationships among those entities and (often implicit) relationships among those attributes. The model describes the organization of the data to some extent irrespective of how data might be represented in a computer system.

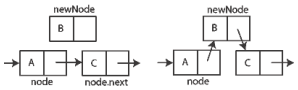
The entities represented by a data model can be the tangible entities, but models that include such concrete entity classes tend to change over time. Robust data models often identify abstractions of such entities. For example, a data model might include an entity class called "Person", representing all the people who interact with an organization. Such an abstract entity class is typically more appropriate than ones called "Vendor" or "Employee", which identify specific roles played by those people.



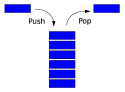
Array



Hash table



Linked list



Stack (data structure)

Data model theory

The term data model can have two meanings:[15]

1. A data model *theory*, i.e. a formal description of how data may be structured and accessed.
2. A data model *instance*, i.e. applying a data model *theory* to create a practical data model *instance* for some particular application.

A data model theory has three main components:

- The structural part: a collection of data structures which are used to create databases representing the entities or objects modeled by the database.
- The integrity part: a collection of rules governing the constraints placed on these data structures to ensure structural integrity.
- The manipulation part: a collection of operators which can be applied to the data structures, to update and query the data contained in the database.

For example, in the relational model, the structural part is based on a modified concept of the mathematical relation; the integrity part is expressed in first-order logic and the manipulation part is expressed using the relational algebra, tuple calculus and domain calculus.

A data model instance is created by applying a data model theory. This is typically done to solve some business enterprise requirement. Business requirements are normally captured by a semantic logical data model. This is transformed into a physical data model instance from which is generated a physical database. For example, a data modeler may use a data modeling tool to create an entity-relationship model of the corporate data repository of some business enterprise. This model is transformed into a relational model, which in turn generates a relational database.

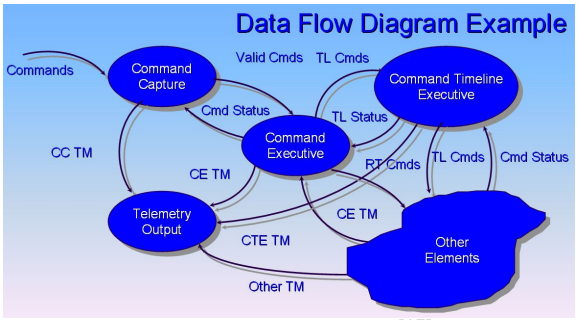
Patterns

Patterns[16] are common data modeling structures that occur in many data models.

Related models

Data flow diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. It differs from the flowchart as it shows the *data* flow instead of the *control* flow of the program. A data flow diagram can also be used for the visualization of data processing (structured design). Data flow diagrams were invented by Larry Constantine, the original developer of structured design,[18] based on Martin and Estrin's "data flow graph" model of computation.

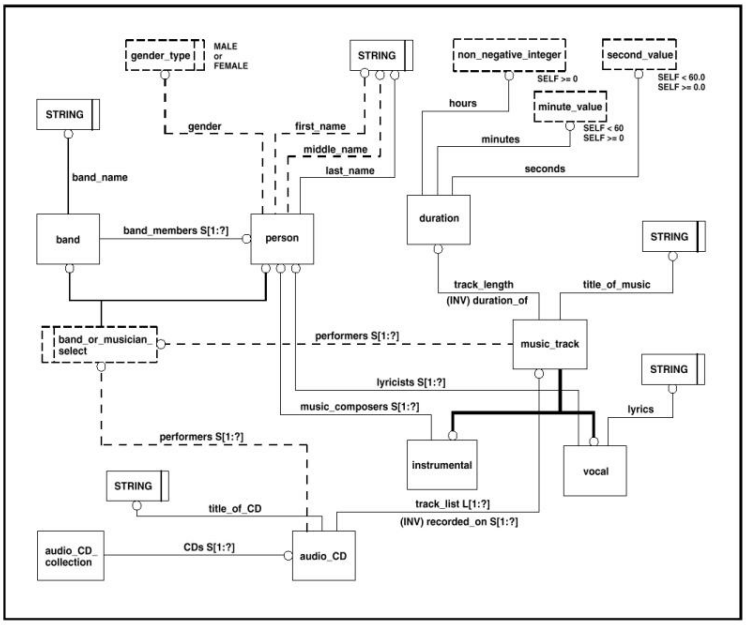


Data Flow Diagram example.[17]

It is common practice to draw a context-level Data flow diagram first which shows the interaction between the system and outside entities. The **DFD** is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts. This context-level Data flow diagram is then "exploded" to show more detail of the system being modeled

Information model

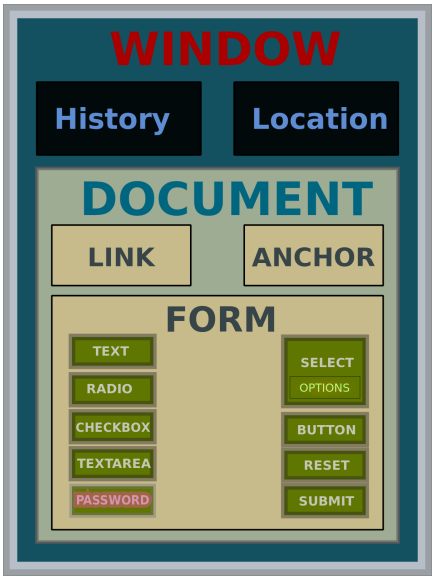
An Information model is not a type of data model, but more or less an alternative model. Within the field of software engineering both a data model and an information model can be abstract, formal representations of entity types that includes their properties, relationships and the operations that can be performed on them. The entity types in the model may be kinds of real-world objects, such as devices in a network, or they may themselves be abstract, such as for the entities used in a billing system. Typically, they are used to model a constrained domain that can be described by a closed set of entity types, properties, relationships and operations.



Example of an EXPRESS G Information model.

According to Lee (1999) an information model is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. It can provide sharable, stable, and organized structure of information requirements for the domain context.[] More in general the term *information model* is used for models of individual things, such as facilities, buildings, process plants, etc. In those cases the concept is specialised to Facility Information Model, Building Information Model, Plant Information Model, etc. Such an information model is an integration of a model of the facility with the data and documents about the facility.

An information model provides formalism to the description of a problem domain without constraining how that description is mapped to an actual implementation in software. There may be many mappings of the information model. Such mappings are called data models, irrespective of whether they are object models (e.g. using UML), entity relationship models or XML schemas.



Document Object Model, a standard object model for representing HTML or XML.

Object model

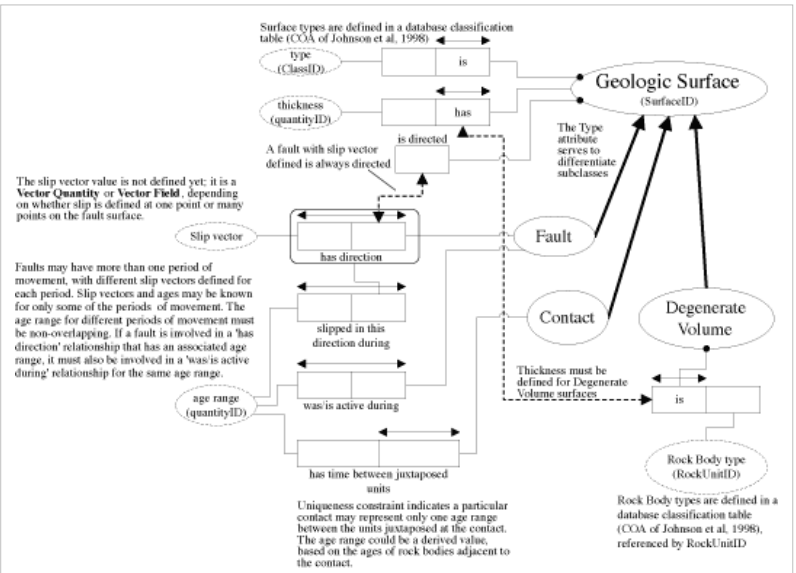
An object model in computer science is a collection of objects or classes through which a program can examine and manipulate some specific parts of its world. In other words, the object-oriented interface to some service or system. Such an interface is said to be the *object model of* the represented service or system. For example, the Document Object Model (DOM) [19] is a collection of objects that represent a page in a web browser, used by script programs to examine and dynamically change the page. There is a Microsoft Excel object model[20] for controlling Microsoft Excel from another program, and the ASCOM Telescope Driver is an object model for controlling an astronomical telescope.

In computing the term *object model* has a distinct second meaning of the general properties of objects in a specific computer programming language, technology, notation or methodology that uses them. For example, the *Java object model*, the *COM object model*, or *the object model of OMT*. Such object models are usually defined using concepts such as class, message, inheritance, polymorphism, and encapsulation. There is an extensive literature on formalized object models as a subset of the formal semantics of programming languages.

Object-Role Model

Object-Role Modeling (ORM) is a method for conceptual modeling, and can be used as a tool for information and rules analysis.[22]

Object-Role Modeling is a fact-oriented method for performing systems analysis at the conceptual level. The quality of a database application depends critically on its design. To help ensure correctness, clarity, adaptability and productivity, information systems are best specified first at the conceptual level, using concepts and language that people can readily understand.



Example of the application of Object-Role Modeling in a "Schema for Geologic Surface", Stephen M. Richard (1999).[21]

The conceptual design may include data, process and behavioral perspectives, and the actual DBMS used to implement the design might be based on one of many logical data models (relational, hierarchic, network, object-oriented etc.).[23]

Unified Modeling Language models

The Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. It is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The Unified Modeling Language offers a standard way to write a system's blueprints, including:[24]

- Conceptual things such as business processes and system functions
- Concrete things such as programming language statements, database schemas, and
- Reusable software components.

UML offers a mix of functional models, data models, and database models.

References

[1] Paul R. Smith & Richard Sarfaty (1993). Creating a strategic plan for configuration management using Computer Aided Software Engineering (CASE) tools. (http://www.osti.gov/energycitations/servlets/purl/10160331-YhIRrY/) Paper For 1993 National DOE/Contractors and Facilities CAD/CAE User's Group.

[2] "Data Modeling Made Simple 2nd Edition", Steve Hoberman, Technics Publications, LLC 2009 (http://www.technicspub.com/product.sc;jsessionid=BDF2DA4FC5B78BDBF732346B13FEE048.qscstrfrnt01?productId=1&categoryId=1)

[3] Michael R. McCaleb (1999). "A Conceptual Data Model of Datum Systems" (http://nvl.nist.gov/pub/nistpubs/jres/104/4/html/j44mac.htm#apa). National Institute of Standards and Technology. August 1999.

[4] Matthew West and Julian Fowler (1999). Developing High Quality Data Models (https://sites.google.com/site/drmatthewwest/publications/princ03.pdf). The European Process Industries STEP Technical Liaison Executive (EPISTLE).

[5] American National Standards Institute. 1975. *ANSI/X3/SPARC Study Group on Data Base Management Systems; Interim Report*. FDT (Bulletin of ACM SIGMOD) 7:2.

[6] "Data Modeling for the Business", Steve Hoberman, Donna Burbank, Chris Bradley, Technics Publications, LLC 2009 (http://www.technicspub.com/product.sc?productId=7&categoryId=1)

[7] Young, J. W., and Kent, H. K. (1958). "Abstract Formulation of Data Processing Problems". In: *Journal of Industrial Engineering*. Nov-Dec 1958. 9(6), pp. 471-479

[8] Janis A. Bubenko jr (2007) "From Information Algebra to Enterprise Modelling and Ontologies - a Historical Perspective on Modelling for Information Systems". In: *Conceptual Modelling in Information Systems Engineering*. John Krogstie et al. eds. pp 1-18

[9] Cornelius T. Leondes (2002). *Database and Data Communication Network Systems: Techniques and Applications*. Page 7

[10] *"Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks"*, E.F. Codd, IBM Research Report, 1969

[11] Jan L. Harrington (2000). *Object-oriented Database Design Clearly Explained*. p.4

[12] FIPS Publication 184 (http://www.itl.nist.gov/fipspubs/idef1x.doc) released of IDEF1X by the Computer Systems Laboratory of the National Institute of Standards and Technology (NIST). 21 December 1993.

[13] Wade, T. and Sommer, S. eds. *A to Z GIS* (http://store.esri.com/esri/showdetl.cfm?SID=2&Product_ID=868&Category_ID=49)

[14] Whitten, Jeffrey L.; Lonnie D. Bentley, Kevin C. Dittman. (2004). *Systems Analysis and Design Methods*. 6th edition. ISBN 0-256-19906-X.

[15] Beynon-Davies P. (2004). Database Systems 3rd Edition. Palgrave, Basingstoke, UK. ISBN 1-4039-1601-2

[16] "The Data Model Resource Book: Universal Patterns for Data Modeling" Len Silverstone & Paul Agnew (2008).

[17] John Azzolini (2000). Introduction to Systems Engineering Practices (http://ses.gsfc.nasa.gov/ses_data_2000/000712_Azzolini.ppt). July 2000.

[18] W. Stevens, G. Myers, L. Constantine, "Structured Design", IBM Systems Journal, 13 (2), 115-139, 1974.

[19] http://www.w3.org/DOM/

[20] Excel Object Model Overview (http://msdn2.microsoft.com/en-us/library/wss56bz7.aspx)

[21] Stephen M. Richard (1999). Geologic Concept Modeling (http://pubs.usgs.gov/of/1999/of99-386/richard.html). U.S. Geological Survey Open-File Report 99-386.

[22] Joachim Rossberg and Rickard Redler (2005). *Pro Scalable .NET 2.0 Application Designs.*. Page 27

[23] Object Role Modeling: An Overview (msdn.microsoft.com) (http://msdn2.microsoft.com/en-us/library/aa290383(VS.71).aspx). Retrieved 19 September 2008.

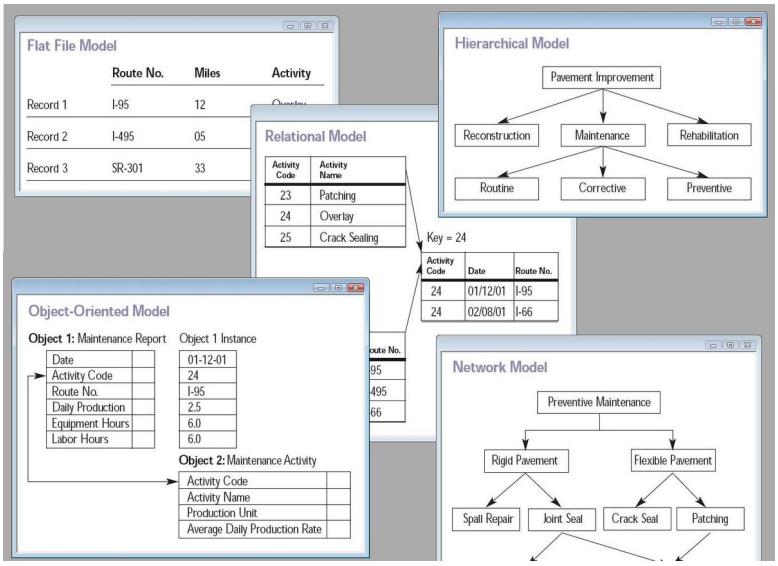
[24] Grady Booch, Ivar Jacobson & Jim Rumbaugh (2000) OMG Unified Modeling Language Specification (http://www.omg.org/docs/formal/00-03-01.pdf), Version 1.3 First Edition: March 2000. Retrieved 12 August 2008.

Further reading

- David C. Hay (1996). *Data Model Patterns: Conventions of Thought*. New York:Dorset House Publishers, Inc.
- Matthew West and Julian Fowler (1999). Developing High Quality Data Models (https://sites.google.com/site/drmatthewwest/publications/princ03.pdf?attredirects=0&d=1). The European Process Industries STEP Technical Liaison Executive (EPISTLE).
- Len Silverston (2001). *The Data Model Resource Book* Volume 1/2. John Wiley & Sons.
- RFC 3444 - On the Difference between Information Models and Data Models
- Len Silverston & Paul Agnew (2008). *The Data Model Resource Book: Universal Patterns for data Modeling* Volume 3. John Wiley & Sons.
- Steve Hoberman, Donna Burbank, & Chris Bradley (2009). Data Modeling for the Business (http://www.technicspub.com/product.sc;jsessionid=E97BA275BDCF51655949A252CEB05824.qscstrfrnt02?productId=7&categoryId=1). Technics Publications, LLC
- Andy Graham (2010), *The Enterprise Data Model: a framework for enterprise data architecture* (http://www.amazon.com/dp/0956582907)
- Wayne Eckerson (2011) Creating an Enterprise Data Strategy: Managing Data as a Corporate Asset (https://www.quest.com/Quest_Site_Assets/WhitePapers/Creating_an_Enterprise_Data_Strategy_Quest.pdf)

Database model

A **database model** is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model, which uses a table-based format.



Collage of five types of database models.

Common logical data models for databases include:

- Hierarchical database model
- Network model
- Relational model
- Entity–relationship model
  - Enhanced entity–relationship model
- Object model
- Document model
- Entity–attribute–value model
- Star schema

An object-relational database combines the two related structures.

Physical data models include:

- Inverted index
- Flat file

Other models include:

- Associative model
- Multidimensional model
- Multivalue model
- Semantic model
- XML database
- Named graph
- Triplestore

Relationships and functions

A given database management system may provide one or more of the five models. The optimal structure depends on the natural organization of the application's data, and on the application's requirements, which include transaction rate (speed), reliability, maintainability, scalability, and cost. Most database management systems are built around one particular data model, although it is possible for products to offer support for more than one model.

Various physical data models can implement any given logical model. Most database software will offer the user some level of control in tuning the physical implementation, since the choices that are made have a significant effect on performance.

A model is not just a way of structuring data: it also defines a set of operations that can be performed on the data. The relational model, for example, defines operations such as select (project) and join. Although these operations may not be explicit in a particular query language, they provide the foundation on which a query language is built.

Flat model

The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. For instance, columns for name and password that might be used as a part of a system security database. Each row would have the specific password associated with an individual user. Columns of the table often have a type associated with them, defining them as character data, date or time information, integers, or floating point numbers. This tabular format is a precursor to the relational model.

Flat File Model

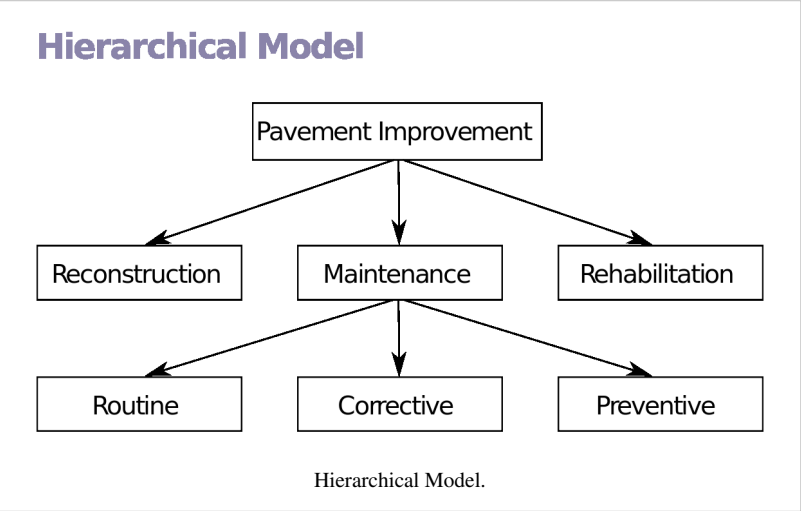
	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

Flat File Model.

Early data models

These models were popular in the 1960s, 1970s, but nowadays can be found primarily in old legacy systems. They are characterized primarily by being navigational with strong connections between their logical and physical representations, and deficiencies in data independence.

Hierarchical model

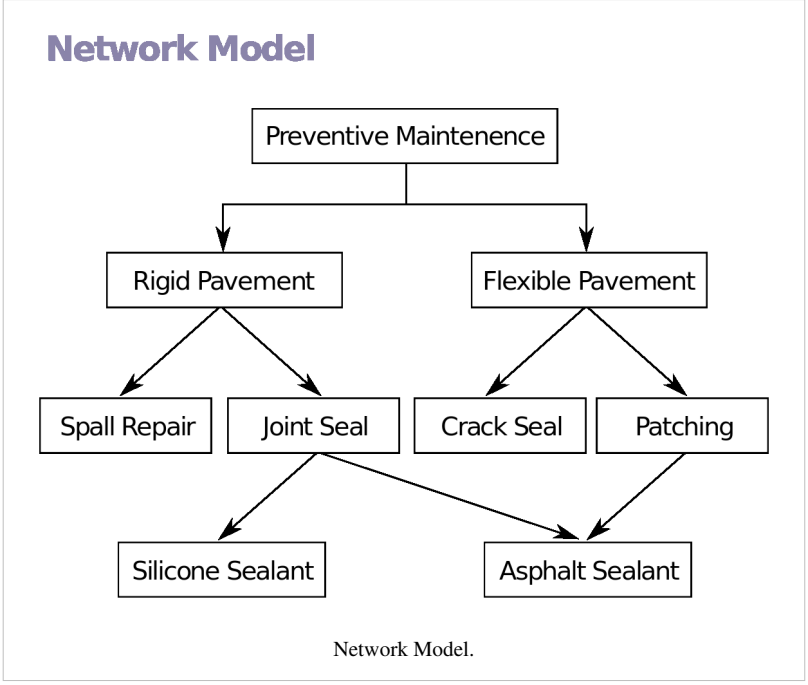


Hierarchical Model.

In a hierarchical model, data is organized into a tree-like structure, implying a single parent for each record. A sort field keeps sibling records in a particular order. Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System (IMS) by IBM, and now describe the structure of XML documents. This structure allows one one-to-many relationship between two types of data. This structure is very efficient to describe many relationships in the real world; recipes, table of contents, ordering of paragraphs/verses, any nested and sorted information.

This hierarchy is used as the physical order of records in storage. Record access is done by navigating through the data structure using pointers combined with sequential accessing. Because of this, the hierarchical structure is inefficient for certain database operations when a full path (as opposed to upward link and sort field) is not also included for each record. Such limitations have been compensated for in later IMS versions by additional logical hierarchies imposed on the base physical hierarchy.

Network model



Network Model.

The network model expands upon the hierarchical structure, allowing many-to-many relationships in a tree-like structure that allows multiple parents. It was the most popular before being replaced by the relational model, and is defined by the CODASYL specification.

The network model organizes data using two fundamental concepts, called *records* and *sets*. Records contain fields (which may be organized hierarchically, as in the programming language COBOL). Sets (not to be confused with mathematical sets) define one-to-many[1] relationships between records: one owner, many members. A record may be an owner in any number of sets, and a member in any number of sets.

A set consists of circular linked lists where one record type, the set owner or parent, appears once in each circle, and a second record type, the subordinate or child, may appear multiple times in each circle. In this way a hierarchy may be established between any two record types, e.g., type A is the owner of B. At the same time another set may be defined where B is the owner of A. Thus all the sets comprise a general directed graph (ownership defines a direction), or *network* construct. Access to records is either sequential (usually in each record type) or by navigation in the circular linked lists.

The network model is able to represent redundancy in data more efficiently than in the hierarchical model, and there can be more than one path from an ancestor node to a descendant. The operations of the network model are navigational in style: a program maintains a current position, and navigates from one record to another by following the relationships in which the record participates. Records can also be located by supplying key values.

Although it is not an essential feature of the model, network databases generally implement the set relationships by means of pointers that directly address the location of a record on disk. This gives excellent retrieval performance, at the expense of operations such as database loading and reorganization.

Popular DBMS products that utilized it were Cincom Systems' Total and Cullinet's IDMS. IDMS gained a considerable customer base; in the 1980s, it adopted the relational model and SQL in addition to its original tools and languages.

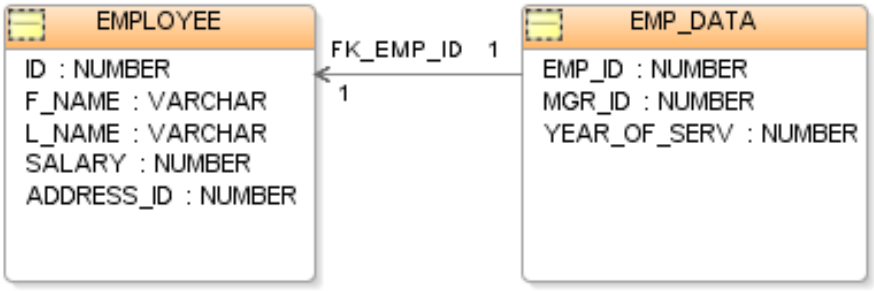
Most object databases (invented in the 1990s) use the navigational concept to provide fast navigation across networks of objects, generally using object identifiers as "smart" pointers to related objects. Objectivity/DB, for instance, implements named one-to-one, one-to-many, many-to-one, and many-to-many named relationships that can cross databases. Many object databases also support SQL, combining the strengths of both models.

Inverted file model

In an *inverted file* or *inverted index*, the contents of the data are used as keys in a lookup table, and the values in the table are pointers to the location of each instance of a given content item. This is also the logical structure of contemporary database indexes, which might only use the contents from a particular columns in the lookup table. The *inverted file data model* can put indexes in a second set of files next to existing flat database files, in order to efficiently directly access needed records in these files.

Notable for using this data model is the ADABAS DBMS of Software AG, introduced in 1970. ADABAS has gained considerable customer base and exists and supported until today. In the 1980s it has adopted the relational model and SQL in addition to its original tools and languages.

Relational model



The relational model was introduced by E.F. Codd in 1970[2] as a way to make database management systems more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory, and systems implementing it have been used by mainframe, midrange and microcomputer systems.

The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. Three key terms are used extensively in relational database models: *relations*, *attributes*, and *domains*. A relation is a table with columns and rows. The named columns of the relation are called attributes, and the domain is the set of values the attributes are allowed to take.

The basic data structure of the relational model is the table, where information about a particular entity (say, an employee) is represented in rows (also called tuples) and columns. Thus, the "relation" in "relational database" refers to the various tables in the database; a relation is a set of tuples. The columns enumerate the various attributes of the entity (the employee's name, address or phone number, for example), and a row is an actual instance of the entity (a specific employee) that is represented by the relation. As a result, each tuple of the employee table represents various attributes of a single employee.

All relations (and, thus, tables) in a relational database have to adhere to some basic rules to qualify as relations. First, the ordering of columns is immaterial in a table. Second, there can't be identical tuples or rows in a table. And third, each tuple will contain a single value for each of its attributes.

A relational database contains multiple tables, each similar to the one in the "flat" database model. One of the strengths of the relational model is that, in principle, any value occurring in two different records (belonging to the same table or to different tables), implies a relationship among those two records. Yet, in order to enforce explicit integrity constraints, relationships between records in tables can also be defined explicitly, by identifying or non-identifying parent-child relationships characterized by assigning cardinality (1:1, (0)1:M, M:M). Tables can also have a designated single attribute or a set of attributes that can act as a "key", which can be used to uniquely identify each tuple in the table.

A key that can be used to uniquely identify a row in a table is called a primary key. Keys are commonly used to join or combine data from two or more tables. For example, an *Employee* table may contain a column named *Location* which contains a value that matches the key of a *Location* table. Keys are also critical in the creation of indexes, which facilitate fast retrieval of data from large tables. Any column can be a key, or multiple columns can be

grouped together into a compound key. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one.

A key that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number) is sometimes called a "natural" key. If no natural key is suitable (think of the many people named *Brown*), an arbitrary or surrogate key can be assigned (such as by giving employees ID numbers). In practice, most databases have both generated and natural keys, because generated keys can be used internally to create links between rows that cannot break, while natural keys can be used, less reliably, for searches and for integration with other databases. (For example, records in two independently developed databases could be matched up by social security number, except when the social security numbers are incorrect, missing, or have changed.)

The most common query language used with the relational model is the Structured Query Language (SQL).

Dimensional model

The dimensional model is a specialized adaptation of the relational model used to represent data in data warehouses in a way that data can be easily summarized using online analytical processing, or OLAP queries. In the dimensional model, a database schema consists of a single large table of facts that are described using dimensions and measures. A dimension provides the context of a fact (such as who participated, when and where it happened, and its type) and is used in queries to group related facts together. Dimensions tend to be discrete and are often hierarchical; for example, the location might include the building, state, and country. A measure is a quantity describing the fact, such as revenue. It is important that measures can be meaningfully aggregated—for example, the revenue from different locations can be added together.

In an OLAP query, dimensions are chosen and the facts are grouped and aggregated together to create a summary.

The dimensional model is often implemented on top of the relational model using a star schema, consisting of one highly normalized table containing the facts, and surrounding denormalized tables containing each dimension. An alternative physical implementation, called a snowflake schema, normalizes multi-level hierarchies within a dimension into multiple tables.

A data warehouse can contain multiple dimensional schemas that share dimension tables, allowing them to be used together. Coming up with a standard set of dimensions is an important part of dimensional modeling.

Its high performance has made the dimensional model the most popular database structure for OLAP.

Post-relational database models

Products offering a more general data model than the relational model are sometimes classified as *post-relational*.[3] Alternate terms include "hybrid database", "Object-enhanced RDBMS" and others. The data model in such products incorporates relations but is not constrained by E.F. Codd's Information Principle, which requires that

> all information in the database must be cast explicitly in terms of values in relations and in no other way

Some of these extensions to the relational model integrate concepts from technologies that pre-date the relational model. For example, they allow representation of a directed graph with trees on the nodes. The German company *sones* implements this concept in its GraphDB.

Some post-relational products extend relational systems with non-relational features. Others arrived in much the same place by adding relational features to pre-relational systems. Paradoxically, this allows products that are historically pre-relational, such as PICK and MUMPS, to make a plausible claim to be post-relational.

The resource space model (RSM) is a non-relational data model based on multi-dimensional classification.

Graph model

Graph databases allow even more general structure than a network database; any node may be connected to any other node.

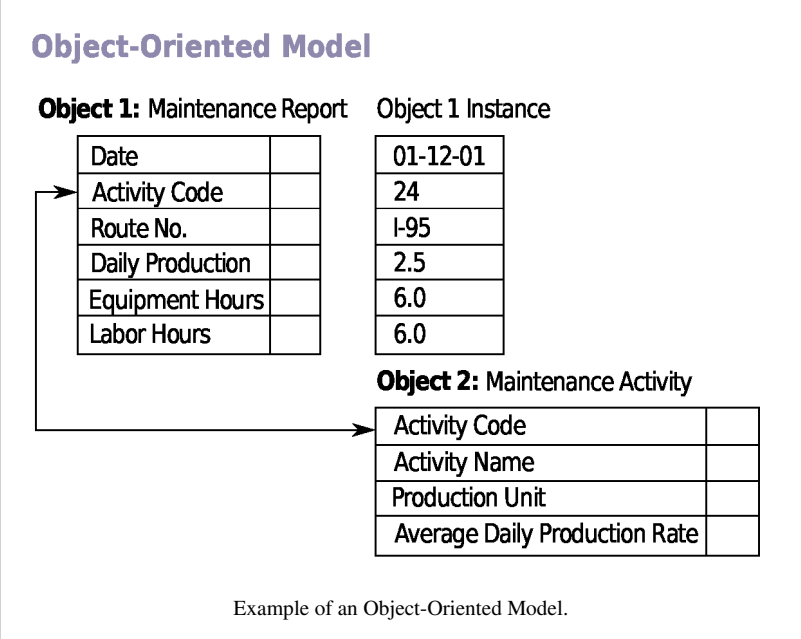
Multivalue model

Multivalue databases are "lumpy" data, in that they can store exactly the same way as relational databases, but they also permit a level of depth which the relational model can only approximate using sub-tables. This is nearly identical to the way XML expresses data, where a given field/attribute can have multiple right answers at the same time. Multivalue can be thought of as a compressed form of XML.

An example is an invoice, which in either multivalue or relational data could be seen as (A) Invoice Header Table - one entry per invoice, and (B) Invoice Detail Table - one entry per line item. In the multivalue model, we have the option of storing the data as on table, with an embedded table to represent the detail: (A) Invoice Table - one entry per invoice, no other tables needed.

The advantage is that the atomicity of the Invoice (conceptual) and the Invoice (data representation) are one-to-one. This also results in fewer reads, less referential integrity issues, and a dramatic decrease in the hardware needed to support a given transaction volume.

Object-oriented database models



Example of an Object-Oriented Model.

In the 1990s, the object-oriented programming paradigm was applied to database technology, creating a new database model known as object databases. This aims to avoid the object-relational impedance mismatch - the overhead of converting information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). Even further, the type system used in a particular application can be defined directly in the database, allowing the database to enforce the same data integrity invariants. Object databases also introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.

A variety of these ways have been tried Wikipedia:Manual of Style/Words to watch#Unsupported attributionsfor storing objects in a database. SomeWikipedia:Avoid weasel words products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. OthersWikipedia:Avoid weasel words have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

Object databases suffered because of a lack of standardization: although standards were defined by ODMG, they were never implemented well enough to ensure interoperability between products. Nevertheless, object databases

have been used successfully in many applications: usually specialized applications such as engineering databases or molecular biology databases rather than mainstream commercial data processing. However, object database ideas were picked up by the relational vendors and influenced extensions made to these products and indeed to the SQL language.

An alternative to translating between objects and relational databases is to use an object-relational mapping (ORM) library.

References

[1] http://toolserver.org/%7Edispenser/cgi-bin/dab_solver.py?page=Database_model&editintro=Template:Disambiguation_needed/editintro&client=Template:Dn

[2] E.F. Codd (1970). "A relational model of data for large shared data banks". In: *Communications of the ACM archive*. Vol 13. Issue 6(June 1970). pp.377-387.

[3] *Introducing databases* by Stephen Chu, in Conrick, M. (2006) *Health informatics: transforming healthcare with technology*, Thomson, ISBN 0-17-012731-1, p. 69.

Database design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).[1]

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

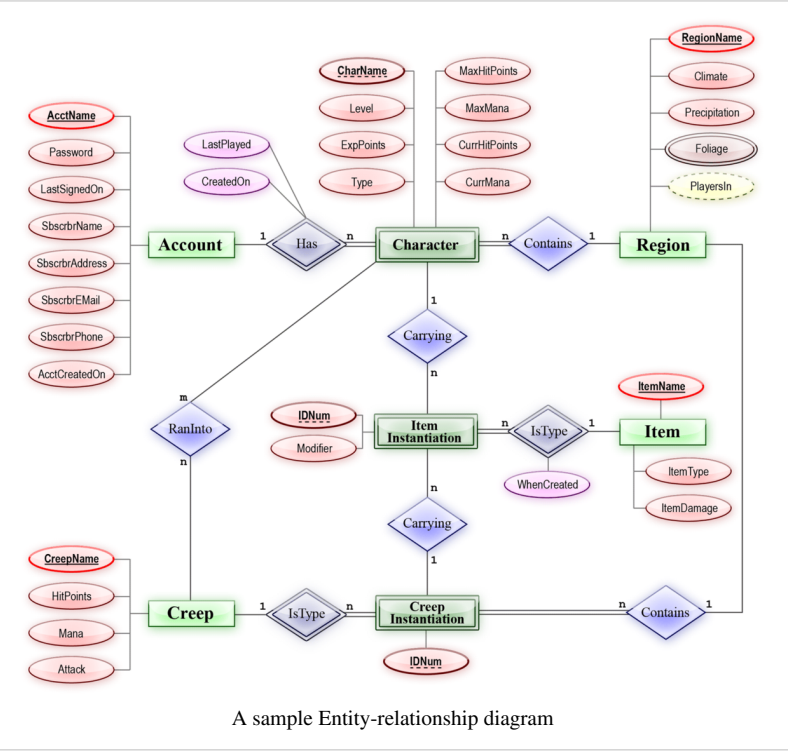
- Determine the relationships between the different data elements.
- Superimpose a logical structure upon the data on the basis of these relationships.[2]

ER diagram (entity-relationship model)

Database designs also include ER (Entity-relationship model) diagrams. An ER diagram is a diagram that helps to design databases in an efficient way.

Attributes in ER diagrams are usually modeled as an oval with the name of the attribute, linked to the entity or relationship that contains the attribute.

Within the relational model the final step can generally be broken down into two further steps, that of determining the grouping of information within the system, generally determining what are the basic objects about which information is being stored, and then determining the relationships between these groups of information, or objects. This step is not necessary with an Object database.



A sample Entity-relationship diagram

Design process[3]

1. **Determine the purpose of the database** - This helps prepare for the remaining steps.
2. **Find and organize the information required** - Gather all of the types of information to record in the database, such as product name and order number.
3. **Divide the information into tables** - Divide information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.
4. **Turn information items into columns** - Decide what information needs to be stored in each table. Each item becomes a field, and is displayed as a column in the table. For example, an Employees table might include fields such as Last Name and Hire Date.
5. **Specify primary keys** - Choose each table's primary key. The primary key is a column, or a set of columns, that is used to uniquely identify each row. An example might be Product ID or Order ID.
6. **Set up the table relationships** - Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.
7. **Refine the design** - Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.
8. **Apply the normalization rules** - Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the tables

Determining data to be stored

In a majority of cases, a person who is doing the design of a database is a person with expertise in the area of database design, rather than expertise in the domain from which the data to be stored is drawn e.g. financial information, biological information etc. Therefore the data to be stored in the database must be determined in cooperation with a person who does have expertise in that domain, and who is aware of what data must be stored within the system.

This process is one which is generally considered part of requirements analysis, and requires skill on the part of the database designer to elicit the needed information from those with the domain knowledge. This is because those with the necessary domain knowledge frequently cannot express clearly what their system requirements for the database are as they are unaccustomed to thinking in terms of the discrete data elements which must be stored. Data to be stored can be determined by Requirement Specification.[4]

Normalization

In the field of relational database design, **normalization** is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics—insertion, update, and deletion anomalies—that could lead to a loss of data integrity.

A standard piece of database design guidance is that the designer should create a fully normalized design; selective denormalization can subsequently be performed, but only for performance reasons. However, some modeling disciplines, such as the dimensional modeling approach to data warehouse design, explicitly recommend non-normalized designs, i.e. designs that in large part do not adhere to 3NF. Normalization consists of normal forms that are 1NF,2NF,3NF,BOYCE-CODD NF (3.5NF),4NF and 5NF

Types of Database design

Conceptual schema

Once a database designer is aware of the data which is to be stored within the database, they must then determine where dependency is within the data. Sometimes when data is changed you can be changing other data that is not visible. For example, in a list of names and addresses, assuming a situation where multiple people can have the same address, but one person cannot have more than one address, the address is dependent upon the name. When provided a name and the list the address can be uniquely determined; however, the inverse does not hold - when given an address and the list, a name cannot be uniquely determined because multiple people can reside at an address. Because an address is determined by a name, an address is considered dependent on a name.

(NOTE: A common misconception is that the relational model is so called because of the stating of relationships between data elements therein. This is not true. The relational model is so named because it is based upon the mathematical structures known as relations.)

Logically structuring data

Once the relationships and dependencies amongst the various pieces of information have been determined, it is possible to arrange the data into a logical structure which can then be mapped into the storage objects supported by the database management system. In the case of relational databases the storage objects are tables which store data in rows and columns.

Each table may represent an implementation of either a logical object or a relationship joining one or more instances of one or more logical objects. Relationships between tables may then be stored as links connecting child tables with parents. Since complex logical relationships are themselves tables they will probably have links to more than one parent.

In an Object database the storage objects correspond directly to the objects used by the Object-oriented programming language used to write the applications that will manage and access the data. The relationships may be defined as attributes of the object classes involved or as methods that operate on the object classes.

Schema Refinement

The Schema Refinement of the database specifies that how the data is Normalized and reduce the Data Insufficiency and Conflicts also.

Physical design

The physical design of the database specifies the physical configuration of the database on the storage media. This includes detailed specification of data elements, data types, indexing options and other parameters residing in the DBMS data dictionary. It is the detailed design of a system that includes modules & the database's hardware & software specifications of the system.

References

[1] Gehani, N. (2006). The Database Book: Principles and practice using MySQL. 1st ed., Summit, NJ.: Silicon Press

[2] Teorey, T.J., Lightstone, S.S., et al., (2009). Database Design: Know it all.1st ed. Burlington, MA.: Morgan Kaufmann Publishers

[3] Database design basics. (n.d.). Database design basics. Retrieved May 1, 2010, from http://office.microsoft.com/en-us/access/HA012242471033.aspx

[4] Teorey, T.; Lightstone, S. and Nadeau, T.(2005) *Database Modeling & Design: Logical Design*, 4th edition, Morgan Kaufmann Press. ISBN 0-12-685352-5

Further reading

- S. Lightstone, T. Teorey, T. Nadeau, “Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more”, Morgan Kaufmann Press, 2007. ISBN 0-12-369389-6
- M. Hernandez, " Database Design for Mere Mortals (http://www.informit.com/store/database-design-for-mere-mortals-a-hands-on-guide-to-9780321884497): A Hands-On Guide to Relational Database Design", 3rd Edition, Addison-Wesley Professional, 2013. ISBN 0-321-88449-3

External links

- (http://www.sqlteam.com/article/database-design-and-modeling-fundamentals)
- (http://office.microsoft.com/en-us/access/HA012242471033.aspx)
- Database Normalization Basics (http://databases.about.com/od/specificproducts/a/normalization.htm) by Mike Chapple (About.com)
- Database Normalization Intro (http://www.databasejournal.com/sqletc/article.php/1428511), Part 2 (http://www.databasejournal.com/sqletc/article.php/26861_1474411_1)
- "An Introduction to Database Normalization" (http://web.archive.org/web/20110606025027/http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html). Archived from the original (http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html) on 2011-06-06. Retrieved 2012-02-25.
- "Normalization" (http://web.archive.org/web/20100106115112/http://www.utexas.edu/its/archive/windows/database/datamodeling/rm/rm7.html). Archived from the original (http://www.utexas.edu/its/windows/database/datamodeling/rm/rm7.html) on 2010-01-06. Retrieved 2012-02-25.
- Efficient Database Design (http://www.sum-it.nl/cursus/enindex.php3#dbdesign)
- Relational database design tutorial (http://en.tekstenuitleg.net/articles/software/database-design-tutorial/intro.html)
- Database design (http://www.dmoz.org/Computers/Data_Formats/Database) on the Open Directory Project

Conceptual schema

This is a high-level description of a business informational needs. It typically includes only the main concepts and the main relationship among them. Typically this is a first-cut model, with insufficient detail to build an actual database.[1]

Overview

A **conceptual schema** or **conceptual data model** is a map of concepts and their relationships used for databases. This describes the semantics of an organization and represents a series of assertions about its nature. Specifically, it describes the things of significance to an organization (*entity classes*), about which it is inclined to collect information, and characteristics of (*attributes*) and associations between pairs of those things of significance (*relationships*).

Because a conceptual schema represents the semantics of an organization, and not a database design, it may exist on various levels of abstraction. The original ANSI four-schema architecture began with the set of *external schemas* that each represent one person's view of the world around him or her. These are consolidated into a single *conceptual schema* that is the superset of all of those external views. A data model can be as concrete as each person's perspective, but this tends to make it inflexible. If that person's world changes, the model must change. Conceptual data models take a more abstract perspective, identifying the fundamental things, of which the things an individual deals with are just examples.

The model does allow for what is called inheritance in object oriented terms. The set of instances of an entity class may be subdivided into entity classes in their own right. Thus, each instance of a *sub-type* entity class is also an instance of the entity class's *super-type*. Each instance of the super-type entity class, then is also an instance of one of the sub-type entity classes.

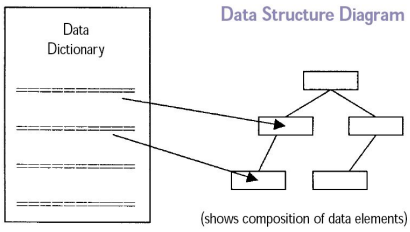
Super-type/sub-type relationships may be *exclusive* or not. A methodology may require that each instance of a super-type may *only* be an instance of *one* sub-type. Similarly, a super-type/sub-type relationship may be *exhaustive* or not. It is exhaustive if the methodology requires that each instance of a super-type *must be* an instance of a sub-type.

Example relationships

- Each PERSON may be *the vendor in* one or more ORDERS..
- Each ORDER must be *from* one and only one PERSON.
- PERSON is *a sub-type of* PARTY. (Meaning that every instance of PERSON is also an instance of PARTY.)
- Each EMPLOYEE may have a *supervisor* who is also an EMPLOYEE.

Data structure diagram

A data structure diagram (DSD) is a data model or diagram used to describe conceptual data models by providing graphical notations which document entities and their relationships, and the constraints that bind them.



Data Structure Diagram.

References

[1] "Data Modelling Tutorial", Stefano Grazioli, Univeristy of Virginia, http://webs.comm.virginia.edu/Grazioli/MSMITMod1/DataModelingTutorial/DMTutorialHome.html, retrieved 20 February 2014

Further reading

- Perez, Sandra K., & Anthony K. Sarris, eds. (1995) Technical Report for IRDS Conceptual Schema, Part 1: Conceptual Schema for IRDS, Part 2: Modeling Language Analysis, X3/TR-14:1995, American National Standards Institute, New York, NY.

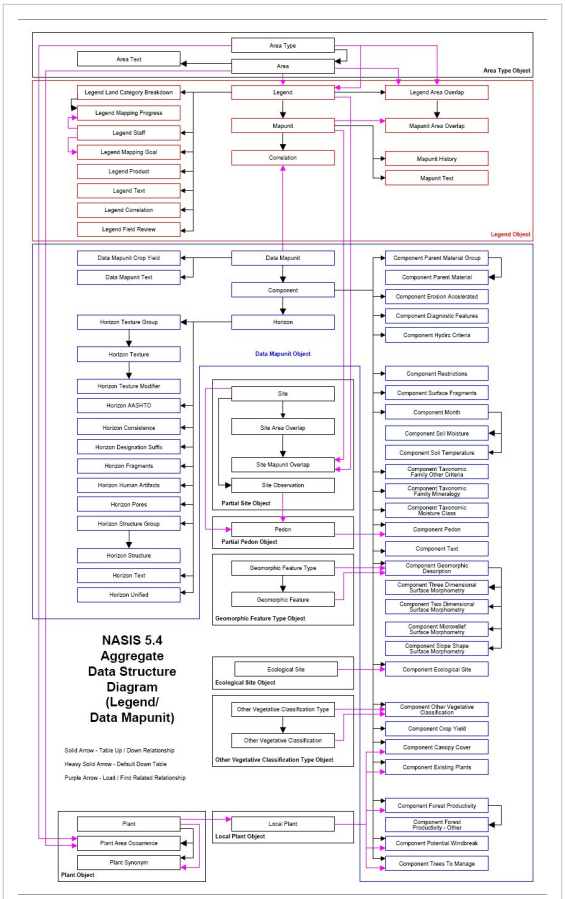
External links

- A different point of view (http://www.agiledata.org/essays/dataModeling101.html), as described by the "agile" community

Data structure diagram

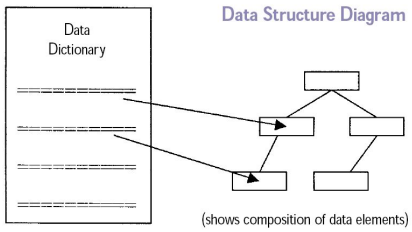
Data Structure Diagram (DSD) is a diagram of the conceptual data model which documents the entities and their relationships, as well as the constraints that connect to them.

The basic graphic notation elements of DSDs are boxes which represent entities.The arrow symbol represents relationships. Data structure diagrams are most useful for documenting complex data entities.



Example of a Data Structure Diagram.

Overview



Data Structure Diagram.

Data Structure Diagram is a diagram type that is used to depict the structure of data elements in the data dictionary. The data structure diagram is a graphical alternative to the composition specifications within such data dictionary entries.[1]

The data structure diagrams is a predecessor of the entity-relationship model (E-R model). In DSDs, attributes are specified inside the entity boxes rather than outside of them, while relationships are drawn as boxes composed of attributes which specify the constraints that bind entities together. DSDs differ from the E-R model in that the E-R model focuses on the relationships between different entities, whereas DSDs focus on the relationships of the elements within an entity.

There are several styles for representing data structure diagrams, with the notable difference in the manner of defining cardinality. The choices are between arrow heads, inverted arrow heads (crow's feet), or numerical representation of the cardinality.

Bachman diagram

A Bachman diagram is a certain type of data structure diagram,[2] and is used to design the data with a network or relational "logical" model, separating the data model from the way the data is stored in the system. The model is named after database pioneer Charles Bachman, and mostly used in computer software design.

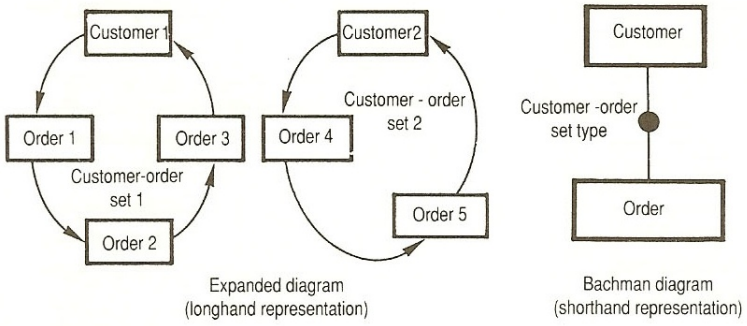


Illustration of set type using a Bachman diagram.

In a relational model, a relation is the cohesion of attributes that are fully and not transitive functional dependent[*clarify*] of every key in that relation. The coupling between the relations is based on accordant attributes. For every relation, a rectangle has to be drawn and every coupling is illustrated by a line that connects the relations. On the edge of each line, arrows indicate the cardinality. We have 1-to-n, 1-to-1 and n-to-n. The latter has to be avoided and must be replaced by two 1-to-n couplings.

References

[1] Data Integration Glossary (http://knowledge.fhwa.dot.gov/tam/aashto.nsf/All+Documents/4825476B2B5C687285256B1F00544258/$FILE/DIGloss.pdf), U.S. Department of Transportation, August 2001.

[2] IRS Resources (http://www.irs.gov/irm/part2/irm_02-005-013.html). Part 2. Information Technology, Chapter 5. Systems Development, Section 13. Database Design Techniques and Deliverables. Retrieved 2 July 2009.

Further reading

- Charles W. Bachman. *Data structure diagrams*. Data Base, 1969, 1(2):4–10.
- Tom DeMarco. *Structured Analysis and System Specification*. ISBN 0-13-854380-1. Prentice Hall. 11 May 1979.
- Edward Yourdon. *Modern Structured Analysis*. ISBN 0-13-598624-9. Prentice Hall. 1 August 1988; now available as the Structured Analysis Wiki (http://www.yourdon.com/strucanalysis/wiki/index.php?title=Introduction).

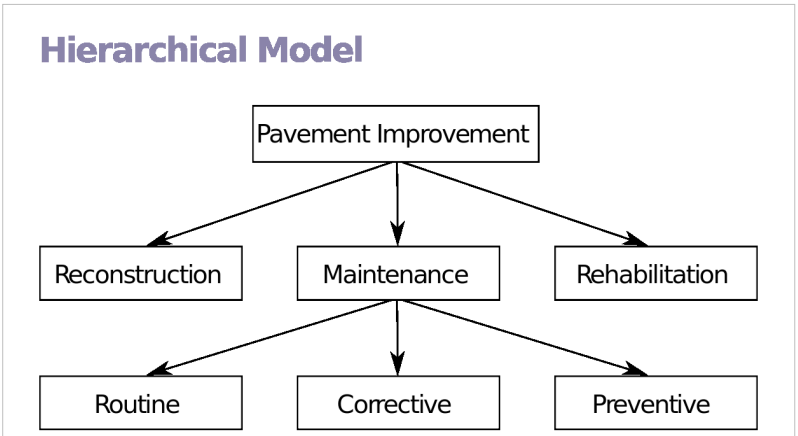
Hierarchical database model

Hierarchical model redirects here. For the statistics usage, see hierarchical linear modeling or hierarchical Bayesian model.

A **hierarchical database model** is a data model in which the data is organized into a tree-like structure. The structure allows representing information using parent/child relationships: each parent can have many children, but each child has only one parent (also known as a **1-to-many relationship**). All attributes of a specific record are listed under an entity type.

In a database an entity type is the equivalent of a table. Each individual record is represented as a row, and each attribute as a column. Entity types are related to each other using *1:N* mappings, also known as one-to-many relationships. This model is recognized as the first database model created by IBM in the 1960s.

Currently the most widely used hierarchical databases are IMS developed by IBM and Windows Registry by Microsoft.



Example of a hierarchical model

History

The hierarchical structure was used in early mainframe DBMS. Records' relationships form a treelike model. This structure is simple but inflexible because the relationship is confined to a one-to-many relationship. The IBM Information Management System (IMS) and the RDM Mobile are examples of a hierarchical database system with multiple hierarchies over the same data. RDM Mobile is a newly designed embedded database for a mobile computer system.[*citation needed*]

The hierarchical data model lost traction as Codd's relational model became the de facto standard used by virtually all mainstream database management systems. A relational-database implementation of a hierarchical model was first discussed in published form in 1992[1] (see also nested set model). Hierarchical data organization schemes resurfaced with the advent of XML in the late 1990s (see also XML database). The hierarchical structure is used primarily today for storing geographic information and file systems.[*citation needed*] Currently the most widely used hierarchical databases are IMS and Windows Registry by Microsoft.[*citation needed*]

Examples of hierarchical data represented as relational tables

An organization could store employee information in a table that contains attributes/columns such as employee number, first name, last name, and Department number. The organization provides each employee with computer hardware as needed, but computer equipment may only be used by the employee to which it is assigned. The organization could store the computer hardware information in a separate table that includes each part's serial number, type, and the employee that uses it. The tables might look like this:

EmpNo	First Name	Last Name	Dept. Num
100	Sally	Baker	10-L
101	Jack	Douglas	10-L
102	Sarah	Schultz	20-B
103	David	Drachmeier	20-B

Serial Num	Type	User EmpNo
3009734-4	Computer	100
3-23-283742	Monitor	100
2-22-723423	Monitor	100
232342	Printer	100

In this model, the employee data table represents the "parent" part of the hierarchy, while the computer table represents the "child" part of the hierarchy. In contrast to tree structures usually found in computer software algorithms, in this model the children point to the parents. As shown, each employee may possess several pieces of computer equipment, but each individual piece of computer equipment may have only one employee owner.

Consider the following structure:

EmpNo	Designation	ReportsTo
10	Director	
20	Senior Manager	10
30	Typist	20
40	Programmer	20

In this, the "child" is the same type as the "parent". The hierarchy stating EmpNo 10 is boss of 20, and 30 and 40 each report to 20 is represented by the "ReportsTo" column. In Relational database terms, the ReportsTo column is a foreign key referencing the EmpNo column. If the "child" data type were different, it would be in a different table, but there would still be a foreign key referencing the EmpNo column of the employees table.

This simple model is commonly known as the adjacency list model, and was introduced by Dr. Edgar F. Codd after initial criticisms surfaced that the relational model could not model hierarchical data.

The Windows Registry is a hierarchical database that stores configuration settings and options on Microsoft Windows operating systems.

References

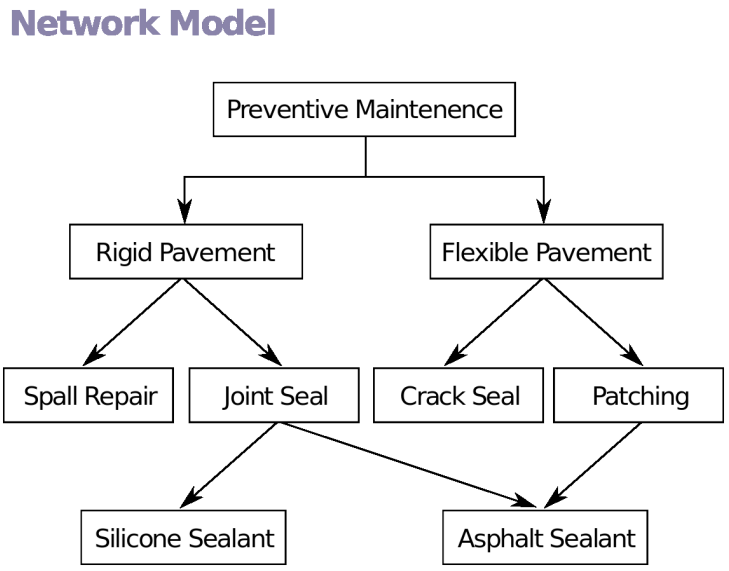
[1] Michael J. Kamfonas/Recursive Hierarchies: The Relational Taboo! (http://www.kamfonas.com/id3.html)--The Relation Journal, October/November 1992

External links

- Troels' links to Hierarchical data in RDBMSs (http://troels.arvin.dk/db/rdbms/links/#hierarchical)
- Managing Hierarchical Data in MySQL (http://web.archive.org/web/20110606032941/http://dev.mysql.com/tech-resources/articles/hierarchical-data.html) (This page is from archive.org as the page has been removed from MySQL.com)
- Hierarchical data in MySQL: parents and children in one query (http://explainextended.com/2009/07/20/hierarchical-data-in-mysql-parents-and-children-in-one-query/)
- Create Hierarchy Chart from Hierarchical Database (http://unifosys.com/hierarchy-chart/Create_Organization_Chart_from_Mutiple_Tables.html)

Network model

The **network model** is a database model conceived as a flexible way of representing objects and their relationships. Its distinguishing feature is that the schema, viewed as a graph in which object types are nodes and relationship types are arcs, is not restricted to being a hierarchy or lattice.



Example of a Network Model.

Overview

While the hierarchical database model structures data as a tree of records, with each record having one parent record and many children, the network model allows each record to have multiple parent and child records, forming a generalized graph structure. This property applies at two levels: the schema is a generalized graph of record types connected by relationship types (called "set types" in CODASYL), and the database itself is a generalized graph of record occurrences connected by relationships (CODASYL "sets"). Cycles are permitted at both levels. The chief argument in favour of the network model, in comparison to the hierarchic model, was that it allowed a more natural modeling of relationships between entities. Although the model was widely implemented and used, it failed to become dominant for two main reasons. Firstly, IBM chose to stick to the hierarchical model with semi-network extensions in their established products such as IMS and DL/I. Secondly, it was eventually displaced by the relational model, which offered a higher-level, more declarative interface. Until the early 1980s the performance benefits of the low-level navigational interfaces offered by hierarchical and network databases were persuasive for many large-scale applications, but as hardware became faster, the extra productivity and flexibility of the relational model led to the gradual obsolescence of the network model in corporate enterprise usage.

Database systems

Some well-known database systems that use the network model include:

- Integrated Data Store (IDS)
- IDMS (Integrated Database Management System)
- RDM Embedded
- RDM Server
- TurboIMAGE
- Univac DMS-1100

History

The network model's original inventor was Charles Bachman, and it was developed into a standard specification published in 1969 by the Conference on Data Systems Languages (CODASYL) Consortium. This was followed by a second publication in 1971, which became the basis for most implementations. Subsequent work continued into the early 1980s, culminating in an ISO specification, but this had little influence on products.

Further reading

- Charles W. Bachman, *The Programmer as Navigator.* ACM Turing Award lecture, Communications of the ACM, Volume 16, Issue 11, 1973, pp. 653–658, ISSN 0001-0782 [1], doi:10.1145/355611.362534 [2]

External links

- CODASYL Systems Committee "Survey of Data Base Systems", 1968 [3] (edited and annotated in 2007 by Ken North)
- Network (CODASYL) Data Model [4]

References

[1] http://www.worldcat.org/search?fq=x0:jrnl&q=n2:0001-0782
[2] http://dx.doi.org/10.1145%2F355611.362534
[3] http://www.sqlsummit.com/PDF/DatabaseSurvey_CODASYL_1968.pdf
[4] http://coronet.iicm.edu/wbtmaster/allcoursescontent/netlib/ndm1.htm

Navigational database

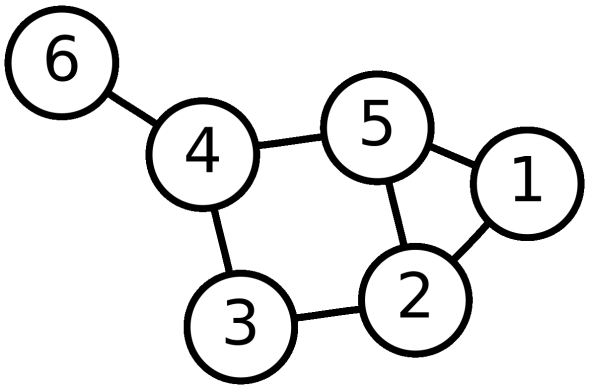
A **navigational database** is a type of database in which records or objects are found primarily by following references from other objects. Navigational interfaces are usually procedural, though some modern systems like XPath can be considered to be simultaneously navigational and declarative.

Navigational access is traditionally associated with the network model and hierarchical model of database interfaces, and some have even acquired set-oriented features. Navigational techniques use "pointers" and "paths" to navigate among data records (also known as "nodes"). This is in contrast to the relational model (implemented in relational databases), which strives to use "declarative" or logic programming techniques that ask the system for *what* to fetch instead of *how* to navigate to it.

For example, to give directions to a house, the navigational approach would resemble something like "Get on highway 25 for 8 miles, turn onto Horse Road, left at the red barn, then stop at the 3rd house down the road", whereas the declarative approach would resemble "Visit the green house(s) within the following coordinates...."

Hierarchical models are also considered navigational because one "goes" up (to parent), down (to leaves), and there are "paths", such as the familiar file/folder paths in hierarchical file systems. In general, navigational systems will use combinations of paths and prepositions such as "next", "previous", "first", "last", "up", "down", "owner", etc.

"Paths" are often formed by concatenation of node names or node addresses. Example:



Sample database nodes: A labeled graph on 6 vertices and 7 edges. (Numbers are used for illustration purposes only. In practice more meaningful names are often used. Other potential attributes are not shown.)

```
Node6.Node4.Node5.Node1
```

Or

```
Node6/Node4/Node5/Node1
```

If there is no link between given nodes, then an error condition is usually triggered with a message such as "Invalid Path". The path "Node6.Node2.Node1" would be invalid in most systems because there is no direct link between Node 6 and Node 2.

The usage of the term "navigational" allegedly is derived from a statement by Charles Bachman in which he describes the "programmer as navigator" while accessing his favored type of database.

Except for hierarchical file systems (which some consider a form of database), navigational techniques fell out of favor by the 1980s. However, object oriented programming and XML have kindled a renewed, but controversial interest in navigational techniques.

Critics of navigational techniques view them as "unstructured spaghetti messes", and liken them to the "goto" of pre-structured programming. In other words, they are allegedly to data organization what goto's were to behavior flow. In this view, relational techniques provide improved discipline and consistency to data organization and usage because of its roots in set theory and predicate calculus.

Some also suggest that navigational database engines are easier to build and take up less memory (RAM) than relational equivalents. However, the existence of relational or relational-based products of the late 1980s that possessed small engines (by today's standards) because they didn't use SQL suggest this is not necessarily the case. Whatever the reason, navigational techniques are still the preferred way to handle smaller-scale structures.

A current example of navigational structuring can be found in the Document Object Model (DOM) often used in web browsers and closely associated with JavaScript. The DOM "engine" is essentially a light-weight navigational database. The World Wide Web itself and Wikipedia could potentially be considered forms of navigational databases, though they focus on human-readable text rather than data (on a large scale, the Web is a network model and on smaller or local scales, such as domain and URL partitioning, it uses hierarchies). In contrast, the Linked Data facet of the Semantic Web is specifically concerned with network-scale machine-readable data, and follows precisely the 'follow your nose' paradigm implied by the navigational idea.

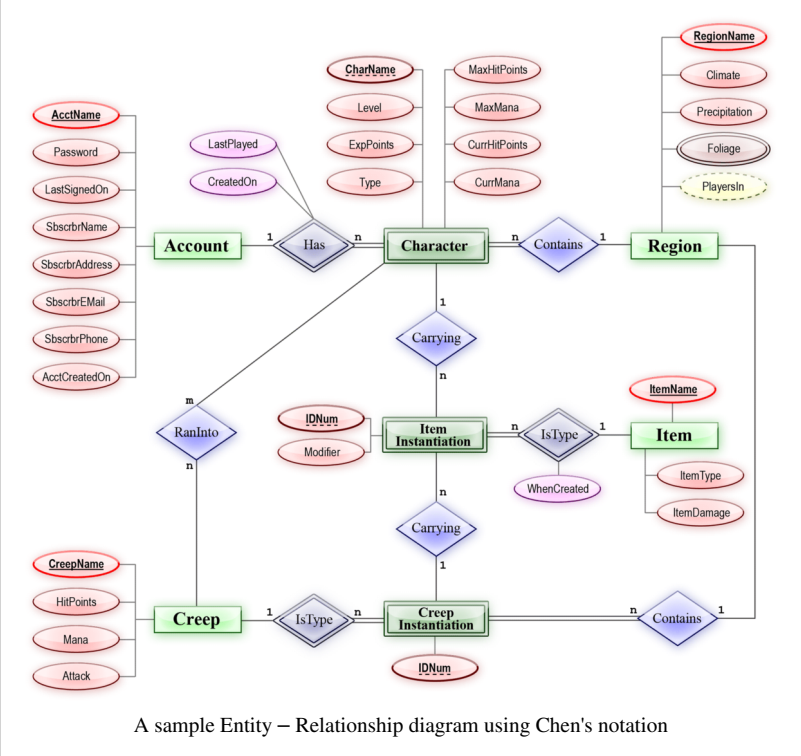
A new kind of navigational databases has recently emerged, the Graph databases. This category of databases is often included as one of the four family of the NoSQL databases.

References

ERD

Entity-relationship model

In software engineering, an **entity–relationship model** (**ER model**) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are *entities* (things) and the *relationships* that can exist among them. This article refers to the techniques proposed in Peter Chen's 1976 paper.[1] However, variants of the idea existed previously,[2] and have been devised subsequently such as supertype and subtype data entities[3] and commonality relationships.



A sample Entity – Relationship diagram using Chen's notation

Overview

An ER model is an abstract way of describing a database. In the case of a relational database, which stores data in tables, some of the data in these tables point to data in other tables - for instance, your entry in the database could point to several entries for each of the phone numbers that are yours. The ER model would say that you are an entity, and each phone number is an entity, and the relationship between you and the phone numbers is 'has a phone number'. Diagrams created to design these entities and relationships are called entity–relationship diagrams or ER diagrams.

Using the three schema approach to software engineering, there are three levels of ER models that may be developed.

Conceptual data model

> This is the highest level ER model in that it contains the least granular detail but establishes the overall scope of what is to be included within the model set. The conceptual ER model normally defines master reference data entities that are commonly used by the organization. Developing an enterprise-wide conceptual ER model is useful to support documenting the data architecture for an organization.
>
> A conceptual ER model may be used as the foundation for one or more *logical data models* (see below). The purpose of the conceptual ER model is then to establish structural metadata commonality for the master data entities between the set of logical ER models. The conceptual data model may be used to form commonality relationships between ER models as a basis for data model integration.

Logical data model

A logical ER model does not require a conceptual ER model, especially if the scope of the logical ER model includes only the development of a distinct information system. The logical ER model contains more detail than the conceptual ER model. In addition to master data entities, operational and transactional data entities are now defined. The details of each data entity are developed and the entity relationships between these data entities are established. The logical ER model is however developed independent of technology into which it will be implemented.

Physical data model

One or more physical ER models may be developed from each logical ER model. The physical ER model is normally developed to be instantiated as a database. Therefore, each physical ER model must contain enough detail to produce a database and each physical ER model is technology dependent since each database management system is somewhat different.

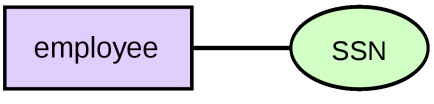
The physical model is normally forward engineered to instantiate the structural metadata into a database management system as relational database objects such as database tables, database indexes such as unique key indexes, and database constraints such as a foreign key constraint or a commonality constraint. The ER model is also normally used to design modifications to the relational database objects and to maintain the structural metadata of the database.

The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database. The data modeling technique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain area of interest. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model; this in turn is mapped to a physical model during physical design. Note that sometimes, both of these phases are referred to as "physical design". It is also used in database management system.

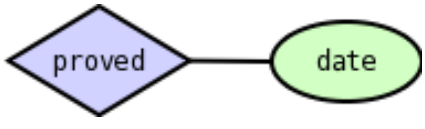
Entity–relationship modelling

An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. An entity is an abstraction from the complexities of a domain. When we speak of an entity, we normally speak of some aspect of the real world which can be distinguished from other aspects of the real world.[4]

Two related entities



An entity with an attribute



A relationship with an attribute

An entity may be a physical object such as a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order. Although the term entity is the one most commonly used, following Chen we should really distinguish between an entity and an entity-type. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type. Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term.

Entities can be thought of as nouns. Examples: a computer, an employee, a song, a mathematical theorem.

A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: an *owns* relationship between a company and a computer, a *supervises* relationship between an employee and a department, a *performs* relationship between an artist and a song, a *proved* relationship between a mathematician and a theorem.

Primary key

The model's linguistic aspect described above is utilized in the declarative database query language ERROL, which mimics natural language constructs. ERROL's semantics and implementation are based on reshaped relational algebra (RRA), a relational algebra which is adapted to the entity–relationship model and captures its linguistic aspect.

Entities and relationships can both have attributes. Examples: an *employee* entity might have a *Social Security Number* (SSN) attribute; the *proved* relationship may have a *date* attribute.

Every entity (unless it is a weak entity) must have a minimal set of uniquely identifying attributes, which is called the entity's primary key.

Entity–relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets and relationship sets. Example: a particular *song* is an entity. The collection of all songs in a database is an entity set. The *eaten* relationship between a child and her lunch is a single relationship. The set of all such child-lunch relationships in a database is a relationship set. In other words, a relationship set corresponds to a relation in mathematics, while a relationship corresponds to a member of the relation.

Certain cardinality constraints on relationship sets may be indicated as well.

Mapping natural language

Chen proposed the following "rules of thumb" for mapping natural language descriptions into ER diagrams:[5]

English grammar structure	ER structure
Common noun	Entity type
Proper noun	Entity
Transitive verb	Relationship type
Intransitive verb	Attribute type
Adjective	Attribute for entity
Adverb	Attribute for relationship

Physical view show how data is actually stored.

Relationships, roles and cardinalities

In Chen's original paper he gives an example of a relationship and its roles. He describes a relationship "marriage" and its two roles "husband" and "wife".

A person plays the role of husband in a marriage (relationship) and another person plays the role of wife in the (same) marriage. These words are nouns. That is no surprise; naming things requires a noun.

However as is quite usual with new ideas, many eagerly appropriated the new terminology but then applied it to their own old ideas. Thus the lines, arrows and crows-feet of their diagrams owed more to the earlier Bachman diagrams than to Chen's relationship diamonds. And they similarly misunderstood other important concepts.[*citation needed*]

In particular, it became fashionable (now almost to the point of exclusivity) to "name" relationships and roles as verbs or phrases.

Role naming

It has also become prevalent to name roles with phrases e.g. is-the-owner-of and is-owned-by etc. Correct nouns in this case are "owner" and "possession". Thus "person plays the role of owner" and "car plays the role of possession" rather than "person plays the role of is-the-owner-of" etc.

The use of nouns has direct benefit when generating physical implementations from semantic models. When a person has two relationships with car then it is possible to very simply generate names such as "owner_person" and "driver_person" which are immediately meaningful.[6]

Cardinalities

Modifications to the original specification can be beneficial. Chen described look-across cardinalities. As an aside, the Barker-Ellis notation, used in Oracle Designer, uses same-side for minimum cardinality (analogous to optionality) and role, but look-across for maximum cardinality (the crows foot).Wikipedia:Please clarify

In Merise,[7] Elmasri & Navathe[8] and others[9] there is a preference for same-side for roles and both minimum and maximum cardinalities. Recent researchers (Feinerer,[10] Dullea et al.[11]) have shown that this is more coherent when applied to n-ary relationships of order > 2.

In Dullea et al. one reads "A 'look across' notation such as used in the UML does not effectively represent the semantics of participation constraints imposed on relationships where the degree is higher than binary."

In Feinerer it says "Problems arise if we operate under the look-across semantics as used for UML associations. Hartmann[12] investigates this situation and shows how and why different transformations fail." *(Although the "reduction" mentioned is spurious as the two diagrams 3.4 and 3.5 are in fact the same)* and also "As we will see on the next few pages, the look-across interpretation introduces several difficulties which prevent the extension of simple mechanisms from binary to n-ary associations."

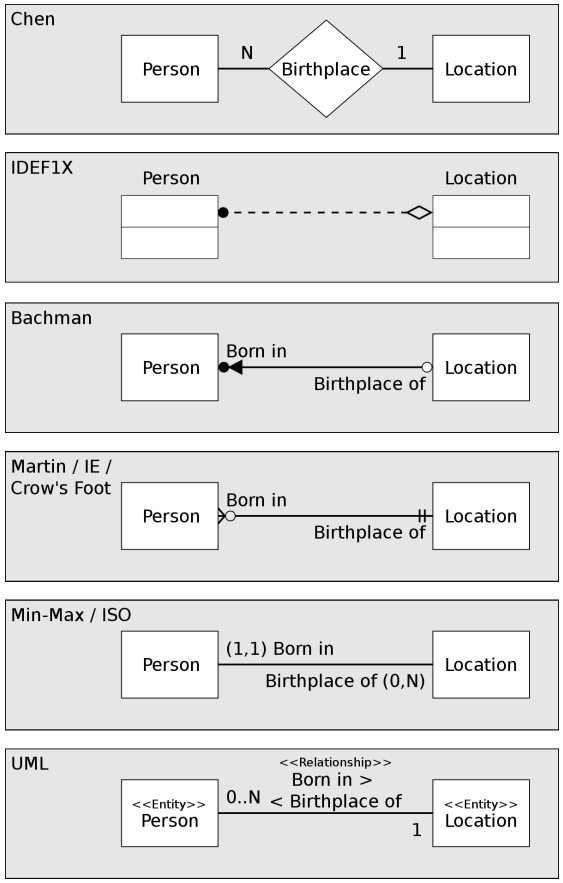
Chen's notation for entity–relationship modeling uses rectangles to represent entity sets, and diamonds to represent relationships appropriate for first-class objects: they can have attributes and relationships of their own. If an entity set participates in a relationship set, they are connected with a line.

Attributes are drawn as ovals and are connected with a line to exactly one entity or relationship set.

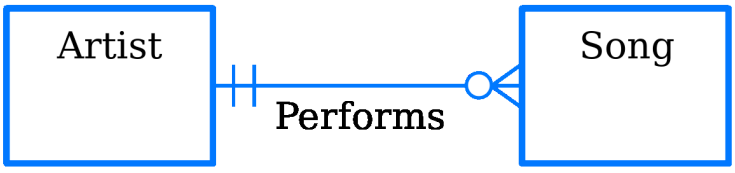
Cardinality constraints are expressed as follows:

- a double line indicates a *participation constraint*, totality or surjectivity: all entities in the entity set must participate in *at least one* relationship in the relationship set;
- an arrow from entity set to relationship set indicates a key constraint, i.e. injectivity: each entity of the entity set can participate in *at most one* relationship in the relationship set;
- a thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in *exactly one* relationship.
- an underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for this attribute.

Attributes are often omitted as they can clutter up a diagram; other diagram techniques often list entity attributes within the rectangles drawn for entity sets.



Various methods of representing the same one to many relationship. In each case, the diagram shows the relationship between a person and a place of birth: each person must have been born at one, and only one, location, but each location may have had zero or more people born at it.



Two related entities shown using Crow's Foot notation. In this example, an optional relationship is shown between Artist and Song; the symbols closest to the song entity represents "zero, one, or many", whereas a song has "one and only one" Artist. The former is therefore read as, an Artist (can) perform(s) "zero, one, or many" song(s).

Related diagramming convention techniques:

- Bachman notation
- Barker's Notation
- EXPRESS
- IDEF1X[13]
- Martin notation
- (min, max)-notation of Jean-Raymond Abrial in 1974

- UML class diagrams
- Merise
- Object-Role Modeling

Crow's Foot Notation

Crow's Foot notation is used in Barker's Notation, SSADM and Information Engineering. Crow's Foot diagrams represent entities as boxes, and relationships as lines between the boxes. Different shapes at the ends of these lines represent the cardinality of the relationship.

Crow's Foot notation was used in the consultancy practice CACI. Many of the consultants at CACI (including Richard Barker) subsequently moved to Oracle UK, where they developed the early versions of Oracle's CASE tools, introducing the notation to a wider audience. The following tools use Crow's Foot notation: ARIS, System Architect, Visio, PowerDesigner, Toad Data Modeler, DeZign for Databases, Devgems Data Modeler, OmniGraffle, MySQL Workbench and SQL Developer Data Modeler. CA's ICASE tool, CA Gen aka Information Engineering Facility also uses this notation. Historically XA Systems Silverrun-LDM (logical data model) also supported this notation.

ER diagramming tools

There are many ER diagramming tools. Free software ER diagramming tools that can interpret and generate ER models and SQL and do database analysis are MySQL Workbench (formerly DBDesigner), and Open ModelSphere (open-source). A freeware ER tool that can generate database and application layer code (webservices) is the RISE Editor. SQL Power Architect while proprietary also has a free community edition.

Proprietary ER diagramming tools are Avolution, ER/Studio, ERwin, DeZign for Databases, MagicDraw, MEGA International, ModelRight, Navicat Data Modeler, OmniGraffle, Oracle Designer, PowerDesigner, Prosa Structured Analysis Tool, Rational Rose, Software Ideas Modeler, Sparx Enterprise Architect, SQLyog, System Architect, Toad Data Modeler, and Visual Paradigm.

Free software diagram tools just draw the shapes without having any knowledge of what they mean, nor do they generate SQL. These include Creately, yEd, LucidChart, Calligra Flow, and Dia.

ER and semantic modelling

Peter Chen, the father of ER modelling said in his seminal paper:

> "*The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. It incorporates some of the important semantic information about the real world.*"

He is here in accord with philosophic and theoretical traditions from the time of the Ancient Greek philosophers: Socrates, Plato and Aristotle (428 BC) through to modern epistemology, semiotics and logic of Peirce, Frege and Russell. Plato himself associates knowledge with the apprehension of unchanging Forms (The forms, according to Socrates, are roughly speaking archetypes or abstract representations of the many types of things, and properties) and their relationships to one another. In his original 1976 article Chen explicitly contrasts entity–relationship diagrams with record modelling techniques:

> "*The data structure diagram is a representation of the organisation of records and is not an exact representation of entities and relationships.*"

Several other authors also support his program:[14][15][16][17][18]

A semantic model is a model of concepts, it is sometimes called a "platform independent model". It is an intensional model. At the latest since Carnap, it is well known that:[19]

> *"...the full meaning of a concept is constituted by two aspects, its intension and its extension. The first part comprises the embedding of a concept in the world of concepts as a whole, i.e. the totality of all relations to*

other concepts. The second part establishes the referential meaning of the concept, i.e. its counterpart in the real or in a possible world".

An extensional model is one which maps to the elements of a particular methodology or technology, and is thus a "platform specific model". The UML specification explicitly states that associations in class models are extensional and this is in fact self-evident by considering the extensive array of additional "adornments" provided by the specification over and above those provided by any of the prior candidate "semantic modelling languages"."UML as a Data Modeling Notation, Part 2" [20]

Limitations

- ER models assume information content that can readily be represented in a relational database. They describe only a relational structure for this information.
- They are inadequate for systems in which the information cannot readily be represented in relational form, such as with semi-structured data.
- For many systems, possible changes to information contained are nontrivial and important enough to warrant explicit specification.
- SomeWikipedia:Avoid weasel words authors have extended ER modeling with constructs to represent change, an approach supported by the original author;[21] an example is Anchor Modeling. An alternative is to model change separately, using a process modeling technique. Additional techniques can be used for other aspects of systems. For instance, ER models roughly correspond to just 1 of the 14 different modeling techniques offered by UML.
- ER modeling is aimed at specifying information from scratch. This suits the design of new, standalone information systems, but is of less help in integrating pre-existing information sources that already define their own data representations in detail.
- Even where it is suitable in principle, ER modeling is rarely used as a separate activity. One reason for this is today's abundance of tools to support diagramming and other design support directly on relational database management systems. These tools can readily extract database diagrams that are very close to ER diagrams from existing databases, and they provide alternative views on the information contained in such diagrams.
- In a survey, Brodie and Liu[22] could not find a single instance of entity–relationship modeling inside a sample of ten Fortune 100 companies. Badia and Lemire[23] blame this lack of use on the lack of guidance but also on the lack of benefits, such as lack of support for data integration.
- The enhanced entity–relationship model (EER modeling) introduces several concepts which are not present in ER modeling, which are closely related to object-oriented design, like is-a relationships.
- For modelling temporal databases, numerous ER extensions have been considered.[24] Similarly, the ER model was found unsuitable for multidimensional databases (used in OLAP applications); no dominant conceptual model has emerged in this field yet, although they generally revolve around the concept of OLAP cube (also known as *data cube* within the field).

References

[1] "The Entity Relationship Model: Toward a Unified View of Data" (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.123.1085) for entity–relationship modeling.

[2] A.P.G. Brown, "Modelling a Real-World System and Designing a Schema to Represent It", in Douque and Nijssen (eds.), *Data Base Description*, North-Holland, 1975, ISBN 0-7204-2833-5.

[3] "Designing a Logical Database: Supertypes and Subtypes" (http://technet.microsoft.com/en-us/library/cc505839.aspx)

[4] Paul Beynon-Davies (2004). Database Systems. Houndmills, Basingstoke, UK: Palgrave

[5] "English, Chinese and ER diagrams" (http://bit.csc.lsu.edu/~chen/pdf/ER_C.pdf) by Peter Chen.

[6] Thomas Basboell: Motion and society. On meaningfulness of concepts (http://pangrammaticon.blogspot.de/2013/01/emotion-and-society.html)

[7] Hubert Tardieu, Arnold Rochfeld and René Colletti La methode MERISE: Principes et outils (Paperback - 1983)

[8] Elmasri, Ramez, B. Shamkant, Navathe, Fundamentals of Database Systems, third ed., Addison-Wesley, Menlo Park, CA, USA, 2000.

[9] ER 2004 : 23rd International Conference on Conceptual Modeling, Shanghai, China, November 8-12, 2004 (http://books.google.com/books?id=odZK99osY1EC&pg=PA52&img=1&pgis=1&dq=genova&sig=ACfU3U3tDC_q8WOMqUJW4EZCa5YQywoYLw&edge=0)

[10] A Formal Treatment of UML Class Diagrams as an Efficient Method for Configuration Management 2007 (http://publik.tuwien.ac.at/files/pub-inf_4582.pdf)

[11] James Dullea, Il-Yeol Song, Ioanna Lamprou - An analysis of structural validity in entity-relationship modeling 2002 (http://www.ischool.drexel.edu/faculty/song/publications/p_DKE_03_Validity.pdf)

[12] Hartmann, Sven. " Reasoning about participation constraints and Chen's constraints (http://crpit.com/confpapers/CRPITV17Hartmann.pdf)". Proceedings of the 14th Australasian database conference-Volume 17. Australian Computer Society, Inc., 2003.

[13] IDEF1X (https://idbms.navo.navy.mil/DataModel/IDEF1X.html)

[14] Kent in "Data and Reality" (http://www.bkent.net/Doc/darxrp.htm) : "One thing we ought to have clear in our minds at the outset of a modelling endeavour is whether we are intent on describing a portion of "reality" (some human enterprise) or a data processing activity."

[15] Abrial in "Data Semantics" : "... the so called "logical" definition and manipulation of data are still influenced (sometimes unconsciously) by the "physical" storage and retrieval mechanisms currently available on computer systems."

[16] Stamper: "They pretend to describe entity types, but the vocabulary is from data processing: fields, data items, values. Naming rules don't reflect the conventions we use for naming people and things; they reflect instead techniques for locating records in files."

[17] In Jackson's words: "The developer begins by creating a model of the reality with which the system is concerned, the reality which furnishes its [the system's] subject matter ..."

[18] Elmasri, Navathe: "The ER model concepts are designed to be closer to the user's perception of data and are not meant to describe the way in which data will be stored in the computer."

[19] http://wenku.baidu.com/view/8048e7bb1a37f111f1855b22.html

[20] http://www.tdan.com/view-articles/8589

[21] P. Chen. Suggested research directions for a new frontier: Active conceptual modeling (http://www.springerlink.com/content/5160x2634402663r/). ER 2006, volume 4215 of Lecture Notes in Computer Science, pages 1–4. Springer Berlin / Heidelberg, 2006.

[22] M. L. Brodie and J. T. Liu. The power and limits of relational technology in the age of information ecosystems (http://www.michaelbrodie.com/documents/The Power and Limits of Relational Technology In the Age of Information Ecosystems V2.pdf). On The Move Federated Conferences, 2010.

[23] A. Badia and D. Lemire. A call to arms: revisiting database design (http://www.sigmod.org/publications/sigmod-record/1109/pdfs/10.openforum.badia.pdf). SIGMOD Record 40, 3 (November 2011), 61-69.

[24] Gregersen, Heidi, and Christian S. Jensen. " Temporal Entity-Relationship models—a survey (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.2497&rep=rep1&type=pdf)." IEEE Transactions on Knowledge and Data Engineering, 11.3 (1999): 464-497.

Further reading

The basic **ER model** is covered in most database textbooks.

- Peter Chen, Peter Pin-Shan (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems* **1** (1): 9–36. doi: 10.1145/320434.320440 (http://dx.doi.org/10.1145/320434.320440). [Original paper by Chen]
- Peter Chen (2002) "Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned" (http://bit.csc.lsu.edu/~chen/pdf/Chen_Pioneers.pdf) in *Software pioneers*, pp. 296–310, Springer-Verlag, ISBN 3-540-43081-4 [A retrospective]

In-depth monographs about ER-based modelling:

- Richard Barker (1990). *CASE Method: Entity Relationship Modelling*, Addison-Wesley, ISBN 0201416964
- Richard Barker (1990). *CASE Method: Tasks and Deliverables*, Addison-Wesley. ISBN 0201416972

- Heikki Mannila, Kari-Jouko Räihä (1992). *The Design of Relational Databases*. Addison-Wesley. ISBN 0201565234
- Bernhard Thalheim (2000). *Entity-Relationship Modeling: Foundations of Database Technology*. Springer. ISBN 978-3-540-65470-4.
- Sikha Bagui; Richard Earp (2011). *Database Design Using Entity-Relationship Diagrams* (2nd ed.). CRC Press. ISBN 978-1-4398-6176-9.

External links

- Entity Relationship Modelling (http://www.databasedesign.co.uk/bookdatabasesafirstcourse/chap3/chap3.htm)
- An Entity Relationship Diagram Example (http://rapidapplicationdevelopment.blogspot.com/2007/06/entity-relationship-diagram-example.html). Demonstrates the crow's feet notation by way of an example.
- Case study: E-R diagram for Acme Fashion Supplies (http://www.cilco.co.uk/briefing-studies/acme-fashion-supplies-feasibility-study/slides/logical-data-structure.html) by Mark H. Ridley.
- Logical Data Structures (LDSs) - Getting started (http://www.cems.uwe.ac.uk/~tdrewry/lds.htm) by Tony Drewry.
- Crow's Foot Notation (http://www2.cs.uregina.ca/~bernatja/crowsfoot.html)

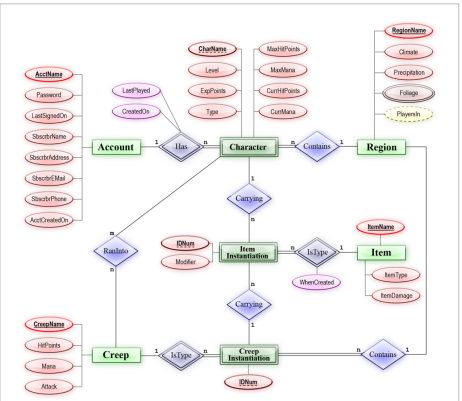
Has-a

In database design and object oriented program architecture, **has-a** is a relationship where one object (often called the composited object) "belongs" to (is a part or member of) another object (called the composite type), and behaves according to the rules of ownership. In simple words, **has-a** relationship in an object is called a member field of an object. Multiple **has-a** relationships will combine to form a possessive hierarchy. This is contrasted with an Is-a relationship which constitutes a different kind of hierarchy (subtyping). The decision whether the most logical relationship for an object and its subordinate is not always clearly *has-a* or *is-a*. Confusion over such decisions have necessitated the creation of these metalinguistic terms. A good example of the *has-a* relationship is containers in the C++ STL.

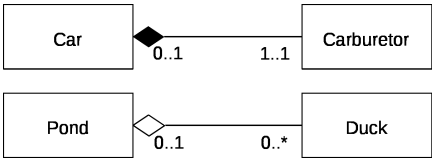
Examples

In databases has-a relationships are usually represented in an Entity-relationship model. As you can see by the diagram on the right an account can have multiple characters. This shows that account has a "has-a" relationship with character.

In object-oriented programming this relationship can be represented with a Unified Modeling Language diagram. This has-a relationship is also known as composition. As you can see from the diagram on the right a car "has-a " carburetor, or a car is "composed of" a carburetor. When the diamond is coloured black it signifies composition, i.e. the object on the side closest to the diamond is made up of or contains the other object. While the white diamond signifies aggregation, which means that the object closest to the diamond can have or possess the other object.



ER Model



Misuses of composition and aggregation

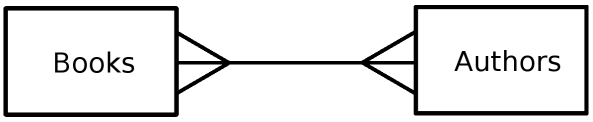
Another way to distinguish between composition and aggregation in modeling the real world, is to consider the relative lifetime of the contained object. For example, if a Car object contains a Chassis object, a Chassis will most likely not be replaced during the lifetime of the Car. It will have the same lifetime as the car itself; so the relationship is one of composition. On the other hand, if the Car object contains a set of Tire objects, these Tire objects may wear out and get replaced several times. Or if the Car becomes unusable, some Tires may be salvaged and assigned to another Car. At any rate, the Tire objects have different lifetimes than the Car object; therefore the relationship is one of aggregation.

The diagram on the right shows a very common misuse of the aggregation concept. It is nowadays common to see almost every relationship in a UML model marked as an aggregation. However it makes no sense to say that "a pond is an aggregation of ducks". The diagram is also incorrect in its example of composition. A carburetor is not necessarily dependent on the car for existence. It could be on a shelf somewhere. The car is an aggregate of its parts. An order-line on the other hand lives and dies with the order. It is therefore correct to say that the order is a composition of order-lines.

If one were to make a C++ software Class to implement the relationships described above, the Car object would contain a complete Chassis object in a data member. This Chassis object would be instantiated in the constructor of the Car class (or defined as the data type of the data member and its properties assigned in the constructor.) And since it would be a wholly contained data member of the Car class, the Chassis object would no longer exist if a Car class object was to be deleted.

On the other hand, the Car class data members that point to Tire objects would most likely be C++ pointers. Tire objects could be instantiated and deleted externally, or even assigned to data members of a different Car object. Tire objects would have an independent lifetime separate from when the Car object was deleted.

Many-to-many (data model)



An Author can write several Books, and a Book can be written by several Authors

In systems analysis, a **many-to-many** relationship is a type of cardinality that refers to the relationship between two entities (see also entity–relationship model) A and B in which A may contain a parent row[*clarify*] for which there are many children[*clarify*] in B and vice versa. For instance, think of A as Authors, and B as Books. An Author can write several Books, and a Book can be written by several Authors. Because most database management systems only support one-to-many relationships, it is necessary to implement such relationships physically via a third junction table (also called *cross-reference table*), say, AB with two one-to-many relationships A -> AB and B -> AB. In this case the logical primary key for AB is formed from the two foreign keys (i.e. copies of the primary keys of A and B).

In web application frameworks such as CakePHP and Ruby on Rails, a **many-to-many** relationship between database tables in a model is sometimes referred to as a HasAndBelongsToMany (HABTM) relationship.[1]

References

[1] 3.7.6.5 hasAndBelongsToMany (HABTM) (http://book.cakephp.org/1.3/view/1044/hasAndBelongsToMany-HABTM). Cakephp.org

External links

- Design pattern: many-to-many (order entry) (http://www.tomjewett.com/dbdesign/dbdesign.php?page=manymany.php), Tomjewett.com

Enhanced Entity-Relationship Model

The **enhanced entity–relationship (EER)** model (or **extended entity-relationship** model) in computer science is a high-level or conceptual data model incorporating extensions to the original entity–relationship (ER) model, used in the design of databases.

It was developed to reflect more precisely the properties and constraints that are found in more complex databases, such as in engineering design and manufacturing (CAD/CAM), telecommunications, complex software systems and geographic information systems (GIS).

The EER model

The EER model includes all of the concepts introduced by the ER model. Additionally it includes the concepts of a subclass and superclass (Is-a), along with the concepts of specialization and generalization. Furthermore, it introduces the concept of a union type or category, which is used to represent a collection of objects that is the union of objects of different entity types.

Subclass and superclass

Entity type Y is a subtype (subclass) of an entity type X if and only if every Y is necessarily an X. A subclass entity inherits all attributes and relationships of its superclass entity. A subclass entity may have its own specific attributes and relationships (together with all the attributes and relationships it inherits from the superclass. One of the most common superclass examples is a vehicle with subclasses of Car and Truck. There are a number of common attributes between a car and a truck, which would be part of the Superclass, while the attributes specific to a car or a truck (such as max payload, truck type...) would make up two subclasses.

Software tools for the EER model

The MySQL Workbench offers creating, editing and exporting EER Models. Exporting to PNG and PDF allows easy sharing for presentations.

Further reading

Textbooks discussing EER and implementation using purely relational databases:

- Elmasri, Ramez; Shamkant B.Navathe (2011). *Fundamentals of Database Systems* (6th ed.). Pearson/Addison Wesley. ISBN 0-136-08620-9. Chapters 8 and 9.
- Carlos Coronel; Steven Morris; Peter Rob (2011). *Database Systems: Design, Implementation, and Management* (9th ed.). Cengage Learning. ISBN 978-0-538-46968-5. Chapter 5.
- Thomas M. Connolly; Carolyn E. Begg (2005). *Database Systems: A Practical Approach to Design, Implementation, and Management* (4th ed.). Addison-Wesley. ISBN 978-0-321-21025-8. Chapters 12 and 16.

Booklet discussing EER and implementation using object-oriented and object-relational databases:

- Suzanne W. Dietrich; Susan D. Urban (2011). *Fundamentals of Object Databases: Object-Oriented and Object-Relational Design*. Morgan & Claypool Publishers. ISBN 978-1-60845-476-1.

Textbook discussing implementation in relational and object-relational databases:

- Catherine Ricardo (2011). *Databases Illuminated* (2nd ed.). Jones & Bartlett Publishers. ISBN 978-1-4496-0600-8. Chapter 8.

Shorter survey articles:

- Teorey, Toby J., Dongqing Yang, and James P. Fry. "A logical design methodology for relational databases using the extended entity-relationship model [1]". ACM Computing Surveys (CSUR) 18.2 (1986): 197-222.
- Sikha Bagui (2006). "Extended Entity Relationship Modeling". In Laura C. Rivero, Jorge H. Doorn, Viviana E. Ferraggine. *Encyclopedia of Database Technologies and Applications*. Idea Group Inc (IGI). pp. 233–239. ISBN 978-1-59140-795-9.

External links

- [2] - Slides for chapter 8 from Fundamentals of Database Systems by Elmasri and Navathe (Pearson, 2011)
- [3] - Lecture notes from the University of Toronto
- [4] - The ER Conference

References

[1] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.7211&rep=rep1&type=pdf
[2] http://tinman.cs.gsu.edu/~raj/4710/f11/Ch08.pdf
[3] http://www.cs.toronto.edu/~jm/2507S/Notes04/EER.pdf
[4] http://www.conceptualmodeling.org/

Weak entity

In a relational database, a **weak entity** is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key. The foreign key is typically a primary key of an entity it is related to.

In entity relationship diagrams a weak entity set is indicated by a bold (or double-lined) rectangle (the entity) connected by a bold (or double-lined) type arrow to a bold (or double-lined) diamond (the relationship). This type of relationship is called an *identifying relationship* and in IDEF1X notation it is represented by an oval entity rather than a square entity for base tables. An identifying relationship is one where the primary key is populated to the child weak entity as a primary key in that entity.

In general (though not necessarily) a weak entity does not have any items in its primary key other than its inherited primary key and a sequence number. There are two types of weak entities: associative entities and subtype entities. The latter represents a crucial type of normalization, where the super-type entity inherits its attributes to subtype entities based on the value of the discriminator.

In IDEF1X, a government standard for capturing requirements, possible sub-type relationships are:

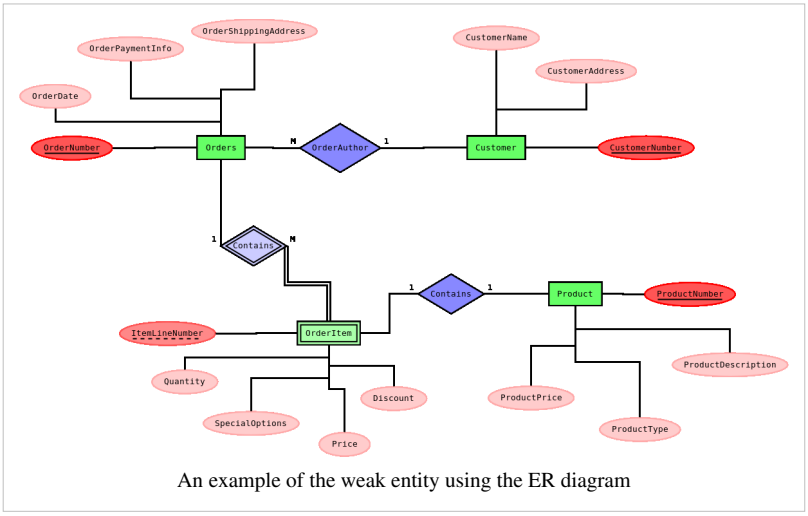
- *Complete subtype relationship*, when all categories are known.
- *Incomplete subtype relationship*, when all categories may not be known.

A classic example of a weak entity without a sub-type relationship would be the "header/detail' records in many real world situations such as claims, orders and invoices, where the header captures information common across all forms and the detail captures information specific to individual items.

The standard example of a **complete subtype relationship** is the *party* entity. Given the discriminator PARTY TYPE (which could be individual, partnership, C Corporation, Sub Chapter S Association, Association, Governmental Unit, Quasi-governmental agency) the two subtype entities are PERSON, which contains individual-specific information such as first and last name and date of birth, and ORGANIZATION, which would contain such attributes as the legal name, and organizational hierarchies such as cost centers.

When sub-type relationships are rendered in a database, the super-type becomes what is referred to as a base table. The sub-types are considered derived tables, which correspond to weak entities. Referential integrity is enforced via cascading updates and deletes.

Example



An example of the weak entity using the ER diagram

Consider a database that records customer orders, where an order is for one or more of the items that the enterprise sells. The database would contain a table identifying customers by a customer number (primary key); another identifying the products that can be sold by a product number (primary key); and it would contain a pair of tables describing orders.

One of the tables could be called Orders and it would have an order number (primary key) to identify this order uniquely, and would contain a customer number (foreign key) to identify who the products are being sold to, plus other information such as the date and time when the order was placed, how it will be paid for, where it is to be shipped to, and so on.

The other table could be called OrderItem; it would be identified by a compound key consisting of both the order number (foreign key) and an item line number; with other non-primary key attributes such as the product number (foreign key) that was ordered, the quantity, the price, any discount, any special options, and so on. There may be zero, one or many OrderItem entries corresponding to an Order entry, but no OrderItem entry can exist unless the corresponding Order entry exists. (The zero OrderItem case normally only applies transiently, when the order is first entered and before the first ordered item has been recorded.)

The OrderItem table stores *weak entities* precisely because an OrderItem has no meaning independent of the Order. Some might argue that an OrderItem does have some meaning on its own; it records that at some time not identified by the record, somebody not identified by the record ordered a certain quantity of a certain product. This information might be of some use on its own, but it is of limited use. For example, as soon as you want to find seasonal or geographical trends in the sales of the item, you need information from the related Order record.

An order would not exist without a product and a person to create the order, so it could be argued that an order would be described as a weak entity and that products ordered would be a multivalue attribute of the order.

Associative Entities

An **associative entity** is an element of the entity–relationship model. The database relational model does not offer direct support to many-to-many relationships, even though such relationships happen frequently in normal usage. The solution to this problem is the creation of another table to hold the necessary information for this relationship. This new table is called an **associative entity**.

An associative entity

To create a relationship, a "child" entity must inherit the primary key of a "parent" entity. However, in a many-to-many relationship, neither entity is the "parent" or the "child"; the relationship is "unresolved". In order to work, these databases require an additional construct to "resolve" the relationship (which is why associative entities are also referred to as "resolving entities").

An associative entity can be thought of as both an entity and a relationship since it encapsulates properties from both. It is a relationship since it is serving to join two or more entities together, but it is also an entity since it may have its own properties. The associative entity must have the primary keys of both adjoining tables as identifiers, but may also contain its own unique identifier and other information about the relationship.

The following guidelines may be used when considering the use of an associative entity:

- All relationships for the associative entity should be many.
- The associative entity could have meaning independent of the other entities.
- The associative entity should not have an additional surrogate key (identity column). The primary key should be a composite of the primary keys from the referenced entities. If the association is temporal, the transaction time may also be part of this composite key. If other entities will be referencing the associative entity and the composite of the referenced primary keys creates a large byte width for the associative entity's primary key, an exception may be made for this rule and an identity column may be added and used as the primary key. In these cases, an alternate key constraint should be implemented on the entity which would be the composite of the referenced primary keys.
- The associative entity may participate in relationships other than the entities of the associated relationship.

References

- Modern Database Management - 7th Edition - Jeffrey A. Hoffer, Mary B. Prescott, Fred R. McFadden

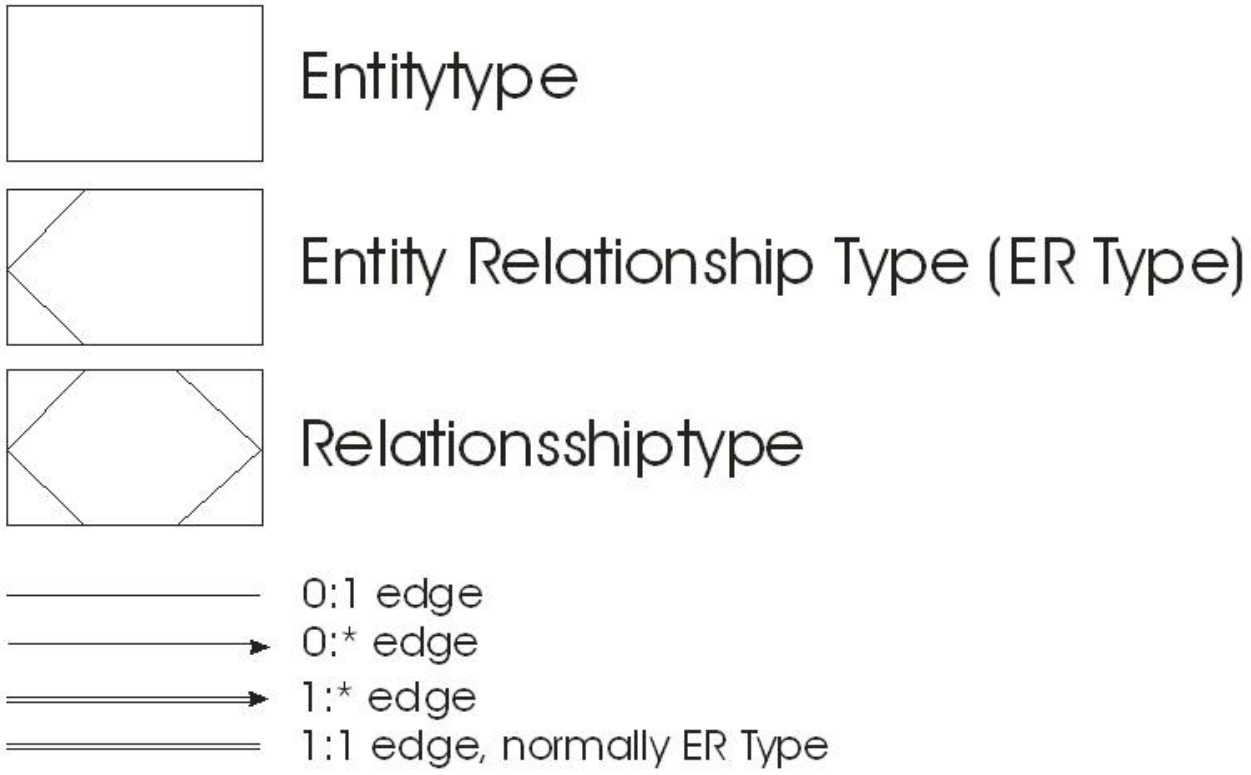
Structured-Entity-Relationship-Model

The **SERM** (Structured Entity Relationship Model) is an amplification of the ERM which is commonly used for data modeling. It was first proposed from Prof. Dr. Sinz in 1989. The SERM is commonly used in the SAP-world for the data modeling.

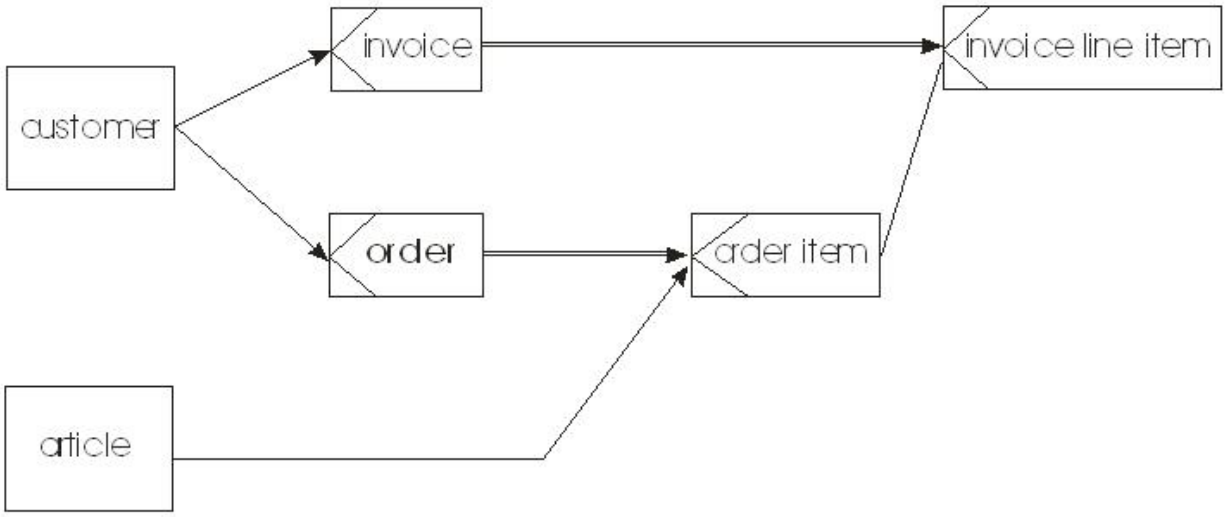
Aims

1. structuring of large schemes
2. visualization of existence dependency
3. avoidance of inconsistencies
4. avoidance of unnecessary relationshiptypes

SERM-Symbols



SERM-Example



- Customer and article are independent entities
- Every order is referred to one customer. Orders without customers are illegal (order is an ER Type). Customers without any orders are legal because they are independent Entities.
- To every order there is belonging at least one order item.
- Every order item is related to exactly one order.
- Every invoice is referred to one customer, as well. Invoices without customers are illegal. Customers without any invoice are legal.
- To every invoice there is belonging at least one invoice line item.
- Every invoice line item is related to exactly order item. An order item could be calculated or not.
- SERM is already in the third normal form

Barker's Notation

Barker's notation refers to the ERD notation developed by Richard Barker, Ian Palmer, Harry Ellis et al. whilst working at the British consulting firm CACI around 1981. The notation was adopted by Barker when he joined Oracle and is effectively defined in his book *Entity Relationship Modelling* as part of the CASE Method series of books. This notation was and still is used by the Oracle CASE modelling tools. It is a variation of the "crows foot" style of data modelling that was favoured by many over the original Chen style of ERD modelling because of its readability and efficient use of drawing space.

The notation has features that represent the properties of relationships including cardinality and optionality (the crows foot and dashing of lines), exclusion (the exclusion arc), recursion (looping structures) and use of abstraction (nested boxes).

Further reading

- Richard Barker (1990). *CASE Method: Entity Relationship Modelling*. Reading, MA: Addison-Wesley Professional. ISBN 0-201-41696-4.

External links

- www.entitymodelling.org [1] A website dedicated to the use of the Barker notation in an extended form.

References

[1] http://www.entitymodelling.org/home-1/

Peter Chen

Peter Chen	
Born	3 January 1947 Taichung, Taiwan, ROC
Fields	Computer Science
Institutions	Carnegie Mellon University Harvard University University of California Massachusetts Institute of Technology Louisiana State University
Alma mater	National Taiwan University Harvard University
Known for	Development of entity-relationship modeling
Notable awards	Stevens Software Innovation Award (2001) ACM/AAAI Allen Newell Award (2002) IEEE Harry H. Goode Memorial Award (2003) DAMA International Achievement Award (2000) Pan Wen-Yuen Research Excellence Award (2004) Software Eng. Society & SDPS Society Transformative Award (2011) Fellow of ACM, IEEE, AAAS, & ER

Peter Pin-Shan Chen (Chinese: 陳 品 山) is an American computer scientist. He is a Distinguished Career Scientist and faculty member at Carnegie Mellon University, who is known for the development of Entity-Relationship Model in 1976.

Biography

Born in Taichung, Taiwan, Peter Chen received a B.S. in electrical engineering in 1968 at the National Taiwan University, and a Ph.D. in computer science/applied mathematics at the Harvard University in 1973. In 1970, he worked one summer at IBM. After graduated from Harvard, he worked one year at Honeywell and one summer at Digital Equipment Corporation.

From 1974 to 1978 Chen was an Assistant Professor at the MIT Sloan School of Management. From 1978 to 1983 he was Associate Professor at the University of California, Los Angeles (UCLA Management School). From 1983 to 2011 Chen held the position of M. J. Foster Distinguished Chair Professor of Computer Science at Louisiana State University and, for several years, Adjunct Professor in its Business School and Medical School (Shreveport).[1] During this time period, he was a Visiting Professor once at Harvard in '89-'90 and three times at Massachusetts Institute of Technology (EECS Dept. in '86-'87, Sloan School in '90-'91, and Division of Engineering Systems in 06-'07). Since 2008, he has been Honorary Chair Professor in the Institute of Service Science at National Tsing Hua University. Currently, Chen is a Distinguished Career Scientist and faculty member at Carnegie Mellon University.

Awards and honors

Chen's original paper[2] is one of the most influential papers in the computer software field based on a survey of more than 1,000 computer science professors documented in a book on "Great Papers in Computer Science".[3][4] Chen's work is also cited in a book *Software Challenges* published by Time-Life Books in 1993 in the series on "Understanding Computers." Chen is recognized as one of the pioneers in a book on "Software Pioneers".[5]

Chen has received many awards in the fields of Information Technology.[6] He received the Data Resource Management Technology Award from the Data Administration Management Association in New York City in 1990. He was elected as a Fellow of the Association for Computing Machinery (ACM), American Association for the Advancement of Science (AAAS), IEEE, and ER.[7] He won the Achievement Award in Information Management in 2000 from DAMA International. He was an inductee into the Data Management Hall of Fame in 2000. He received the Stevens Award in Software Method Innovation in 2001. In 2003, Chen received the IEEE Harry H. Goode Memorial Award at the IEEE-CS Board of Governors meeting in San Diego. He was presented with the ACM/AAAI Allen Newell Award at the ACM Banquet in San Diego in June 2003 and International Joint Conference on Artificial Intelligence (IJCAI) in Acapulco in August 2003. Chen is also the recipient of the Pan Wen-Yuan Outstanding Research Award in 2004.[8] In June 2011 in Jeju Island, Korea, Chen received the Transformative Achievement Medal from Software Engineering Society and the Society for Design and Process Science.

Conceptual modeling field and annual conceptual modeling (ER) conference

His innovative work initiated a new field of research and practice called Conceptual Modeling. Since 1979, an annual international professional meeting, the International Conference on Conceptual Modeling has been held in different countries.

Peter P. Chen Award

To recognize Chen's pioneering leadership role, "Peter P. Chen Award" was established in 2008, to be given to one excellent researcher/educator in the conceptual modeling field each year. The recipients of the Peter P. Chen Award are:

- 2008: Bernhard Thalheim, Professor, University of Kiel, Germany
- 2009: David W. Embley, Professor, Brigham Young University (BYU), U.S.A.
- 2010: John Mylopoulos, Professor, University of Toronto, Canada, and University of Trento, Italy
- 2011: Tok Wang Ling, Professor, University of Singapore, Singapore
- 2012: Stefano Spaccapietra, Swiss Federal Institute of Technology (EPFL), Switzerland
- 2013: Carlo Batini, Professor, University of Milano-Bicocca, Italy

Work

Entity-relationship Modeling and data/conceptual modeling

The entity-relationship model model serves as the foundation of many systems analysis and design methodologies, computer-aided software engineering (CASE) tools, and repository systems. The ER Model is the basis for IBM's Repository Manager/MVS and DEC's CDD/Plus.

Dr. Peter Chen's original paper[9] is commonly cited as the definitive reference for Entity Relationship Modelling. Chen is one of the pioneers of using the entity relationship concepts in software and information system modeling and design. Before Chen's paper, the basic entity relationship ideas were used mostly informally by practitioners. Chen first published an abstract and presented his ER model in the First Very Large Database Conference in 1975, the same year of a paper with similar concepts written by A. P. G. Brown.[10] Chen's main contributions are: formalized the concepts, developed a theory with a set of data definition and manipulation operations, and specified

the translation rules from the ER model to several major types of databases (including the Relational Database). He also popularized the model and introduced it to the academic literature.

The ER Model was adopted as the meta model ANSI Standard in Information Resource Directory System (IRDS), and the ER approach has been ranked at the top methodology for database design and one of the top methodologies in systems development by several surveys of Fortune 500 companies.

Computer-aided software engineering

Chen's work is a cornerstone of software engineering,[] in particular Computer-Aided Software Engineering (CASE). In the late 80's and early 90's, IBM's Application Development Cycle (AD/Cycle) framework and DB2 repository (RM/MVS) were based on the ER model. Other vendors' repository systems such as Digital's CDD+ were also based on the ER model. Chen has had a significant impact on the CASE industry through his research and his lecturing around the world on structured system development methodologies. The ER model has influenced most of the major CASE tools, including Computer Associates' ERWIN, Oracle Corporation's Designer/2000, and Sybase's PowerDesigner (and even a general drawing tool like Microsoft Visio), as well as the IDEF1X standard. The ER model is also the basis for Microsoft's ADO.NET Entity Framework.

The hypertext concept, which makes the World Wide Web extremely popular, is very similar to the main concept in the ER model. Chen investigated this linkage as an invited expert of several XML working groups of the World Wide Web Consortium (W3C).

The ER model also serves as the foundation of some of the recent work on Object-oriented analysis and design methodologies and Semantic Web. The UML modeling language has its roots in the ER model.

Computer performance modeling

In his early career, he was active in R&D activities in computer system performance. He was the program chair of an ACM SIGMETRICS conference. He developed a computer performance model for a major computer vendor. His innovative research results were adopted in commercial compter performance tuning and capacity planning tools.

Memory/storage hierarchy, storage technology, CD-ROM, firmware, micro-programming

His Ph.D. thesis at Harvard was one of the first studies of cost/performance optimization models of memory/storage hierarchies. He was also one of the early micro-programmers developing the firmware for a file control unit for an IBM mainframe computer. His article on "CD-ROM" in IEEE Proceedings journal in the 80s was one of the first articles explaining how CD-ROM worked when CD-ROMs became popular. He was a co-author of the "storage technology" article in a major computer encyclopedia.

Cyber security and terrorist detection

In recent years, he led a multidisciplinary research team in developing new efficient and effective techniques in identifying terrorists and malicious cyber transactions. At CMU, he is active in the R&D activities of CERT Coordination Center and Software Engineering Institute (SEI).

Publications

Peter P. Chen published several books, papers and articles. Books, a selection:

- 2007. *Active Conceptual Modeling of Learning: Next Generation Learning-Base System Development*. With Leah Y. Wong (Eds.). Springer.
- 1999. *Advances in Conceptual Modeling: ER'99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World ... (Lecture Notes in Computer Science)*. With David W. Embley, Jacques Kouloumdjian, Stephen W. Liddle and John F. Roddick (Eds.) Springer Verlag.

- 1999. *Conceptual Modeling: Current Issues and Future Directions (Lecture Notes in Computer Science)* With Jacky Akoka, Hannu Kangassalo, and Bernhard Thalheim.
- 1985. *Data & Knowledge Engineering*, Volume 1, Number 1, 1985.
- 1981. *Entity Relationship Approach to Information Modeling and Analysis*.
- 1980. *Entity relationship approach to systems analysis and design*. North-Holland.

Articles (a selection)

- 1976. "The Entity-Relationship Model--Toward a Unified View of Data" [11]. In: *ACM Transactions on Database Systems* 1/1/1976 ACM-Press ISSN 0362-5915, S. 9–36
- 2002. "Entity-Relationship Modeling--Historical Events, Future Trends, and Lessons Learned" [12]. In: *Software Pioneers: Contributions to Software Engineering*. Broy M. and Denert, E. (eds.), Springer-Verlag, Berlin, Lecturing Notes in Computer Sciences, June 2002, pp. 100–114.

References

[1] Dr. Peter Chen home page (http://bit.csc.lsu.edu/~chen/chen.html) Section Education & Experience.(Retrieved 1 October 2008

[2] The Entity Relationship Model - Toward A Unified View of Data (http://citeseer.ist.psu.edu/519283.html)

[3] See Great Papers in Computer Science (http://bit.csc.lsu.edu/~chen/GreatPapers.html)

[4] Laplante, P., ed. *Great Papers in Computer Science.* West Publishing Co. 1996. ISBN 0-314-06365

[5] Software Pioneers: Contributions to Software Engineering. Broy M. and Denert, E. (eds.), Springer-Verlag, Berlin, Lecturing Notes in Computer Sciences, June 2002.

[6] Dr. Peter Chen home page (http://bit.csc.lsu.edu/~chen/chen.html) Section Honors & Professional Activities. Retrieved 1 October 2008

[7] ER/Conceptual Modeling Main Website (http://www.conceptualmodeling.org)

[8] Dr. Peter Pin-Shan Chen at Louisiana State University Received Outstanding Research Award from Pan Wen-Yuan Foundation (http://stn.nsc.gov.tw/en/view_detail.asp?doc_uid=0930812007). 2004.08.18.

[9] The Entity Relationship Model - Toward A Unified View of Data (http://www.csc.lsu.edu/news/erd.pdf)

[10] A. P. G. Brown, Modelling a Real-World System and Designing a Schema to Represent It, in Data Base Description, ed Douque and Nijssen, North-Holland, 1975, ISBN 0-7204-2833-5

[11] http://csc.lsu.edu/news/erd.pdf

[12] http://bit.csc.lsu.edu/~chen/pdf/Chen_Pioneers.pdf

External links

- Home page of Dr. Peter Chen at Louisiana State University (http://bit.csc.lsu.edu/~chen/chen.html)

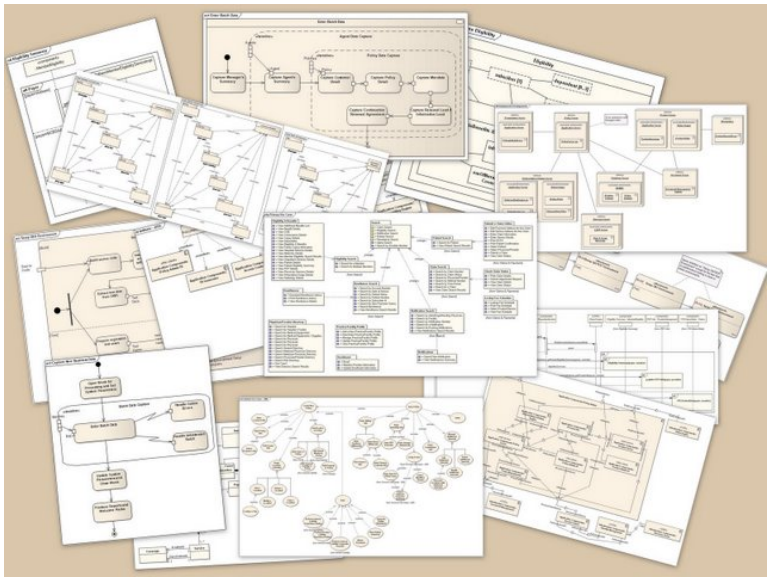
Other Data Modeling Techniques

Unified Modeling Language

The **Unified Modeling Language** (**UML**) is a general-purpose modeling language in the field of software engineering. It provides a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. It was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in the 1990s.[1] It was adopted by the Object Management Group (OMG) in 1997, and has been managed by this organization ever since. In 2000 the Unified Modeling Language was accepted by the International Organization for Standardization (ISO) as a standard for modeling software-intensive systems.

Overview

Unified Modeling Language (UML) combines techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.[2]



A collage of UML diagrams.

The Unified Modeling Language (UML) offers a standard way to visualize a system's architectural blueprints, including elements such as:

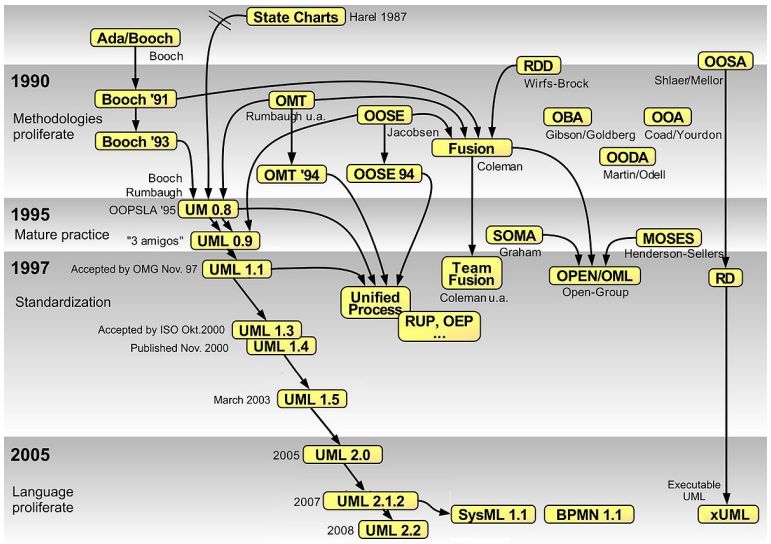
- activities
- actors
- business processes
- database schemas
- (logical) components
- programming language statements
- reusable software components.[3]

UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

UML models may be automatically transformed to other representations (e.g. Java) by means of QVT-like transformation languages. UML is extensible, with two mechanisms for customization: profiles and stereotypes.

History

UML has been evolving since the second half of the 1990s and has its roots in the object-oriented methods developed in the late 1980s and early 1990s. The timeline (see image) shows the highlight of the history of object-oriented modeling methods and notation.



History of object-oriented methods and notation.

Before UML 1.x

After Rational Software Corporation hired James Rumbaugh from General Electric in 1994, the company became the source for two of the most popular object-oriented modeling approaches of the day:[4] Rumbaugh's Object-modeling technique (OMT) and Grady Booch's method known as Object-oriented design (OOD). They were soon assisted in their efforts by Ivar Jacobson, the creator of the object-oriented software engineering (OOSE) method. Jacobson joined Rational in 1995, after his company, Objectory AB,[5] was acquired by Rational. The three methodologists were collectively referred to as the "Three Amigos".

In 1996, Rational concluded that the abundance of modeling languages was slowing the adoption of object technology,[*citation needed*] so repositioning the work on a unified method, they tasked the "Three Amigos" with the development of a non-proprietary Unified Modeling Language. Representatives of competing object technology companies were consulted during OOPSLA '96;[*citation needed*] they chose *boxes* for representing classes rather than the *cloud* symbols that were used in Booch's notation.

Under the technical leadership of the "Three Amigos", an international consortium called the UML Partners was organized in 1996 to complete the *Unified Modeling Language (UML)* specification, and propose it as a response to the OMG RFP. The UML Partners' UML 1.0 specification draft was proposed to the OMG in January 1997. During the same month the UML Partners formed a Semantics Task Force, chaired by Cris Kobryn and administered by Ed Eykholt, to finalize the semantics of the specification and integrate it with other standardization efforts. The result of this work, UML 1.1, was submitted to the OMG in August 1997 and adopted by the OMG in November 1997.

UML 1.x

As a modeling notation, the influence of the OMT notation dominates (e. g., using rectangles for classes and objects). Though the Booch "cloud" notation was dropped, the Booch capability to specify lower-level design detail was embraced. The use case notation from Objectory and the component notation from Booch were integrated with the rest of the notation, but the semantic integration was relatively weak in UML 1.1, and was not really fixed until the UML 2.0 major revision.[*citation needed*]

Concepts from many other OO methods were also loosely integrated with UML with the intent that UML would support all OO methods. Many others also contributed, with their approaches flavouring the many models of the day, including: Tony Wasserman and Peter Pircher with the "Object-Oriented Structured Design (OOSD)" notation (not a method), Ray Buhr's "Systems Design with Ada", Archie Bowen's use case and timing analysis, Paul Ward's data analysis and David Harel's "Statecharts"; as the group tried to ensure broad coverage in the real-time systems domain. As a result, UML is useful in a variety of engineering problems, from single process, single user

applications to concurrent, distributed systems, making UML rich but also large.

The Unified Modeling Language is an international standard:

> ISO/IEC 19501:2005 Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2

UML 2.x

UML has matured significantly since UML 1.1. Several minor revisions (UML 1.3, 1.4, and 1.5) fixed shortcomings and bugs with the first version of UML, followed by the UML 2.0 major revision that was adopted by the OMG in 2005.

Although UML 2.1 was never released as a formal specification, versions 2.1.1 and 2.1.2 appeared in 2007, followed by UML 2.2 in February 2009. UML 2.3 was formally released in May 2010. UML 2.4.1 was formally released in August 2011. UML 2.5 was released in October 2012 as an "In process" version and has yet to become formally released.

There are four parts to the UML 2.x specification:

1. The Superstructure that defines the notation and semantics for diagrams and their model elements
2. The Infrastructure that defines the core metamodel on which the Superstructure is based
3. The Object Constraint Language (OCL) for defining rules for model elements
4. The UML Diagram Interchange that defines how UML 2 diagram layouts are exchanged

The current versions of these standards follow: UML Superstructure version 2.4.1, UML Infrastructure version 2.4.1, OCL version 2.3.1, and UML Diagram Interchange version 1.0.

Although many UML tools support some of the new features of UML 2.x, the OMG provides no test suite to objectively test compliance with its specifications.

Topics

Software development methods

UML is not a development method by itself;[6] however, it was designed to be compatible with the leading object-oriented software development methods of its time (for example OMT, Booch method, Objectory). Since UML has evolved, some of these methods have been recast to take advantage of the new notations (for example OMT), and new methods have been created based on UML, such as IBM Rational Unified Process (RUP). Others include Abstraction Method and Dynamic Systems Development Method.

Modeling

It is important to distinguish between the UML model and the set of diagrams of a system. A diagram is a partial graphic representation of a system's model. The model also contains documentation that drives the model elements and diagrams (such as written use cases).

UML diagrams represent two different views of a system model:[7]

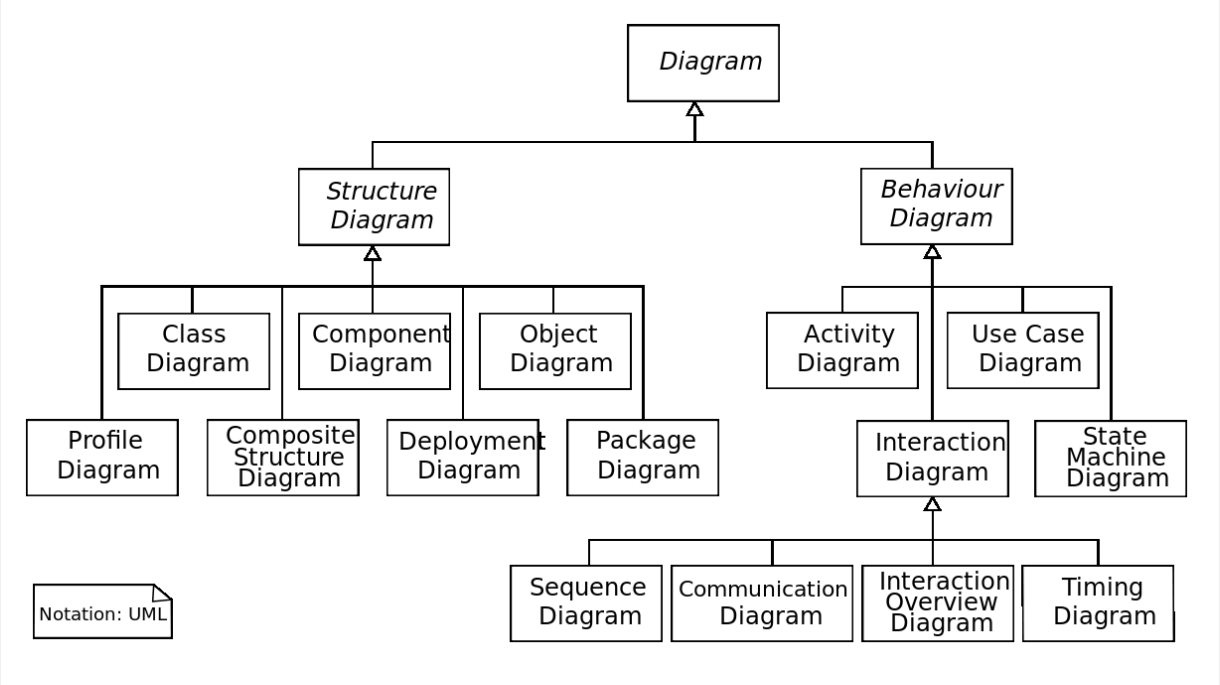
- Static (or *structural*) view: emphasizes the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams and composite structure diagrams.
- Dynamic (or *behavioral*) view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.

UML models can be exchanged among UML tools by using the XML Metadata Interchange (XMI) interchange format.

Diagrams overview

UML diagrams
Structural UML diagrams
• Class diagram • Component diagram • Composite structure diagram • Deployment diagram • Object diagram • Package diagram • Profile diagram
Behavioral UML diagrams
• Activity diagram • Communication diagram • Interaction overview diagram • Sequence diagram • State diagram • Timing diagram • Use case diagram
• v • t • e [8]

UML 2.2 has 14 types of diagrams divided into two categories.[9] Seven diagram types represent *structural* information, and the other seven represent general types of *behavior*, including four that represent different aspects of *interactions*. These diagrams can be categorized hierarchically as shown in the following class diagram:



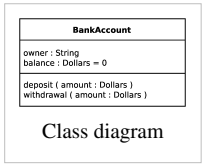
UML does not restrict UML element types to a certain diagram type. In general, every UML element may appear on almost all types of diagrams; this flexibility has been partially restricted in UML 2.0. UML profiles may define additional diagram types or extend existing diagrams with additional notations.

In keeping with the tradition of engineering drawings,[*citation needed*] a comment or note explaining usage, constraint, or intent is allowed in a UML diagram.

Structure diagrams

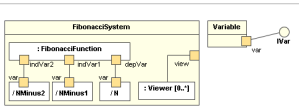
Structure diagrams emphasize the things that must be present in the system being modeled. Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.

- Class diagram: describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.
- Component diagram: describes how a software system is split up into components and shows the dependencies among these components.
- Composite structure diagram: describes the internal structure of a class and the collaborations that this structure makes possible.
- Deployment diagram: describes the hardware used in system implementations and the execution environments and artifacts deployed on the hardware.
- Object diagram: shows a complete or partial view of the structure of an example modeled system at a specific time.
- Package diagram: describes how a system is split up into logical groupings by showing the dependencies among these groupings.
- Profile diagram: operates at the metamodel level to show stereotypes as classes with the <<stereotype>> stereotype, and profiles as packages with the <<profile>> stereotype. The extension relation (solid line with closed, filled arrowhead) indicates what metamodel element a given stereotype is extending.



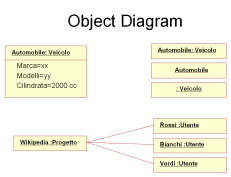
Class diagram


Component diagram

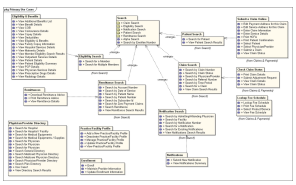


Composite structure diagram


Deployment diagram



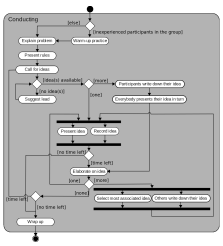
Object diagram


Package diagram

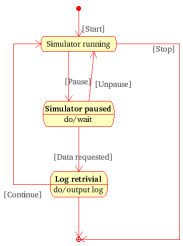
Behavior diagrams

Behavior diagrams emphasize what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

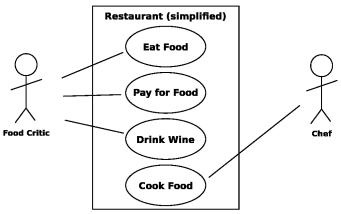
- Activity diagram: describes the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
- UML state machine diagram: describes the states and state transitions of the system.
- Use Case Diagram: describes the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases.



UML Activity Diagram



State Machine diagram

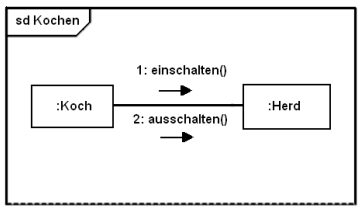


Use Case Diagram

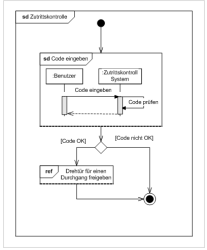
Interaction diagrams

Interaction diagrams, a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled:

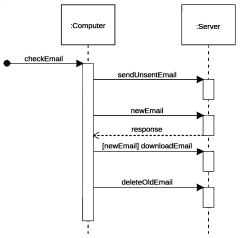
- Communication diagram: shows the interactions between objects or parts in terms of sequenced messages. They represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.
- Interaction overview diagram: provides an overview in which the nodes represent communication diagrams.
- Sequence diagram: shows how objects communicate with each other in terms of a sequence of messages. Also indicates the lifespans of objects relative to those messages.
- Timing diagrams: a specific type of interaction diagram where the focus is on timing constraints.



Communication diagram



Interaction overview diagram



Sequence diagram

The Protocol State Machine is a sub-variant of the State Machine. It may be used to model network communication protocols.

Meta modeling

The Object Management Group (OMG) has developed a metamodeling architecture to define the Unified Modeling Language (UML), called the Meta-Object Facility (MOF).[10] The Meta-Object Facility is designed as a four-layered architecture, as shown in the image at right. It provides a meta-meta model at the top layer, called the M3 layer. This M3-model is the language used by Meta-Object Facility to build metamodels, called M2-models.

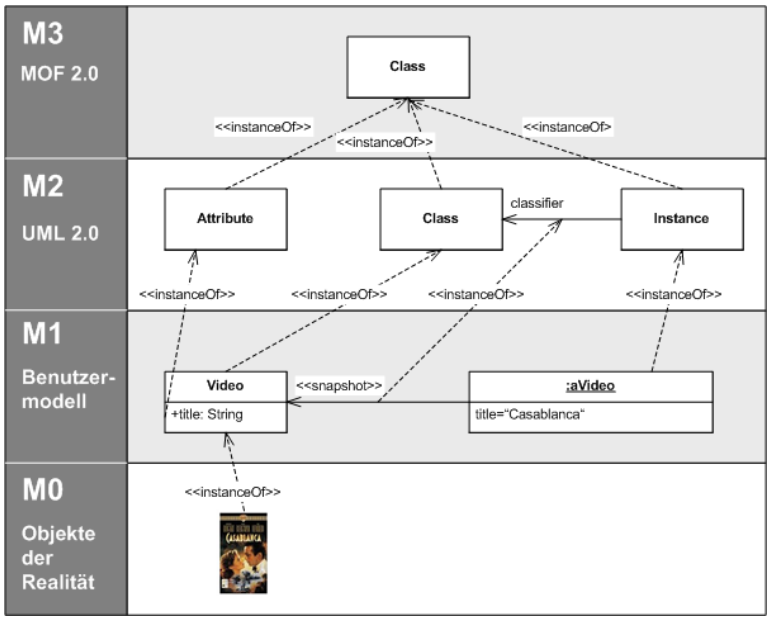


Illustration of the Meta-Object Facility.

The most prominent example of a Layer 2 Meta-Object Facility model is the UML metamodel, the model that describes the UML itself. These M2-models describe elements of the M1-layer, and thus M1-models. These would be, for example, models written in UML. The last layer is the M0-layer or data layer. It is used to describe runtime instance of the system.

Beyond the M3-model, the Meta-Object Facility describes the means to create and manipulate models and metamodels by defining Common Object Request Broker Architecture (CORBA) interfaces that describe those operations. Because of the similarities between the Meta-Object Facility M0-model and UML structure models, Meta-Object Facility metamodels are usually modeled as UML class diagrams. A supporting standard of the Meta-Object Facility is XMI, which defines an XML-based exchange format for models on the M3-, M2-, or M1-Layer.

Criticisms

Although UML is a widely recognized and used modeling standard, it is frequently[*citation needed*] criticized for the following:

Standards bloat

> Bertrand Meyer, in a satirical essay framed as a student's request for a grade change, apparently criticized UML as of 1997 for being unrelated to object-oriented software development; a disclaimer was added later pointing out that his company nevertheless supports UML. Ivar Jacobson, a co-architect of UML, said that objections to UML 2.0's size were valid enough to consider the application of intelligent agents to the problem.[11] It contains many diagrams and constructs that are redundant or infrequently used.

Problems in learning and adopting

> The problems cited in this section make learning and adopting UML problematic, especially when required of engineers lacking the prerequisite skills.[12] In practice, people often draw diagrams with the symbols provided by their CASE tool, but without the meanings those symbols are intended to provide. Simple user narratives e.g. "what I do at work ..." have shown to be much simpler to record and more immediately useful.[*citation needed*]

Linguistic incoherence

The standards have been cited as being ambiguous and inconsistent.[13] The UML 2.0 standard still suffers many issues.

Capabilities of UML and implementation language mismatch

Typical of other notational systems, UML is able to represent some systems more concisely or efficiently than others. Thus a developer gravitates toward solutions that reside at the intersection of the capabilities of UML and the implementation language. This problem is particularly pronounced if the implementation language does not adhere to orthodox object-oriented doctrine, since the intersection set between UML and implementation language may be that much smaller or equal in size.[*citation needed*]

Dysfunctional interchange format

While the XMI (XML Metadata Interchange) standard is designed to facilitate the interchange of UML models, it has been largely ineffective in the practical interchange of UML 2.x models. This interoperability ineffectiveness is attributable to several reasons. Firstly, XMI 2.x is large and complex in its own right, since it purports to address a technical problem more ambitious than exchanging UML 2.x models. In particular, it attempts to provide a mechanism for facilitating the exchange of any arbitrary modeling language defined by the OMG's Meta-Object Facility (MOF). Secondly, the UML 2.x Diagram Interchange specification lacks sufficient detail to facilitate reliable interchange of UML 2.x notations between modeling tools. Since UML is a visual modeling language, this shortcoming is substantial for modelers who don't want to redraw their diagrams. The Diagram Definition OMG project is another alternative.

Cardinality notation

As with database Chen, Bachman, and ISO ER diagrams, class models are specified to use "look-across" cardinalities, even though several authors (Merise,[14] Elmasri & Navathe [15] amongst others [16]) prefer same-side or "look-here" for roles and both minimum and maximum cardinalities. Recent researchers (Feinerer, Dullea et. alia) have shown that the "look-across" technique used by UML and ER diagrams is less effective and less coherent when applied to n-ary relationships of order >2.

In Feinerer it says "Problems arise if we operate under the look-across semantics as used for UML associations. Hartmann investigates this situation and shows how and why different transformations fail." *(Although the "reduction" mentioned is spurious as the two diagrams 3.4 and 3.5 are in fact the same)* and also "As we will see on the next few pages, the look-across interpretation introduces several difficulties which prevent the extension of simple mechanisms from binary to n-ary associations."

Exclusive

The term "unified" applies only to the unification of the many prior existing and competing object-oriented languages. Important well known and popular techniques, almost universally used in industry, such as data flow diagrams and structure charts were not included in the specification.[*citation needed*]

Modeling experts have written criticisms of UML, including Brian Henderson-Sellers and Cesar Gonzalez-Perez in "Uses and Abuses of the Stereotype Mechanism in UML 1.x and 2.0".[17]

Complexity

UML has been criticized for being extremely complex compared to other tools.

References

This article is based on material taken from the Free On-line Dictionary of Computing prior to 1 November 2008 and incorporated under the "relicensing" terms of the GFDL, version 1.3 or later.

[1] Marc Hamilton (1999) *Software Development: A Guide to Building Reliable Systems* p.48

[2] Satish Mishra (1997). "Visual Modeling & Unified Modeling Language (UML) : Introduction to UML" (http://www2.informatik.hu-berlin.de/~hs/Lehre/2004-WS_SWQS/20050107_Ex_UML.ppt). Rational Software Corporation. Accessed 9 November 2008.

[3] Grady Booch, Ivar Jacobson & James Rumbaugh (2000) OMG Unified Modeling Language Specification (http://www.omg.org/docs/formal/00-03-01.pdf), Version 1.3 First Edition: March 2000. Retrieved 12 August 2008.

[4] Andreas Zendler (1997) *Advanced Concepts, Life Cycle Models and Tools for Objeckt-Oriented Software Development*. p.122

[5] Objectory AB, known as Objectory System, was founded in 1987 by Ivar Jacobson. In 1991, It was acquired and became a subsidiary of Ericsson.

[6] John Hunt (2000). *The Unified Process for Practitioners: Object-oriented Design, UML and Java*. Springer, 2000. ISBN 1-85233-275-1. p.5.door

[7] Jon Holt Institution of Electrical Engineers (2004). *UML for Systems Engineering: Watching the Wheels* IET, 2004, ISBN 0-86341-354-4. p.58

[8] http://en.wikipedia.org/w/index.php?title=Template:UML_diagram_types&action=edit

[9] *UML Superstructure Specification Version 2.2*. OMG, February 2009.

[10] Iman Poernomo (2006) " The Meta-Object Facility Typed (http://calcium.dcs.kcl.ac.uk/1259/1/acm-paper.pdf)" in: *Proceeding SAC '06 Proceedings of the 2006 ACM symposium on Applied computing*. pp. 1845-1849

[11] "Ivar Jacobson on UML, MDA, and the future of methodologies" (http://www.infoq.com/interviews/Ivar_Jacobson) (video of interview, transcript available), 24 October 2006. Retrieved 2009-05-22

[12] See the ACM article *"Death by UML Fever"* (http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=130) for an amusing account of such issues.

[13] Génova et alia 2004 "Open Issues in Industrial Use Case Modeling"

[14] Hubert Tardieu, Arnold Rochfeld and René Colletti La methode MERISE: Principes et outils (Paperback - 1983)

[15] Elmasri, Ramez, B. Shamkant, Navathe, Fundamentals of Database Systems, third ed., Addison-Wesley, Menlo Park, CA, USA, 2000.

[16] ER 2004 : 23rd International Conference on Conceptual Modeling, Shanghai, China, 8-12 November 2004 (http://books.google.com/books?id=odZK99osY1EC&pg=PA52&img=1&pgis=1&dq=genova&sig=ACfU3U3tDC_q8WOMqUJW4EZCa5YQywoYLw&edge=0)

[17] B. Henderson-Sellers; C. Gonzalez-Perez (2006). "Uses and Abuses of the Stereotype Mechanism in UML 1.x and 2.0". in: *Model Driven Engineering Languages and Systems*. Springer Berlin / Heidelberg.

Further reading

- Ambler, Scott William (2004). *The Object Primer: Agile Model Driven Development with UML 2* (http://www.ambysoft.com/books/theObjectPrimer.html). Cambridge University Press. ISBN 0-521-54018-6.
- Chonoles, Michael Jesse; James A. Schardt (2003). *UML 2 for Dummies*. Wiley Publishing. ISBN 0-7645-2614-6.
- Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Addison-Wesley. ISBN 0-321-19368-7.
- Jacobson, Ivar; Grady Booch; James Rumbaugh (1998). *The Unified Software Development Process*. Addison Wesley Longman. ISBN 0-201-57169-2.
- Martin, Robert Cecil (2003). *UML for Java Programmers*. Prentice Hall. ISBN 0-13-142848-9.
- Noran, Ovidiu S. "Business Modelling: UML vs. IDEF" (http://www.cit.gu.edu.au/~noran/Docs/UMLvsIDEF.pdf) (PDF). Retrieved 2005-12-28.
- Penker, Magnus; Hans-Erik Eriksson (2000). *Business Modeling with UML*. John Wiley & Sons. ISBN 0-471-29551-5.

External links

- UML Resource Page (http://www.uml.org/) of the Object Management Group – Resources that include the latest version of the UML specification from the group in charge of defining the UML specification
- Death by UML Fever (http://queue.acm.org/detail.cfm?id=984495) – An ACM queue article about the abuse of UML
- Understanding the Unified Modeling Language (UML) (http://www.methodsandtools.com/archive/archive.php?id=76) – Introductory article for UML. Retrieved 2011-01-29
- UML 2.5 resource site (http://www.uml-diagrams.org/) - Includes latest notation documentation.
- ITU-T standard UML profile (http://www.itu.int/rec/T-REC-Z.109/en)

The Third Manifesto

The Third Manifesto (1995) is Christopher J. Date's and Hugh Darwen's proposal for future database management systems, a response to two earlier Manifestos with the same purpose. The theme of the manifestos is how to avoid the 'object-relational impedance mismatch' between object-oriented programming languages and relational database management systems. The Third Manifesto proposes to maintain the relational model for databases and to support objects as user-defined types.

A major theme of the manifesto is to explain how the inadequacies of existing relational database management systems are not shortcomings of the relational database model *per se*, but rather, of implementation decisions in those systems, and of the SQL query language that most of these systems use.

The manifesto describes an alternative to SQL, named D. D is a specification of desirable characteristics of a database language, rather than a specific syntax or grammar. As such, it describes a family of languages rather than any particular language. However, as an example, a particular member of the hypothetical D "family" called Tutorial D is described in detail, including significant portions of its grammar.

Several partial implementations of D exist, including:

- Alphora Dataphor, an open source product which implements D atop SQL databases.
- Rel, an Open Source implementation of Tutorial D in Java.
- Muldis Rosetta (Muldis D), an Open Source implementation in Perl

Bibliography

- Darwen, Hugh; Date, C. J. (March 1995). "The third manifesto" [1] (PostScript). *ACM SIGMOD Record* (New York, NY, USA: ACM Press) **24** (1): 39–49. doi:10.1145/202660.202667 [2]. ISSN 0163-5808 [3].
- Date, C. J. (August 1998). "Preview of The Third Manifesto" [4]. *Database Programming & Design* (San Francisco, CA: Miller Freeman Publications) **11** (8): 67. ISSN 0895-4518 [5]. OCLC 89297479 [6]. Retrieved 2007-06-18.
- Date, C. J.; Darwen, Hugh (1998). *Foundation for object/relational databases: the third manifesto: a detailed study of the impact of objects and type theory on the relational model of data including a comprehensive proposal for type inheritance* (1st ed.). Reading, MA: Addison-Wesley. xxi, 496. ISBN 0-201-30978-5. LCCN 9810364 [7]. OCLC 38431501 [8]. LCC QA76.9.D3 D15994 1998 [9].
- Date, C. J.; Darwen, Hugh (2000). *Foundation for future database systems: the third manifesto: a detailed study of the impact of type theory on the relational model of data, including a comprehensive model of type inheritance* (2nd ed.). Reading, MA: Addison-Wesley Professional. xxiii, 547. ISBN 0-201-70928-7. LCCN 0035527 [10]. OCLC 43662285 [11]. LCC QA76.9.D3 D3683 2000 [12].
- Date, C. J.; Darwen, Hugh (2006). *Databases, types and the relational model: the third manifesto* (3rd ed.). Reading, MA: Addison-Wesley. p. 572. ISBN 0-321-39942-0. OCLC 70044091 [13].

External links

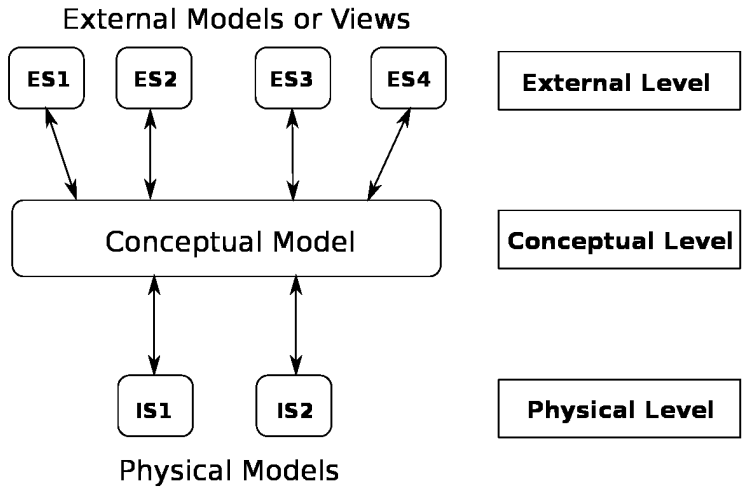
- Official website [14] - including errata, related materials, and a PDF version of The Third Manifesto.
- PDF version [15] of the February 7, 2013 version of The Third Manifesto

References

[1] http://acm.org/sigmod/record/issues/9503/manifesto.ps
[2] http://dx.doi.org/10.1145%2F202660.202667
[3] http://www.worldcat.org/issn/0163-5808
[4] http://www.dbpd.com/vault/9808date.html
[5] http://www.worldcat.org/issn/0895-4518
[6] http://www.worldcat.org/oclc/89297479
[7] http://lccn.loc.gov/9810364
[8] http://www.worldcat.org/oclc/38431501
[9] http://catalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=QA76.9.D3+D15994+1998&Search_Code=CALL_&CNT=5
[10] http://lccn.loc.gov/0035527
[11] http://www.worldcat.org/oclc/43662285
[12] http://catalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=QA76.9.D3+D3683+2000&Search_Code=CALL_&CNT=5
[13] http://www.worldcat.org/oclc/70044091
[14] http://www.thethirdmanifesto.com/
[15] http://www.dcs.warwick.ac.uk/~hugh/TTM/TTM-2013-02-07.pdf

Three schema approach

The **three-schema approach**, or the *Three Schema Concept*, in software engineering is an approach to building information systems and systems information management from the 1970s. It proposes to use three different views in systems development, in which conceptual modelling is considered to be the key to achieving data integration.[2]



The notion of a three-schema model was first introduced in 1975 by the ANSI/X3/SPARC three level architecture, which determined three levels to model data.[1]

Overview

The three-schema approach offers three types of schemas with schema techniques based on formal language descriptions:[3]

- External schema for user views
- Conceptual schema integrates external schemata
- Internal schema that defines physical storage structures

At the center, the conceptual schema defines the ontology of the concepts as the users think of them and talk about them. The physical schema according to Sowa (2004) "describes the internal formats of the data stored in the database, and the external schema defines the view of the data presented to the application programs".[4] The framework attempted to permit multiple data models to be used for external schemata.[5]

Over the years, the skill and interest in building information systems has grown tremendously. However, for the most part, the traditional approach to building systems has only focused on defining data from two distinct views, the "user view" and the "computer view". From the user view, which will be referred to as the “external schema,” the definition of data is in the context of reports and screens designed to aid individuals in doing their specific jobs. The required structure of data from a usage view changes with the business environment and the individual preferences of the user. From the computer view, which will be referred to as the “internal schema,” data is defined in terms of file structures for storage and retrieval. The required structure of data for computer storage depends upon the specific computer technology employed and the need for efficient processing of data.[6]

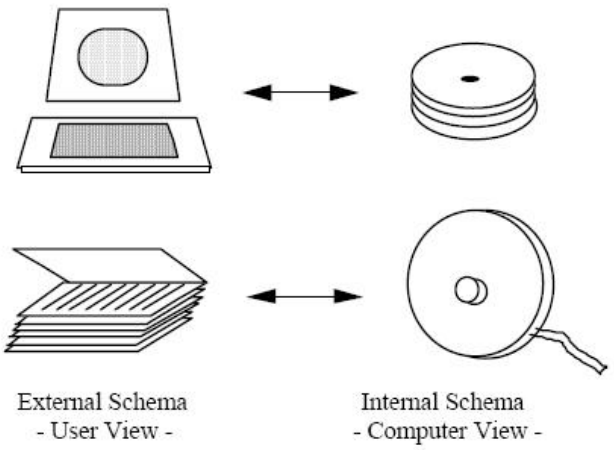


Figure 1: Traditional View of Data

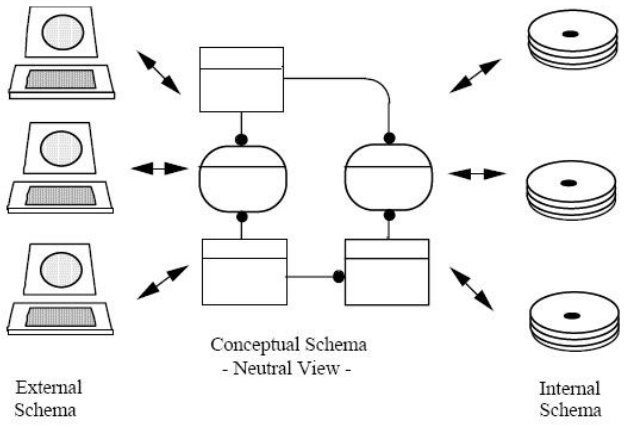


Figure 2: Three schema approach

These two traditional views of data have been defined by analysts over the years on an application by application basis as specific business needs were addressed, see Figure 1. Typically, the internal schema defined for an initial application cannot be readily used for subsequent applications, resulting in the creation of redundant and often inconsistent definition of the same data. Data was defined by the layout of physical records and processed sequentially in early information systems. The need for flexibility, however, led to the introduction of Database Management Systems (DBMSs), which allow for random access of logically connected pieces of data. The logical data structures within a DBMS are typically defined as either hierarchies, networks or relations. Although DBMSs have greatly improved the shareability of data, the use of a DBMS alone does not guarantee a consistent definition of data. Furthermore, most large companies have had to develop multiple databases which are often under the control of different DBMSs and still have the problems of redundancy and inconsistency.

The recognition of this problem led the ANSI/X3/SPARC Study Group on Database Management Systems to conclude that in an ideal data management environment a third view of data is needed. This view, referred to as a “conceptual schema” is a single integrated definition of the data within an enterprise which is unbiased toward any single application of data and is independent of how the data is physically stored or accessed, see Figure 2. The primary objective of this conceptual schema is to provide a consistent definition of the meanings and interrelationship of data which can be used to integrate, share, and manage the integrity of data.

History

The notion of a three-schema model consisting of a conceptual model, an external model, and an internal or physical model was first introduced by the ANSI/X3/SPARC Standards Planning and Requirements Committee directed by Charles Bachman in 1975. The ANSI/X3/SPARC Report characterized DBMSs as having a two schema organization. That is, DBMSs utilize an internal schema, which represents the structure of the data as viewed by the DBMS, and an external schema, which represents various structures of the data as viewed by the end user. The concept of a third schema (conceptual) was introduced in the report. The conceptual schema represents the basic underlying structure of data as viewed by the enterprise as a whole.

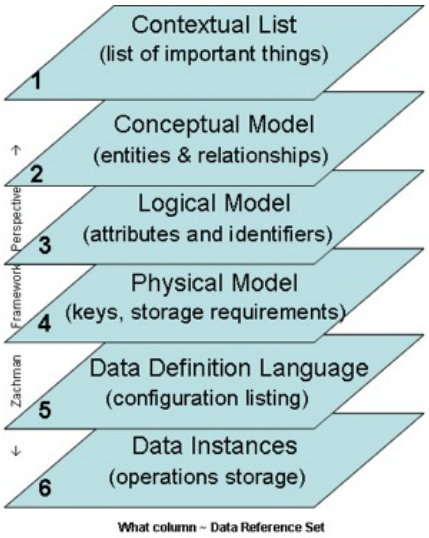


Image of the six layers in the Zachman Framework.

The ANSI/SPARC report was intended as a basis for interoperable computer systems. All database vendors adopted the three-schema terminology, but they implemented it in incompatible ways. Over the next twenty years, various groups attempted to define standards for the conceptual schema and its mappings to databases and programming languages. Unfortunately, none of the vendors had a strong incentive to make their formats compatible with their competitors'. A few reports were produced, but no standards.

As the practice of Data Administration has evolved and more graphical techniques have evolved, the term "schema" has given way to the term "model". The conceptual model represents the view of data that is negotiated between end users and database administrators covering those entities about which it is important to keep data, the meaning of the data, and the relationships of the data to each other.

One further development is the IDEF1X information modeling methodology, which is based on the three-schema concept. Another is the Zachman Framework, proposed by John Zachman in 1987 and developed ever since in the field of Enterprise Architecture. In this framework, the three-schema model has evolved into a layer of six perspectives. In other Enterprise Architecture frameworks some kind of view model is incorporated.

References

This article incorporates public domain material from websites or documents of the National Institute of Standards and Technology.

[1] Matthew West and Julian Fowler (1999). High Quality Data Models (http://www.matthew-west.org.uk/publications/princ03.pdf). The European Process Industries STEP Technical Liaison Executive (EPISTLE).

[2] STRAP SECTION 2 APPROACH (http://www.fas.org/irp/doddir/army/strap/strpsec2.htm). Retrieved 30 September 2008.

[3] Mary E.S. Loomis (1987). *The Database Book*. p. 26.

[4] John F. Sowa (2004). ["The Challenge of Knowledge Soup"]. published in: *Research Trends in Science, Technology and Mathematics Education*. Edited by J. Ramadas & S. Chunawala, Homi Bhabha Centre, Mumbai, 2006.

[5] Gad Ariav & James Clifford (1986). *New Directions for Database Systems: Revised Versions of the Papers*. New York University Graduate School of Business Administration. Center for Research on Information Systems, 1986.

[6] itl.nist.gov (1993) *Integration Definition for Information Modeling (IDEFIX)* (http://www.itl.nist.gov/fipspubs/idef1x.doc). 21 Dec 1993.

External links

- Information Designing (http://www.walden3d.com/w3d_old/papers/lynk_part1_90/sld008.htm) presentation by Walden 3d Inc.
- Example of an application (http://www.nws.noaa.gov/oh/hrl/ihfs/logmodel/ihfsdatamodel.php) at the Office of Hydrologic Development at the US National Weather Service.

White pages schema

A **white pages schema** is a data model, specifically a logical schema, for organizing the data contained in entries in a directory service, database, or application, such as an address book. In a white pages directory, each entry typically represents an individual person that makes use of network resources, such as by receiving email or having an account to log into a system. In some environments, the schema may also include the representation of organizational divisions, roles, groups, and devices. The term is derived from the white pages, the listing of individuals in a telephone directory, typically sorted by the individual's home location (e.g. city) and then by their name.

While many telephone service providers have for decades published a list of their subscribers in a telephone directory, and similarly corporations published a list of their employees in an internal directory, it was not until the rise of electronic mail systems that a requirement for standards for the electronic exchange of subscriber information between different systems appeared.

A white pages schema typically defines, for each real-world object being represented:

- what attributes of that object are to be represented in the entry for that object
- what relationships of that object to other objects are to be represented
- how is the entry to be named in a DIT
- how an entry is to be located by a client searching for it
- how similar entries are to be distinguished
- how are entries to be ordered when displayed in a list

One of the earliest attempts to standardize a white pages schema for electronic mail use was in X.520 and X.521, part of the X.500 specifications, that was derived from the addressing requirements of X.400 and defined a Directory Information Tree that mirrored the international telephone system, with entries representing residential and organizational subscribers. This evolved into the Lightweight Directory Access Protocol standard schema in RFC 2256. One of the most widely deployed white pages schemas used in LDAP for representing individuals in an organizational context is **inetOrgPerson**, defined in RFC 2798, although versions of Active Directory require a different object class, **User**. Many large organizations have also defined their own white pages schemas for their employees or customers, as part of their Identity management architecture. Converting between data bases and directories using different schemas is often the function of a Metadirectory, and data interchange standards such as Common Indexing Protocol.

Some early directory deployments suffered due to poor design choices in their white pages schema, such as:

- attributes used for naming purposes were non-unique in large environments (such as a person's common name)
- attributes used for naming purposes were likely to change (such as surnames)
- attributes were included which could lead to Identity theft, such as a Social security number
- users were required during provisioning to choose attributes which are unique but still memorable to them

Numerous other proposed schemas exist, both as standalone definitions suitable for use with general purpose directories, or as embedded into network protocols.

Examples of other generic white pages schemas include vCard, defined in RFC 2426, and FOAF.

Anchor Modeling

Anchor Modeling is an agile database modeling technique suited for information that change over time both in structure and content. It provides a graphical notation used for conceptual modeling similar to that of entity—relationship modeling, with extensions for working with temporal data. The modeling technique is based around four modeling constructs: the anchor, attribute, tie and knot, each capturing different aspects of the domain being modeled.[1] The resulting models can be translated to physical database designs using formalized rules. When such a translation is done the tables in the relational database will mostly be in the sixth normal form.

Philosophy and history

Anchor Modeling was created in order to take advantage of the benefits from a high degree of normalization while avoiding its drawbacks. Advantages such as being able to non-destructively evolve the model, avoid null values, and keep the information free from redundancies are gained. Performance issues due to extra joins are largely avoided thanks to a feature in modern database engines called 'table elimination'. In order to handle changes in the information content Anchor Modeling emulates aspects of a temporal database in the resulting relational database schema.

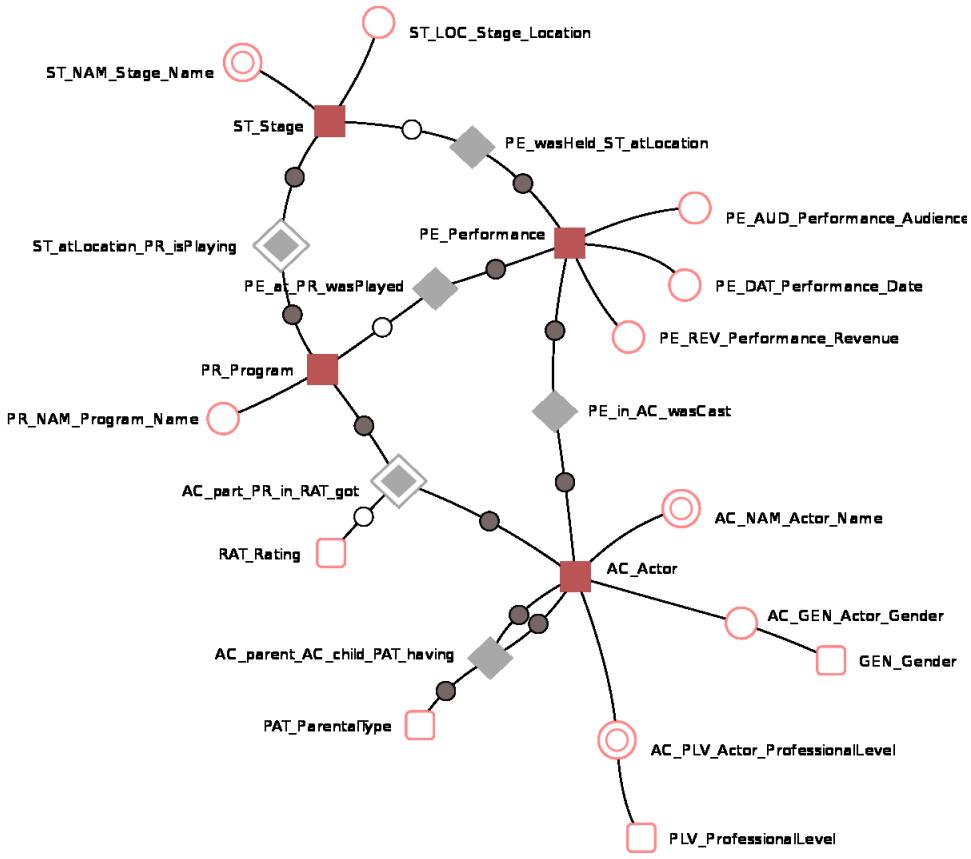
The earliest installations using Anchor Modeling were made in Sweden with the first dating back to 2004, when a data warehouse for an insurance company was built using the technique. In 2007 the technique was being used in a few data warehouses and one OLTP system, and it was presented internationally by Lars Rönnbäck at the TDWI (The Data Warehousing Institute) conference in Amsterdam.[2] This stirred enough interest for the technique to warrant a more formal description. Since then research concerning Anchor Modeling is being done in a collaboration between the creators Olle Regardt and Lars Rönnbäck and a team at the Department of Computer and Systems Sciences, Stockholm University. The first paper,[3] in which Anchor Modeling is formalized, was presented at the 28th International Conference on Conceptual Modeling[4] and won the best paper award.

The research can be followed at www.anchormodeling.com [5], where material on Anchor Modeling is made public and free to use under a Creative Commons license. An online modeling tool is also available, which is free to use and Open Source.

Basic notions

Anchor Modeling has four basic modeling concepts, anchors, attributes, ties, and knots. Anchors are used to model entities and events, attributes are used to model properties of anchors, ties model the relationships between anchors, and knots are used to model shared properties, such as states. Attributes and ties can be historized when changes in the information they model need to be kept.

An example model showing the different graphical symbols for all the concepts can be seen below. The symbols resemble those used in Entity-Relationship modeling, with a couple of extensions. A double outline on an attribute or tie indicates that a history of changes is kept and the knot symbol (an outlined square with rounded edges) is also available.



Temporal aspects

Anchor Modeling handles two types of informational evolution, structural changes and content changes. Changes to the structure of information is represented through extensions. The high degree of normalization makes it possible to non-destructively add the necessary modeling concepts needed to capture a change, in such a way that every previous schema always remains as a subset of the current schema. Since the existing schema is not touched, this gives the benefit of being able to evolve the database in a highly iterative manner and without causing any downtime.

Changes in the content of information is done by emulating similar features of a temporal database in a relational database. In Anchor Modeling, pieces of information can be tied to points in time or to intervals of time (both open and closed). The time points when events occur are modeled using attributes, e g the birth dates of persons or the time of a purchase. The intervals of time in which a value is valid are captured through the historization of attributes and ties, e g the changes of hair color of a person or the period of time during which a person was married. In a relational database this is achieved by adding a single column, with a data type granular enough to capture the speed of the changes, to the table corresponding to the historized attribute or tie. This adds a slight complexity as more than one row in the table have to be examined in order to know if an interval is closed or not.

Points or intervals of time not directly related to the domain being modeled, such as the points of time information entered the database, are handled through the use of metadata in Anchor Modeling, rather than any of the above mentioned constructs. If information about such changes to the database needs to be kept Bitemporal Anchor Modeling can be used, where in addition to updates, also delete statements become non-destructive.

Relational representation

In Anchor Modeling there is a one-to-one mapping between the symbols used in the conceptual model and tables in the relational database. Every anchor, attribute, tie, and knot have a corresponding table in the database with an unambiguously defined structure. A conceptual model can thereby be translated to a relational database schema using simple automated rules, and vice versa. This is different from many other modeling techniques in which there are complex and sometimes subjective translation steps between the conceptual, logical, and physical levels.

Anchor tables contain a single column in which identities are stored. An identity is assumed to be the only property of an entity that is always present and immutable. As identities are rarely available from the domain being modeled, they are instead technically generated, e g from an incrementing number sequence.

An example of an anchor for the identities of the nephews of Donald Duck is a set of 1-tuples:
{⟨#42⟩, ⟨#43⟩, ⟨#44⟩}

Knots can be thought of as the combination of an anchor and a single attribute. Knot tables contain two columns, one for an identity and one for a value. Due to storing identities and values together, knots cannot be historized. Their usefulness comes from being able to reduce storage requirements and improve performance, since tables referencing knots can store a short value rather than a long string.

An example of a knot for genders is a set of 2-tuples:
{⟨#1, 'Male'⟩, ⟨#2, 'Female'⟩}

Static attribute tables contain two columns, one for the identity of the entity to which the value belongs and one for the actual property value. Historized attribute tables have an extra column for storing the starting point of a time interval. In a knotted attribute table, the value column is an identity that references a knot table.

An example of a static attribute for their names is a set of 2-tuples:
{⟨#42, 'Huey'⟩, ⟨#43, 'Dewey'⟩, ⟨#44, 'Louie'⟩}

An example of a knotted static attribute for their genders is a set of 2-tuples:
{⟨#42, #1⟩, ⟨#43, #1⟩, ⟨#44, #1⟩}

An example of a historized attribute for the (changing) colors of their outfits is a set of 3-tuples:
{⟨#44, 'Orange', 1938-04-15⟩, ⟨#44, 'Green', 1939-04-28⟩, ⟨#44, 'Blue', 1940-12-13⟩}

Static tie tables relate two or more anchors to each other, and contain two or more columns for storing the identities. Historized tie tables have an extra column for storing the starting point of a time interval. Knotted tie tables have an additional column for each referenced knot.

An example of a static tie for the sibling relationship is a set of 2-tuples:
{⟨#42, #43⟩, ⟨#42, #44⟩, ⟨#43, #42⟩, ⟨#43, #44⟩, ⟨#44, #42⟩, ⟨#44, #43⟩}

The resulting tables will all be in sixth normal form except for ties in which not all columns are part of the primary key.

References

[1] (Preprint available here (http://api.ning.com/files/O03Gglrgkj7tPERDEbcWXEmDI1VDNzZx9EZc6pikGGG0uewENqy-l-xLFL62LHnpw0WkU7afbudzj2ftuc5AIQ__/AnchorModelingDKEpreprint.pdf))
[2] 6th TDWI European Conference - TDWI homepage (https://www.tdwi.eu/en/events/conferences/tdwi-2007-europe.html?tx_fhconference_pi1_conference_view[show]=online_program&tx_fhconference_pi1_conference_view[session]=39)
[3] (Preprint available here (http://api.ning.com/files/R0UON79E5Kh7ell4QTUVt0Aty-EEblPNcpyaTWiHnLMBXgte-CCu7smzUYIvHtcQwtab90phqS2vOLETydOEHIfEId88ND9q/AnchorModeling09.pdf))
[4] 28th International Conference on Conceptual Modeling - ER'09 homepage (http://www.inf.ufrgs.br/er2009/)
[5] http://www.anchormodeling.com

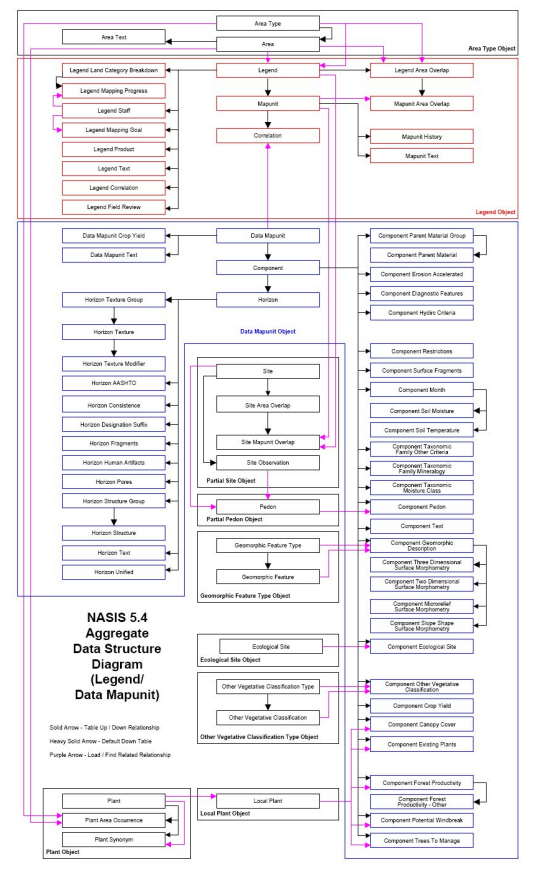
External links

- Anchor Modeling blog, with video tutorials and research information (http://www.anchormodeling.com)
- Online Anchor Modeling tool (http://www.anchormodeling.com/modeler)

Bachman diagram

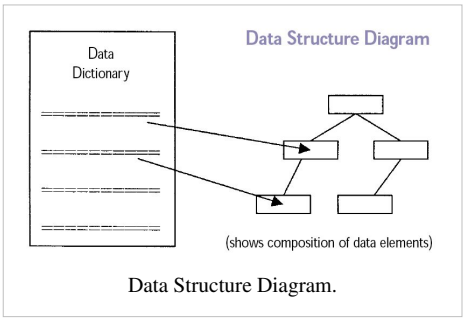
Data Structure Diagram (DSD) is a diagram of the conceptual data model which documents the entities and their relationships, as well as the constraints that connect to them.

The basic graphic notation elements of DSDs are boxes which represent entities.The arrow symbol represents relationships. Data structure diagrams are most useful for documenting complex data entities.



Example of a Data Structure Diagram.

Overview



Data Structure Diagram.

Data Structure Diagram is a diagram type that is used to depict the structure of data elements in the data dictionary. The data structure diagram is a graphical alternative to the composition specifications within such data dictionary entries.[1]

The data structure diagrams is a predecessor of the entity-relationship model (E-R model). In DSDs, attributes are specified inside the entity boxes rather than outside of them, while relationships are drawn as boxes composed of attributes which specify the constraints that bind entities together. DSDs differ from the E-R model in that the E-R model focuses on the relationships between different entities, whereas DSDs focus on the relationships of the elements within an entity.

There are several styles for representing data structure diagrams, with the notable difference in the manner of defining cardinality. The choices are between arrow heads, inverted arrow heads (crow's feet), or numerical representation of the cardinality.

Bachman diagram

A Bachman diagram is a certain type of data structure diagram,[2] and is used to design the data with a network or relational "logical" model, separating the data model from the way the data is stored in the system. The model is named after database pioneer Charles Bachman, and mostly used in computer software design.

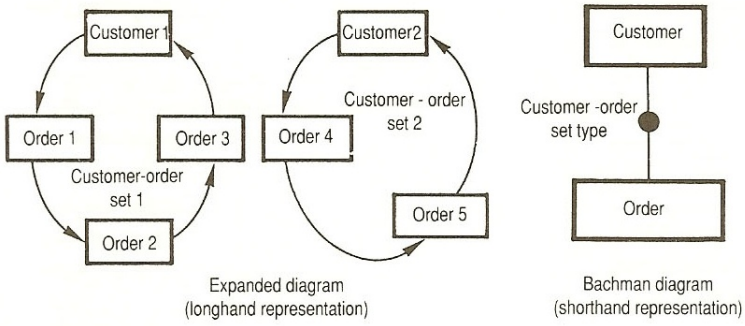


Illustration of set type using a Bachman diagram.

In a relational model, a relation is the cohesion of attributes that are fully and not transitive functional dependent[*clarify*] of every key in that relation. The coupling between the relations is based on accordant attributes. For every relation, a rectangle has to be drawn and every coupling is illustrated by a line that connects the relations. On the edge of each line, arrows indicate the cardinality. We have 1-to-n, 1-to-1 and n-to-n. The latter has to be avoided and must be replaced by two 1-to-n couplings.

References

[1] Data Integration Glossary (http://knowledge.fhwa.dot.gov/tam/aashto.nsf/All+Documents/4825476B2B5C687285256B1F00544258/$FILE/DIGloss.pdf), U.S. Department of Transportation, August 2001.

[2] IRS Resources (http://www.irs.gov/irm/part2/irm_02-005-013.html). Part 2. Information Technology, Chapter 5. Systems Development, Section 13. Database Design Techniques and Deliverables. Retrieved 2 July 2009.

Further reading

- Charles W. Bachman. *Data structure diagrams*. Data Base, 1969, 1(2):4–10.
- Tom DeMarco. *Structured Analysis and System Specification*. ISBN 0-13-854380-1. Prentice Hall. 11 May 1979.
- Edward Yourdon. *Modern Structured Analysis*. ISBN 0-13-598624-9. Prentice Hall. 1 August 1988; now available as the Structured Analysis Wiki (http://www.yourdon.com/strucanalysis/wiki/index.php?title=Introduction).

Bitemporal Modeling

Bitemporal Modeling is an information modeling technique designed to handle historical data along two different timelines. This makes it possible to rewind the information to "as it actually was" in combination with "as it was recorded" at some point in time. In order to able do so, information cannot be discarded even if it is erroneous. Within, for example, financial reporting it is often desirable to be able to recreate an old report both as it actually looked at the time of creation and as it should have looked given corrections made to the data after its creation.

Implementations of Bitemporal Modeling are mostly done using relational databases. As such, Bitemporal Modeling is considered different from Dimensional Modeling and complementary to database normalization. The upcoming SQL standard SQL:2011 will provide language constructs for working with bitemporal data. As no standard is yet in place, current solutions are vendor specific.

Philosophy

Bitemporal modeling uses bitemporal structures as the basic components. This results in the databases which have a consistent type of temporality for all data.

Benefits of Bitemporal Modeling

By focusing on completness and accuracy of data Bitemporal Modeling facilitates the creation of complete audit trails of data. Specifically this allows for queries which provide:

1. The most accurate data possible as we know it now
2. Data as we knew it at any point in time
3. When and why the most accurate data we had changed

Bitemporal data

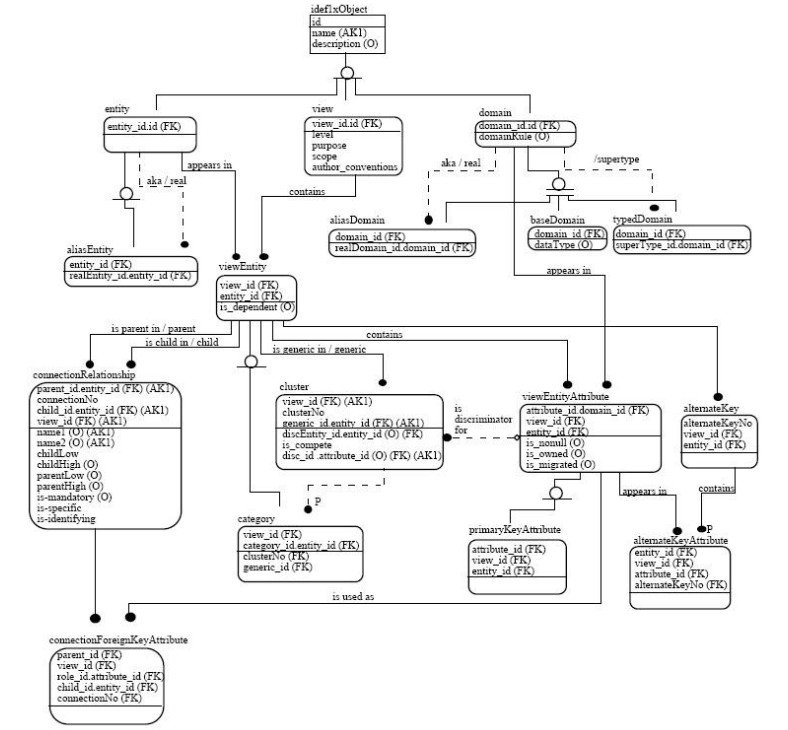
Bitemporal data is a concept used in a temporal database. It denotes both the valid time and transaction time of the data.

In a database table bitemporal data is often represented by four extra table-columns StartVT and EndVT, StartTT and EndTT. Normally each time interval is closed at its lower bound, and open at its upper bound.

IDEF1X

Integration DEFinition for Information Modeling (IDEF1X) is a data modeling language for the development of semantic data models. IDEF1X is used to produce a graphical information model which represents the structure and semantics of information within an environment or system.[1]

IDEF1X permits the construction of semantic data models which may serve to support the management of data as a resource, the integration of information systems, and the building of computer databases. This standard is part of the IDEF family of modeling languages in the field of software engineering.



Example of an IDEF1X Diagram.

Overview

A data modeling technique is used to model data in a standard, consistent and predictable manner in order to manage it as a resource. It can be used in projects requiring a standard means of defining and analyzing the data resources within an organization. Such projects include the incorporation of a data modeling technique into a methodology, managing data as a resource, integrating information systems, or designing computer databases. The primary objectives of the IDEF1X standard are to provide:

- Means for completely understanding and analyzing an organization's data resources
- Common means of representing and communicating the complexity of data
- A technique for presenting an overall view of the data required to run an enterprise
- Means for defining an application-independent view of data which can be validated by users and transformed into a physical database design
- A technique for deriving an integrated data definition from existing data resources.

A principal objective of IDEF1X is to support integration. The approach to integration focuses on the capture, management, and use of a single semantic definition of the data resource referred to as a “Conceptual schema.” The “conceptual schema” provides a single integrated definition of the data within an enterprise which is not biased toward any single application of data and is independent of how the data is physically stored or accessed. The

primary objective of this conceptual schema is to provide a consistent definition of the meanings of and interrelationships between data that can be used to integrate, share, and manage the integrity of data. A conceptual schema must have three important characteristics:

- Consistent with the infrastructure of the business and true across all application areas
- Extendible, such that new data can be defined without altering previously defined data
- Transformable to both the required user views and to a variety of data storage and access structures.

History

The need for semantic data models was first recognized by the U.S. Air Force in the mid-1970s as a result of the Integrated Computer Aided Manufacturing (ICAM) Program. The objective of this program was to increase manufacturing productivity through the systematic application of computer technology. The ICAM Program identified a need for better analysis and communication techniques for people involved in improving manufacturing productivity. As a result, the ICAM Program developed a series of techniques known as the IDEF (ICAM Definition) Methods which included the following:

- IDEF0 used to produce a “function model” which is a structured representation of the activities or processes within the environment or system
- IDEF1 used to produce an “information model” which represents the structure and semantics of information within the environment or system
- IDEF2 used to produce a “dynamics model”.

The initial approach to IDEF information modeling (IDEF1) was published by the ICAM program in 1981, based on current research and industry needs. The theoretical roots for this approach stemmed from the early work of Edgar F. Codd on relational theory and Peter Chen on the entity-relationship model. The initial IDEF1 technique was based on the work of Dr R. R. Brown and Mr T. L. Ramey of Hughes Aircraft and Mr D. S. Coleman of D. Appleton Company (DACOM), with critical review and influence by Charles Bachman, Peter Chen, Dr M. A. Melkanoff, and Dr G.M. Nijssen.

In 1983, the U.S. Air Force initiated the Integrated Information Support System (I2S2) project under the ICAM program. The objective of this project was to provide the enabling technology to logically and physically integrate a network of heterogeneous computer hardware and software. As a result of this project, and industry experience, the need for an enhanced technique for information modeling was recognized.

From the point of view of the contract administrators of the Air Force IDEF program, IDEF1X was a result of the ICAM IISS-6201 project and was further extended by the IISS-6202 project. To satisfy the data modeling enhancement requirements that were identified in the IISS-6202 project, a sub-contractor, DACOM, obtained a license to the Logical Database Design Technique (LDDT) and its supporting software (ADAM). From the point of view of the technical content of the modeling technique, IDEF1X is a renaming of LDDT.

Logical Database Design Technique

The Logical Database Design Technique (LDDT) had been developed in 1982 by Robert G. Brown of The Database Design Group entirely outside the IDEF program and with no knowledge of IDEF1. Nevertheless, the central goal of IDEF1 and LDDT was the same: to produce a database-neutral model of the persistent information needed by an enterprise by modeling the real-world entities involved. LDDT combined elements of the relational data model, the E-R model, and data generalization in a way specifically intended to support data modeling and the transformation of the data models into database designs.

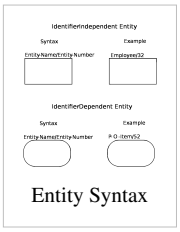
LDDT included an environmental (namespace) hierarchy, multiple levels of model, the modeling of generalization/specialization, and the explicit representation of relationships by primary and foreign keys, supported by a well defined role naming facility. The primary keys and unambiguously role-named foreign keys expressed

sometimes subtle uniqueness and referential integrity constraints that needed to be known and honored by whatever type of database was ultimately designed. Whether the database design used the integrity constraint based keys of the LDDT model as database access keys or indexes was an entirely separate decision. The precision and completeness of the LDDT models was an important factor in enabling the relatively smooth transformation of the models into database designs. Early LDDT models were transformed into database designs for IBM's hierarchical database, IMS. Later models were transformed into database designs for Cullinet's network database, IDMS, and many varieties of relational database.

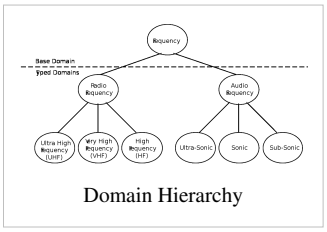
The LDDT software, ADAM, supported view (model) entry, view merging, selective (subset) viewing, namespace inheritance, normalization, a quality assurance analysis of views, entity relationship graph and report generation, transformation to a relational database expressed as SQL data declaration statements, and referential integrity checking SQL. Logical models were serialized with a structural modeling language.

The graphic syntax of LDDT differed from that of IDEF1 and, more importantly, LDDT contained many interrelated modeling concepts not present in IDEF1. Therefore, instead of extending IDEF1, Mary E. Loomis of DACOM wrote a concise summary of the syntax and semantics of a substantial subset of LDDT, using terminology compatible with IDEF1 wherever possible. DACOM labeled the result IDEF1X and supplied it to the ICAM program, which published it in 1985. (IEEE 1998, p. iii) (Bruce 1992, p. xii) DACOM also converted the ADAM software to C and sold it under the name Leverage.

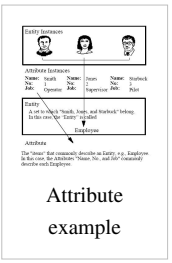
IDEF1X Building blocks



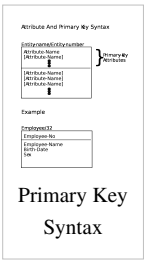
Entity Syntax



Domain Hierarchy



Attribute example



Primary Key Syntax

Entities

The representation of a class of real or abstract things (people, objects, places, events, ideas, combination of things, etc.) that are recognized as instances of the same class because they share the same characteristics and can participate in the same relationships.

Domains

A named set of data values (fixed, or possibly infinite in number) all of the same data type, upon which the actual value for an attribute instance is drawn. Every attribute must be defined on exactly one underlying domain. Multiple attributes may be based on the same underlying domain.

Attributes

A property or characteristic that is common to some or all of the instances of an entity. An attribute represents the use of a domain in the context of an entity.

Keys

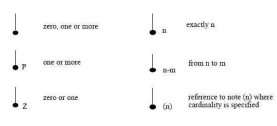
An attribute, or combination of attributes, of an entity whose values uniquely identify each entity instance. Each such set constitutes a candidate key.

Primary Keys

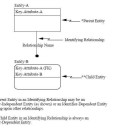
The candidate key selected as the unique identifier of an entity.

Foreign Keys

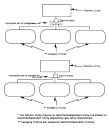
An attribute, or combination of attributes of a child or category entity instance whose values match those in the primary key of a related parent or generic entity instance. A foreign key can be viewed as the result of the "migration" of the primary key of the parent or generic entity through a specific connection or categorization relationship. An attribute or combination of attributes in the foreign key can be assigned a role name reflecting its role in the child or category entity.



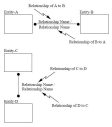
Relationship Cardinality Syntax



Identifying Relationship Syntax



Categorization Relationship Syntax



Non-Specific Relationship Syntax

Relationships

An association between the instances of two entities or between instances of the same entity.

Connection Relationships

A relationship having no semantics in addition to association. See Constraint, Cardinality.

Categorization Relationships

A relationship in which instances of both entities represent the same real or abstract thing. One entity (generic entity) represents the complete set of things, the other (category entity) represents a sub-type or sub-classification of those things. The category entity may have one or more characteristics, or a relationship with instances of another entity, not shared by all generic entity instances. Each instance of the category entity is simultaneously an instance of the generic entity.

Non-Specific Relationships

A relationship in which an instance of either entity can be related to any number of instances of the other.

View Levels

Three levels of view are defined in IDEF1X: Entity Relationship (ER), Key Based (KB), and Fully Attributed (FA). They differ in level of abstraction. The ER level is the most abstract. It models the most fundamental elements of the subject area - the entities and their relationships. It is usually broader in scope than the other levels. The KB level adds keys and the FA level adds all the attributes.

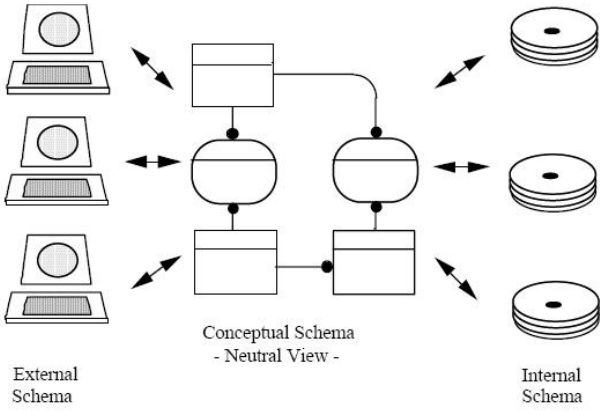
IDEF1X Topics

The Three Schema Approach

The three-schema approach in software engineering is an approach to building information systems and systems information management, that promotes the conceptual model as the key to achieving data integration.[3]

A schema is a model, usually depicted by a diagram and sometimes accompanied by a language description. The three schemas used in this approach are:[4]

- External schema for user views
- Conceptual schema integrates external schemata
- Internal schema that defines physical storage structures.



The three schema approach .[2]

At the center, the conceptual schema defines the ontology of the concepts as the users think of them and talk about them. The physical schema describes the internal formats of the data stored in the database, and the external schema defines the view of the data presented to the application programs.[5] The framework attempted to permit multiple data models to be used for external schemata.[6]

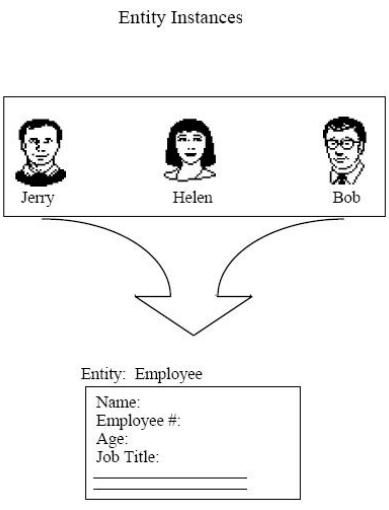
Modeling Guidelines

The modeling process can be divided into five stages of model developing.

Phase Zero – Project Initiation

The objectives of the Project Initiation phase include:

- Project definition – a general statement of what has to be done, why, and how it will get done
- Source material – a plan for the acquisition of source material, including indexing and filing
- Author conventions – a fundamental declaration of the conventions (optional methods) by which the author chooses to make and manage the model.



Synthesizing an Entity in Phase One – Entity Definition

Phase One – Entity Definition

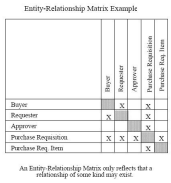
The objective of the Entity Definition phase is to identify and define the entities that fall within the problem domain being modeled.

Phase Two – Relationship Definition

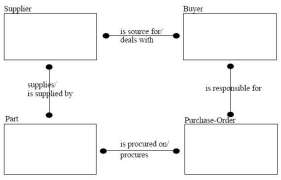
The objective of the Relationship Definition phase is to identify and define the basic relationships between entities. At this stage of modeling, some relationships may be non-specific and will require additional refinement in subsequent phases. The primary outputs from Phase Two are:

- Relationship matrix

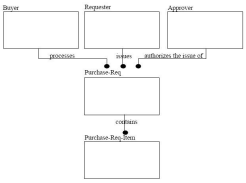
- Relationship definitions
- Entity-level diagrams.



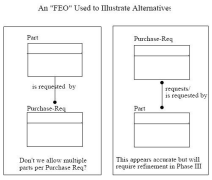
Entity Relationship Matrix



Entity Level Diagram



Entity Level Diagram Example

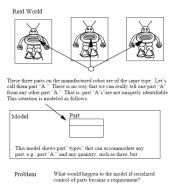


Reference Diagram

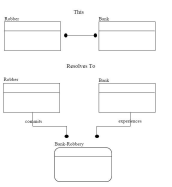
Phase Three - Key Definitions

The objectives of the Key Definitions phase are to:

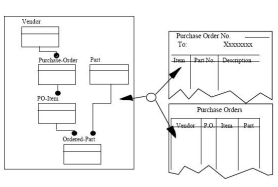
- Refine the non-specific relationships from Phase Two
- Define key attributes for each entity
- Migrate primary keys to establish foreign keys
- Validate relationships and keys.



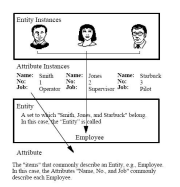
Example Reference Diagram



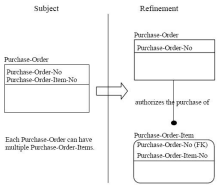
Non-Specific Relationship Refinement



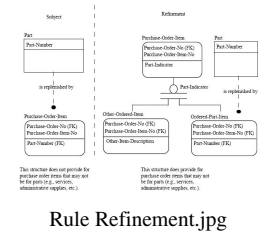
Scope of a Function View



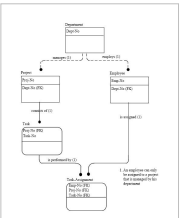
Attribute Examples



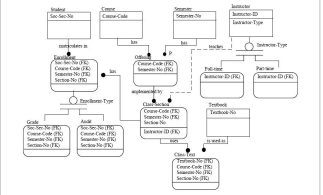
No-Repeat Rule Refinement



Rule Refinement.jpg



Path Assertions

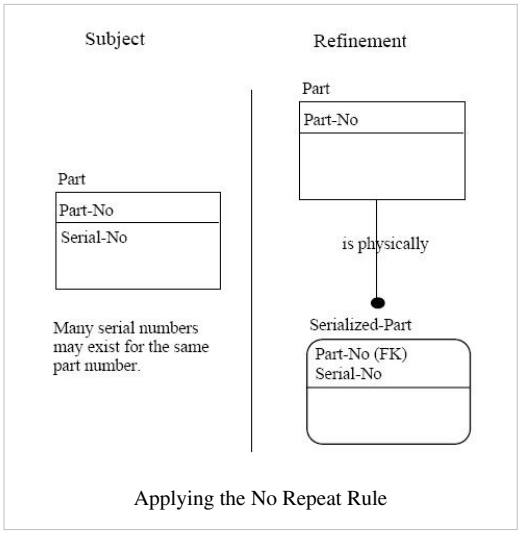


Example of Phase Three Function View Diagram

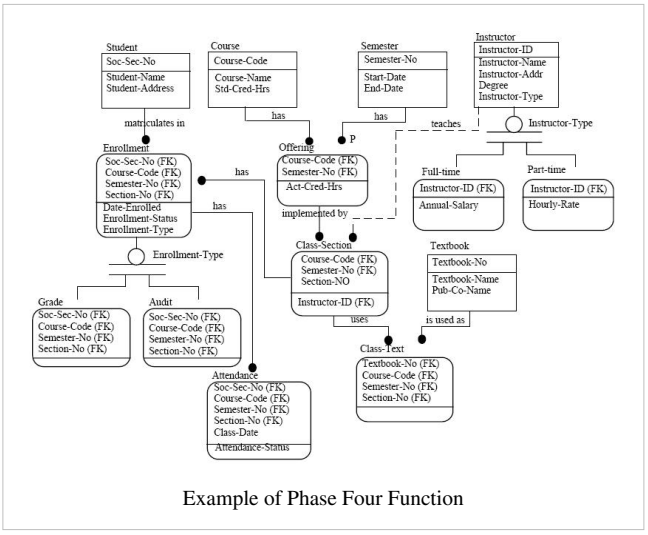
Phase Four - Attribute Definition

The objectives of the Attribute Definition phase are to:

- Develop an attribute pool
- Establish attribute ownership
- Define nonkey attributes
- Validate and refine the data structure.



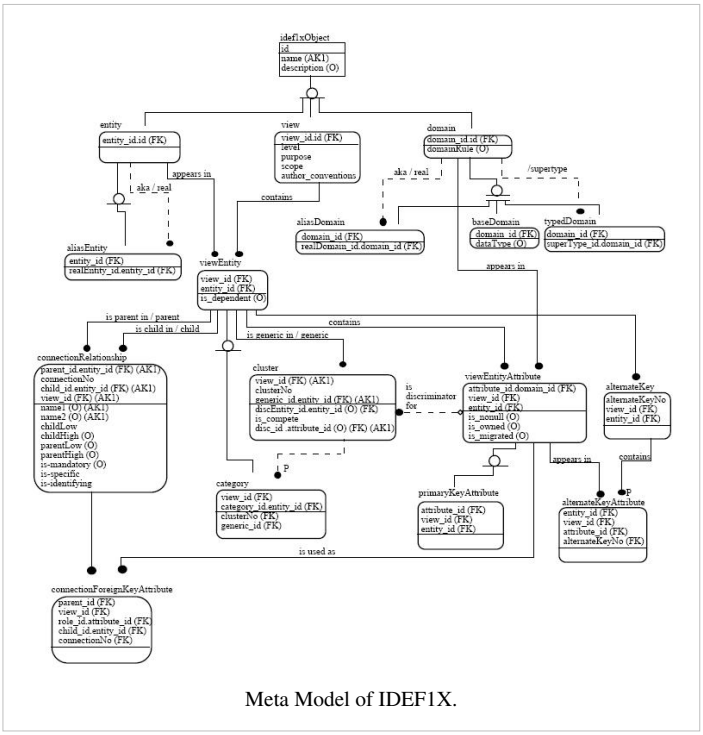
Applying the No Repeat Rule



Example of Phase Four Function

IDEF1X Meta Model

A meta model is a model of the constructs of a modeling system. Like any model, it is used to represent and reason about the subject of the model - in this case IDEF1X. The meta model is used to reason about IDEF1X, i.e., what the constructs of IDEF1X are and how they relate to one another. The model shown is an IDEF1X model of IDEF1X. Such meta models can be used for various purposes, such as repository design, tool design, or in order to specify the set of valid IDEF1X models. Depending on the purpose, somewhat different models result. There is no “one right model.” For example, a model for a tool that supports building models incrementally must allow incomplete or even inconsistent models. The meta model for formalization, however, emphasizes alignment with the concepts of the formalization and hence incomplete or inconsistent models are not allowed.



Meta Model of IDEF1X.

Meta models have two important limitations. First, they specify syntax but not semantics. Second, a meta model must be supplemented with constraints in natural or formal language. The formal theory of IDEF1X provides both the semantics and a means to precisely express the necessary constraints.

A meta model for IDEF1X is given in the adjacent figure. The name of the view is *mm*. The domain hierarchy and constraints are also given. The constraints are expressed as sentences in the formal theory of the meta model. The meta model informally defines the set of valid IDEF1X models in the usual way, as the sample instance tables that correspond to a valid IDEF1X model. The meta model also formally defines the set of valid IDEF1X models in the following way. The meta model, as an IDEF1X model, has a corresponding formal theory. The semantics of the theory are defined in the standard way. That is, an interpretation of a theory consists of a domain of individuals and a set of assignments:

- To each constant in the theory, an individual in the domain is assigned
- To each n-ary function symbol in the theory, an n-ary function over the domain is assigned
- To each n-ary predicate symbol in the theory, an n-ary relation over the domain is assigned.

In the intended interpretation, the domain of individuals consists of views, such as production; entities, such as part and vendor; domains, such as qty_on_hand; connection relationships; category clusters; and so on. If every axiom in the theory is true in the interpretation, then the interpretation is called a model for the theory. Every model for the IDEF1X theory corresponding to the IDEF1X meta model and its constraints is a valid IDEF1X model.

References

This article incorporates public domain material from websites or documents of the National Institute of Standards and Technology.

[1] FIPS Publication 184 (http://www.itl.nist.gov/fipspubs/idef1x.doc) released of IDEF1X by the Computer Systems Laboratory of the National Institute of Standards and Technology (NIST). 21 December 1993.

[2] itl.nist.gov (1993) *Integration Definition for Information Modeling (IDEFIX)* (http://www.itl.nist.gov/fipspubs/idef1x.doc). 21 Dec 1993.

[3] STRAP SECTION 2 APPROACH (http://www.fas.org/irp/doddir/army/strap/strpsec2.htm). Retrieved 30 September 2008.

[4] Mary E.S. Loomis (1987). *The Database Book*. p. 26.

[5] John F. Sowa (2004). ["The Challenge of Knowledge Soup"]. published in: *Research Trends in Science, Technology and Mathematics Education*. Edited by J. Ramadas & S. Chunawala, Homi Bhabha Centre, Mumbai, 2006.

[6] Gad Ariav & James Clifford (1986). *New Directions for Database Systems: Revised Versions of the Papers*. New York University Graduate School of Business Administration. Center for Research on Information Systems, 1986.

Further reading

- Thomas A. Bruce (1992). *Designing Quality Databases With Idef1X Information Models*. Dorset House Publishing.
- Y. Tina Lee & Shigeki Umeda (2000). "An IDEF1x Information Model for a Supply Chain Simulation" (http://www.mel.nist.gov/msidlibrary/doc/idefx.pdf).
- U.S. Department of Interior (2005). "Data Reference Model Overview" (http://www.doi.gov/ocio/architecture/drmsg/doi_drm_overview_5_2005.doc). May 4, 2005.

External links

- FIPS Publication 184 (http://www.itl.nist.gov/fipspubs/idef1x.doc) Announcing the IDEF1X Standard December 1993 by the Computer Systems Laboratory of the National Institute of Standards and Technology (NIST). (Withdrawn by NIST 08 Sep 02 see Withdrawn FIPS by Numerical Order Index (http://www.itl.nist.gov/fipspubs/withdraw.htm))
- Overview of IDEF1X (http://www.idef.com/idef1X.htm) at www.idef.com
- IDEF1X (http://www.essentialstrategies.com/publications/modeling/idef1x.htm) Overview from Essential Strategies, Inc.

Universal Data Element Framework

The **Universal Data Element Framework** (**UDEF**) provides the foundation for building an enterprise-wide controlled vocabulary. It is a standard way of indexing enterprise information that can produce big cost savings. UDEF simplifies information management through consistent classification and assignment of a global standard identifier to the data names and then relating them to similar data element concepts defined by other organizations. Though this approach is a small part of the overall picture, it is potentially a crucial enabler of semantic interoperability.

How UDEF works

UDEF provides semantic links, through assigning an intelligent, derived ID as an attribute of the data element, essentially labeling the element as a specific data element concept. When this UDEF ID exists in both source and target formats, it can then be used as an easy analysis point via a match report, and then as the primary pivot point for transformations between source and target.

UDEF takes a list of high-level root object classes and assigns an integer to each class plus alpha characters to each specialization modifier. It then also assigns integers to property word plus integers to each specialization modifier. These object class alpha-integers are concatenated together with the property integers to form a dewey-decimal like code for each data element concept.

For the following examples, go to http:// www. opengroup. org/ udefinfo/ htm/ en_defs. htm and expand the applicable UDEF object and property trees

Assuming an application used by a hospital needs to map the data element concepts to the UDEF, the last name and first name (within UDEF you will find Family Name and Given Name under the UDEF property Name) of several people that are likely to appear on a medical record that could include the following example data element concepts –

Patient Person Family Name – find the word “Patient” under the UDEF object “Person” and find the word “Family” under the UDEF property “Name”

Patient Person Given Name – find the word “Patient” under the UDEF object “Person” and find the word “Given” under the UDEF property “Name”

Doctor Person Family Name – find the word “Doctor” under the UDEF object “Person” and find the word “Family” under the UDEF property “Name”

Doctor Person Given Name – find the word “Doctor” under the UDEF object “Person” and find the word “Given” under the UDEF property “Name”

The associated UDEF IDs for the above are derived by walking up each tree respectively and using an underscore to separate the object from the property. For the examples above, the following data element concepts are available within the current UDEF – see http:// www. opengroup. org/ udefinfo/ htm/ en_ob5. htm and http:// www. opengroup.org/udefinfo/htm/en_pr10.htm

“Patient Person Family Name” the UDEF ID is “au.5_11.10”

“Patient Person Given Name” the UDEF ID is “au.5_12.10”

“Doctor Person Family Name” the UDEF ID is “aq.5_11.10”

“Doctor Person Given Name” the UDEF ID is “aq.5_12.10”

Six basic steps to map enterprise data to the UDEF

There are six basic steps to follow when mapping data element concepts to the UDEF.

1. Identify the applicable UDEF property word that characterizes the dominant attribute (property) of the data element concept. For example: Name, Identifier, Date, etc.

2. Identify the dominant UDEF object word that the dominant property (selected in step 1) is describing. For example, Person_Name, Product_Identifier, Document_Date, etc.

3. By reviewing the UDEF tree for the selected property identified in step 1, identify applicable qualifiers that are necessary to describe the property word term unambiguously. For example, Family Name.

4. By reviewing the UDEF tree for the selected object identified in step 2, identify applicable qualifiers that are necessary to describe the object word term unambiguously. For example, Customer Person.

5. Concatenate the object term and the property term to create a UDEF naming convention compliant name where it is recognized that the name may seem artificially long. For example, Customer Person_Family Name.

6. Derive a structured ID based on the UDEF taxonomy that carries the UDEF inherited indexing scheme. For example <CustomerPersonFamilyName UDEFID=”as.5_11.10”>.

The Open Group UDEF Project objectives

The UDEF Project aims to establish the Universal Data Element Framework (UDEF) as the universally-used classification system for data element concepts. It focuses on developing and maintaining the UDEF as an open standard, advocating and promoting it, putting in place a technical infrastructure to support it, implementing a Registry for it, and setting up education programs to train information professionals in its use.

Organizations that implement UDEF will likely realize the greatest benefit by defining their controlled vocabulary based on the UDEF. To help an organization manage its UDEF based controlled vocabulary, it should seriously consider a metadata registry that is based on ISO/IEC 11179-5.

History of UDEF

Ron Schuldt, Sr. Enterprise Data Architect, Lockheed Martin, originated the UDEF concept based on ISO/IEC 11179 Metadata standards in the early 1990s. Currently, he is a Senior Partner with UDEF-IT, LLC.

Ownership of UDEF intellectual property

The Open Group assumed from AFEI the right to grant public use licensing of the UDEF.

The Supplier Management Council Electronic Enterprise Working Group of the Aerospace Industry Association (AIA) supports the UDEF as the naming convention solution to XML interoperability between standards that include all functions throughout a product's life-cycle and is working through a well defined process to obtain approval of this position from AIA and its member companies.

Criticism

Classification in UDEF is not sometimes hampered by ad hoc decisions that might produce problems. Example:

- b.be.5 is "United-Kingdom Citizen Person" and
- c.be.5 is "European Union Citizen Person"

As the United Kingdom is part of the European Union, the classification is not unique. Response: The UDEF is flexible and is designed to match the semantics and behaviour of existing systems. Therefore, if one system has a table for United Kingdom Citizens and a different system has a table for European Union Citizens, the UDEF can handle both situations.

Some of the concepts in UDEF are not as universal as it is claimed. They show a lot of bias to Anglo-American tradition and way of thinking and are not easily transferable to other languages. Example: The following part of the hierarchy shows the concept of an officer.

- j.5 Officer.Person
- a.j.5 Contracting.Officer.Person
- a.a.j.5 Procuring.Contracting.Officer.Person
- a.a.a.j.5 Government.Procuring.Contracting.Officer.Person
- b.a.j.5 Administrative.Contracting.Officer.Person
- b.j.5 Police.Officer.Person
- c.j.5 Military.Officer.Person

In many cultures, the part of the tree below "a.j.5 Contracting Officer Person" would not be placed under j.5 (see officer) as b.j.5 (see Law enforcement officer) or c.j.5 (see Officer (armed forces)).

External links

- UDEF Project of The Open Group [1]
- YouTube - UDEF Tutorial, Part 1 [2]
- YouTube - UDEF Tutorial, Part 2 [3]
- UDEF Frequently Asked Questions [4]
- - Obtain Enhanced UDEF Gap Analysis Tool in English, Dutch, or French [5]

Further reading

- Ronald Schuldt (November 15, 2011). *UDEF - Six Steps to Cost Effective Data Integration* [6]. CreateSpace. ISBN 978-1-4664-6762-0.
- Ronald Schuldt and Roberta Shauger (January 16, 2012). *UDEF - Six Steps to Cost Effective Data Integration* [7]. Amazon Digital Services. ISBN 1-4664-6762-2.
- Roberta Shauger (December 20, 2011). *UDEF Concepts Defined - Reference Guide* [8]. CreateSpace. ISBN 978-1-4681-1483-6.
- Roberta Shauger and Ronald Schuldt (January 14, 2012). *UDEF Concepts Defined - Reference Guide* [9]. Amazon Digital Services. ISBN 1-4681-1483-2.

References

[1] http://www.opengroup.org/udefinfo/
[2] http://www.youtube.com/embed/y6hID5qzAzQ
[3] http://www.youtube.com/embed/d6dH_U8TqhY
[4] http://www.opengroup.org/udefinfo/faq.htm
[5] https://udef-it.com/UDEF_Tools.html
[6] https://www.createspace.com/3711806
[7] http://www.amazon.com/dp/B006YK6YOQ
[8] https://www.createspace.com/3753707
[9] http://www.amazon.com/dp/B006XXMLQE

Terminology model

A terminology model is a refinement of a concept system.[1] Within a terminology model the concepts (object types) of a specific problem or subject area are defined by subject matter experts in terms of concept (object type) definitions and definitions of subordinated concepts or characteristics (properties). Besides object types, the terminology model allows defining hierarchical classifications, definitions for object type and property behavior and definition of casual relations.

The terminology model is a means for subject matter experts to express their knowledge about the subject in subject specific terms. Since the terminology model is structured rather similar to an object-oriented database schema, is can be transformed without loss of information into an object-oriented database schema. Thus, the terminology model is a method for problem analysis on the one side and a mean of defining database schema on the other side.

Several terminology models have been developed and published in the field of statistics:

- Terminology model for classifications[2]
- Terminology model for statistical variables[3]
- Reference model for statistical metadata[4]

References

[1] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=38109 ISO 704:2009 - Terminology work -- Principles and methods
[2] http://www1.unece.org/stat/platform/download/attachments/14319930/Part+I+Neuchatel_version+2_1.pdf?version=1 Neuchâtel Terminology Model PART I: Classification database object types and their attributes
[3] http://www1.unece.org/stat/platform/download/attachments/14319930/Neuchatel+Model+V1.pdf?version=1 Neuchâtel Terminology Model PART II: Variables and related concepts
[4] http://www.epros.ed.ac.uk/metanet/working_groups/Reference_model/ReferenceModel.doc METANET - Reference Model

Georelational data model

A **georelational data model** is a geographic data model that represents geographic features as an interrelated set of spatial and attribute data. The georelational model is the fundamental data model used in coverages.[1]

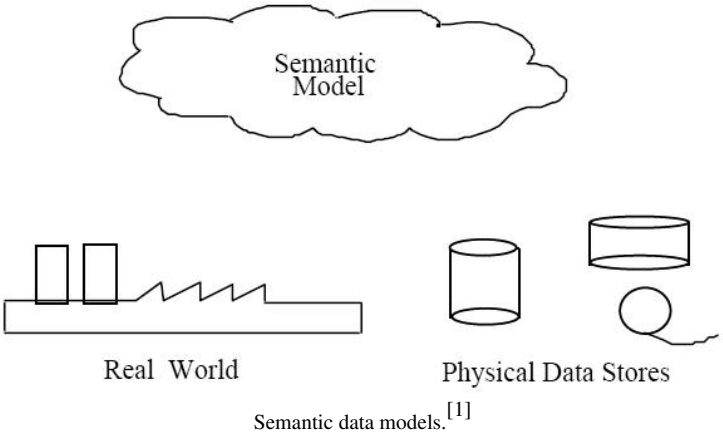
References

[1] Wade, T. and Sommer, S. eds. *A to Z GIS* (http://store.esri.com/esri/showdetl.cfm?SID=2&Product_ID=868&Category_ID=49)

Semantic data model

A **semantic data model** in software engineering has various meanings:

1. It is a conceptual data model in which semantic information is included. This means that the model describes the meaning of its instances. Such a semantic data model is an abstraction that defines how the stored symbols (the instance data) relate to the real world.
2. It is a conceptual data model that includes the capability to express information that enables parties to the information exchange to interpret meaning (semantics) from the instances, without the need to know the meta-model. Such semantic models are fact oriented (as opposed to object oriented). Facts are typically expressed by binary relations between data elements, whereas higher order relations are expressed as collections of binary relations. Typically binary relations have the form of triples: Object-RelationType-Object. For example: the Eiffel Tower <is located in> Paris.



Semantic data models.[1]

Typically the instance data of semantic data models explicitly include the kinds of relationships between the various data elements, such as <is located in>. To interpret the meaning of the facts from the instances it is required that the meaning of the kinds of relations (relation types) is known. Therefore, semantic data models typically standardise such relation types. This means that the second kind of semantic data models enable that the instances express facts that include their own meaning. The second kind of semantic data models are usually meant to create semantic databases. The ability to include meaning in semantic databases facilitates building distributed databases that enable applications to interpret the meaning from the content. This implies that semantic databases can be integrated when they use the same (standard) relation types. This also implies that in general they have a wider applicability than relational or object oriented databases.

Overview

The logical data structure of a database management system (DBMS), whether hierarchical, network, or relational, cannot totally satisfy the requirements for a conceptual definition of data, because it is limited in scope and biased toward the implementation strategy employed by the DBMS. Therefore, the need to define data from a conceptual view has led to the development of semantic data modeling techniques. That is, techniques to define the meaning of data within the context of its interrelationships with other data. As illustrated in the figure. The real world, in terms

of resources, ideas, events, etc., are symbolically defined within physical data stores. A semantic data model is an abstraction which defines how the stored symbols relate to the real world. Thus, the model must be a true representation of the real world.

According to Klas and Schrefl (1995), the "overall goal of semantic data models is to capture more meaning of data by integrating relational concepts with more powerful abstraction concepts known from the Artificial Intelligence field. The idea is to provide high level modeling primitives as integral part of a data model in order to facilitate the representation of real world situations".[2]

History

The need for semantic data models was first recognized by the U.S. Air Force in the mid-1970s as a result of the Integrated Computer-Aided Manufacturing (ICAM) Program. The objective of this program was to increase manufacturing productivity through the systematic application of computer technology. The ICAM Program identified a need for better analysis and communication techniques for people involved in improving manufacturing productivity. As a result, the ICAM Program developed a series of techniques known as the IDEF (ICAM Definition) Methods which included the following:

- IDEF0 used to produce a “function model” which is a structured representation of the activities or processes within the environment or system.
- IDEF1 used to produce an “information model” which represents the structure and semantics of information within the environment or system.
  - IDEF1X is a semantic data modeling technique. It is used to produce a graphical information model which represents the structure and semantics of information within an environment or system. Use of this standard permits the construction of semantic data models which may serve to support the management of data as a resource, the integration of information systems, and the building of computer databases.
- IDEF2 used to produce a “dynamics model” which represents the time varying behavioral characteristics of the environment or system.

During the 1990s the application of semantic modelling techniques resulted in the semantic data models of the second kind. An example of such is the semantic data model that is standardised as ISO 15926-2 (2002), which is further developed into the semantic modelling language Gellish (2005). The definition of the Gellish language is documented in the form of a semantic data model. Gellish itself is a semantic modelling language, that can be used to create other semantic models. Those semantic models can be stored in Gellish Databases, being semantic databases.

Applications

A semantic data model can be used to serve many purposes. Some key objectives include:

- Planning of Data Resources: A preliminary data model can be used to provide an overall view of the data required to run an enterprise. The model can then be analyzed to identify and scope projects to build shared data resources.
- Building of Shareable Databases: A fully developed model can be used to define an application independent view of data which can be validated by users and then transformed into a physical database design for any of the various DBMS technologies. In addition to generating databases which are consistent and shareable, development costs can be drastically reduced through data modeling.
- Evaluation of Vendor Software: Since a data model actually represents the infrastructure of an organization, vendor software can be evaluated against a company’s data model in order to identify possible inconsistencies between the infrastructure implied by the software and the way the company actually does business.
- Integration of Existing Databases: By defining the contents of existing databases with semantic data models, an integrated data definition can be derived. With the proper technology, the resulting conceptual schema can be

used to control transaction processing in a distributed database environment. The U.S. Air Force Integrated Information Support System (I2S2) is an experimental development and demonstration of this type of technology applied to a heterogeneous DBMS environment.

References

[1] FIPS Publication 184 (http://www.itl.nist.gov/fipspubs/idef1x.doc) released of IDEF1X by the Computer Systems Laboratory of the National Institute of Standards and Technology (NIST). 21 December 1993.

[2] Wolfgang Klas, Michael Schrefl (1995). "Semantic data modeling" In: *Metaclasses and Their Application*. Book Series Lecture Notes in Computer Science. Publisher Springer Berlin / Heidelberg. Volume Volume 943/1995.

Further reading

- Database Design - The Semantic Modelling Approach (http://hpdrc.cs.fiu.edu/library/books/datades-book/)
- Johan ter Bekke (1992). *Semantic Data Modeling*. Prentice Hall.
- Alfonso F. Cardenas and Dennis McLeod (1990). *Research Foundations in Object-Oriented and Semantic Database Systems*. Prentice Hall.
- Peter Gray, Krishnarao G. Kulkarni and, Norman W. Paton (1992). *Object-Oriented Databases: A Semantic Data Model Approach*. Prentice-Hall International Series in Computer Science.
- Michael Hammer and Dennis McLeod (1978). "The Semantic Data Model: a Modeling Mechanism for Data Base Applications." In: *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*. Austin, Texas, May 31 - June 2, 1978, pp. 26–36.

External links

- Semantic Data Modeling (http://www.jhterbekke.net/SemanticDataModeling.html) Johan ter Bekke tribute site.

Relational Model/Tasmania

Relational Model/Tasmania (RM/T) was published by E.F. Codd in 1979 and is the name given to a number of extensions to his original relational model (RM) published in 1970. The overall goal of the RM/T was to define some fundamental semantic units, at "atomic" and "molecular" levels, for data modelling. Codd writes: "*the result is a model with a richer variety of objects than the original relational model, additional insert-update-delete rules and some additional operators that make the algebra more powerful*".

RM History

Between 1968 and 1988 Codd published over 30 papers on the relational model (RM) - the most famous of which is his 1970 paper. Up to 1978 the papers describe RM Version 1 (RM/V1). In early 1979 Codd first presented some new ideas, called RM/T ('T' for Tasmania), at an invited talk for the *Australian Computer Science Conference* in Hobart, Tasmania. Later that year the ACM journal published a paper on RM/T, in which Codd acknowledges the influence of Schmid & Swensen (1975) and Wiederhold (1977).

A later version of RM/T (we shall call it here "RM/D") was described in Date (1983) in which Date and Codd improved and refined RM/T, adding an entity type called *designative*. Although Codd writes nothing about this new type, Date offers a rationale in Date (1983, page 262). Date revised this 1983 article in Date (1995), which additionally compares the RM/T model with the E/R model.

Following a disappointing uptake of RM/T by the database industry, Codd decided to introduce the RM/T model more gradually. He planned to release a sequence of RM versions: RM/V2, RM/V3 etc. each time progressively including some of the ideas of the original RM/T into the new version. Perhaps this explains why there is no obvious mapping of concepts between RM/T and RM/V2. For example, there is no reference to *associative* or *designative* entity types in Codd's 1990 book that defines RM/V2. On the other hand, the book extends and builds on the existing body of query language issues, many of which were addressed by Codd in several papers throughout the 1980s.

Summary of RM/T

First we shall introduce some of the new concepts of RM/T:

Surrogates A surrogate is a unique value assigned to each entity. If two relations use the same surrogate value then they represent the same entity in the modelled universe. The surrogate *value* can be any unique string or number but cannot be assigned or changed by the database user. For example, a SQL SEQUENCE is often used to generate *numerical* surrogate values.

Entities and Nonentities An *entity* is some *thing* in the modelled universe and is typically identified by a surrogate. A *nonentity* is some *thing* that is not an entity and does not have its own identifying surrogate. An *independent* entity has its own surrogate. A *dependent* entity has a surrogate but it belongs to another entity, i.e. the surrogate is a foreign key.

Atomic Semantics The RM/T addresses *atomic* semantics by describing how the original RM *relation* can be used to describe entities with attributes. An entity is represented as an *Entity-relation* or *E-relation* and its attributes (or immediate properties) are stored in separate *Property-relations* or *P-relations*. Each *E-relation* shares its surrogate with the associated *P-relation*s.

E-relations mark the *existence* of an entity. An *E-relation* is a relation (table) storing only the *surrogates* for a particular entity type. A surrogate value entered into the *E-relation* table implies the corresponding existence of an entity of that type in the *modelled world*. For example, the *E-relation* "Employee" is a table containing the surrogates of all entities of type *Employee*.

P-relations store the *attribute values* of an entity. A *P-relation* is a relation (table) storing the surrogate and one or more attributes of an entity. The surrogate value of a *P-relation* is that of the corresponding *E-relation*; it plays the role (K-role) of the primary key for that *P-relation*. For example, the *P-relation* "Employee_Number" is a table with two columns: one containing the surrogate value of the "Employee" *E-relation*, the other containing the employee *number*.

Note that by performing an OUTER NATURAL JOIN on the RM/T "Employee" *E-relation* and "Employee_Person" *P-relation* we can construct the RM/V1 "Employee" relation. This illustrates why the *E-relation* and *P-relation* concepts of RM/T are more *atomic* than the *relation* concept of RM/V1.

Molecular Semantics The RM/T addresses *molecular* semantics by taking the original RM and categorising the relations into several entity types, increasing the information captured by the semantic data model. However Codd does not define a notation for diagramming his new semantics. Each entity may play several roles at once and thus belong to one or more of the following entity types:

- *Characteristic* - subordinate entities that describe kernel entities.
- *Associative* - superordinate entities that interrelate kernel entities.
- *Kernel* - entities that are neither characteristic or associative.

Codd goes on to introduce subtyping of entities, giving yet another qualifier for entities:

- *Inner* - entities that are not subtypes of another entity.

Hence Codd speaks of *inner kernel* and *inner associative* entities.

The following definition is based on the RM/D model in Date (1983); it does *not* appear in Codd (1979):

- *Designative* - entities that contain a designation. A designative entity is at the *many* end of a *one-to-many* relationship between two independent entities. For example, a writer may write many books, hence a one-to-many relationship between writer and book entities; the book is the *designative* entity because it contains a designation (or *designative reference*) to the writer - namely the primary key of the writer entity. Note that an *associative* entity contains at least two designations. For example, we can regard a booking as either an entity that *associates* a person with a flight, or as an entity that *designates* a person and *designates* a flight. Hence a designative entity must contain at least *one* designation whereas an associative entity must contain at least *two* designations.

Associations These are what we might otherwise call *relationships* between entities or non-entities. The value *E-null* is used when deleteting entities from the RM/T model; all associations that have surrogates referring to a non-existing entity are assigned the value *E-null*, meaning the entity is unknown.

Associative Entity and Nonentity Association An *associative entity* is an entity that represents an association between two independent entities; the *associative entity* is an entity in itself because it has a surrogate. A *nonentity association* is similar to an *associative entity* however it has *no* surrogate. This lack of a surrogate stops the *nonentity association* from having, for example, any descriptive *characteristic entities*.

Directed Graph Relations Several *directed graph* relations are defined to capture further semantic features of the RM/T model. These graphs are named as follows:

- PG-relation (Property Graph) stores *property* relationships
- CG-relation (Characteristic Graph) stores *characteristic* relationships
- AG-relation (Association Graph) stores *association* relationships
- UGI-relation (Unconditional Generalisation by Inclusion) stores *generalisation by inclusion* relationships
- AGI-relation (Alternative Generalisation by Inclusion) stores *generalisation by alternative* relationships
- US-relation (Unconditional Successor) stores *unconditional successor* relationships
- AS-relation (Alternative Successor) stores *alternative successor* relationships
- KG-relation (Cover Membership) stores *cover membership* relationships
- UP-relation (Unconditional Precedence) stores *unconditional succession of event* relationships
- AP-relation (Alternative Precedence) stores *alternative succession of eve*nt relationships

RM/T Catalog The Catalog is a meta-model storing the descriptions of the relations themselves. The RM/T Catalog comprises the following relations:

- CATR (R-surrogate, *relname*, RelType) describes relations
- CATRA (RA-surrogate, R-surrogate, A-surrogate) relates relations and attributes
- CATA (A-surrogate, *attname*, UserKey) describes attributes
- CATAD (AD-surrogate, A-surrogate, D-surrogate) relates attributes and domains
- CATD (D-surrogate, *domname*, VType, Ordering) describes domains
- CATC (C-surrogate, *pername*) describes categories
- CATRC (RC-surrogate, R-surrogate, C-surrogate) relates relations and categories

where

- *relname* is the textual name of a relation. e.g. "Address"
- *attname* is the textual name of an attribute. e.g. "Street"
- *domname* is the textual name of a domain. e.g. "Salary"
- *pername* is the category label (from the PER-domain)
- RN-domain is the domain of all relnames in the database
- PER-domain is the domain of all category labels
- E-domain is the domain of all surrogates in the database
- E-attribute is any attribute that plays the role of a surrogate (from the E-domain)
- E-null is the "entity unknown" surrogate (from the E-domain)
- R-surrogate is the relation surrogate (from the E-domain)
- A-surrogate is the attribute surrogate (from the E-domain)
- D-surrogate is the domain surrogate (from the E-domain)
- C-surrogate is the category label surrogate (from the E-domain)
- RA-surrogate is the relation-attribute surrogate (from the E-domain)
- AD-surrogate is the attribute-domain surrogate (from the E-domain)
- RC-surrogate is the relation-category-label surrogate (from the E-domain)
- RelType is the type of object represented by the relation
- UserKey shows whether the attribute participates in a user-defined key
- VType is the syntactic type of the value
- Ordering shows whether the operator > is applicable between values of the domain

Operators Numerous operators are defined on names, sets and graphs. See Codd's 1979 paper for details.

RM/T Today

There is little mention of RM/T today and no articles have appeared recently. Peckam and Maryanski (1988) wrote about RM/T in their study of semantic data models. Codd published his book in 1990 but wrote nothing more about RM/T. RM/V1 and RM/V2 have a chapter each in *Date and Darwen* (1992) and the Date (1983) article was updated in (1995) and now contains a long overdue comparison of the E/R model and RM/T. Date's most recent reflections can be found on the Web at Date (1999), *The Database Relational Model* (2001) and *Date on RM/T* (2003).

RM/T contributed to the body of knowledge called semantic data modeling and semantic object modeling and continues to influence new data modellers. See the paper by Hammer and McLeod (1981), the book by Knoenke (2001) and implementation by Grabczewski et alia (2004).

External links

- Extending the database relational model to capture more meaning [1] by E.F. Codd (1979)
- On the semantics of the relational data model [2] by H.A. Schmid and J.R. Swensen (1975)
- Database Design [3] by G. Wiederhold (1977)
- The Relational Model for Data Base Management: Version 2 [4] by E.F. Codd (1990)
- An Introduction to Data Base Systems: Vol 2 [5] by C.J. Date (1983)
- Relational Database Writings 1989–1991 [6] by C.J. Date with Hugh Darwen (1992)
- Relational Database Writings 1991–1994 [7] by C.J. Date (1995)
- Thirty Years of Relational: Extending the Relational Model [8] by C.J. Date (1999)
- Semantic Data Models [9] by J. Peckam and F. Maryanski (1988); a useful survey that includes RM/T
- Codd's Extended Relational Model [10] by Dr Arthur Cater of University College Dublin
- A relational model of data for large shared data banks [11] by E.F. Codd (1970)
- Database Description with SDM: A Semantic Database Model [12] by M. Hammer and D. McLeod (1981)
- Database Processing (Eighth Edition) [13] by David M. Kroenke (2001)
- A Corporate Data Repository for CCLRC using CERIF [14] by E. Grabczewski, S.Crompton, S.K. Robinson and T.H. Hall (2004)
- "A Practical Approach to Database Design" in *Relational Database: Selected Writings* by C.J.Date (1986)
- *The Database Relational Model: A Retrospective Review and Analysis* by C.J.Date (2001)

References

[1] http://portal.acm.org/citation.cfm?id=320109
[2] http://portal.acm.org/citation.cfm?id=500110&dl=ACM&coll=&CFID=15151515&CFTOKEN=6184618
[3] http://portal.acm.org/citation.cfm?id=542883&dl=ACM&coll=GUIDE&CFID=11111111&CFTOKEN=2222222
[4] http://www.amazon.co.uk/dp/0201141922
[5] http://www.amazon.co.uk/dp/0201144743
[6] http://www.amazon.com/dp/0201543036
[7] http://www.amazon.com/dp/0201824590/
[8] http://www.intelligententerprise.com/db_area/archives/1999/990106/online1.jhtml;jsessionid=DYSOTINHQQI5OQSNDLQSKH0CJUNN2JVN
[9] http://delivery.acm.org/10.1145/70000/62062/p153-peckham.pdf?key1=62062&key2=4927854611&coll=GUIDE&dl=GUIDE&CFID=5476987&CFTOKEN=80047688
[10] http://www.cs.ucd.ie/staff/acater/home/comp4002/rmt.pdf
[11] http://portal.acm.org/citation.cfm?id=362685
[12] http://portal.acm.org/citation.cfm?doid=319587.319588
[13] http://www.amazon.com/dp/0130648396/
[14] http://epubs.cclrc.ac.uk/work-details?w=35234

Relational Schema

Relational model

The **relational model** for database management is a database model based on first-order predicate logic, first formulated and proposed in 1969 by Edgar F. Codd.[1] In the relational model of a database, all data is represented in terms of tuples, grouped into relations. A database organized in terms of the relational model is a relational database.

The purpose of the relational model is to provide a declarative method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.

Most relational databases use the SQL data definition and query language; these systems implement what can be regarded as an engineering approximation to the relational model. A *table* in an SQL database schema corresponds to a predicate variable; the contents of a table to a relation; key constraints, other constraints, and SQL queries correspond to predicates. However, SQL databases, including DB2, deviate from the relational model in many details, and Codd fiercely argued against deviations that compromise the original principles.[3]

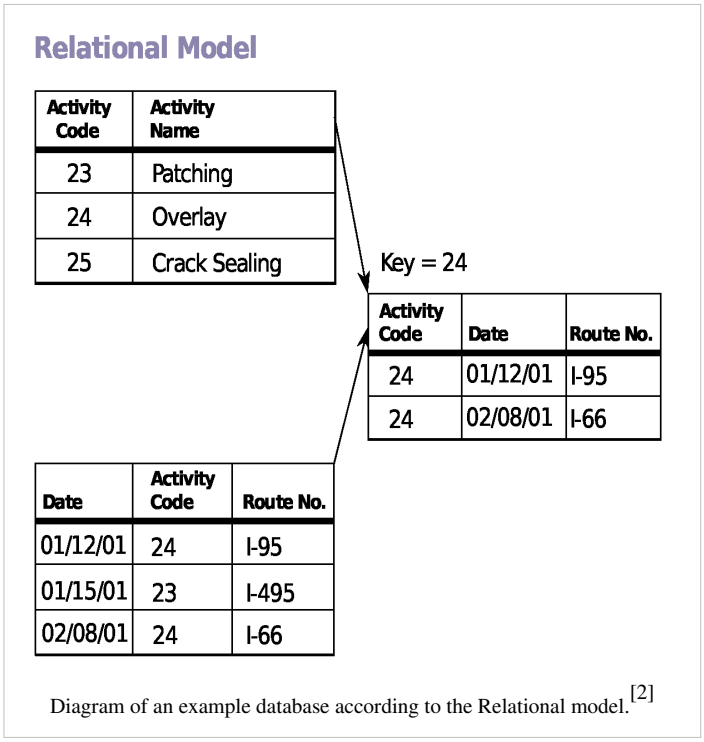
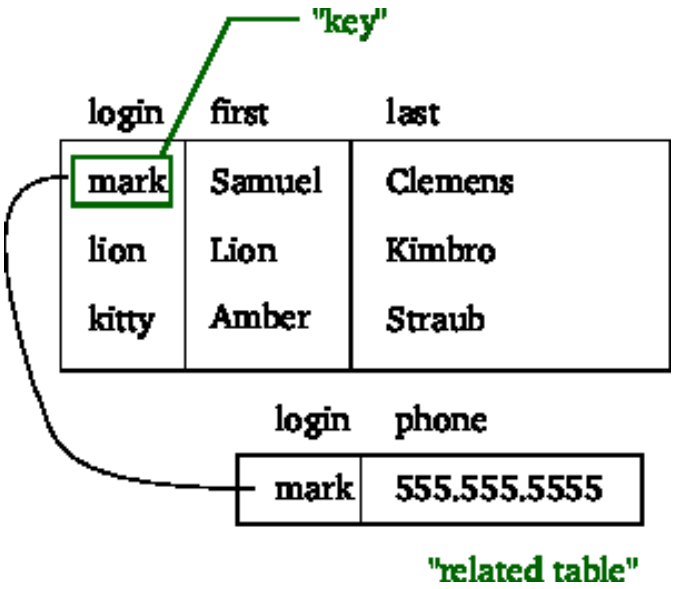


Diagram of an example database according to the Relational model.[2]



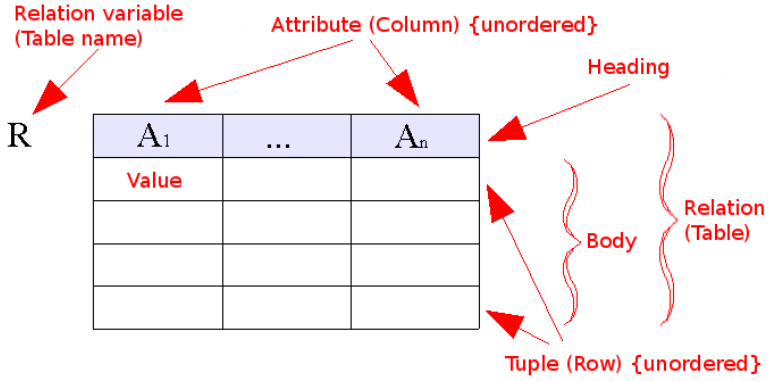
In the relational model, related records are linked together with a "key".

Overview

The relational model's central idea is to describe a database as a collection of predicates over a finite set of predicate variables, describing constraints on the possible values and combinations of values. The content of the database at any given time is a finite (logical) model of the database, i.e. a set of relations, one per predicate variable, such that all predicates are satisfied. A request for information from the database (a database query) is also a predicate.

Alternatives to the relational model

Other models are the hierarchical model and network model. Some systems using these older architectures are still in use today in data centers with high data volume needs, or where existing systems are so complex and abstract it would be cost-prohibitive to migrate to systems employing the relational model; also of note are newer object-oriented databases.



Relational model concepts.

Implementation

There have been several attempts to produce a true implementation of the relational database model as originally defined by Codd and explained by Date, Darwen and others, but none have been popular successes so far. Rel is one of the more recent attempts to do this.

The relational model was the first database model to be described in formal mathematical terms. Hierarchical and network databases existed before relational databases, but their specifications were relatively informal. After the relational model was defined, there were many attempts to compare and contrast the different models, and this led to the emergence of more rigorous descriptions of the earlier models; though the procedural nature of the data manipulation interfaces for hierarchical and network databases limited the scope for formalization.[*citation needed*]

History

The relational model was invented by E.F. (Ted) Codd as a general model of data, and subsequently maintained and developed by Chris Date and Hugh Darwen among others. In The Third Manifesto (first published in 1995) Date and Darwen show how the relational model can accommodate certain desired object-oriented features.

Controversies

Codd himself, some years after publication of his 1970 model, proposed a three-valued logic (True, False, Missing or NULL) version of it to deal with missing information, and in his *The Relational Model for Database Management Version 2* (1990) he went a step further with a four-valued logic (True, False, Missing but Applicable, Missing but Inapplicable) version. But these have never been implemented, presumably because of attending complexity. SQL's NULL construct was intended to be part of a three-valued logic system, but fell short of that due to logical errors in the standard and in its implementations.[*citation needed*]

Relational model topics

The model

The fundamental assumption of the relational model is that all data is represented as mathematical *n*-ary **relations**, an *n*-ary relation being a subset of the Cartesian product of *n* domains. In the mathematical model, reasoning about such data is done in two-valued predicate logic, meaning there are two possible evaluations for each proposition: either *true* or *false* (and in particular no third value such as *unknown*, or *not applicable*, either of which are often associated with the concept of NULL). Data are operated upon by means of a relational calculus or relational

algebra, these being equivalent in expressive power.

The relational model of data permits the database designer to create a consistent, logical representation of information. Consistency is achieved by including declared ***constraints*** in the database design, which is usually referred to as the logical schema. The theory includes a process of database normalization whereby a design with certain desirable properties can be selected from a set of logically equivalent alternatives. The access plans and other implementation and operation details are handled by the DBMS engine, and are not reflected in the logical model. This contrasts with common practice for SQL DBMSs in which performance tuning often requires changes to the logical model.

The basic relational building block is the domain or data type, usually abbreviated nowadays to ***type***. A *tuple* is an ordered set of ***attribute values***. An attribute is an ordered pair of ***attribute name*** and ***type name***. An attribute value is a specific valid value for the type of the attribute. This can be either a scalar value or a more complex type.

A relation consists of a ***heading*** and a ***body***. A heading is a set of attributes. A body (of an *n*-ary relation) is a set of *n*-tuples. The heading of the relation is also the heading of each of its tuples.

A relation is defined as a set of *n*-tuples. In both mathematics and the relational database model, a set is an *unordered* collection of unique, non-duplicated items, although some DBMSs impose an order to their data. In mathematics, a tuple has an order, and allows for duplication. E.F. Codd originally defined tuples using this mathematical definition. Later, it was one of E.F. Codd's great insights that using attribute names instead of an ordering would be so much more convenient (in general) in a computer language based on relations [*citation needed*]. This insight is still being used today. Though the concept has changed, the name "tuple" has not. An immediate and important consequence of this distinguishing feature is that in the relational model the Cartesian product becomes commutative.

A table is an accepted visual representation of a relation; a tuple is similar to the concept of a *row*.

A *relvar* is a named variable of some specific relation type, to which at all times some relation of that type is assigned, though the relation may contain zero tuples.

The basic principle of the relational model is the Information Principle: all information is represented by data values in relations. In accordance with this Principle, a relational database is a set of relvars and the result of every query is presented as a relation.

The consistency of a relational database is enforced, not by rules built into the applications that use it, but rather by *constraints*, declared as part of the logical schema and enforced by the DBMS for all applications. In general, constraints are expressed using relational comparison operators, of which just one, "is subset of" ($\subseteq$), is theoretically sufficient[*citation needed*]. In practice, several useful shorthands are expected to be available, of which the most important are candidate key (really, superkey) and foreign key constraints.

Interpretation

To fully appreciate the relational model of data it is essential to understand the intended *interpretation* of a relation.

The body of a relation is sometimes called its extension. This is because it is to be interpreted as a representation of the extension of some predicate, this being the set of true propositions that can be formed by replacing each free variable in that predicate by a name (a term that designates something).

There is a one-to-one correspondence between the free variables of the predicate and the attribute names of the relation heading. Each tuple of the relation body provides attribute values to instantiate the predicate by substituting each of its free variables. The result is a proposition that is deemed, on account of the appearance of the tuple in the relation body, to be true. Contrariwise, every tuple whose heading conforms to that of the relation, but which does not appear in the body is deemed to be false. This assumption is known as the closed world assumption: it is often violated in practical databases, where the absence of a tuple might mean that the truth of the corresponding proposition is unknown. For example, the absence of the tuple ('John', 'Spanish') from a table of language skills

cannot necessarily be taken as evidence that John does not speak Spanish.

For a formal exposition of these ideas, see the section Set-theoretic Formulation, below.

Application to databases

A **data type** as used in a typical relational database might be the set of integers, the set of character strings, the set of dates, or the two boolean values *true* and *false*, and so on. The corresponding **type names** for these types might be the strings "int", "char", "date", "boolean", etc. It is important to understand, though, that relational theory does not dictate what types are to be supported; indeed, nowadays provisions are expected to be available for *user-defined* types in addition to the *built-in* ones provided by the system.

Attribute is the term used in the theory for what is commonly referred to as a **column**. Similarly, **table** is commonly used in place of the theoretical term **relation** (though in SQL the term is by no means synonymous with relation). A table data structure is specified as a list of column definitions, each of which specifies a unique column name and the type of the values that are permitted for that column. An **attribute *value*** is the entry in a specific column and row, such as "John Doe" or "35".

A **tuple** is basically the same thing as a **row**, except in an SQL DBMS, where the column values in a row are ordered. (Tuples are not ordered; instead, each attribute value is identified solely by the **attribute name** and never by its ordinal position within the tuple.) An attribute name might be "name" or "age".

A **relation** is a **table** structure definition (a set of column definitions) along with the data appearing in that structure. The structure definition is the **heading** and the data appearing in it is the **body**, a set of rows. A database **relvar** (relation variable) is commonly known as a **base table**. The heading of its assigned value at any time is as specified in the table declaration and its body is that most recently assigned to it by invoking some **update operator** (typically, INSERT, UPDATE, or DELETE). The heading and body of the table resulting from evaluation of some query are determined by the definitions of the operators used in the expression of that query. (Note that in SQL the heading is not always a set of column definitions as described above, because it is possible for a column to have no name and also for two or more columns to have the same name. Also, the body is not always a set of rows because in SQL it is possible for the same row to appear more than once in the same body.)

SQL and the relational model

SQL, initially pushed as the standard language for relational databases, deviates from the relational model in several places. The current ISO SQL standard doesn't mention the relational model or use relational terms or concepts. However, it is possible to create a database conforming to the relational model using SQL if one does not use certain SQL features.

The following deviations from the relational model have been notedWikipedia:Avoid weasel words in SQL. Note that few database servers implement the entire SQL standard and in particular do not allow some of these deviations. Whereas NULL is ubiquitous, for example, allowing duplicate column names within a table or anonymous columns is uncommon.

Duplicate rows

: The same row can appear more than once in an SQL table. The same tuple cannot appear more than once in a relation.

Anonymous columns

: A column in an SQL table can be unnamed and thus unable to be referenced in expressions. The relational model requires every attribute to be named and referenceable.

Duplicate column names

: Two or more columns of the same SQL table can have the same name and therefore cannot be referenced, on account of the obvious ambiguity. The relational model requires every attribute to be referenceable.

Column order significance

The order of columns in an SQL table is defined and significant, one consequence being that SQL's implementations of Cartesian product and union are both noncommutative. The relational model requires there to be no significance to any ordering of the attributes of a relation.

Views without CHECK OPTION

Updates to a view defined without CHECK OPTION can be accepted but the resulting update to the database does not necessarily have the expressed effect on its target. For example, an invocation of INSERT can be accepted but the inserted rows might not all appear in the view, or an invocation of UPDATE can result in rows disappearing from the view. The relational model requires updates to a view to have the same effect as if the view were a base relvar.

Columnless tables unrecognized

SQL requires every table to have at least one column, but there are two relations of degree zero (of cardinality one and zero) and they are needed to represent extensions of predicates that contain no free variables.

NULL

This special mark can appear instead of a value wherever a value can appear in SQL, in particular in place of a column value in some row. The deviation from the relational model arises from the fact that the implementation of this *ad hoc* concept in SQL involves the use of three-valued logic, under which the comparison of NULL with itself does not yield *true* but instead yields the third truth value, *unknown*; similarly the comparison NULL with something other than itself does not yield *false* but instead yields *unknown*. It is because of this behaviour in comparisons that NULL is described as a mark rather than a value. The relational model depends on the law of excluded middle under which anything that is not true is false and anything that is not false is true; it also requires every tuple in a relation body to have a value for every attribute of that relation. This particular deviation is disputed by some if only because E.F. Codd himself eventually advocated the use of special marks and a 4-valued logic, but this was based on his observation that there are two distinct reasons why one might want to use a special mark in place of a value, which led opponents of the use of such logics to discover more distinct reasons and at least as many as 19 have been noted, which would require a 21-valued logic. [*citation needed*] SQL itself uses NULL for several purposes other than to represent "value unknown". For example, the sum of the empty set is NULL, meaning zero, the average of the empty set is NULL, meaning undefined, and NULL appearing in the result of a LEFT JOIN can mean "no value because there is no matching row in the right-hand operand".

Relational operations

Users (or programs) request data from a relational database by sending it a query that is written in a special language, usually a dialect of SQL. Although SQL was originally intended for end-users, it is much more common for SQL queries to be embedded into software that provides an easier user interface. Many Web sites, such as Wikipedia, perform SQL queries when generating pages.

In response to a query, the database returns a result set, which is just a list of rows containing the answers. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted.

Often, data from multiple tables are combined into one, by doing a join. Conceptually, this is done by taking all possible combinations of rows (the Cartesian product), and then filtering out everything except the answer. In practice, relational database management systems rewrite ("optimize") queries to perform faster, using a variety of techniques.

There are a number of relational operations in addition to join. These include project (the process of eliminating some of the columns), restrict (the process of eliminating some of the rows), union (a way of combining two tables

with similar structures), difference (that lists the rows in one table that are not found in the other), intersect (that lists the rows found in both tables), and product (mentioned above, which combines each row of one table with each row of the other). Depending on which other sources you consult, there are a number of other operators – many of which can be defined in terms of those listed above. These include semi-join, outer operators such as outer join and outer union, and various forms of division. Then there are operators to rename columns, and summarizing or aggregating operators, and if you permit relation values as attributes (RVA – relation-valued attribute), then operators such as group and ungroup. The SELECT statement in SQL serves to handle all of these except for the group and ungroup operators.

The flexibility of relational databases allows programmers to write queries that were not anticipated by the database designers. As a result, relational databases can be used by multiple applications in ways the original designers did not foresee, which is especially important for databases that might be used for a long time (perhaps several decades). This has made the idea and implementation of relational databases very popular with businesses.

Database normalization

Relations are classified based upon the types of anomalies to which they're vulnerable. A database that's in the first normal form is vulnerable to all types of anomalies, while a database that's in the domain/key normal form has no modification anomalies. Normal forms are hierarchical in nature. That is, the lowest level is the first normal form, and the database cannot meet the requirements for higher level normal forms without first having met all the requirements of the lesser normal forms.[4]

Examples

Database

An idealized, very simple example of a description of some relvars (relation variables) and their attributes:

- Customer (**Customer ID**, Tax ID, Name, Address, City, State, Zip, Phone, Email)
- Order (**Order No**, Customer ID, Invoice No, Date Placed, Date Promised, Terms, Status)
- Order Line (**Order No**, **Order Line No**, Product Code, Qty)
- Invoice (**Invoice No**, Customer ID, Order No, Date, Status)
- Invoice Line (**Invoice No**, **Invoice Line No**, Product Code, Qty Shipped)
- Product (**Product Code**, Product Description)

In this design we have six relvars: Customer, Order, Order Line, Invoice, Invoice Line and Product. The bold, underlined attributes are *candidate keys*. The non-bold, underlined attributes are *foreign keys*.

Usually one candidate key is arbitrarily chosen to be called the primary key and used in preference over the other candidate keys, which are then called alternate keys.

A *candidate key* is a unique identifier enforcing that no tuple will be duplicated; this would make the relation into something else, namely a bag, by violating the basic definition of a set. Both foreign keys and superkeys (that includes candidate keys) can be composite, that is, can be composed of several attributes. Below is a tabular depiction of a relation of our example Customer relvar; a relation can be thought of as a value that can be attributed to a relvar.

Customer relation

Customer ID	Tax ID	Name	Address	[More fields…]
1234567890	555-5512222	Munmun	323 Broadway	…
2223344556	555-5523232	Wile E.	1200 Main Street	…
3334445563	555-5533323	Ekta	871 1st Street	…
4232342432	555-5325523	E. F. Codd	123 It Way	…

If we attempted to *insert* a new customer with the ID *1234567890*, this would violate the design of the relvar since **Customer ID** is a *primary key* and we already have a customer *1234567890*. The DBMS must reject a transaction such as this that would render the database inconsistent by a violation of an integrity constraint.

Foreign keys are integrity constraints enforcing that the value of the attribute set is drawn from a *candidate key* in another relation. For example in the Order relation the attribute **Customer ID** is a foreign key. A *join* is the operation that draws on information from several relations at once. By joining relvars from the example above we could *query* the database for all of the Customers, Orders, and Invoices. If we only wanted the tuples for a specific customer, we would specify this using a restriction condition.

If we wanted to retrieve all of the Orders for Customer *1234567890*, we could query the database to return every row in the Order table with **Customer ID** *1234567890* and join the Order table to the Order Line table based on **Order No**.

There is a flaw in our database design above. The Invoice relvar contains an Order No attribute. So, each tuple in the Invoice relvar will have one Order No, which implies that there is precisely one Order for each Invoice. But in reality an invoice can be created against many orders, or indeed for no particular order. Additionally the Order relvar contains an Invoice No attribute, implying that each Order has a corresponding Invoice. But again this is not always true in the real world. An order is sometimes paid through several invoices, and sometimes paid without an invoice. In other words there can be many Invoices per Order and many Orders per Invoice. This is a **many-to-many** relationship between Order and Invoice (also called a *non-specific relationship*). To represent this relationship in the database a new relvar should be introduced whose role is to specify the correspondence between Orders and Invoices:

OrderInvoice(**Order No**,**Invoice No**)

Now, the Order relvar has a *one-to-many relationship* to the OrderInvoice table, as does the Invoice relvar. If we want to retrieve every Invoice for a particular Order, we can query for all orders where **Order No** in the Order relation equals the **Order No** in OrderInvoice, and where **Invoice No** in OrderInvoice equals the **Invoice No** in Invoice.

Set-theoretic formulation

Basic notions in the relational model are *relation names* and *attribute names*. We will represent these as strings such as "Person" and "name" and we will usually use the variables $r, s, t, \ldots$ and a, b, c to range over them. Another basic notion is the set of *atomic values* that contains values such as numbers and strings.

Our first definition concerns the notion of *tuple*, which formalizes the notion of row or record in a table:

Tuple

: A tuple is a partial function from attribute names to atomic values.

Header

: A header is a finite set of attribute names.

Projection

The projection of a tuple t on a finite set of attributes A is $t[A] = \{(a, v) : (a, v) \in t, a \in A\}$.

The next definition defines *relation* that formalizes the contents of a table as it is defined in the relational model.

Relation

A relation is a tuple (H, B) with H, the header, and B, the body, a set of tuples that all have the domain H.

Such a relation closely corresponds to what is usually called the extension of a predicate in first-order logic except that here we identify the places in the predicate with attribute names. Usually in the relational model a database schema is said to consist of a set of relation names, the headers that are associated with these names and the constraints that should hold for every instance of the database schema.

Relation universe

A relation universe U over a header H is a non-empty set of relations with header H.

Relation schema

A relation schema (H, C) consists of a header H and a predicate $C(R)$ that is defined for all relations R with header H. A relation satisfies a relation schema (H, C) if it has header H and satisfies C.

Key constraints and functional dependencies

One of the simplest and most important types of relation constraints is the *key constraint*. It tells us that in every instance of a certain relational schema the tuples can be identified by their values for certain attributes.

Superkey

A superkey is written as a finite set of attribute names.

A superkey K holds in a relation (H, B) if:

- $K \subseteq H$ and
- there exist no two distinct tuples $t_1, t_2 \in B$ such that $t_1[K] = t_2[K]$.

A superkey holds in a relation universe U if it holds in all relations in U.

Theorem: A superkey K holds in a relation universe U over H if and only if $K \subseteq H$ and $K \to H$ holds in U.

Candidate key

A superkey K holds as a candidate key for a relation universe U if it holds as a superkey for U and there is no proper subset of K that also holds as a superkey for U.

Functional dependency

A functional dependency (FD for short) is written as $X \to Y$ for X, Y finite sets of attribute names.

A functional dependency $X \to Y$ holds in a relation (H, B) if:

- $X, Y \subseteq H$ and
- $\forall$ tuples $t_1, t_2 \in B$, $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$

A functional dependency $X \to Y$ holds in a relation universe U if it holds in all relations in U.

Trivial functional dependency

A functional dependency is trivial under a header H if it holds in all relation universes over H.

Theorem: An FD $X \to Y$ is trivial under a header H if and only if $Y \subseteq X \subseteq H$.

Closure

Armstrong's axioms: The closure of a set of FDs S under a header H, written as S^+, is the smallest superset of S such that:

- $Y \subseteq X \subseteq H \Rightarrow X \to Y \in S^+$ (reflexivity)

- $X \to Y \in S^+ \wedge Y \to Z \in S^+ \Rightarrow X \to Z \in S^+$ (transitivity) and
- $X \to Y \in S^+ \wedge Z \subseteq H \Rightarrow (X \cup Z) \to (Y \cup Z) \in S^+$ (augmentation)

  Theorem: Armstrong's axioms are sound and complete; given a header H and a set S of FDs that only contain subsets of H, $X \to Y \in S^+$ if and only if $X \to Y$ holds in all relation universes over H in which all FDs in S hold.

Completion

The completion of a finite set of attributes X under a finite set of FDs S, written as X^+, is the smallest superset of X such that:

- $Y \to Z \in S \wedge Y \subseteq X^+ \Rightarrow Z \subseteq X^+$

  The completion of an attribute set can be used to compute if a certain dependency is in the closure of a set of FDs.

  Theorem: Given a set S of FDs, $X \to Y \in S^+$ if and only if $Y \subseteq X^+$.

Irreducible cover

An irreducible cover of a set S of FDs is a set T of FDs such that:

- $S^+ = T^+$
- there exists no $U \subset T$ such that $S^+ = U^+$
- $X \to Y \in T \Rightarrow Y$ is a singleton set and
- $X \to Y \in T \wedge Z \subset X \Rightarrow Z \to Y \notin S^+$.

Algorithm to derive candidate keys from functional dependencies

```
INPUT: a set S of FDs that contain only subsets of a header H
OUTPUT: the set C of superkeys that hold as candidate keys in
        all relation universes over H in which all FDs in S hold
begin
  C := ∅;          // found candidate keys
  Q := { H };      // superkeys that contain candidate keys
  while Q <> ∅ do
    let K be some element from Q;
    Q := Q − { K };
    minimal := true;
    for each X->Y in S do
      K' := (K − Y) ∪ X;   // derive new superkey
      if K' ⊂ K then
        minimal := false;
        Q := Q ∪ { K' };
      end if
    end for
    if minimal and there is not a subset of K in C then
      remove all supersets of K from C;
      C := C ∪ { K };
    end if
  end while
end
```

References

[1] *"Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks"*, E.F. Codd, IBM Research Report, 1969

[2] Data Integration Glossary (http://knowledge.fhwa.dot.gov/tam/aashto.nsf/All+Documents/4825476B2B5C687285256B1F00544258/$FILE/DIGloss.pdf), U.S. Department of Transportation, August 2001.

[3] E. F. Codd, The Relational Model for Database Management, Addison-Wesley Publishing Company, 1990, ISBN 0-201-14192-2

[4] David M. Kroenke, *Database Processing: Fundamentals, Design, and Implementation* (1997), Prentice-Hall, Inc., pages 130–144

Further reading

- Date, C. J.; Darwen, Hugh (2000). *Foundation for future database systems : the third manifesto ; a detailed study of the impact of type theory on the relational model of data, including a comprehensive model of type inheritance* (2. ed. ed.). Reading, Mass. [u.a.]: Addison-Wesley. ISBN 0-201-70928-7.
- Date, C. J. (2007). *An Introduction to Database Systems* (8 ed.). Boston [u.a.]: Pearson Education. ISBN 0-321-19784-4.

External links

- Feasibility of a set-theoretic data structure : a general structure based on a reconstituted definition of relation (http://hdl.handle.net/2027.42/4164) (Childs' 1968 research cited by Codd's 1970 paper)
- The Third Manifesto (TTM) (http://www.thethirdmanifesto.com/)
- Relational Databases (http://www.dmoz.org/Computers/Software/Databases/Relational) on the Open Directory Project
- Relational Model (http://c2.com/cgi/wiki?RelationalModel)
- Binary relations and tuples compared with respect to the semantic web (http://blogs.sun.com/bblfish/entry/why_binary_relations_beat_tuples)

Relational database

A **relational database** is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model. Data in a single table represents a relation, from which the name of the database type comes. In typical solutions, tables may have additionally defined relationships with each other.

In the relational model, each table schema must identify a column or group of columns, called the *primary key*, to uniquely identify each row. A relationship can then be established between each row in the table and a row in another table by creating a *foreign key*, a column or group of columns in one table that points to the primary key of another table. The relational model offers various levels of refinement of table organization and reorganization called database normalization. (See Normalization below.) The database management system (DBMS) of a relational database is called an RDBMS, and is the software of a relational database.

In relational databases, each data item has a row of attributes, so the database displays a fundamentally tabular organization. The table goes down a row of items (the records) and across many columns of attributes or fields. The same data (along with new and different attributes) can be organized into different tables.

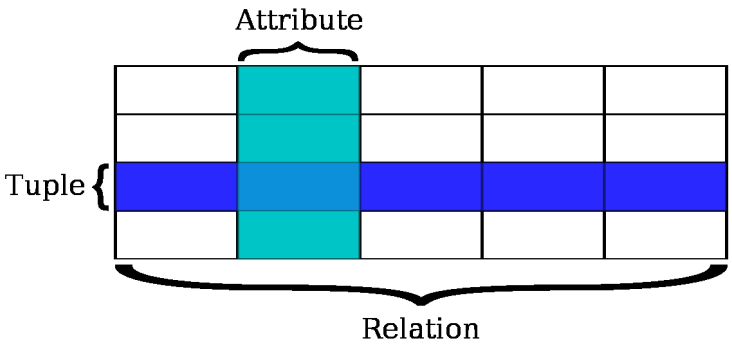
The term relational does not just refer to relationships between tables: firstly, it refers to the table itself[*citation needed*], or rather, the relationship between columns within a table; and secondly, it refers to links between tables.

Important columns in any relational database's tables will be a column whose entry (customer ID, serial number) can uniquely identify any particular item or record (the primary key), and any column(s) that link to other tables (the foreign key(s)). The size and complexity of relational databases typically requires stored procedures to support the relationships and provide access (interfaces) to external programs which, for example, "query" the relational database to retrieve and present selected data.

Relational databases are both created and queried by DataBase Management Systems (DBMSs). Relational databases displaced hierarchical databases because the ability to add new relations made it possible to add new information that was valuable but "broke" a database's original hierarchical conception. The trend continues as a networked planet and social media create the world of "big data" which is larger and less structured than the datasets and tasks that relational databases handle well (it is instructive to compare Hadoop).

Terminology

The relational database was first defined in June 1970 by Edgar Codd, of IBM's San Jose Research Laboratory. Codd's view of what qualifies as an RDBMS is summarized in Codd's 12 rules. A relational database has become the predominant choice in storing data. Other models besides the *relational model* include the hierarchical database model and the network model.



Relational database terminology.

The table below summarizes some of the most important relational database terms and their SQL equivalents.

SQL term	Relational database term	Description
Row	***Tuple*** or ***record***	A data set representing a single item
Column	***Attribute*** or ***field***	A labeled element of a tuple, e.g. "Address" or "Date of birth"
Table	***Relation*** or ***Base relvar***	A set of tuples sharing the same attributes; a set of columns and rows
View or ***result set***	***Derived relvar***	Any set of tuples; a data report from the RDBMS in response to a query

Relations or Tables

A *relation* is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints.

The relational model specifies that the tuples of a relation have no specific order and that the tuples, in turn, impose no order on the attributes. Applications access data by specifying queries, which use operations such as *select* to identify tuples, *project* to identify attributes, and *join* to combine relations. Relations can be modified using the *insert*, *delete*, and *update* operators. New tuples can supply explicit values or be derived from a query. Similarly, queries identify tuples for updating or deleting.

Tuples by definition are unique. If the tuple contains a candidate or primary key then obviously it is unique; however, a primary key need not be defined for a row or record to be a tuple. The definition of a tuple requires that it be unique, but does not require a primary key to be defined. Because a tuple is unique, its attributes by definition constitute a superkey.

Base and derived relations

In a relational database, all data are stored and accessed via relations. Relations that store data are called "base relations", and in implementations are called "tables". Other relations do not store data, but are computed by applying relational operations to other relations. These relations are sometimes called "derived relations". In implementations these are called "views" or "queries". Derived relations are convenient in that they act as a single relation, even though they may grab information from several relations. Also, derived relations can be used as an abstraction layer.

Domain

A domain describes the set of possible values for a given attribute, and can be considered a constraint on the value of the attribute. Mathematically, attaching a domain to an attribute means that any value for the attribute must be an element of the specified set. The character data value 'ABC', for instance, is not in the integer domain, but the integer value 123 is in the integer domain.

Constraints

Constraints make it possible to further restrict the domain of an attribute. For instance, a constraint can restrict a given integer attribute to values between 1 and 10. Constraints provide one method of implementing business rules in the database. SQL implements constraint functionality in the form of check constraints. Constraints restrict the data that can be stored in relations. These are usually defined using expressions that result in a boolean value, indicating whether or not the data satisfies the constraint. Constraints can apply to single attributes, to a tuple (restricting combinations of attributes) or to an entire relation. Since every attribute has an associated domain, there are constraints (**domain constraints**). The two principal rules for the relational model are known as **entity integrity**

and **referential integrity**.

Primary key

A primary key uniquely specifies a tuple within a table. In order for an attribute to be a good primary key it must not repeat. While natural attributes (attributes used to describe the data being entered) are sometimes good primary keys, surrogate keys are often used instead. A surrogate key is an artificial attribute assigned to an object which uniquely identifies it (for instance, in a table of information about students at a school they might all be assigned a student ID in order to differentiate them). The surrogate key has no intrinsic (inherent) meaning, but rather is useful through its ability to uniquely identify a tuple. Another common occurrence, especially in regards to N:M cardinality is the composite key. A composite key is a key made up of two or more attributes within a table that (together) uniquely identify a record. (For example, in a database relating students, teachers, and classes. Classes *could* be uniquely identified by a composite key of their room number and time slot, since no other class could have exactly the same combination of attributes. In fact, use of a composite key such as this can be a form of data verification, albeit a weak one.)

Foreign key

A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables. Foreign keys need not have unique values in the referencing relation. Foreign keys effectively use the values of attributes in the referenced relation to restrict the domain of one or more attributes in the referencing relation. A foreign key could be described formally as: "For all tuples in the referencing relation projected over the referencing attributes, there must exist a tuple in the referenced relation projected over those same attributes such that the values in each of the referencing attributes match the corresponding values in the referenced attributes."

Stored procedures

A stored procedure is executable code that is associated with, and generally stored in, the database. Stored procedures usually collect and customize common operations, like inserting a tuple into a relation, gathering statistical information about usage patterns, or encapsulating complex business logic and calculations. Frequently they are used as an application programming interface (API) for security or simplicity. Implementations of stored procedures on SQL RDBMSs often allow developers to take advantage of procedural extensions (often vendor-specific) to the standard declarative SQL syntax. Stored procedures are not part of the relational database model, but all commercial implementations include them.

Index

An index is one way of providing quicker access to data. Indices can be created on any combination of attributes on a relation. Queries that filter using those attributes can find matching tuples randomly using the index, without having to check each tuple in turn. This is analogous to using the index of a book to go directly to the page on which the information you are looking for is found, so that you do not have to read the entire book to find what you are looking for. Relational databases typically supply multiple indexing techniques, each of which is optimal for some combination of data distribution, relation size, and typical access pattern. Indices are usually implemented via B+ trees, R-trees, and bitmaps. Indices are usually not considered part of the database, as they are considered an implementation detail, though indices are usually maintained by the same group that maintains the other parts of the database. It should be noted that use of efficient indexes on both primary and foreign keys can dramatically improve query performance. This is because B-tree indexes result in query times proportional to log(n) where n is the number of rows in a table and hash indexes result in constant time queries (no size dependency so long as the relevant part of the index fits into memory).

Relational operations

Queries made against the relational database, and the derived relvars in the database are expressed in a relational calculus or a relational algebra. In his original relational algebra, Codd introduced eight relational operators in two groups of four operators each. The first four operators were based on the traditional mathematical set operations:

- The union operator combines the tuples of two relations and removes all duplicate tuples from the result. The relational union operator is equivalent to the SQL UNION operator.
- The intersection operator produces the set of tuples that two relations share in common. Intersection is implemented in SQL in the form of the INTERSECT operator.
- The difference operator acts on two relations and produces the set of tuples from the first relation that do not exist in the second relation. Difference is implemented in SQL in the form of the EXCEPT or MINUS operator.
- The cartesian product of two relations is a join that is not restricted by any criteria, resulting in every tuple of the first relation being matched with every tuple of the second relation. The cartesian product is implemented in SQL as the CROSS JOIN operator.

The remaining operators proposed by Codd involve special operations specific to relational databases:

- The selection, or restriction, operation retrieves tuples from a relation, limiting the results to only those that meet a specific criterion, i.e. a subset in terms of set theory. The SQL equivalent of selection is the SELECT query statement with a WHERE clause.
- The projection operation extracts only the specified attributes from a tuple or set of tuples.
- The join operation defined for relational databases is often referred to as a natural join. In this type of join, two relations are connected by their common attributes. SQL's approximation of a natural join is the INNER JOIN operator. In SQL, an INNER JOIN prevents a cartesian product from occurring when there are two tables in a query. For each table added to a SQL Query, one additional INNER JOIN is added to prevent a cartesian product. Thus, for N tables in a SQL query, there must be N-1 INNER JOINS to prevent a cartesian product.
- The relational division operation is a slightly more complex operation, which involves essentially using the tuples of one relation (the dividend) to partition a second relation (the divisor). The relational division operator is effectively the opposite of the cartesian product operator (hence the name).

Other operators have been introduced or proposed since Codd's introduction of the original eight including relational comparison operators and extensions that offer support for nesting and hierarchical data, among others.

Normalization

Normalization was first proposed by Codd as an integral part of the relational model. It encompasses a set of procedures designed to eliminate nonsimple domains (non-atomic values) and the redundancy (duplication) of data, which in turn prevents data manipulation anomalies and loss of data integrity. The most common forms of normalization applied to databases are called the normal forms.

References

Relational database management system

A **relational database management system** (**RDBMS**) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd, of IBM's San Jose Research Laboratory. Many popular databases currently in use are based on the relational database model.

RDBMSs have become a predominant choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data, and much more since the 1980s. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use. However, relational databases have been challenged by object databases, which were introduced in an attempt to address the object-relational impedance mismatch in relational database, and XML databases.[*citation needed*]

Market share

According to research company Gartner, the five leading commercial relational database vendors by revenue in 2011 were Oracle (48.8%), IBM (20.2%), Microsoft (17.0%), SAP including Sybase (4.6%), and Teradata (3.7%).

The three leading open source implementations are MySQL, PostgreSQL, and SQLite. MariaDB is a prominent fork of MySQL prompted by Oracle's acquisition of MySQL AB.

According to Gartner, in 2008, the percentage of database sites using any given technology were (a given site may deploy multiple technologies):

- Oracle Database - 70%
- Microsoft SQL Server - 68%
- MySQL (Oracle Corporation) - 50%
- IBM DB2 - 39%
- IBM Informix - 18%
- SAP Sybase Adaptive Server Enterprise - 15%
- SAP Sybase IQ - 14%
- Teradata - 11%

According to DB-Engines, the most popular systems are Oracle, MySQL, Microsoft SQL Server, PostgreSQL and IBM DB2.

History

In 1974, IBM began developing System R, a research project to develop a prototype RDBMS. Its first commercial product was SQL/DS, released in 1981. However, the first commercially available RDBMS was Oracle, released in 1979 by Relational Software, now Oracle Corporation. Other examples of an RDBMS include DB2, SAP Sybase ASE, and Informix.

Historical usage of the term

The term "relational database" was invented by E. F. Codd at IBM in 1970, Codd introduced the term in his seminal paper "A Relational Model of Data for Large Shared Data Banks".[1] In this paper and later papers, he defined what he meant by "relational". One well-known definition of what constitutes a relational database system is composed of Codd's 12 rules. However, many of the early implementations of the relational model did not conform to all of Codd's rules, so the term gradually came to describe a broader class of database systems, which at a minimum:

- Present the data to the user as relations (a presentation in tabular form, i.e. as a *collection* of tables with each table consisting of a set of rows and columns);

- Provide relational operators to manipulate the data in tabular form.

The first systems that were relatively faithful implementations of the relational model were from the University of Michigan; Micro DBMS (1969), the Massachusetts Institute of Technology;[2] (1971), and from IBM UK Scientific Centre at Peterlee; IS1 (1970–72) and its followon PRTV (1973–79). The first system sold as an RDBMS was Multics Relational Data Store, first sold in 1978. Others have been Berkeley Ingres QUEL and IBM BS12. The most popular definition of an RDBMS is a product that presents a view of data as a collection of rows and columns, even if it is not based strictly upon relational theory. By this definition, RDBMS products typically implement some but not all of Codd's 12 rules.

A second school of thought argues that if a database does not implement all of Codd's rules (or the current understanding on the relational model, as expressed by Christopher J Date, Hugh Darwen and others), it is not relational. This view, shared by many theorists and other strict adherents to Codd's principles, would disqualify most DBMSs as not relational. For clarification, they often refer to some RDBMSs as *Truly-Relational Database Management Systems* (TRDBMS), naming others *Pseudo-Relational Database Management Systems* (PRDBMS).

As of 2009, most commercial relational DBMSes employ SQL as their query language.[*citation needed*] Alternative query languages have been proposed and implemented, notably the pre-1996 implementation of Berkeley Ingres QUEL.

References

[1] "A Relational Model of Data for Large Shared Data Banks" (http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf)

[2] SIGFIDET '74 Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control

Life cycle of a relational database

The **life cycle of a relational database** is the cycle of development and changes that a relational database goes through during the course of its life. The cycle typically consists of several stages. There is a possibility that the database designer/developer can go back to any of the previous stages. This represents an admission that a full understanding of a problem, and its solution is likely to evolve as the various stages of design and implementation proceed. The typical eleven stages involved in the life cycle of a relational database are as follows:

Process

1. The designer must try to obtain as complete as possible an understanding of the real world problem that is going to be helped by the introduction of a database. This understanding of the nature of the problem and the constraints and outline feasible solutions is often performed using some systems analysis methodology.
2. The entity relationship diagram is drawn, and this diagram in its modified form serves as an essential part of the logical schema. Attributes of the entity types so produced are then added. Primary and foreign keys are specified.
3. Normalization is used to check the entity-relationship model. Some splitting and even recombination of entity types may result from normalization and the entity relationship model will have to be updated accordingly. The entity relationship model and the table definitions resulting from normalization should be consistent.
4. Set of Table(s) definition for the required schema is finalized.
5. The database tables are created. Primary, Foreign keys, database constraints and database integrity rules are specified at this stage.
6. At this stage, the file organization is performed. File organization is the way the database relations are to be stored on the storage medium. The file organization is decided on the basis of maximum speed of access, the type of access required and storage space considerations. There are two factors to consider; firstly how the records are to be physically mapped onto the storage medium, and secondly which indexes are to be used and if so, which

fields (attributes, columns) are to be indexed. Indexes are designed to increase the speed of access to required records. Views can also be defined at this stage. Views are used to limit access to parts of database only, when used in conjunction with access privileges. Views also make programming simpler.

7. The designer will be able to design the required queries at this stage. The designer should have a good idea of the main types of query and reports the database will have to accommodate.
8. At this stage, application screens are designed. The application screens are used to capture the input information that will be kept in the database. Screen design is partially determined by the data items that must be input and output by particular applications and partially in human-computer interface terms. When designing screens, special consideration is given to the suggestions given by the application end users. There are published standards which can be exactly followed for screens design or organization can develop their own screen design standards as per their requirements.
9. Report design is another area where input from users is paramount. They will specify what they want to see on the reports and the format of the reports and in the case of regular reports, when they should be produced. Nowadays most of the application design tools provide easy to use friendly tools for quick reports development. e.g. report builder in Oracle, Crystal Reports, R&R Report Writer etc.
10. Testing is performed at this stage. Application screens, various functions offered by the application screens, data validations through screens and reports are tested and it serves as the ultimate test of the correctness of the database schema and the viability of the system as a whole. It is recommended to create a test database separate of the production database. The test database will be useful for testing any schema changes and new and modified application before applying the changes to the production (live) database. Careful testing of the system before handover will minimize the expense of later modifications to the schema and major applications.
11. The final stage is Handover. This is the stage where the users receive the finished database and applications and begin data entry. In practice, it is likely that the core of the system will be handed over to users and later extensions to the system will be implemented.

Sources

- Open University Data base development life cycle [1]

References

[1] http://www.open.edu/openlearn/science-maths-technology/computing-and-ict/information-and-communication-technologies/the-database-development-life-cycle/content-section-0

Logical data model

In computing and more specifically systems engineering, a **logical data model** (**LDM**) is a type of data model showing a detailed representation of some or all of an organization's data, independent of any particular data management technology, and described in business language. It is typically represented as a diagram, organized in terms of entities and relationships, with underlying definitions.

Overview

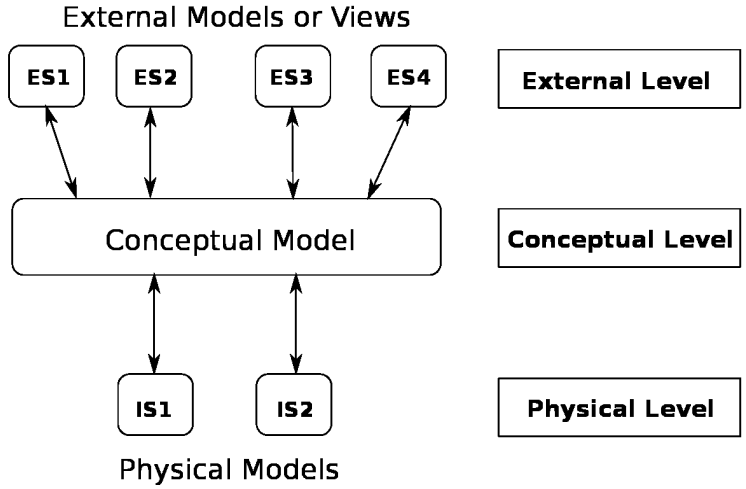
Logical data models represent the abstract structure of a domain of information. They are often diagrammatic in nature and are most typically used in business processes that seek to capture things of importance to an organization and how they relate to one another. Once validated and approved, the logical data model can become the basis of a physical data model and inform the design of a database.

Logical data models should be based on the structures identified in a preceding conceptual data model, since this describes the semantics of the information context, which the logical model should also reflect. Even so, since the logical data model anticipates implementation on a specific computing system, the content of the logical data model is adjusted to achieve certain efficiencies.

The term 'Logical Data Model' is sometimes used as a synonym of 'domain model' or as an alternative to the domain model. While the two concepts are closely related, and have overlapping goals, a domain model is more focused on capturing the concepts in the problem domain rather than the structure of the data associated with that domain.

History

When ANSI first laid out the idea of a *logical schema* in 1975,[2] the choices were *hierarchical* and *network*. The relational model – where data is described in terms of tables and columns – had just been recognized as a data organization theory but no software existed to support that approach. Since that time, an object-oriented approach to data modelling – where data is described in terms of classes, attributes, and associations – has also been introduced.



The ANSI/SPARC three level architecture, which "shows that a data model can be an external model (or view), a conceptual model, or a physical model. This is not the only way to look at data models, but it is a useful way, particularly when comparing models".[1]

Logical data model topics

Reasons for building a logical data model

- Helps common understanding of business data elements and requirements
- Provides foundation for designing a database
- Facilitates avoidance of data redundancy and thus prevent data & business transaction inconsistency
- Facilitates data re-use and sharing
- Decreases development and maintenance time and cost

- Confirms a logical process model and helps impact analysis.

Conceptual, logical and physical data model

A logical data model is sometimes incorrectly called a physical data model, which is not what the ANSI people had in mind. The physical design of a database involves deep use of particular database management technology. For example, a table/column design could be implemented on a collection of computers, located in different parts of the world. That is the domain of the physical model.

Conceptual, Logical and physical data models are very different in their objectives, goals and content. Key differences noted below.

Conceptual Data Model (CDM)	Logical Data Model (LDM)	Physical Data Model (PDM)
Includes high-level data constructs	Includes entities (tables), attributes (columns/fields) and relationships (keys)	Includes tables, columns, keys, data types, validation rules, database triggers, stored procedures, domains, and access constraints
Non-technical names, so that executives and managers at all levels can understand the data basis of Architectural Description	Uses business names for entities & attributes	Uses more defined and less generic specific names for tables and columns, such as abbreviated column names, limited by the database management system (DBMS) and any company defined standards
Uses general high-level data constructs from which Architectural Descriptions are created in non-technical terms	Is independent of technology (platform, DBMS)	Includes primary keys and indices for fast data access.
May not be normalized	Is normalized to fourth normal form (4NF)	May be de-normalized to meet performance requirements based on the nature of the database. If the nature of the database is Online Transaction Processing (OLTP) or Operational Data Store (ODS) it is usually not de-normalized. De-normalization is common in Datawarehouses.
Represented in the DIV-1 Viewpoint (DoDAF V2.0)	Represented in the DIV-2 Viewpoint (DoDAF V2.0), and OV-7 View (DoDAF V1.5)	Represented in the DIV-3 Viewpoint (DoDAF V2.0), and SV-11 View (DoDAF V1.5)

References

[1] Matthew West and Julian Fowler (1999). Developing High Quality Data Models (http://www.matthew-west.org.uk/documents/princ03.pdf). The European Process Industries STEP Technical Liaison Executive (EPISTLE).

[2] American National Standards Institute. 1975. “ANSI/X3/SPARC Study Group on Data Base Management Systems; Interim Report”. FDT(Bulletin of ACM SIGMOD) 7:2.

External links

- Building a Logical Data Model (http://replay.web.archive.org/20080509063521/http://www.dbmsmag.com/9506d16.html) By George Tillmann, DBMS, June 1995.

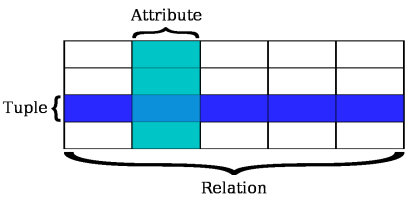
Logical schema

A **logical schema** is a data model of a specific problem domain expressed in terms of a particular data management technology.

Without being specific to a particular database management product, it is in terms of either relational tables and columns, object-oriented classes, or XML tags. This is as opposed to a conceptual data model, which describes the semantics of an organization without reference to technology, or a physical data model, which describes the particular physical mechanisms used to capture data in a storage medium.

The next step in creating a database, after the logical schema is produced, is to create the physical schema.

Relation (database)



Relation, tuple, and attribute represented as table, row, and column.

In relational database theory, a **relation**, as originally defined by E.F. Codd, is a set of tuples $(d_1, d_2, ..., d_n)$, where each element d_j is a member of D_j, a data domain. Codd's original definition notwithstanding, and contrary to the usual definition in mathematics, there is no ordering to the elements of the tuples of a relation. Instead, each element is termed an **attribute value**. An **attribute** is a name paired with a domain (nowadays more commonly referred to as **type** or **data type**). An **attribute value** is an attribute name paired with an element of that attribute's domain, and a tuple is a *set* of attribute values in which no two distinct elements have the same name. Thus, in some accounts, a tuple is described as a function, mapping names to values.

A set of attributes in which no two distinct elements have the same name is called a **heading**. A set of tuples having the same heading is called a **body**. A relation is thus a heading paired with a body, the heading of the relation being also the heading of each tuple in its body. The number of attributes constituting a heading is called the **degree**, which term also applies to tuples and relations. The term *n*-tuple refers to a tuple of degree *n* (n>=0).

E. F. Codd used the term relation in its mathematical sense of a finitary relation, a set of tuples on some set of *n* sets $S_1, S_2, ... ,S_n$. Thus, an *n*-ary relation is interpreted, under the Closed World Assumption, as the extension of some *n*-adic predicate: all and only those *n*-tuples whose values, substituted for corresponding free variables in the predicate, yield propositions that hold true, appear in the relation.

The term relation schema refers to a heading paired with a set of constraints defined in terms of that heading. A relation can thus be seen as an instantiation of a relation schema if it has the heading of that schema and it satisfies the applicable constraints.

Sometimes a relation schema is taken to include a name. A relational database definition (database schema, sometimes referred to as a relation*al* schema) can thus be thought of as a collection of named relation schemas.

In implementations, the domain of each attribute is effectively a data type and a named relation schema is effectively a relation variable or relvar for short (see **Relation Variables** below).

In SQL, a database language for relational databases, relations are represented by tables, where each row of a table represents a single tuple, and where the values of each attribute form a column.

Examples

Below is an example of a relation having three named attributes: 'ID' from the domain of integers, and 'Name' and 'Address' from the domain of strings:

ID (Integer)	Name (String)	Address (String)
102	Yonezawa Akinori	Naha, Okinawa
202	Murata Makoto	Sendai, Miyagi
104	Sakamura Ken	Kumamoto, Kumamoto
152	Matsumoto Yukihiro	Okinawa, Okinawa

A predicate for this relation, using the attribute names to denote free variables, might be "Employee number *ID* is known as *Name* and lives at *Address*". Examination of the relation tells us that there are just four tuples for which the predicate holds true. So, for example, employee 102 is known only by that name, Yonezawa Akinori, and does not live anywhere else but in Naha, Okinawa. Also, apart from the four employees shown,there is no other employee who has both a name and an address.

Under the definition of **body**, the tuples of a body do not appear in any particular order - one cannot say "The tuple of 'Murata Makoto' is above the tuple of 'Matsumoto Yukihiro'", nor can one say "The tuple of 'Yonezawa Akinori' is the first tuple." A similar comment applies to the rows of an SQL table.

Under the definition of **heading** the elements of a element do not appear in any particular order either, nor, therefore do the elements of a tuple. A similar comment does *not* apply here to SQL, which does define an ordering to the columns of a table.

Relation Variables

A relational database consists of named relation variables (relvars) for the purposes of updating the database in response to changes in the real world. An update to a single relvar causes the body of the relation assigned to that variable to be replaced by a different set of tuples. Such variables are classified into two classes: **base relation variables** and **derived relation variables**, the latter also known as **virtual relvars** but usually referred to by the short term **view**.

A **base relation variable** is a relation variable which is not derived from any other relation variables. In SQL the term **base table** equates approximately to base relation variable.

A view can be defined by an expression using the operators of the relational algebra or the relational calculus. Such an expression operates on one or more relations and when evaluated yields another relation. The result is sometimes referred to as a "derived" relation when the operands are relations assigned to database variables. A view is defined by giving a name to such an expression, such that the name can subsequently be used as a variable name. (Note that the expression must then mention at least one base relation variable.)

By using a Data Definition Language (DDL), it is able to define base relation variables. In SQL, `CREATE TABLE` syntax is used to define base tables. The following is an example.

```
CREATE TABLE List_of_people (
 ID INTEGER,
 Name CHAR(40),
 Address CHAR(200),
 PRIMARY KEY (ID)
)
```

The Data Definition Language (DDL) is also used to define derived relation variables. In SQL, `CREATE VIEW` syntax is used to define a derived relation variable. The following is an example.

```
CREATE VIEW List_of_Okinawa_people AS (
 SELECT ID, Name, Address
  FROM List_of_people
  WHERE Address LIKE '%, Okinawa'
)
```

References

- Date, C. J. (2004). *An Introduction to Database Systems* (8 ed.). Addison–Wesley. ISBN 0-321-19784-4.

Table (database)

In relational databases and flat file databases, a **table** is an organized set of data elements (values) using a model of vertical columns (which are identified by their name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a unique key index.

Table is another term for relation; although there is the difference in that a table is usually a multiset (bag) of rows whereas a relation is a set and does not allow duplicates. Besides the actual data rows, tables generally have associated with them some metadata, such as constraints on the table or on the values within particular columns.Wikipedia:Disputed statement|

The data in a table does not have to be physically stored in the database. Views are also relational tables, but their data are calculated at query time. Another example are nicknames[*clarify*], which represent a pointer to a table in another database.

Comparisons

In non-relational systems, hierarchical databases, the distant counterpart of a table is a structured file, representing the rows of a table in each record of the file and each column in a record.This structure implies that a record can have repeating information, Generally in the child data segments.Data are stored in sequence of records which are equivalent to table term of a relational database.with each record having equivalent rows.

Unlike a spreadsheet, the datatype of field is ordinarily defined by the schema describing the table. Some SQL systems, such as SQLite, are less strict about field datatype definitions.

Tables versus relations

In terms of the relational model of databases, a table can be considered a convenient representation of a relation, but the two are not strictly equivalent. For instance, an SQL table can potentially contain duplicate rows, whereas a true relation cannot contain duplicate tuples. Similarly, representation as a table implies a particular ordering to the rows and columns, whereas a relation is explicitly unordered. However, the database system does not guarantee any ordering of the rows unless an `ORDER BY` clause is specified in the `SELECT` statement that queries the table.

An equally valid representations of a relation is as an n-dimensional chart, where n is the number of attributes (a table's columns). For example, a relation with two attributes and three values can be represented as a table with two columns and three rows, or as a two-dimensional graph with three points. The table and graph representations are only equivalent if the ordering of rows is not significant, and the table has no duplicate rows.

Table types

Two types of tables exist:

- A relational table, which is the basic structure to hold user data in a relational database.
- An object table, which is a table that uses an object type to define a column. It is defined to hold instances of objects of a defined type.

In SQL, the `CREATE TABLE` statement creates these tables.

References

Tuple

In mathematics, computer science, linguistics,[1] and philosophy[2] a **tuple** is an ordered list of elements. In set theory, an **(ordered) n-tuple** is a sequence (or ordered list) of n elements, where n is a non-negative integer. There is only one 0-tuple, an empty sequence. An n-tuple is defined inductively using the construction of an ordered pair. Tuples are usually written by listing the elements within parentheses " $(\ )$ " and separated by commas; for example, $(2, 7, 4, 1, 7)$ denotes a 5-tuple. Sometimes other delimiters are used, such as square brackets " $[\ ]$ " or angle brackets " $\langle\ \rangle$ ". Braces " $\{\}$ " are almost never used for tuples, as they are the standard notation for sets. Tuples are often used to describe other mathematical objects, such as vectors. In computer science, tuples are directly implemented as product types in most functional programming languages. More commonly, they are implemented as record types, where the components are labeled instead of being identified by position alone. This approach is also used in relational algebra. Tuples are also used in relation to programming the semantic web with Resource Description Framework or RDF.

Etymology

The term originated as an abstraction of the sequence: single, double, triple, quadruple, quintuple, sextuple, septuple, octuple, ..., *n*-tuple, ..., where the prefixes are taken from the Latin names of the numerals. The unique 0-tuple is called the null tuple. A 1-tuple is called a singleton, a 2-tuple is called an ordered pair and a 3-tuple is a triple or triplet. *n* can be any nonnegative integer. For example, a complex number can be represented as a 2-tuple, a quaternion can be represented as a 4-tuple, an octonion can be represented as an 8-tuple and a sedenion can be represented as a 16-tuple.

Although these uses treat *-tuple* as the suffix, the original suffix was *-ple* as in "triple" (three-fold) or "decuple" (ten-fold). This originates from a medieval Latin suffix *-plus* (meaning "more") related to Greek -πλοῦς, which replaced the classical and late antique *-plex* (meaning "folded"), as in "duplex".[3]

Names for tuples of specific lengths

Tuple Length n	Name	Alternate names
0	empty tuple	unit
1	single	Monad
2	double	pair / dual / twin
3	triple	treble
4	quadruple	quad
5	quintuple	pentuple
6	sextuple	hextuple
7	septuple	
8	octuple	
9	nonuple	
10	decuple	
11	undecuple	hendecuple
12	duodecuple	
13	tredecuple	
100	centuple	

Properties

The general rule for the identity of two n-tuples is

$$(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n) \text{ if and only if } a_1 = b_1,\ a_2 = b_2,\ \ \ldots,\ \ a_n = b_n.$$

Thus a tuple has properties that distinguish it from a set.

1. A tuple may contain multiple instances of the same element, so tuple $(1, 2, 2, 3) \neq (1, 2, 3)$; but set $\{1, 2, 2, 3\} = \{1, 2, 3\}$.
2. Tuple elements are ordered: tuple $(1, 2, 3) \neq (3, 2, 1)$, but set $\{1, 2, 3\} = \{3, 2, 1\}$.
3. A tuple has a finite number of elements, while a set or a multiset may have an infinite number of elements.

Definitions

There are several definitions of tuples that give them the properties described in the previous section.

Tuples as functions

An n-tuple can be regarded as a function, *F*, whose domain is the tuple's implicit set of element indices, *X*, and whose codomain, *Y*, is the tuple's set of elements. Formally:

$$(a_1, a_2, \ldots, a_n) \equiv (X, Y, F)$$

where:

$$X = \{1, 2, \ldots, n\}$$
$$Y = \{a_1, a_2, \ldots, a_n\}$$
$$F = \{(1, a_1), (2, a_2), \ldots, (n, a_n)\}$$

Tuples as nested ordered pairs

Another way of formalizing tuples is as nested ordered pairs. This approach assumes that the notion of ordered pair has already been defined; thus a 2-tuple

1. The 0-tuple (i.e. the empty tuple) is represented by the empty set $\emptyset$.
2. An n -tuple, with $n > 0$, can be defined as an ordered pair of its first entry and an $(n-1)$-tuple (which contains the remaining entries when $n > 1$):

$$(a_1, a_2, a_3, \ldots, a_n) = (a_1, (a_2, a_3, \ldots, a_n))$$

This definition can be applied recursively to the $(n-1)$-tuple:

$$(a_1, a_2, a_3, \ldots, a_n) = (a_1, (a_2, (a_3, (\ldots, (a_n, \emptyset)\ldots))))$$

Thus, for example:

$$(1, 2, 3) = (1, (2, (3, \emptyset)))$$
$$(1, 2, 3, 4) = (1, (2, (3, (4, \emptyset))))$$

A variant of this definition starts "peeling off" elements from the other end:

1. The 0-tuple is the empty set $\emptyset$.
2. For $n > 0$:

$$(a_1, a_2, a_3, \ldots, a_n) = ((a_1, a_2, a_3, \ldots, a_{n-1}), a_n)$$

This definition can be applied recursively:

$$(a_1, a_2, a_3, \ldots, a_n) = ((\ldots(((\emptyset, a_1), a_2), a_3), \ldots), a_n)$$

Thus, for example:

$$(1, 2, 3) = (((\emptyset, 1), 2), 3)$$
$$(1, 2, 3, 4) = ((((\emptyset, 1), 2), 3), 4)$$

Tuples as nested sets

Using Kuratowski's representation for an ordered pair, the second definition above can be reformulated in terms of pure set theory:

1. The 0-tuple (i.e. the empty tuple) is represented by the empty set $\emptyset$;
2. Let x be an n -tuple $(a_1, a_2, \ldots, a_n)$, and let $x \to b \equiv (a_1, a_2, \ldots, a_n, b)$. Then, $x \to b \equiv \{\{x\}, \{x, b\}\}$. (The right arrow, $\to$, could be read as "adjoined with".)

In this formulation:

$$\begin{aligned}
() &= \emptyset \\
(1) &= () \to 1 = \{\{()\}, \{(), 1\}\} \\
&= \{\{\emptyset\}, \{\emptyset, 1\}\} \\
(1, 2) &= (1) \to 2 = \{\{(1)\}, \{(1), 2\}\} \\
&= \{\{\{\emptyset\}, \{\emptyset, 1\}\}\}, \\
&\quad \{\{\{\emptyset\}, \{\emptyset, 1\}\}, 2\}\} \\
(1, 2, 3) &= (1, 2) \to 3 = \{\{(1, 2)\}, \{(1, 2), 3\}\} \\
&= \{\{\{\{\{\emptyset\}, \{\emptyset, 1\}\}\}\}, \\
&\quad \{\{\{\emptyset\}, \{\emptyset, 1\}\}, 2\}\}\}, \\
&\quad \{\{\{\{\emptyset\}, \{\emptyset, 1\}\}\}, \\
&\quad \{\{\{\emptyset\}, \{\emptyset, 1\}\}, 2\}\}, 3\}\}
\end{aligned}$$

Relational model

In database theory, the relational model uses a tuple definition similar to tuples as functions, but each tuple element is identified by a distinct name, called an *attribute*, instead of a number; this leads to a more user-friendly and practical notation.[4] A tuple in the relational model is formally defined as a finite function that maps attributes to values. For example:

(player : "Harry", score : 25)

In this notation, attribute–value pairs may appear in any order. The distinction between tuples in the relational model and those in set theory is only superficial; the above example can be interpreted as a 2-tuple if an arbitrary total order is imposed on the attributes (e.g. $player \leq score$) and then the elements are distinguished by this ordering rather than by the attributes themselves. Conversely, a 2-tuple may be interpreted as relational model tuple over the attributes $\{1, 2\}$.

In the relational model, a relation is a (possibly empty) finite set of tuples all having the same finite set of attributes. This set of attributes is more formally called the sort of the relation, or more casually referred to as the set of column names. A tuple is usually implemented as a row in a database table, but see relational algebra for means of deriving tuples not physically represented in a table.

Type theory

In type theory, commonly used in programming languages, a tuple has a product type; this fixes not only the length, but also the underlying types of each component. Formally:

$$(x_1, x_2, \ldots, x_n) : \mathsf{T}_1 \times \mathsf{T}_2 \times \ldots \times \mathsf{T}_n$$

and the projections are term constructors:

$$\pi_1(x) : \mathsf{T}_1,\ \pi_2(x) : \mathsf{T}_2,\ \ \ldots,\ \pi_n(x) : \mathsf{T}_n$$

The tuple with labeled elements used in the relational model has a record type. Both of these types can be defined as simple extensions of the simply typed lambda calculus.

The notion of a tuple in type theory and that in set theory are related in the following way: If we consider the natural model of a type theory, and use the Scott brackets to indicate the semantic interpretation, then the model consists of some sets $S_1, S_2, \ldots, S_n$ (note: the use of italics here that distinguishes sets from types) such that:

$$[\![\mathsf{T}_1]\!] = S_1,\ [\![\mathsf{T}_2]\!] = S_2,\ \ \ldots,\ [\![\mathsf{T}_n]\!] = S_n$$

and the interpretation of the basic terms is:

$$[\![x_1]\!] \in [\![\mathsf{T}_1]\!],\ [\![x_2]\!] \in [\![\mathsf{T}_2]\!],\ \ \ldots,\ [\![x_n]\!] \in [\![\mathsf{T}_n]\!].$$

The n-tuple of type theory has the natural interpretation as an n-tuple of set theory:[5]

$$[\![(x_1, x_2, \ldots, x_n)]\!] = ([\![x_1]\!], [\![x_2]\!], \ldots, [\![x_n]\!])$$

The unit type has as semantic interpretation the 0-tuple.

Notes

[1] http://www.oxfordreference.com/view/10.1093/acref/9780199202720.001.0001/acref-9780199202720-e-2276
[2] http://www.oxfordreference.com/view/10.1093/acref/9780199541430.001.0001/acref-9780199541430-e-2262
[3] *OED*, *s.v.* "triple", "quadruple", "quintuple", "decuple"
[4] Serge Abiteboul, Richard Hull, Victor Vianu, *Foundations of databases*, Addison-Wesley, 1995, ISBN 0-201-53771-0, p. 29–33
[5] Steve Awodey, *From sets, to types, to categories, to sets* (http://www.andrew.cmu.edu/user/awodey/preprints/stcsFinal.pdf), 2009, preprint

References

The set theory definitions herein are found in any textbook on the topic, e.g.

- Abraham Adolf Fraenkel, Yehoshua Bar-Hillel, Azriel Lévy, *Foundations of set theory* (http://books.google.com/books?q=Foundations+of+set+theory&btnG=Search+Books), Elsevier Studies in Logic Vol. 67, Edition 2, revised, 1973, ISBN 0-7204-2270-1, p. 33
- Gaisi Takeuti, W. M. Zaring, *Introduction to Axiomatic Set Theory*, Springer GTM 1, 1971, ISBN 978-0-387-90024-7, p. 14
- George J. Tourlakis, *Lecture Notes in Logic and Set Theory. Volume 2: Set theory* (http://books.google.com/books?as_isbn=9780521753746), Cambridge University Press, 2003, ISBN 978-0-521-75374-6, pp. 182–193
- Keith Devlin, *The Joy of Sets*. Springer Verlag, 2nd ed., 1993, ISBN 0-387-94094-4, pp. 7–8

Row (database)

In the context of a relational database, a **row**—also called a **record** —represents a single, implicitly structured data item in a table. In simple terms, a database table can be thought of as consisting of *rows* and columns or fields. Each row in a table represents a set of related data, and every row in the table has the same structure.

For example, in a table that represents companies, each row would represent a single company. Columns might represent things like company name, company street address, whether the company is publicly held, its VAT number, etc.. In a table that represents *the association* of employees with departments, each row would associate one employee with one department.

In a less formal usage, e.g. for a database which is not formally relational, a record is equivalent to a row as described above, but is not usually referred to as a row.

The implicit structure of a row, and the meaning of the data values in a row, requires that the row be understood as providing a succession of data values, one in each column of the table. The row is then interpreted as a relvar composed of a set of tuples, with each tuple consisting of the two items: the name of the relevant column and the value this row provides for that column.

Each column expects a data value of a particular type. For example, one column might require a unique identifier, another might require text representing a person's name, another might require an integer representing hourly pay in cents.

-	Column 1	Column 2
Row (Record) 1	Row 1, Column (Field)1	Row 1, Column 2
Row 2	Row 2, Column 1	Row 2, Column 2
Row 3	Row 3, Column 1	Row 3, Column 2

Attribute domain

In computing, the **attribute domain** is the set of values allowed in an attribute.

For example:

```
Rooms in hotel (1-300)
Age (1-99)
Married (yes or no)
Nationality (Nepalese, Indian, American, or British)
Colors(Red , Yellow , Green)
```

For the relational model it is a requirement that each part of a tuple be atomic. The consequence is that each value in the tuple must be of some basic type, like a string or an integer. For the elementary type to be atomic it cannot be broken into more pieces. Alas, the domain is an elementary type, and attribute domain the domain a given attribute belongs to an abstraction belonging to or characteristic of an entity.

Candidate key

In the relational model of databases, a **candidate key** of a relation is a minimal superkey for that relation; that is, a set of attributes such that

1. the relation does not have two distinct tuples (i.e. rows or records in common database language) with the same values for these attributes (which means that the set of attributes is a superkey)
2. there is no proper subset of these attributes for which (1) holds (which means that the set is minimal).

The constituent attributes are called **prime attributes**. Conversely, an attribute that does not occur in ANY candidate key is called a **non-prime attribute**.

Since a relation contains no duplicate tuples, the set of all its attributes is a superkey if NULL values are not used. It follows that every relation will have at least one candidate key.

The candidate keys of a relation tell us all the possible ways we can identify its tuples. As such they are an important concept for the design of database schema.

Example

The definition of candidate keys can be illustrated with the following (abstract) example. Consider a relation variable (relvar) *R* with attributes (*A*, *B*, *C*, *D*) that has only the following two legal values *r1* and *r2*:

r1

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d1
a2	b1	c2	d1

r2

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d1
a1	b1	c2	d2

Here *r2* differs from *r1* only in the **A** and **D** values of the last tuple.

For *r1* the following sets have the uniqueness property, i.e., there are no two distinct tuples in the instance with the same values for the attributes in the set:

{A,B}, {A,C}, {B,C}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}

For *r2* the uniqueness property holds for the following sets;

{B,C}, {B,D}, {C,D}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}

Since superkeys of a relvar are those sets of attributes that have the uniqueness property for *all* legal values of that relvar and because we assume that *r1* and *r2* are all the legal values that *R* can take, we can determine the set of superkeys of *R* by taking the intersection of the two lists:

{B,C}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}

Finally we need to select those sets for which there is no proper subset in the list, which are in this case:

{B,C}, {A,B,D}, {A,C,D}

These are indeed the candidate keys of relvar *R*.

We have to consider *all* the relations that might be assigned to a relvar to determine whether a certain set of attributes is a candidate key. For example, if we had considered only *r1* then we would have concluded that {A,B} is a candidate key, which is incorrect. However, we *might* be able to conclude from such a relation that a certain set is *not* a candidate key, because that set does not have the uniqueness property (example {A,D} for *r1*). Note that the existence of a proper subset of a set that has the uniqueness property *cannot* in general be used as evidence that the superset is not a candidate key. In particular, note that in the case of an empty relation, every subset of the heading has the uniqueness property, including the empty set.

Determining candidate keys

The set of all candidate keys can be computed e.g. from the set of functional dependencies. To this end we need to define the attribute closure α^+ for an attribute set α. The set α^+ contains all attributes that are functionally implied by α.

It is quite simple to find a single candidate key. We start with a set α of attributes and try to remove successively each attribute. If after removing an attribute the attribute closure stays the same, then this attribute is not necessary and we can remove it permanently. We call the result $\text{minimize}(\alpha)$. If α is the set of all attributes, then $\text{minimize}(\alpha)$ is a candidate key.

Actually we can detect every candidate key with this procedure by simply trying every possible order of removing attributes. However there are much more permutations of attributes ($n!$) than subsets (2^n). That is, many attribute orders will lead to the same candidate key.

There is a fundamental difficulty for efficient algorithms for candidate key computation: Certain sets of functional dependencies lead to exponentially many candidate keys. Consider the $2 \cdot n$ functional dependencies $\{A_i \to B_i : i \in \{1, \ldots, n\}\} \cup \{B_i \to A_i : i \in \{1, \ldots, n\}\}$ which yields 2^n candidate keys: $\{A_1, B_1\} \times \ldots \times \{A_n, B_n\}$. That is, the best we can expect is an algorithm that is efficient with respect to the number of candidate keys.

The following algorithm actually runs in polynomial time in the number of candidate keys and functional dependencies:

```
K[0] := minimize(A);  /* A is the set of all attribute */
n := 1; /* Number of Keys known so far */
i := 0; /* Currently processed key */
while i < n do
  foreach α → β ∈ F do
    S := α ∪ (K[i] − β);
    found := false;
    for j := 0 to n-1 do
      if K[j] ⊆ S then found := true;
    if not found then
      K[n] := minimize(S);
      n := n + 1;
```

The idea behind the algorithm is that given a candidate key K_i and a functional dependency $\alpha \to \beta$, the reverse application of the functional dependency yields the set $\alpha \cup (K_i \setminus \beta)$, which is a key, too. It may however be covered by other already known candidate keys. (The algorithm checks this case using the 'found' variable.) If not, then minimizing the new key yields a new candidate key. The key insight is (pun not intended) that all candidate keys can be created this way.

References

- Date, Christopher (2003). "5: Integrity". *An Introduction to Database Systems*. Addison-Wesley. pp. 268–276. ISBN 978-0-321-18956-1.

External links

- Relational Database Management Systems - Database Design - Terms of Reference - Keys (http://rdbms.opengrass.net/2_Database Design/2.1_TermsOfReference/2.1.2_Keys.html): An overview of the different types of keys in an RDBMS (Relational Database Management System).

Unique key

A **unique key** or **primary key** is a key that uniquely defines the characteristics of each row. The **primary key** has to consist of characteristics that cannot collectively be duplicated by any other row. A Primary and Foreign key is needed in order to link tables.

In an entity relationship diagram of a data model, one or more **unique keys** may be declared for each data entity. Each unique key is composed from one or more data attributes of that data entity. The set of unique keys declared for a data entity is often referred to as the candidate keys for that data entity. From the set of candidate keys, a single unique key is selected and declared the primary key for that data entity. In an entity relationship diagram, each entity relationship uses a unique key, most often the primary key, of one data entity and copies the unique key data attributes to another data entity to which it relates. This inheritance of the unique key data attributes is referred to as a foreign key and is used to provide data access paths between data entities. Once the data model is instantiated into a database, each data entity usually becomes a database table, unique keys become unique indexes associated with their assigned database tables, and entity relationships become foreign key constraints. In integrated data models,[1] commonality relationships[2] do not become foreign key constraints since commonality relationships are a peer-to-peer type of relationship.

The primary key may consist of a single attribute or a multiple attributes in combination. For example, a birthday could be shared by many people and so would not be a prime candidate for the Primary Key, but a social security number or Driver's License number would be ideal since it correlates to one single data value. Another unique characteristic of a Primary Key as it pertains to a relational database, is that a Primary Key must also serve as a Foreign Key on a related table[*citation needed*]. For example:

```
Author Table Schema:

AuthorTable(AUTHOR_ID,AuthorName,CountryBorn,YearBorn)

Book Table Schema:

Book(ISBN,Author_ID,Title,Publisher,Price)
```

Here we can see that AUTHOR_ID serves as the Primary Key in AuthorTable but also serves as the Foreign Key on the BookTable. The Foreign Key serves as the link and therefore the connection between the two "related" tables in this sample database.

In a relational database, a **unique key** index can uniquely identify each row of data values in a database table. A unique key index comprises a single column or a set of columns in a single database table. No two distinct rows or data records in a database table can have the same data value (or combination of data values) in those unique key index columns if NULL values are not used. Depending on its design, a database table may have many unique key

indexes but at most one primary key index.

A unique key constraint does not imply the `NOT NULL` constraint in practice. Because NULL is not an actual value (it represents the lack of a value), when two rows are compared, and both rows have NULL in a column, the column values are not considered to be equal. Thus, in order for a unique key to uniquely identify each row in a table, NULL values must not be used. According to the SQL[3] standard and Relational Model theory, a unique key (unique constraint) should accept NULL in several rows/tuples — however not all RDBMS implement this feature correctly.[4][5]

A unique key should uniquely identify all *possible* rows that exist in a table and not only the currently existing rows [*citation needed*]. Examples of unique keys are Social Security numbers (associated with a specific person[6]) or ISBNs (associated with a specific book). Telephone books and dictionaries cannot use names, words, or Dewey Decimal system numbers as candidate keys because they do not uniquely identify telephone numbers or words.

A table can have at most one **primary key**, but more than one unique key. A primary key is a combination of columns which uniquely specify a row. It is a special case of unique keys. One difference is that primary keys have an implicit `NOT NULL` constraint while unique keys do not. Thus, the values in unique key columns may or may not be NULL, and in fact such a column may contain at most one NULL fields.[7] Another difference is that primary keys must be defined using another syntax.

The relational model, as expressed through relational calculus and relational algebra, does not distinguish between primary keys and other kinds of keys. Primary keys were added to the SQL standard mainly as a convenience to the application programmer.[*citation needed*]

Unique keys as well as primary keys can be referenced by foreign keys.

Defining primary keys

Primary keys are defined in the ANSI SQL Standard, through the PRIMARY KEY constraint. The syntax to add such a constraint to an existing table is defined in SQL:2003 like this:

```
ALTER TABLE <table identifier>
    ADD [ CONSTRAINT <constraint identifier> ]
    PRIMARY KEY ( <column expression> {, <column expression>}... )
```

The primary key can also be specified directly during table creation. In the SQL Standard, primary keys may consist of one or multiple columns. Each column participating in the primary key is implicitly defined as NOT NULL. Note that some DBMS require explicitly marking primary-key columns as `NOT NULL`.[*citation needed*]

```
CREATE TABLE table_name (

   ...
)
```

If the primary key consists only of a single column, the column can be marked as such using the following syntax:

```
CREATE TABLE table_name (
   id_col  INT  PRIMARY KEY,
   col2    CHARACTER VARYING(20),
   ...
)
```

Differences between Primary Key and Unique Key:

Primary Key

1. A primary key *cannot allow* null values. (You cannot define a primary key on columns that allow nulls.)

2. Each table can have *at most one* primary key.
3. On some RDBMS a primary key automatically generates a *clustered* table index by default.

Unique Key

1. A unique key *can allow* null values. (You can define a unique key on columns that allow nulls.)
2. Each table can have *multiple* unique keys.
3. On some RDBMS a unique key automatically generates a *non-clustered* table index by default.

Defining unique keys

The definition of unique keys is syntactically very similar to primary keys.

```
ALTER TABLE <table identifier>
    ADD [ CONSTRAINT <constraint identifier> ]
    UNIQUE ( <column expression> {, <column expression>}... )
```

Likewise, unique keys can be defined as part of the `CREATE TABLE` SQL statement.

```
CREATE TABLE table_name (
   id_col   INT,
   col2     CHARACTER VARYING(20),
   key_col  SMALLINT,
   ...
   CONSTRAINT key_unique UNIQUE(key_col),
   ...
)
```

```
CREATE TABLE table_name (
   id_col  INT  PRIMARY KEY,
   col2    CHARACTER VARYING(20),
   ...
   key_col  SMALLINT UNIQUE,
   ...
)
```

Surrogate keys

In some design situations the natural key that uniquely identifies a tuple in a relation is difficult to use for software development. For example, it may involve multiple columns or large text fields. A surrogate key can be used as the primary key. In other situations there may be more than one candidate key for a relation, and no candidate key is obviously preferred. A surrogate key may be used as the primary key to avoid giving one candidate key artificial primacy over the others.

Since primary keys exist primarily as a convenience to the programmer, surrogate primary keys are often used—in many cases exclusively—in database application design.

Due to the popularity of surrogate primary keys, many developers and in some cases even theoreticians have come to regard surrogate primary keys as an inalienable part of the relational data model. This is largely due to a migration of principles from the Object-Oriented Programming model to the relational model, creating the hybrid object-relational model. In the ORM, these additional restrictions are placed on primary keys:

- Primary keys should be immutable, that is, not changed until the record is destroyed.
- Primary keys should be anonymous integer or numeric identifiers.

However, neither of these restrictions is part of the relational model or any SQL standard. Due diligence should be applied when deciding on the immutability of primary key values during database and application design. Some database systems even imply that values in primary key columns cannot be changed using the `UPDATE` SQL statement[*citation needed*].

Alternate key

It is commonplace in SQL databases to declare a single **primary key**, the most important unique key. However, there could be further unique keys that could serve the same purpose. These should be marked as 'unique' keys. This is done to prevent incorrect data from entering a table (a duplicate entry is not valid in a unique column) and to make the database more complete and useful. These could be called alternate keys.[8]

References

[1] Data Model Integration | The Integration of Data Models (http://www.strins.com/data-model-integration.html)
[2] Commonality Relationships | Commonality Constraints (http://www.strins.com/commonality-relationships.html)
[3] Summary of ANSI/ISO/IEC SQL (http://www.xcdsql.org/Summary of SQL.html#chapter-Table constraints)
[4] Constraints - SQL Database Reference Material - Learn sql, read an sql manual, follow an sql tutorial, or learn how to structure an SQL query (http://www.sql.org/sql-database/postgresql/manual/ddl-constraints.html#AEN1832)
[5] Comparison of different SQL implementations (http://troels.arvin.dk/db/rdbms/#constraints-unique)
[6] **SSN uniqueness:** Rare SSN duplicates do exist in the field, a condition that led to problems with early commercial computer systems that relied on SSN uniqueness. Practitioners are taught that well-known duplications in SSN assignments occurred in the early days of the SSN system. This situation points out the complexity of designing systems that assume unique keys in real-world data.
[7] MySQL 5.5 Reference Manual :: 12.1.14. CREATE TABLE Syntax (http://dev.mysql.com/doc/refman/5.5/en/create-table.html) "For all engines, a UNIQUE index permits multiple NULL values for columns that can contain NULL."
[8] Alternate key - Oracle FAQ (http://www.orafaq.com/wiki/Alternate_key)

External links

- Relation Database terms of reference, Keys (http://rdbms.opengrass.net/2_Database Design/2.1_TermsOfReference/2.1.2_Keys.html): An overview of the different types of keys in an RDBMS

Natural key

In relational model database design, a **natural key** is a key that is formed of attributes that already exist in the real world. For example, a USA citizen's social security number could be used as a natural key. In other words, a natural key is a candidate key that has a logical relationship to the attributes within that row. A natural key is sometimes called *domain key*.

The main advantage of a natural key over a surrogate key, which has no meaning outside the database environment, is that it already exists; there is no need to add a new, artificial column to the schema. Using a natural key (when one can be identified) also simplifies data quality: It ensures that there can only be one row for a key; this "one version of the truth" can be verified, because the natural key is based on a real-world observation.

The main disadvantage of choosing a natural key is that its value may change and the relational database engine may not be able to propagate that change across the related foreign keys. For example, if `person_name` is used as the primary key for the person table, and a person gets married and changes name, then all of the one-to-many related tables need to be updated also. The secondary disadvantage of choosing a natural key is identifying uniqueness. The primary key must consist of the attributes that uniquely identify a row. However, it may be difficult (or it may add constraints) to create a natural key on a table. For example, if `person_name` is used as a primary key for the person table, many persons may share the same name and all but the first entry will be rejected as a duplication. The uniqueness constraint may be overcome by adding an additional column to the primary key, like `street_address`, to increase the likelihood of uniqueness.

External links

- "Intelligent Versus Surrogate Keys" [1], *B Carter*.
- "Avoid Unique Indexes – (Mistake 3 of 10)" [2], *Near Infinity*, Create Data Disaster.
- "Natural versus surrogate keys" [3], *c2*.

References

[1] http://www.bcarter.com/intsurr1.htm
[2] http://www.nearinfinity.com/blogs/page/lrichard?entry=create_data_disaster_avoid_unique
[3] http://c2.com/cgi/wiki?AutoKeysVersusDomainKeys

Key field

A **unique key** or **primary key** is a key that uniquely defines the characteristics of each row. The **primary key** has to consist of characteristics that cannot collectively be duplicated by any other row. A Primary and Foreign key is needed in order to link tables.

In an entity relationship diagram of a data model, one or more **unique keys** may be declared for each data entity. Each unique key is composed from one or more data attributes of that data entity. The set of unique keys declared for a data entity is often referred to as the candidate keys for that data entity. From the set of candidate keys, a single unique key is selected and declared the primary key for that data entity. In an entity relationship diagram, each entity relationship uses a unique key, most often the primary key, of one data entity and copies the unique key data attributes to another data entity to which it relates. This inheritance of the unique key data attributes is referred to as a foreign key and is used to provide data access paths between data entities. Once the data model is instantiated into a database, each data entity usually becomes a database table, unique keys become unique indexes associated with their assigned database tables, and entity relationships become foreign key constraints. In integrated data models,[1] commonality relationships[2] do not become foreign key constraints since commonality relationships are a peer-to-peer type of relationship.

The primary key may consist of a single attribute or a multiple attributes in combination. For example, a birthday could be shared by many people and so would not be a prime candidate for the Primary Key, but a social security number or Driver's License number would be ideal since it correlates to one single data value. Another unique characteristic of a Primary Key as it pertains to a relational database, is that a Primary Key must also serve as a Foreign Key on a related table[*citation needed*]. For example:

```
Author Table Schema:

AuthorTable(AUTHOR_ID,AuthorName,CountryBorn,YearBorn)

Book Table Schema:

Book(ISBN,Author_ID,Title,Publisher,Price)
```

Here we can see that AUTHOR_ID serves as the Primary Key in AuthorTable but also serves as the Foreign Key on the BookTable. The Foreign Key serves as the link and therefore the connection between the two "related" tables in this sample database.

In a relational database, a **unique key** index can uniquely identify each row of data values in a database table. A unique key index comprises a single column or a set of columns in a single database table. No two distinct rows or data records in a database table can have the same data value (or combination of data values) in those unique key index columns if NULL values are not used. Depending on its design, a database table may have many unique key indexes but at most one primary key index.

A unique key constraint does not imply the `NOT NULL` constraint in practice. Because NULL is not an actual value (it represents the lack of a value), when two rows are compared, and both rows have NULL in a column, the column values are not considered to be equal. Thus, in order for a unique key to uniquely identify each row in a table, NULL values must not be used. According to the SQL[3] standard and Relational Model theory, a unique key (unique constraint) should accept NULL in several rows/tuples — however not all RDBMS implement this feature correctly.[4][5]

A unique key should uniquely identify all *possible* rows that exist in a table and not only the currently existing rows [*citation needed*]. Examples of unique keys are Social Security numbers (associated with a specific person[6]) or ISBNs (associated with a specific book). Telephone books and dictionaries cannot use names, words, or Dewey Decimal

system numbers as candidate keys because they do not uniquely identify telephone numbers or words.

A table can have at most one **primary key**, but more than one unique key. A primary key is a combination of columns which uniquely specify a row. It is a special case of unique keys. One difference is that primary keys have an implicit `NOT NULL` constraint while unique keys do not. Thus, the values in unique key columns may or may not be NULL, and in fact such a column may contain at most one NULL fields.[7] Another difference is that primary keys must be defined using another syntax.

The relational model, as expressed through relational calculus and relational algebra, does not distinguish between primary keys and other kinds of keys. Primary keys were added to the SQL standard mainly as a convenience to the application programmer.[*citation needed*]

Unique keys as well as primary keys can be referenced by foreign keys.

Defining primary keys

Primary keys are defined in the ANSI SQL Standard, through the PRIMARY KEY constraint. The syntax to add such a constraint to an existing table is defined in SQL:2003 like this:

```
ALTER TABLE <table identifier>
    ADD [ CONSTRAINT <constraint identifier> ]
    PRIMARY KEY ( <column expression> {, <column expression>}... )
```

The primary key can also be specified directly during table creation. In the SQL Standard, primary keys may consist of one or multiple columns. Each column participating in the primary key is implicitly defined as NOT NULL. Note that some DBMS require explicitly marking primary-key columns as `NOT NULL`.[*citation needed*]

```
CREATE TABLE table_name (

   ...
)
```

If the primary key consists only of a single column, the column can be marked as such using the following syntax:

```
CREATE TABLE table_name (
   id_col  INT  PRIMARY KEY,
   col2    CHARACTER VARYING(20),
   ...
)
```

Differences between Primary Key and Unique Key:

Primary Key

1. A primary key *cannot allow* null values. (You cannot define a primary key on columns that allow nulls.)
2. Each table can have *at most one* primary key.
3. On some RDBMS a primary key automatically generates a *clustered* table index by default.

Unique Key

1. A unique key *can allow* null values. (You can define a unique key on columns that allow nulls.)
2. Each table can have *multiple* unique keys.
3. On some RDBMS a unique key automatically generates a *non-clustered* table index by default.

Defining unique keys

The definition of unique keys is syntactically very similar to primary keys.

```
ALTER TABLE <table identifier>
    ADD [ CONSTRAINT <constraint identifier> ]
    UNIQUE ( <column expression> {, <column expression>}... )
```

Likewise, unique keys can be defined as part of the `CREATE TABLE` SQL statement.

```
CREATE TABLE table_name (
   id_col   INT,
   col2     CHARACTER VARYING(20),
   key_col  SMALLINT,
   ...
   CONSTRAINT key_unique UNIQUE(key_col),
   ...
)
```

```
CREATE TABLE table_name (
   id_col  INT  PRIMARY KEY,
   col2    CHARACTER VARYING(20),
   ...
   key_col  SMALLINT UNIQUE,
   ...
)
```

Surrogate keys

In some design situations the natural key that uniquely identifies a tuple in a relation is difficult to use for software development. For example, it may involve multiple columns or large text fields. A surrogate key can be used as the primary key. In other situations there may be more than one candidate key for a relation, and no candidate key is obviously preferred. A surrogate key may be used as the primary key to avoid giving one candidate key artificial primacy over the others.

Since primary keys exist primarily as a convenience to the programmer, surrogate primary keys are often used—in many cases exclusively—in database application design.

Due to the popularity of surrogate primary keys, many developers and in some cases even theoreticians have come to regard surrogate primary keys as an inalienable part of the relational data model. This is largely due to a migration of principles from the Object-Oriented Programming model to the relational model, creating the hybrid object-relational model. In the ORM, these additional restrictions are placed on primary keys:

- Primary keys should be immutable, that is, not changed until the record is destroyed.
- Primary keys should be anonymous integer or numeric identifiers.

However, neither of these restrictions is part of the relational model or any SQL standard. Due diligence should be applied when deciding on the immutability of primary key values during database and application design. Some database systems even imply that values in primary key columns cannot be changed using the `UPDATE` SQL statement[*citation needed*].

Alternate key

It is commonplace in SQL databases to declare a single **primary key**, the most important unique key. However, there could be further unique keys that could serve the same purpose. These should be marked as 'unique' keys. This is done to prevent incorrect data from entering a table (a duplicate entry is not valid in a unique column) and to make the database more complete and useful. These could be called alternate keys.[8]

References

[1] Data Model Integration | The Integration of Data Models (http://www.strins.com/data-model-integration.html)
[2] Commonality Relationships | Commonality Constraints (http://www.strins.com/commonality-relationships.html)
[3] Summary of ANSI/ISO/IEC SQL (http://www.xcdsql.org/Summary of SQL.html#chapter-Table constraints)
[4] Constraints - SQL Database Reference Material - Learn sql, read an sql manual, follow an sql tutorial, or learn how to structure an SQL query (http://www.sql.org/sql-database/postgresql/manual/ddl-constraints.html#AEN1832)
[5] Comparison of different SQL implementations (http://troels.arvin.dk/db/rdbms/#constraints-unique)
[6] **SSN uniqueness:** Rare SSN duplicates do exist in the field, a condition that led to problems with early commercial computer systems that relied on SSN uniqueness. Practitioners are taught that well-known duplications in SSN assignments occurred in the early days of the SSN system. This situation points out the complexity of designing systems that assume unique keys in real-world data.
[7] MySQL 5.5 Reference Manual :: 12.1.14. CREATE TABLE Syntax (http://dev.mysql.com/doc/refman/5.5/en/create-table.html) "For all engines, a UNIQUE index permits multiple NULL values for columns that can contain NULL."
[8] Alternate key - Oracle FAQ (http://www.orafaq.com/wiki/Alternate_key)

External links

- Relation Database terms of reference, Keys (http://rdbms.opengrass.net/2_Database Design/2. 1_TermsOfReference/2.1.2_Keys.html): An overview of the different types of keys in an RDBMS

Compound key

In database design, a **compound key** is a key that consists of 2 or more attributes that uniquely identify an entity occurrence. Each attribute that makes up the compound key is a **simple key** in its own right.

This is often confused with a **composite key** whereby even though this is also a key that consists of 2 or more attributes that uniquely identify an entity occurrence, at least one attribute that makes up the composite key is not a **simple key** in its own right.

An example might be an entity that represents the modules each student is attending at University. The entity has a studentId and a moduleCode as its primary key. Each of the attributes that make up the primary key are simple keys because each represents a unique reference when identifying a student in one instance and a module in the other.

In contrast, using the same example, imagine we identified a student by their firstName + lastName. In our table representing students on modules our primary key would now be firstName + lastName + moduleCode. Because firstName + lastName represent a unique reference to a student, it is not a simple key, it is a combination of attributes used to uniquely identify a student. Therefore the primary key for this entity is a composite key.

No restriction is applied to the attributes regarding their (initial) ownership within the data model. This means that any one, none, or all, of the multiple attributes within the compound key can be foreign keys. Indeed, a foreign key may itself be a compound key.

Compound keys almost always originate from attributive or an associative entity (tables) within the model, but this is not an absolute.

External links

- Composite Inverse Functional Properties [1]: for an equivalent notion in the Semantic Web
- Relation Database terms of reference, Keys [2]: An overview of the different types of keys in an RDBMS

References

[1] http://esw.w3.org/topic/CIFP
[2] http://rdbms.opengrass.net

Foreign key

In the context of relational databases, a **foreign key** is a field (or collection of fields) in one table that uniquely identifies a row of another table. In other words, a foreign key is a column or a combination of columns that is used to establish and enforce a link between two tables.

The table containing the foreign key is called the **referencing** or **child table**, and the table containing the candidate key is called the **referenced** or **parent table**.

Since the purpose of the foreign key is to identify a particular row of the referenced table, it is generally required that the foreign key is equal to the candidate key in some row of the primary table, or else have no value (the NULL value.). This rule is called a **referential integrity constraint** between the two tables. Because violations of these constraints can be the source of many database problems, most database management systems provide mechanisms to ensure that every non-null foreign key corresponds to a row of the referenced table.

For example, consider a database with two tables: a CUSTOMER table that includes all customer data and an ORDER table that includes all customer orders. Suppose the business requires that each order must refer to a single customer. To reflect this in the database, a foreign key column is added to the ORDER table (e.g., CUSTOMERID), which references the primary key of CUSTOMER (e.g. ID). Because the primary key of a table must be unique, and because CUSTOMERID only contains values from that primary key field, we may assume that, when it has a value, CUSTOMERID will identify the particular customer which placed the order. However, this can no longer be assumed if the ORDER table is not kept up to date when rows of the CUSTOMER table are deleted or the ID column altered, and working with these tables may become more difficult. Many real world databased work around this problem by 'inactivating' rather than physically deleting master table foreign keys, or by complex update programs that modify all references to a foreign key when a change is needed.

Foreign keys play an essential role in database design. One important part of database design is making sure that relationships between real-world entities are reflected in the database by references, using foreign keys to refer from one table to another. Another important part of database design is database normalization, in which tables are broken apart and foreign keys make it possible for them to be reconstructed.

Multiple rows in the referencing (or child) table may refer to the same row in the referenced (or parent) table. In this case, the relationship between the two tables is called a one to many relationship between the referenced table and the referencing table.

In addition, the child and parent table may, in fact, be the same table, i.e. the foreign key refers back to the same table. Such a foreign key is known in SQL:2003 as a **self-referencing** or **recursive** foreign key. In database management systems, this is often accomplished by linking a first and second reference to the same table.

A table may have multiple foreign keys, and each foreign key can have a different parent table. Each foreign key is enforced independently by the database system. Therefore, cascading relationships between tables can be established using foreign keys.

Defining foreign keys

Foreign keys are defined in the ISO SQL Standard, through a FOREIGN KEY constraint. The syntax to add such a constraint to an existing table is defined in SQL:2003 as shown below. Omitting the column list in the REFERENCES clause implies that the foreign key shall reference the primary key of the referenced table.

```
ALTER TABLE <table identifier>
   ADD [ CONSTRAINT <constraint identifier> ]
      FOREIGN KEY ( <column expression> )
      REFERENCES <table identifier> [ ( <column expression> {, <column expression>}... ) ]
      [ ON UPDATE <referential action> ]
      [ ON DELETE <referential action> ]
```

Likewise, foreign keys can be defined as part of the `CREATE TABLE` SQL statement.

```
CREATE TABLE table_name (
   id    INTEGER  PRIMARY KEY,
   col2  CHARACTER VARYING(20),
   col3  INTEGER,
   ...
   FOREIGN KEY(col3)
      REFERENCES other_table(key_col) ON DELETE CASCADE,
   ... )
```

If the foreign key is a single column only, the column can be marked as such using the following syntax:

```
CREATE TABLE table_name (
   id    INTEGER  PRIMARY KEY,
   col2  CHARACTER VARYING(20),
   col3  INTEGER REFERENCES other_table(column_name),
   ... )
```

Foreign keys can be defined with a stored procedure statement.Wikipedia:Please clarify

```
sp_foreignkey tabname, pktabname, col1 [, col2] ...  [, col8]
```

- **tabname**: the name of the table or view that contains the foreign key to be defined.
- **pktabname**: the name of the table or view that has the primary key to which the foreign key applies. The primary key must already be defined.
- **col1**: the name of the first column that makes up the foreign key. The foreign key must have at least one column and can have a maximum of eight columns.

Referential actions

Because the database management system enforces referential constraints, it must ensure data integrity if rows in a referenced table are to be deleted (or updated). If dependent rows in referencing tables still exist, those references have to be considered. SQL:2003 specifies 5 different **referential actions** that shall take place in such occurrences:

- CASCADE
- RESTRICT
- NO ACTION
- SET NULL
- SET DEFAULT

CASCADE

Whenever rows in the master (referenced) table are deleted (resp. updated), the respective rows of the child (referencing) table with a matching foreign key column will get deleted (resp. updated) as well. This is called a cascade delete (resp. update).

RESTRICT

A value cannot be updated or deleted when a row exists in a referencing or child table that references the value in the referenced table.

Similarly, a row cannot be deleted as long as there is a reference to it from a referencing or child table.

To understand RESTRICT (and CASCADE) better, it may be helpful to notice the following difference, which might not be immediately clear. The referential action CASCADE modifies the "behavior" of the (child) table itself where the word CASCADE is used. For example, ON DELETE CASCADE effectively says "When the referenced row is deleted from the other table (master table), then delete *also from me*". However, the referential action RESTRICT modifies the "behavior" of the master table, *not* the child table, although the word RESTRICT appears in the child table and not in the master table! So, ON DELETE RESTRICT effectively says: "When someone tries to delete the row from the other table (master table), prevent deletion *from that other table* (and of course, also don't delete from me, but that's not the main point here)."

RESTRICT is not supported by Microsoft SQL 2012 and earlier.

NO ACTION

NO ACTION and RESTRICT are very much alike. The main difference between NO ACTION and RESTRICT is that with NO ACTION the referential integrity check is done after trying to alter the table. RESTRICT does the check before trying to execute the UPDATE or DELETE statement. Both referential actions act the same if the referential integrity check fails: the UPDATE or DELETE statement will result in an error.

In other words, when an UPDATE or DELETE statement is executed on the referenced table using the referential action NO ACTION, the DBMS verifies at the end of the statement execution that none of the referential relationships are violated. This is different from RESTRICT, which assumes at the outset that the operation will violate the constraint. Using NO ACTION, the triggers or the semantics of the statement itself may yield an end state in which no foreign key relationships are violated by the time the constraint is finally checked, thus allowing the statement to complete successfully.

SET DEFAULT , SET NULL

In general, the action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE or ON UPDATE: The value of the affected referencing attributes is changed to NULL for SET NULL, and to the specified default value for SET DEFAULT.

Triggers

Referential actions are generally implemented as implied triggers (i.e. triggers with system-generated names, often hidden.) As such, they are subject to the same limitations as user-defined triggers, and their order of execution relative to other triggers may need to be considered; in some cases it may become necessary to replace the referential action with its equivalent user-defined trigger to ensure proper execution order, or to work around mutating-table limitations.

Another important limitation appears with transaction isolation: your changes to a row may not be able to fully cascade because the row is referenced by data your transaction cannot "see", and therefore cannot cascade onto. An example: while your transaction is attempting to renumber a customer account, a simultaneous transaction is attempting to create a new invoice for that same customer; while a CASCADE rule may fix all the invoice rows your transaction can see to keep them consistent with the renumbered customer row, it won't reach into another transaction to fix the data there; because the database cannot guarantee consistent data when the two transactions commit, one of them will be forced to roll back (often on a first-come-first-served basis.)

Example

As a first example to illustrate foreign keys, suppose an accounts database has a table with invoices and each invoice is associated with a particular supplier. Supplier details (such as name and address) are kept in a separate table; each supplier is given a 'supplier number' to identify it. Each invoice record has an attribute containing the supplier number for that invoice. Then, the 'supplier number' is the primary key in the Supplier table. The foreign key in the Invoices table points to that primary key. The relational schema is the following. Primary keys are marked in bold, and foreign keys are marked in italics.

```
Supplier ( SupplierNumber, Name, Address, Type )
Invoices ( InvoiceNumber, SupplierNumber, Text )
```

The corresponding Data Definition Language statement is as follows.

```
CREATE TABLE Supplier (
   SupplierNumber  INTEGER NOT NULL,
   Name            VARCHAR(20) NOT NULL,
   Address         VARCHAR(50) NOT NULL,
   Type            VARCHAR(10),
   CONSTRAINT supplier_pk PRIMARY KEY(SupplierNumber),
   CONSTRAINT number_value CHECK (SupplierNumber > 0) )

CREATE TABLE Invoices (
   InvoiceNumber   INTEGER NOT NULL,
   SupplierNumber  INTEGER NOT NULL,
   Text            VARCHAR(4096),
   CONSTRAINT invoice_pk PRIMARY KEY(InvoiceNumber),
   CONSTRAINT inumber_value CHECK (InvoiceNumber > 0),
   CONSTRAINT supplier_fk FOREIGN KEY(SupplierNumber)
      REFERENCES Supplier(SupplierNumber)
```

```
        ON UPDATE CASCADE ON DELETE RESTRICT )
```

References

External links

- SQL-99 Foreign Keys (https://mariadb.com/kb/en/constraint_type-foreign-key-constraint/)
- PostgreSQL Foreign Keys (http://www.postgresql.org/docs/current/static/tutorial-fk.html)
- MySQL Foreign Keys (http://dev.mysql.com/doc/refman/5.1/en/create-table-foreign-keys.html)
- FirebirdSQL Foreign Keys (http://www.firebirdsql.org/manual/nullguide-keys.html#nullguide-keys-fk)
- SQLite support for Foreign Keys (http://www.sqlite.org/foreignkeys.html)
- Microsoft SQL 2012 table_constraint (Transact-SQL) (http://technet.microsoft.com/en-us/library/ms188066.aspx)

Persistent Object Identifier

In database design, a **Persistent Object Identifier** (**POID**) is a unique identifier of a record on a table, used as the primary key. Important characteristics of a POID are that it does not carry business information and are not generally exported or otherwise made visible to data users; as such a POID has many of the characteristics of a surrogate key. The only purpose of the POID is to act as the primary key on the table where it is defined and to be referenced as the foreign key by other tables. Because POIDs, like surrogate keys, do not carry business information, they are immune to changes in the form or meaning of business data.

External links

- Persistent Object ID Service [1]
- What is a Persistent Object Identifier and why should I care? [2]
- Persistent Object [3]

References

[1] http://www.ibm.com/developerworks/websphere/techjournal/0306_biernat/biernat.html
[2] http://blog.telemapics.com/?p=92
[3] http://devnet.objectivity.com/objects_faq

Cardinality (data modeling)

In database design, the **cardinality** or fundamental principle of one data table with respect to another is a critical aspect. The relationship of one to the other must be precise and exact between each other in order to explain how each table links together.

In the relational model, tables can be related as any of "one to many" or "many-to-many." This is said to be the **cardinality** of a given table in relation to another.

For example, consider a database designed to keep track of hospital records. Such a database could have many tables like:

- a *doctor* table with information about physicians;
- a *patient* table for medical subjects undergoing treatment;
- and a *department* table with an entry for each division of a hospital.

In that model:

- There is a **many-to-many** relationship between the records in the doctor table and records in the patient table because doctors have many patients, and a patient could have several doctors;
- A **one-to-many** relation between the department table and the doctor table because each doctor may work for only one department, but one department could have many doctors.

A "one-to-one" relationship is mostly used to split a table in two in order to provide information concisely and make it more understandable. In the hospital example, such a relationship could be used to keep apart doctors' own unique professional information from administrative details.

In data modeling, collections of data elements are grouped into "data tables" which contain groups of data field names called "database attributes". Tables are linked by "key fields". A "primary key" assigns a field to its "special order table". For example, the "Doctor Last Name" field might be assigned as a primary key of the Doctor table with all people having same last name organized alphabetically according to the first three letters of their first name. A table can also have a **foreign key** which indicates that field is linked to the primary key of another table.

A complex data model can involve hundreds of related tables. A renowned computer scientist, C.J. Date, created a systematic method to organize database models. Date's steps for organizing database tables and their keys is called **Database Normalization**. Database normalization avoids certain hidden database design errors (**delete anomalies** or **update anomalies**). In real life the process of database normalization ends up breaking tables into a larger number of smaller tables, so there are common sense data modeling tactics called **de-normalization** which combine tables in practical ways.

In real world data models careful design is critical because as the data grows voluminous, tables linked by keys must be used to speed up programmed retrieval of data. If data modeling is poor, even a computer applications system with just a million records will give the end-users unacceptable response time delays. For this reason data modeling is a keystone in the skills needed by a modern software developer.

Formal Database Modeling Technologies

UML class diagram may be used for data modeling. In that case, relationship are modeled using UML associations, and multiplicity is used on those associations to denote **cardinality**. Here are some examples:

left	right		example
1	**1**	one-to-one	person <-> weight
0..1	**1**	*optional on one side* one-to-one	date of death <-> person
0..* or *****	**0..*** or *****	*optional on both sides* many-to-many	person <-> book
1	**1..***	one-to-many	person <-> language

As an alternative to UML, Entity Relationship Diagrams (ERDs) can be used to capture information about data model cardinality. A crow's foot shows a **one-to-many** relationship. Alternatively a single line represents a one-to-one relationship.

External links

- UML multiplicity as data model cardinality [1] - http://www.agiledata.org

References

[1] http://www.agiledata.org/essays/umlDataModelingProfile.html#Relationships

Recordset

A **recordset** is a data structure that consists of a group of database records, and can either come from a base table or as the result of a query to the table.

The concept is common to a number of platforms, notably Microsoft's Data Access Objects (DAO) and ActiveX Data Objects (ADO). The Recordset object contains a Fields collection, and a Properties collection. At any time, the Recordset object refers to only a single record within the set as the current record.

External links

- Microsoft definition of a Recordset object in ADO [1]
- [2]

References

[1] http://msdn.microsoft.com/en-us/library/ms681510.aspx
[2] http://www.w3schools.com/ado/ado_ref_recordset.asp

Superkey

A **superkey** is defined in the relational model of database organization as a set of attributes of a relation variable for which it holds that in all relations assigned to that variable, there are no two distinct tuples (rows) that have the same values for the attributes in this set. Equivalently a superkey can also be defined as a set of attributes of a relation schema upon which all attributes of the schema are functionally dependent.

Note that the set of **all** attributes is a trivial superkey, because in relational algebra duplicate rows are not permitted.

Also note that if attribute set *K* is a superkey of relation *R*, then at all times it is the case that the projection of *R* over *K* has the same cardinality as *R* itself.

Informally, a superkey is a set of attributes within a table whose values can be used to uniquely identify a tuple. A candidate key is a minimal set of attributes necessary to identify a tuple, this is also called a minimal superkey. For example, given an employee schema, consisting of the attributes employeeID, name, job, and departmentID, we could use the employeeID in combination with any or all other attributes of this table to uniquely identify a tuple in the table. Examples of superkeys in this schema would be {employeeID, Name}, {employeeID, Name, job}, and {employeeID, Name, job, departmentID}. The last example is known as trivial superkey, because it uses all attributes of this table to identify the tuple.

In a real database we do not need values for all of those attributes to identify a tuple. We only need, per our example, the set {employeeID}. This is a **minimal superkey –** that is, a minimal set of attributes that can be used to identify a single tuple. So, employeeID is a candidate key.

Example

English Monarchs

Monarch Name	Monarch Number	Royal House
Edward	II	Plantagenet
Edward	III	Plantagenet
Richard	III	Plantagenet
Henry	IV	Lancaster

First, list out all the (non-empty) sets of attributes:

- {Monarch Name}
- {Monarch Number}
- {Royal House}
- {Monarch Name, Monarch Number}
- {Monarch Name, Royal House}
- {Monarch Number, Royal House}
- {Monarch Name, Monarch Number, Royal House}

Second, eliminate all the sets which **do not** meet superkey's requirement. For example, {Monarch Name, Royal House} cannot be a superkey because for the same attribute values (Edward, Plantagenet), there are two distinct tuples:

- (Edward, **II**, Plantagenet)
- (Edward, **III**, Plantagenet)

Finally, after elimination, the remaining sets of attributes are the only possible superkeys in this example:

- {Monarch Name, Monarch Number} **(Candidate Key)**
- {Monarch Name, Monarch Number, Royal House}

In real situations, however, superkeys are normally not determined by this method, which is very tedious and time-consuming, but by analyzing functional dependencies.

References

- Silberschatz, Abraham (2011). *Database System Concepts (6th ed.)*. McGraw-Hill. pp. 45–46. ISBN 978-0-07-352332-3.

External links

- Relation Database terms of reference, Keys (http://rdbms.opengrass.net/2_Database Design/2.1_TermsOfReference/2.1.2_Keys.html): An overview of the different types of keys in an RDBMS

Integrity constraints

Integrity constraints are used to ensure accuracy and consistency of data in a relational database.[*citation needed*]

Types

Codd initially defined two sets of constraints but, in his second version of the relational model, he came up with four integrity constraints: [*citation needed*]

Entity Integrity

The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples.This also specifies that there may not be any duplicate entries in primary key column key word.

Referential Integrity

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations in dbms.

Domain Integrity

Domain constraints allows us to test the values inserted into the database and to test the queries to make sure comparisons made are appropriate.

User Defined Integrity

A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. E.g.: Age>=18 && Age<=60

Check Constraint

A **check constraint** is applied to each row in the table.Wikipedia:Please clarify The constraint must be a predicate. It can refer to a single or multiple columns of the table. The result of the predicate can be either `TRUE`, `FALSE`, or `UNKNOWN`, depending on the presence of NULLs. If the predicate evaluates to `UNKNOWN`, then the constraint is not violated and the row can be inserted or updated in the table. This is contrary to predicates in WHERE clauses in SELECT or UPDATE statements.

For example, in a table containing products, one could add a check constraint such that the price of a product and quantity of a product is a non-negative value:

```
PRICE >= 0
```

```
QUANTITY >= 0
```

If these constraints were not in place, it would be possible to have a negative price (-$30) or quantity (-3 items).

Check constraints are used to ensure the validity of data in a database and to provide data integrity. If they are used at the database level, applications that use the database will not be able to add invalid data or modify valid data so the data becomes invalid, even if the application itself accepts invalid data.

Definition

Each check constraint has to be defined in the `CREATE TABLE` or `ALTER TABLE` statement using the syntax:

```
CREATE TABLE table_name (
   ...,
   CONSTRAINT constraint_name CHECK ( predicate ),
   ...
)
```

```
ALTER TABLE table_name
   ADD CONSTRAINT constraint_name CHECK ( predicate )
```

If the check constraint refers to a single column only, it is possible to specify the constraint as part of the column definition.

```
CREATE TABLE table_name (
   ...
   column_name type CHECK ( predicate ),
   ...
)
```

NOT NULL constraint

A `NOT NULL` constraint is functionally equivalent to the following check constraint with an `IS NOT NULL` predicate:

```
CHECK (column IS NOT NULL)
```

Some relational database management systems are able to optimize performance when the `NOT NULL` constraint syntax is used as opposed to the `CHECK` constraint syntax given above.[1]

Common restrictions

Most database management systems restrict check constraints to a single row, with access to constants and deterministic functions, but not to data in other tables, or to data invisible to the current transaction because of transaction isolation.

Such constraints are not truly *table check constraints* but rather *row check constraints*. Because these constraints are generally only verified when a row is directly updated (for performance reasons,) and often implemented as implied `INSERT` or `UPDATE` triggers, integrity constraints could be violated by indirect action were it not for these limitations. Furthermore, otherwise-valid modifications to these records would then be prevented by the `CHECK` constraint. Some examples of dangerous constraints include:

- `CHECK ((select count(*) from invoices where invoices.customerId = customerId) < 1000)`
- `CHECK (dateInserted = CURRENT_DATE)`
- `CHECK (countItems = RAND())`

User-defined triggers can be used to work around these restrictions. Although similar in implementation, it is semantically clear that triggers will only be fired when the table is directly modified, and that it is the designer's responsibility to handle indirect, important changes in other tables; constraints on the other hand are intended to be "true at all times" regardless of the user's actions or the designer's lack of foresight.

References

[1] PostgreSQL 8.3devel Documentation, Chapter 5. *Data Definition*, Section 5.3.2. *Not-Null Constraints*, Website: http://developer.postgresql.org/pgdocs/postgres/ddl-constraints.html, Accessed on May 5, 2007

Propagation constraint

In database systems, a **propagation constraint** "details what should happen to a related table when we update a row or rows of a target table" (Paul Beynon-Davies, 2004, p.108). Tables are linked using primary key to foreign key relationships. It is possible for users to update one table in a relationship in such a way that the relationship is no longer consistent and this is known as breaking referential integrity. An example of breaking referential integrity: if a table of employees includes a department number for 'Housewares' which is a foreign key to a table of departments and a user deletes that department from the department table then Housewares employees records would refer to a non-existent department number.

Propagation constraints are methods used by relational database management systems (RDBMS) to solve this problem by ensuring that relationships between tables are preserved without error. In his database textbook, Beynon-Davies explains the three ways that RDBMS handle deletions of target and related tuples:

- Restricted Delete - the user cannot delete the target row until all rows that point to it (via foreign keys) have been deleted. This means that all Housewares employees would need to be deleted, or their departments changed, before removing the department from the departmental table.
- Cascades Delete - can delete the target row and all rows that point to it (via foreign keys) are also deleted. The process is the same as a restricted delete, except that the RDBMS would delete the Houseware employees automatically before removing the department.
- Nullifies Delete - can delete the target row and all foreign keys (pointing to it) are set to null. In this case, after removing the housewares department, employees who worked in this department would have a NULL (unknown) value for their department.

Bibliography

- Beynon-Davies, P. (2004) *Database Systems* Third Edition, Palgrave Macmillan.

Transition constraint

A **transition constraint** is a way of enforcing that the data does not enter an impossible state because of a previous state. For example, it should not be possible for a person to change from being "married" to being "single, never married". The only valid states after "married" might be "divorced", "widowed", or "deceased".

Transition constraint is commonly used in database models such as relational databases.

References

- Modelling Transition Constraints (ResearchIndex) [1]

References

[1] http://citeseer.ist.psu.edu/612731.html

Wide and narrow data

Wide and **narrow** (sometimes un-stacked and stacked) are terms used to describe two different presentations for tabular data.[1][2][3]

Wide

Wide, or unstacked data is presented with each different data variable in a separate column.

Person	Age	Weight
Bob	32	128
Alice	24	86
Steve	64	95

Narrow

Narrow, or stacked data is presented with one column containing all the values and another column listing the context of the value

Person	Variable	Value
Bob	Age	32
Bob	Weight	128
Alice	Age	24
Alice	Weight	86
Steve	Age	64
Steve	Weight	95

This is often easier to implement, addition of a new field does not require any changes to the structure of the table, however it can be harder for people to understand.

Implementations

Many statistical and data processing systems have functions to convert between these two presentations, for instance the R programming language has several packages such as the reshape [4] package

References

[1] Thompson, M. E. (1997), *Theory of sample surveys*, Chapman & Hall, London. ISBN 0-412-31780-X

[2] Kitchenham, B. A. & Pfleeger, S. L. (2003), "Principles of survey research part 6: data analysis" (http://doi.acm.org/10.1145/638750.638758), *SIGSOFT Softw.Eng.Notes*, 28 (2), 24–27

[3] Chantala, K. (2006) "Using STATA to Analyze data from a Sample Survey" (http://www.cpc.unc.edu/services/computer/presentations/statatutorial/statasvy.pdf). 1-10-2001. UNC Chapel Hill, Carolina Population Center. 10-1-2006.

[4] http://had.co.nz/reshape/

External links

- http://cran.r-project.org/web/packages/reshape (http://cran.r-project.org/web/packages/reshape)

Universal relation assumption

In relational databases, the **universal relation assumption** states that one can place all data attributes into a (possibly very wide) table, which may then be decomposed into smaller tables as needed.[1]

However, the assumption that a single large table can capture real database designs is often plagued with a number of difficulties.[2] The "nested universal relation" model has attempted to address some of the problems and offer improvements.[3]

References

[1] *Database analysis and design* by I. T. Hawryszkiewycz, 1984 ISBN 0-574-21485-2 pages 59-62

[2] *Advances in database technology--EDBT 2000* by Carlo Zaniolo 2000 ISBN 3-540-67227-3 page 276

[3] *The nested universal relation database model* by Mark Levene 1992 ISBN 3-540-55493-9 pages 1-5

External links

- http://infolab.stanford.edu/jdu-symposium/talks/mendelzon.pdf

Reference table

A **reference table** (or table of reference) may mean a set of references that an author may have cited or gained inspiration from whilst writing an article, similar to a bibliography.

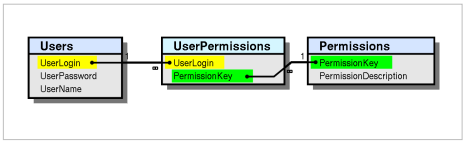
It can also mean an information table that is used as a quick and easy reference for things that are difficult to remember such as comparing imperial with metric measurements.

In the context of database design a reference table is a table into which an enumerated set of possible values of a certain field data type is divested. For example, in a relational database model of a warehouse the entity 'Item' may have a field called 'status' with a predefined set of values such as 'sold', 'reserved', 'out of stock'. In a purely designed database these values would be divested into an extra entity or Reference Table called 'status' in order to achieve database normalisation. The entity 'status' in this case has no true representative in the real world but rather would an exceptional case where the attribute of a certain database entity is divested into its own table. The advantage of doing this is that internal functionality and optional conditions within the database and the software which utilizes it are easier to modify and extend on that particular aspect. Establishing an enterprise-wide view of reference tables is called master data management.

Junction table

In database management systems following the relational model, a **junction table** is a database table that contains common fields from two or more other database tables within the same database. It is on the many side of a one-to-many relationship with each of the other tables. Junction tables are known under many names, among them **cross-reference table**, **bridge table**, **join table**, **map table**, **intersection table**, **linking table**, **many-to-many resolver**, **link table**, **pairing table**, **pivot table**, **transition table**, or **association table**.

Junction tables are employed when dealing with many-to-many relationships in a database. A practical use of a junction table would be to assign permissions to users. There can be multiple users, and each user can be assigned 0 or more permissions.




```sql
CREATE TABLE Users (
    UserLogin varchar(50) PRIMARY KEY,
    UserPassword varchar(50) NOT NULL,
    UserName varchar(50) NOT NULL
)

CREATE TABLE Permissions (
    PermissionKey varchar(50) PRIMARY KEY,
    PermissionDescription varchar(500) NOT NULL
)

-- This is the junction table.
CREATE TABLE UserPermissions (
    UserLogin varchar(50) REFERENCES Users (UserLogin),
    PermissionKey varchar(50) REFERENCES Permissions (PermissionKey),
```

```
    PRIMARY KEY (UserLogin, PermissionKey)
)
```

Using junction tables

A SELECT-statement on a junction table usually involves joining the main table with the junction table:

```
SELECT * FROM Users
JOIN UserPermissions USING (UserLogin);
```

This will return a list of all users and their permissions.

Inserting into a junction table involves two steps: first inserting into the main table (for example, a new User), then updating the junction table.

```
-- Creating a new User
INSERT INTO Users (UserLogin, UserPassword, UserName)
VALUES ('SomeUser', 'SecretPassword', 'UserName');

-- Creating a new Permission
INSERT INTO Permissions (PermissionKey, PermissionDescription)
VALUES ('TheKey', 'A key used for several permissions');

-- Finally, updating the junction
INSERT INTO UserPermissions (UserLogin, PermissionKey)
VALUES ('SomeUser', 'TheKey');
```

Using foreign keys, the database will automatically dereference the values of the UserPermissions table to their own table.

Nested set model

The **nested set model** is a particular technique for representing nested sets (also known as trees or hierarchies) in relational databases. The term was apparently introduced by Joe Celko; others describe the same technique without naming it [1] or using different terms.[2]

Motivation

The technique is an answer to the problem that the standard relational algebra and relational calculus, and the SQL operations based on them, are unable to express all desirable operations on hierarchies directly. A hierarchy can be expressed in terms of a parent-child relation - Celko calls this the adjacency list model - but if it can have arbitrary depth, this does not allow the expression of operations such as comparing the contents of hierarchies of two elements, or determining whether an element is somewhere in the subhierarchy of another element. When the hierarchy is of fixed or bounded depth, the operations are possible, but expensive, due to the necessity of performing one relational join per level. This is often known as the bill of materials problem.[*citation needed*]

Several resolutions exist and are available in some relational database management systems:

- support for a dedicated hierarchy data type, such as in SQL's hierarchical query facility;
- extending the relational language with hierarchy manipulations, such as in the nested relational algebra.
- extending the relational language with transitive closure, such as SQL's CONNECT statement; this allows a parent-child relation to be used, but execution remains expensive;
- the queries can be expressed in a language that supports iteration and is wrapped around the relational operations, such as PL/SQL, T-SQL or a general-purpose programming language

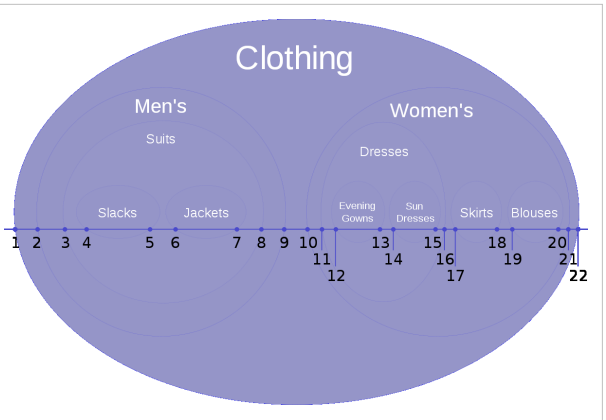
When these solutions are not available or not feasible, another approach must be taken.

The technique

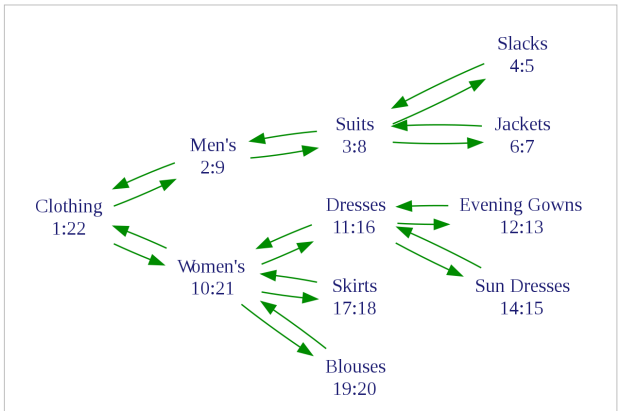
The **nested set model** is to number the nodes according to a tree traversal, which visits each node twice, assigning numbers in the order of visiting, and at both visits. This leaves two numbers for each node, which are stored as two attributes. Querying becomes inexpensive: hierarchy membership can be tested by comparing these numbers. Updating requires renumbering and is therefore expensive. Refinements that use rational numbers instead of integers can avoid renumbering, and so are faster to update, although much more complicated.

Example

In a clothing store catalog, clothing may be categorized according to the hierarchy given on the left:



A hierarchy: types of clothing



The numbering assigned by tree traversal

Node	Left	Right
Clothing	1	22
Men's	2	9
Women's	10	21
Suits	3	8
Slacks	4	5
Jackets	6	7
Dresses	11	16
Skirts	17	18
Blouses	19	20
Evening Gowns	12	13
Sun Dresses	14	+ The resulting representation

The "Clothing" category, with the highest position in the hierarchy, encompasses all subordinating categories. It is therefore given left and right domain values of 1 and 22, the latter value being the double of the total number of nodes being represented. The next hierarchical level contains "Men's" and "Women's", both containing levels within themselves that must be accounted for. Each level's data node is assigned left and right domain values according to the number of sublevels contained within, as shown in the table data.

Performance

Queries using nested sets can be expected to be faster than queries using a stored procedure to traverse an adjacency list, and so are the faster option for databases which lack native recursive query constructs, such as MySQL. However, recursive SQL queries can be expected to perform comparably for 'find immediate descendants' queries, and much faster for other depth search queries, and so are the faster option for databases which provide them, such as PostgreSQL, Oracle, and Microsoft SQL Server.

Drawbacks

Nested sets are very slow for inserts because it requires updating left and right domain values for all records in the table after the insert. This can cause a lot of database thrash[*citation needed*] as many rows are rewritten and indexes rebuilt.

The nested interval model does not suffer from this problem, but is more complex to implement, and is not as well known. The nested interval model stores the position of the nodes as rational numbers expressed as quotients (n/d). [3]

Variations

Using the nested set model as described above has some performance limitations during certain tree traversal operations. For example, trying to find the immediate child nodes given a parent node requires pruning the subtree to a specific level as in the following SQL code example:

```
SELECT Child.Node, Child.Left, Child.Right
FROM Tree as Parent, Tree as Child
WHERE
      Child.Left BETWEEN Parent.Left AND Parent.Right
      AND NOT EXISTS (    -- No Middle Node
```

```
            SELECT *
            FROM Tree as Mid
            WHERE Mid.Left BETWEEN Parent.Left AND Parent.Right
                        AND Child.Left BETWEEN Mid.Left AND Mid.Right
                  AND Mid.Node NOT IN (Parent.Node AND Child.Node)
      )
      AND Parent.Left = 1  -- Given Parent Node Left Index
```

Or, equivalently:

```
SELECT DISTINCT Child.Node, Child.Left, Child.Right
FROM Tree as Child, Tree as Parent
WHERE Parent.Left < Child.Left AND Parent.Right > Child.Right  -- associate Child Nodes with ancestors
GROUP BY Child.Node
HAVING max(Parent.Left) = 1  -- Subset for those with the given Parent
Node as the nearest ancestor
```

The query will be more complicated when searching for children more than one level deep. To overcome this limitation and simplify tree traversal an additional column is added to the model to maintain the depth of a node within a tree.

Node	Left	Right	Depth
Clothing	1	22	0
Men's	2	9	1
Women's	10	21	1
Suits	3	8	2
Slacks	4	5	3
Jackets	6	7	3
Dresses	11	16	2
Skirts	17	18	2
Blouses	19	20	2
Evening Gowns	12	13	3
Sun Dresses	14	15	+ The resulting representation

In this model, finding the immediate children given a parent node can be accomplished with the following SQL code:

```
SELECT Child.Node, Child.Left, Child.Right
FROM Tree as Child, Tree as Parent
WHERE
      Child.Depth = Parent.Depth + 1
      AND Child.Left > Parent.Left
      AND Child.Right < Parent.Right
      AND Parent.Left = 1  -- Given Parent Node Left Index
```

References

[1] *Recursive Hierarchies: The Relational Taboo!* (http://www.kamfonas.com/id3.html), by Michael J. Kamfonas, in: *The Relational Journal* - October/November 1992,

[2] Storing Hierarchical Data in a Database: *Modified Pre-order Tree Traversal* (http://articles.sitepoint.com/article/hierarchical-data-database/2), by Gijs van Tulder, at articles.sitepoint.com

[3] http://www.sigmod.org/publications/sigmod-record/0506/p47-article-tropashko.pdf

External links

- Troels' links to Hierarchical data in RDBMSs (http://troels.arvin.dk/db/rdbms/links/#hierarchical)
- Managing hierarchical data in relational databases (http://mikehillyer.com/articles/managing-hierarchical-data-in-mysql/)
- PHP PEAR Implementation for Nested Sets (http://pear.php.net/package/DB_NestedSet) - by Daniel Khan
- Interpreting Nested Sets in PHP (http://semlabs.co.uk/journal/converting-nested-set-model-data-in-to-multi-dimensional-arrays-in-php)
- Understanding Nested Sets (http://www.evanpetersen.com/item/nested-sets.html)

Information schema

In relational databases, the **information schema** is an ANSI standard set of read-only views which provide information about all of the tables, views, columns, and procedures in a database. It can be used as a source of the information which some databases make available through non-standard commands, such as the `SHOW` command of MySQL, the `DESCRIBE` command of Oracle, and the `\d` command of PostgreSQL.

```
=> select count(table_name) from information_schema.tables;
 count
-------
    99
(1 row)
=> select column_name, data_type, column_default, is_nullable
      from information_schema.columns where table_name='alpha';
 column_name | data_type | column_default | is_nullable
-------------+-----------+----------------+-------------
 foo         | integer   |                | YES
 bar         | character |                | YES
(2 rows)
=> select * from information_schema.information_schema_catalog_name;
 catalog_name
--------------
 johnd
(1 row)
```

External links

- Information schema in MySQL 5.7 [1]
- Information schema in PostgreSQL (current version) [2]
- Information schema in SQLite [3]
- Information schema in Microsoft SQL Server 2005 [4]
- Information schema in Microsoft SQL Server Compact 4.0 [5]

References

[1] http://dev.mysql.com/doc/refman/5.7/en/information-schema.html
[2] http://www.postgresql.org/docs/current/interactive/information-schema.html
[3] http://www.sqlite.org/cvstrac/wiki?p=InformationSchema
[4] http://msdn.microsoft.com/en-us/library/ms186778.aspx
[5] http://msdn.microsoft.com/en-us/library/ms174156

Codd's 12 rules

Codd's twelve rules are a set of thirteen rules (numbered zero to twelve) proposed by Edgar F. Codd worked under a pioneer of the relational model for databases, designed to define what is required from a database management system in order for it to be considered *relational*, i.e., a relational database management system (RDBMS). They are sometimes jokingly referred to as "Codd's Twelve Commandments".

Details

Codd produced these rules as part of a personal campaign to prevent his vision of the relational database being diluted, as database vendors scrambled in the early 1980s to repackage existing products with a relational veneer. Rule 12 was particularly designed to counter such a positioning. Even if such repackaged non-relational products eventually gave way to SQL DBMSs, no popular "relational" DBMSs are actually relational, be it by Codd's twelve rules or by the more formal definitions in his papers, in his books or in succeeding works in the academia or by its coworkers and successors, Christopher J. Date, Hugh Darwen, David McGoveran and Fabian Pascal. Only less known DBMSs, most of them academic, strive to comply. The only commercial example, as of December 2010[1], is Dataphor. Some rules are controversial, especially rule three, because of the debate on three-valued logic.

Rules

Rule 0: The *Foundation rule*:

> A relational database management system must manage its stored data using only its relational capabilities. The system must qualify as *relational*, as a *database*, and as a *management system*. For a system to qualify as a relational database management system (RDBMS), that system must use its *relational* facilities (exclusively) to *manage* the *database*.

Rule 1: The *information rule*:

> All information in a relational database (including table and column names) is represented in only one way, namely as a value in a table.

Rule 2: The *guaranteed access rule*:

> All data must be accessible. This rule is essentially a restatement of the fundamental requirement for primary keys. It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column and the primary key value of the containing

row.

Rule 3: *Systematic treatment of null values*:

The DBMS must allow each field to remain null (or empty). Specifically, it must support a representation of "missing information and inapplicable information" that is systematic, distinct from all regular values (for example, "distinct from zero or any other number", in the case of numeric values), and independent of data type. It is also implied that such representations must be manipulated by the DBMS in a systematic way.

Rule 4: *Active online catalog based on the relational model*:

The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

Rule 5: The *comprehensive data sublanguage rule*:

The system must support at least one relational language that

1. Has a linear syntax
2. Can be used both interactively and within application programs,
3. Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).

Rule 6: The *view updating rule*:

All views that are theoretically updatable must be updatable by the system.

Rule 7: *High-level insert, update, and delete*:

The system must support set-at-a-time *insert*, *update*, and *delete* operators. This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8: *Physical data independence*:

Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

Rule 9: *Logical data independence*:

Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure. Logical data independence is more difficult to achieve than physical data independence.

Rule 10: *Integrity independence*:

Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

Rule 11: *Distribution independence*:

The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully:

1. when a distributed version of the DBMS is first introduced; and
2. when existing distributed data are redistributed around the system.

Rule 12: The *nonsubversion rule*:

If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.

References

[1] http://en.wikipedia.org/w/index.php?title=Codd%27s_12_rules&action=edit

Further reading

- Codd, Edgar F. (1990). *The relational model for database management: Version 2*. Addison-Wesley. ISBN 9780201141924.
- Harrington, Jan L. (2002). "Codd's Rules". *Relational Database Design Clearly Explained*. The Morgan Kaufmann Series in Data Management Systems (2nd ed.). Morgan Kaufmann. ISBN 9781558608207.
- Krishna, S. (1992). "Criteria for Evaluating Relational Database Systems". *Introduction to Database and Knowledge-Base Systems*. Computer Science **28**. World Scientific. pp. 91 et seq. ISBN 9789810206192.

Edgar F. Codd

Edgar Frank "Ted" Codd	
Born	August 19, 1923 Isle of Portland, England
Died	April 18, 2003 (aged 79) Williams Island, Aventura, Florida, USA
Fields	Computer Science
Institutions	University of Oxford University of Michigan IBM
Alma mater	Exeter College, Oxford University of Michigan
Thesis	*Propagation, Computation, and Construction in Two-dimensional cellular spaces* [1] (1965)
Doctoral advisor	John Henry Holland
Known for	OLAP Relational model Codd's cellular automaton Codd's 12 rules Boyce–Codd normal form
Notable awards	Turing Award

Edgar Frank "Ted" Codd (August 19, 1923 – April 18, 2003) was an English computer scientist who, while working for IBM, invented the relational model for database management, the theoretical basis for relational databases. He made other valuable contributions to computer science, but the relational model, a very influential general theory of data management, remains his most mentioned achievement.

Biography

Edgar Frank Codd was born on the Isle of Portland in England. After attending Poole Grammar School, he studied mathematics and chemistry at Exeter College, Oxford, before serving as a pilot in the Royal Air Force during the Second World War. In 1948, he moved to New York to work for IBM as a mathematical programmer. In 1953, angered by Senator Joseph McCarthy, Codd moved to Ottawa, Canada. A decade later he returned to the U.S. and received his doctorate in computer science from the University of Michigan in Ann Arbor. Two years later he moved to San Jose, California, to work at IBM's San Jose Research Laboratory, where he continued to work until the 1980s.[2] He was appointed IBM Fellow in 1976. During the 1990s, his health deteriorated and he ceased work.

Codd received the Turing Award in 1981, and in 1994 he was inducted as a Fellow of the Association for Computing Machinery.[3]

Codd died of heart failure at his home in Williams Island, Florida, at the age of 79 on April 18, 2003.[4]

Work

Codd received a PhD in 1965 from the University of Michigan, Ann Arbor advised by John Henry Holland. His thesis was about self-replication in cellular automata, extending on work of von Neumann and showing that a set of eight states was sufficient for universal computation and construction. His design for a self-replicating computer was only implemented in 2010.

In the 1960s and 1970s he worked out his theories of data arrangement, issuing his paper "A Relational Model of Data for Large Shared Data Banks" in 1970, after an internal IBM paper one year earlier.[5] To his disappointment, IBM proved slow to exploit his suggestions until commercial rivals started implementing them.

Initially, IBM refused to implement the relational model in order to preserve revenue from IMS/DB. Codd then showed IBM customers the potential of the implementation of its model, and they in turn pressured IBM. Then IBM included in its Future Systems project a System R subproject — but put in charge of it developers who were not thoroughly familiar with Codd's ideas, and isolated the team from Codd[*citation needed*]. As a result, they did not use Codd's own Alpha language but created a non-relational one, SEQUEL. Even so, SEQUEL was so superior to pre-relational systems that it was copied, in 1979, based on pre-launch papers presented at conferences, by Larry Ellison, of Relational software Inc, in his Oracle Database, which actually reached market before SQL/DS — because of the then-already proprietary status of the original name, SEQUEL had been renamed SQL.

Codd continued to develop and extend his relational model, sometimes in collaboration with Chris Date. One of the normalized forms, the Boyce–Codd normal form, is named after him.

Codd's theorem, a result proven in his seminal work on the relational model, equates the expressive power of relational algebra and relational calculus (which, in essence, is equivalent to first-order logic).

As the relational model started to become fashionable in the early 1980s, Codd fought a sometimes bitter campaign to prevent the term being misused by database vendors who had merely added a relational veneer to older technology. As part of this campaign, he published his 12 rules to define what constituted a relational database. This made his position in IBM increasingly difficult, so he left to form his own consulting company with Chris Date and others.

Codd coined the term *Online analytical processing (OLAP)* and wrote the "twelve laws of online analytical processing".[6] Controversy erupted, however, after it was discovered that this paper had been sponsored by Arbor Software (subsequently Hyperion, now acquired by Oracle), a conflict of interest that had not been disclosed, and ComputerWorld withdrew the paper.[7]

In 2004, SIGMOD renamed its highest prize to the SIGMOD Edgar F. Codd Innovations Award, in his honour.

Publications

- Codd, E.F. (1970). "Relational Completeness of Data Base Sublanguages". *Database Systems*: 65–98. CiteSeerX: 10.1.1.86.9277 [8].
- Codd, E.F. (1981-11-09). "1981 Turing Award Lecture - Relational Database: A Practical Foundation for Productivity" [9].
- Codd, E.F. (1990). *The Relational Model for Database Management* (Version 2 ed.). Addison Wesley Publishing Company. ISBN 0-201-14192-2.
- Codd, E.F.; Codd S.B. and Salley C.T. (1993). "Providing OLAP to User-Analysts: An IT Mandate" [10].

References

[1] http://search.proquest.com/docview/302172044

[2] Rubenstein, Steve. "Edgar F. Codd -- computer pioneer in databases." San Francisco Chronicle 24 Apr. 2003: A21. Gale Biography In Context. Web. 1 Dec. 2011.

[3] ACM Fellows (http://fellows.acm.org/homepage.cfm?alpha=C&srt=alpha)

[4] Edgar F Codd Passes Away (http://www.research.ibm.com/resources/news/20030423_edgarpassaway.shtml), IBM Research, 2003 Apr 23.

[5] Michael Owens. The Definitive Guide to SQLite, p.47. New York: Apress (Springer-Verlag) 2006. ISBN 978-1-59059-673-9.

[6] Providing OLAP to User-Analysts: An IT Mandate by E F Codd, S B Codd and C T Salley, ComputerWorld, July 26, 1993.

[7] Mark Whitehorn. OLAP and the need for speed. URL: http://www.theregister.co.uk/2007/01/26/olap_speed/Accessed: 2012-5-16.

[8] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.9277

[9] http://delivery.acm.org/10.1145/1290000/1283937/a1981-codd.pdf?ip=178.25.80.110&acc=OPEN&key=1B55DF923F77674F55057ED4F3766CA0&CFID=221194362&CFTOKEN=58865617&__acm__=1370110684_b7591082d2746b1ffcc3d8188edfbad3

[10] http://dev.hyperion.com/resource_library/white_papers/providing_olap_to_user_analysts.pdf

Further reading

- National Academy of Sciences (1999). "Chapt. 6: The Rise of Relational Databases" (http://www.nap.edu/readingroom/books/far/ch6.html). *Funding a Revolution: Government Support for Computing Research* (http://www.nap.edu/readingroom/books/far/). Washington DC, USA: National Academy Press.
- Date, C.J. (2000). *The Database Relational Model: A Retrospective Review and Analysis: A Historical Account and Assessment of E. F. Codd's Contribution to the Field of Database Technology*. Addison Wesley Longman. ISBN 0-201-61294-1.

Relational algebra

Relational algebra

In computer science, **relational algebra** is an offshoot of first-order logic and of algebra of sets concerned with operations over finitary relations, usually made more convenient to work with by identifying the components of a tuple by a name (called attribute) rather than by a numeric column index, which is called a relation in database terminology.

The main application of relational algebra is providing a theoretical foundation for relational databases, particularly query languages for such databases, chief among which is SQL.

Introduction

Relational algebra received little attention outside of pure mathematics until the publication of E.F. Codd's relational model of data in 1970. Codd proposed such an algebra as a basis for database query languages. (See section Implementations.)

Both a named and an unnamed perspective are possible for relational algebra, depending on whether the tuples are endowed with component names or not. In the unnamed perspective, a tuple is simply a member of a Cartesian product. In the named perspective, tuples are functions from a finite set U of attributes (of the relation) to a domain of values (assumed distinct from U).[1] The relational algebras obtained from the two perspectives are equivalent.[2] The typical undergraduate textbooks present only the named perspective though, and this article follows suit.

Relational algebra is essentially equivalent in expressive power to relational calculus (and thus first-order logic); this result is known as Codd's theorem. One must be careful to avoid a mismatch that may arise between the two languages because negation, applied to a formula of the calculus, constructs a formula that may be true on an infinite set of possible tuples, while the difference operator of relational algebra always returns a finite result. To overcome these difficulties, Codd restricted the operands of relational algebra to finite relations only and also proposed restricted support for negation (NOT) and disjunction (OR). Analogous restrictions are found in many other logic-based computer languages. Codd defined the term ***relational completeness*** to refer to a language that is complete with respect to first-order predicate calculus apart from the restrictions he proposed. In practice the restrictions have no adverse effect on the applicability of his relational algebra for database purposes.

Primitive operations

As in any algebra, some operators are primitive and the others are derived in terms of the primitive ones. It is useful if the choice of primitive operators parallels the usual choice of primitive logical operators.

Five primitive operators of Codd's algebra are the *selection*, the *projection*, the *Cartesian product* (also called the *cross product* or *cross join*), the *set union*, and the *set difference*. Another operator, *rename* was not noted by Codd, but the need for it is shown by the inventors of ISBL. These six operators are fundamental in the sense that omitting any one of them causes a loss of expressive power. Many other operators have been defined in terms of these six. Among the most important are set intersection, division, and the natural join. In fact ISBL made a compelling case for replacing the Cartesian product with the natural join, of which the Cartesian product is a degenerate case.

Altogether, the operators of relational algebra have an expressive power identical to that of domain relational calculus or tuple relational calculus. However, for the reasons given in section Introduction, relational algebra is less expressive than first-order predicate calculus without function symbols. Relational algebra corresponds to a subset of

first-order logic, namely Horn clauses without recursion and negation.

Set operators

The relational algebra uses set union, set difference, and Cartesian product from set theory, but adds additional constraints to these operators.

For set union and set difference, the two relations involved must be *union-compatible*—that is, the two relations must have the same set of attributes. Because set intersection can be defined in terms of set difference, the two relations involved in set intersection must also be union-compatible.

For the Cartesian product to be defined, the two relations involved must have disjoint headers—that is, they must not have a common attribute name.

In addition, the Cartesian product is defined differently from the one in set theory in the sense that tuples are considered to be "shallow" for the purposes of the operation. That is, the Cartesian product of a set of *n*-tuples with a set of *m*-tuples yields a set of "flattened" $(n + m)$-tuples (whereas basic set theory would have prescribed a set of 2-tuples, each containing an *n*-tuple and an *m*-tuple). More formally, $R \times S$ is defined as follows:

$$R \times S = \{(r_1, r_2, ..., r_n, s_1, s_2, ..., s_m) \mid (r_1, r_2, ..., r_n) \in R, (s_1, s_2, ..., s_m) \in S\}$$

The cardinality of the Cartesian product is the product of the cardinalities of its factors, i.e., $|R \times S| = |R| \times |S|$.

Projection (π)

A **projection** is a unary operation written as $\pi_{a_1,\ldots,a_n}(R)$ where $a_1, \ldots, a_n$ is a set of attribute names. The result of such projection is defined as the set that is obtained when all tuples in *R* are restricted to the set $\{a_1, \ldots, a_n\}$. This specifies the specific subset of columns (attributes of each tuple) to be retrieved. To obtain the names and phone numbers from an address book, the projection might be written $\pi_{\text{contactName, contactPhoneNumber}}(\text{addressBook})$. The result of that projection would be a relation which contains only the contactName and contactPhoneNumber attributes for each unique entry in addressBook.

Selection (σ)

A **generalized selection** is a unary operation written as $\sigma_{\varphi}(R)$ where φ is a propositional formula that consists of atoms as allowed in the normal selection and the logical operators $\wedge$ (and), $\vee$ (or) and $\neg$ (negation). This selection selects all those tuples in *R* for which φ holds.

To obtain a listing of all friends or business associates in an address book, the selection might be written as $\sigma_{\text{isFriend} = \text{true} \vee \text{isBusinessContact} = \text{true}}(\text{addressBook})$. The result would be a relation containing every attribute of every unique record where isFriend is true or where isBusinessContact is true.

In Codd's 1970 paper, selection is called restriction.

Rename (ϱ)

A **rename** is a unary operation written as $\rho_{a/b}(R)$ where the result is identical to *R* except that the *b* attribute in all tuples is renamed to an *a* attribute. This is simply used to rename the attribute of a relation or the relation itself.
To rename the 'isFriend' attribute to 'isBusinessContact' in a relation, $\rho_{\text{isBusinessContact / isFriend}}(\text{addressBook})$ might be used.

Joins and join-like operators

Natural join (⋈)

Natural join (⋈) is a binary operator that is written as (*R* ⋈ *S*) where *R* and *S* are relations.[3] The result of the natural join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names. For an example consider the tables *Employee* and *Dept* and their natural join:

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

This can also be used to define composition of relations. For example, the composition of *Employee* and *Dept* is their join as shown above, projected on all but the common attribute *DeptName*. In category theory, the join is precisely the fiber product.

The natural join is arguably one of the most important operators since it is the relational counterpart of logical AND. Note carefully that if the same variable appears in each of two predicates that are connected by AND, then that variable stands for the same thing and both appearances must always be substituted by the same value. In particular, natural join allows the combination of relations that are associated by a foreign key. For example, in the above example a foreign key probably holds from *Employee.DeptName* to *Dept.DeptName* and then the natural join of *Employee* and *Dept* combines all employees with their departments. Note that this works because the foreign key holds between attributes with the same name. If this is not the case such as in the foreign key from *Dept.manager* to *Employee.Name* then we have to rename these columns before we take the natural join. Such a join is sometimes also referred to as an **equijoin** (see θ-join).

More formally the semantics of the natural join are defined as follows:

$$R \bowtie S = \{t \cup s \mid t \in R \ \wedge \ s \in S \ \wedge \ Fun(t \cup s)\}$$

where *Fun* is a predicate that is true for a relation *r* if and only if *r* is a function. It is usually required that *R* and *S* must have at least one common attribute, but if this constraint is omitted, and *R* and *S* have no common attributes, then the natural join becomes exactly the Cartesian product.

The natural join can be simulated with Codd's primitives as follows. Assume that $c_1,...,c_m$ are the attribute names common to *R* and *S*, $r_1,...,r_n$ are the attribute names unique to *R* and $s_1,...,s_k$ are the attribute unique to *S*. Furthermore assume that the attribute names $x_1,...,x_m$ are neither in *R* nor in *S*. In a first step we can now rename the common attribute names in *S*:

$$T = \rho_{x_1/c_1,\ldots,x_m/c_m}(S) = \rho_{x_1/c_1}(\rho_{x_2/c_2}(\ldots \rho_{x_m/c_m}(S)\ldots))$$

Then we take the Cartesian product and select the tuples that are to be joined:

$$P = \sigma_{c_1=x_1,\ldots,c_m=x_m}(R \times T) = \sigma_{c_1=x_1}(\sigma_{c_2=x_2}(\ldots \sigma_{c_m=x_m}(R \times T)\ldots))$$

Finally we take a projection to get rid of the renamed attributes:

$$U = \pi_{r_1,\ldots,r_n,c_1,\ldots,c_m,s_1,\ldots,s_k}(P)$$

θ-join and equijoin

Consider tables *Car* and *Boat* which list models of cars and boats and their respective prices. Suppose a customer wants to buy a car and a boat, but she does not want to spend more money for the boat than for the car. The θ-join ($\bowtie_\theta$) on the relation *CarPrice* ≥ *BoatPrice* produces a table with all the possible options. When using a condition where the attributes are equal, for example Price, then the condition may be specified as *Price=Price* or alternatively *(Price)* itself.

CarModel	CarPrice
CarA	20,000
CarB	30,000
CarC	50,000

BoatModel	BoatPrice
Boat1	10,000
Boat2	40,000
Boat3	60,000

CarModel	CarPrice	BoatModel	BoatPrice
CarA	20,000	Boat1	10,000
CarB	30,000	Boat1	10,000
CarC	50,000	Boat1	10,000
CarC	50,000	Boat2	40,000

If we want to combine tuples from two relations where the combination condition is not simply the equality of shared attributes then it is convenient to have a more general form of join operator, which is the θ-join (or theta-join). The θ-join is a binary operator that is written as $R \underset{a\ \theta\ b}{\bowtie} S$ or $R \underset{a\ \theta\ v}{\bowtie} S$ where *a* and *b* are attribute names, θ is a binary relation in the set {<, ≤, =, >, ≥}, *v* is a value constant, and *R* and *S* are relations. The result of this operation consists of all combinations of tuples in *R* and *S* that satisfy the relation θ. The result of the θ-join is defined only if the headers of *S* and *R* are disjoint, that is, do not contain a common attribute.

The simulation of this operation in the fundamental operations is therefore as follows:

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$

In case the operator θ is the equality operator (=) then this join is also called an **equijoin**.

Note, however, that a computer language that supports the natural join and rename operators does not need θ-join as well, as this can be achieved by selection from the result of a natural join (which degenerates to Cartesian product when there are no shared attributes).

Semijoin (▯)(▯)

The left semijoin is joining similar to the natural join and written as $R \ltimes S$ where *R* and *S* are relations.[4] The result of this semijoin is the set of all tuples in *R* for which there is a tuple in *S* that is equal on their common attribute names. For an example consider the tables *Employee* and *Dept* and their semi join:

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production

DeptName	Manager
Sales	Bob
Sales	Thomas
Production	Katie
Production	Mark

Name	EmpId	DeptName
Sally	2241	Sales
Harriet	2202	Production

More formally the semantics of the semijoin can be defined as follows:

$$R \ltimes S = \{\, t : t \in R \wedge \exists\, s \in S(Fun\,(t \cup s)) \,\}$$

where $Fun(r)$ is as in the definition of natural join.

The semijoin can be simulated using the natural join as follows. If a_1, ..., a_n are the attribute names of R, then

$$R \ltimes S = \pi_{a1,..,a_n}(R \bowtie S).$$

Since we can simulate the natural join with the basic operators it follows that this also holds for the semijoin.

Antijoin (▷)

The antijoin, written as $R \triangleright S$ where R and S are relations, is similar to the semijoin, but the result of an antijoin is only those tuples in R for which there is *no* tuple in S that is equal on their common attribute names.[5]

For an example consider the tables *Employee* and *Dept* and their antijoin:

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production

DeptName	Manager
Sales	Sally
Production	Harriet

Name	EmpId	DeptName
Harry	3415	Finance
George	3401	Finance

The antijoin is formally defined as follows:

$$R \triangleright S = \{ t : t \in R \land \neg\exists\, s \in S(Fun\,(t \cup s)) \}$$

or

$$R \triangleright S = \{ t : t \in R, \text{there is no tuple } s \text{ of } S \text{ that satisfies } Fun\,(t \cup s) \}$$

where $Fun(r)$ is as in the definition of natural join.

The antijoin can also be defined as the complement of the semijoin, as follows:

$$R \triangleright S = R - R \ltimes S$$

Given this, the antijoin is sometimes called the anti-semijoin, and the antijoin operator is sometimes written as semijoin symbol with a bar above it, instead of ▷.

Division (÷)

The division is a binary operation that is written as $R \div S$. The result consists of the restrictions of tuples in R to the attribute names unique to R, i.e., in the header of R but not in the header of S, for which it holds that all their combinations with tuples in S are present in R. For an example see the tables *Completed*, *DBProject* and their division:

Student	Task
Fred	Database1
Fred	Database2
Fred	Compiler1
Eugene	Database1
Eugene	Compiler1
Sarah	Database1
Sarah	Database2

Task
Database1
Database2

Student
Fred
Sarah

If *DBProject* contains all the tasks of the Database project, then the result of the division above contains exactly the students who have completed both of the tasks in the Database project.

More formally the semantics of the division is defined as follows:

$$R \div S = \{\ t[a_1,...,a_n] : t \in R \wedge \forall\ s \in S\ (\ (t[a_1,...,a_n] \cup s) \in R)\ \}$$

where $\{a_1,...,a_n\}$ is the set of attribute names unique to R and $t[a_1,...,a_n]$ is the restriction of t to this set. It is usually required that the attribute names in the header of S are a subset of those of R because otherwise the result of the operation will always be empty.

The simulation of the division with the basic operations is as follows. We assume that $a_1,...,a_n$ are the attribute names unique to R and $b_1,...,b_m$ are the attribute names of S. In the first step we project R on its unique attribute names and construct all combinations with tuples in S:

$$T := \pi_a 1,...,a_n(R) \times S$$

In the prior example, T would represent a table such that every Student (because Student is the unique key / attribute of the Completed table) is combined with every given Task. So Eugene, for instance, would have two rows, Eugene -> Database1 and Eugene -> Database2 in T.

In the next step we subtract R from this relation:

$$U := T - R$$

Note that in U we have the possible combinations that "could have" been in R, but weren't. So if we now take the projection on the attribute names unique to R then we have the restrictions of the tuples in R for which not all combinations with tuples in S were present in R:

$$V := \pi_a 1,...,a_n(U)$$

So what remains to be done is take the projection of R on its unique attribute names and subtract those in V:

$$W := \pi_a 1,...,a_n(R) - V$$

Common extensions

In practice the classical relational algebra described above is extended with various operations such as outer joins, aggregate functions and even transitive closure.

Outer joins

Whereas the result of a join (or inner join) consists of tuples formed by combining matching tuples in the two operands, an outer join contains those tuples and additionally some tuples formed by extending an unmatched tuple in one of the operands by "fill" values for each of the attributes of the other operand. Note that outer joins are not considered part of the classical relational algebra discussed so far.

The operators defined in this section assume the existence of a *null* value, ω, which we do not define, to be used for the fill values; in practice this corresponds to the NULL in SQL. In order to make subsequent selection operations on the resulting table meaningful, a semantic meaning needs to be assigned to nulls; in Codd's approach the propositional logic used by the selection is extended to a three-valued logic, although we elide those details in this article.

Three outer join operators are defined: left outer join, right outer join, and full outer join. (The word "outer" is sometimes omitted.)

Left outer join (⟕)

The left outer join is written as $R \mathbin{⟕} S$ where R and S are relations.[6] The result of the left outer join is the set of all combinations of tuples in R and S that are equal on their common attribute names, in addition (loosely speaking) to tuples in R that have no matching tuples in S.

For an example consider the tables *Employee* and *Dept* and their left outer join:

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

DeptName	Manager
Sales	Harriet
Production	Charles

Name	EmpId	DeptName	Manager
Harry	3415	Finance	ω
Sally	2241	Sales	Harriet
George	3401	Finance	ω
Harriet	2202	Sales	Harriet
Tim	1123	Executive	ω

In the resulting relation, tuples in S which have no common values in common attribute names with tuples in R take a *null* value, ω.

Since there are no tuples in *Dept* with a *DeptName* of *Finance* or *Executive*, ωs occur in the resulting relation where tuples in *Employee* have a *DeptName* of *Finance* or *Executive*.

Let r_1, r_2, ..., r_n be the attributes of the relation R and let $\{(\omega, ..., \omega)\}$ be the singleton relation on the attributes that are *unique* to the relation S (those that are not attributes of R). Then the left outer join can be described in terms of the natural join (and hence using basic operators) as follows:

$$(R \bowtie S) \cup ((R - \pi_{r_1, r_2, \ldots, r_n}(R \bowtie S)) \times \{(\omega, \ldots \omega)\})$$

Right outer join (⟖)

The right outer join behaves almost identically to the left outer join, but the roles of the tables are switched.

The right outer join of relations R and S is written as $R \mathbin{⟖} S$.[7] The result of the right outer join is the set of all combinations of tuples in R and S that are equal on their common attribute names, in addition to tuples in S that have no matching tuples in R.

For example consider the tables *Employee* and *Dept* and their right outer join:

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

DeptName	Manager
Sales	Harriet
Production	Charles

Name	EmpId	DeptName	Manager
Sally	2241	Sales	Harriet
Harriet	2202	Sales	Harriet
ω	ω	Production	Charles

In the resulting relation, tuples in R which have no common values in common attribute names with tuples in S take a *null* value, ω.

Since there are no tuples in *Employee* with a *DeptName* of *Production*, ωs occur in the Name attribute of the resulting relation where tuples in *DeptName* had tuples of *Production*.

Let s_1, s_2, ..., s_n be the attributes of the relation S and let $\{(\omega, ..., \omega)\}$ be the singleton relation on the attributes that are *unique* to the relation R (those that are not attributes of S). Then, as with the left outer join, the right outer join

can be simulated using the natural join as follows:

$$(R \bowtie S) \cup (\{(\omega, \ldots, \omega)\} \times (S - \pi_{s_1, s_2, \ldots, s_n}(R \bowtie S)))$$

Full outer join ( )

The **outer join** or **full outer join** in effect combines the results of the left and right outer joins.

The full outer join is written as R S where R and S are relations.[8] The result of the full outer join is the set of all combinations of tuples in R and S that are equal on their common attribute names, in addition to tuples in S that have no matching tuples in R and tuples in R that have no matching tuples in S in their common attribute names.

For an example consider the tables *Employee* and *Dept* and their full outer join:

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

DeptName	Manager
Sales	Harriet
Production	Charles

Name	EmpId	DeptName	Manager
Harry	3415	Finance	ω
Sally	2241	Sales	Harriet
George	3401	Finance	ω
Harriet	2202	Sales	Harriet
Tim	1123	Executive	ω
ω	ω	Production	Charles

In the resulting relation, tuples in R which have no common values in common attribute names with tuples in S take a *null* value, ω. Tuples in S which have no common values in common attribute names with tuples in R also take a *null* value, ω.

The full outer join can be simulated using the left and right outer joins (and hence the natural join and set union) as follows:

R $S = (R$ $S) \cup (R$ $S)$

Operations for domain computations

There is nothing in relational algebra introduced so far that would allow computations on the data domains (other than evaluation of propositional expressions involving equality). For example, it's not possible using only the algebra introduced so far to write an expression that would multiply the numbers from two columns, e.g. a unit price with a quantity to obtain a total price. Practical query languages have such facilities, e.g. the SQL SELECT allows arithmetic operations to define new columns in the result `SELECT unit_price * quantity AS total_price FROM t`, and a similar facility is provided more explicitly by Tutorial D's `EXTEND` keyword. In database theory, this is called **extended projection**.:213

Aggregation

Furthermore, computing various functions on a column, like the summing up its elements, is also not possible using the relational algebra introduced so far. There are five aggregate functions that are included with most relational database systems. These operations are Sum, Count, Average, Maximum and Minimum. In relational algebra the aggregation operation over a schema $(A_1, A_2, \ldots A_n)$ is written as follows:

$G_1, G_2, \ldots, G_m\ g\ f_1(A_1'), f_2(A_2'), \ldots, f_k(A_k')$ (**r**)

where each A_j', $1 \le j \le k$, is one of the original attributes A_i, $1 \le i \le n$.

The attributes preceding the g are grouping attributes, which function like a "group by" clause in SQL. Then there are an arbitrary number of aggregation functions applied to individual attributes. The operation is applied to an arbitrary relation **r**. The grouping attributes are optional, and if they are not supplied, the aggregation functions are

applied across the entire relation to which the operation is applied.

Let's assume that we have a table named Account with three columns, namely Account_Number, Branch_Name and Balance. We wish to find the maximum balance of each branch. This is accomplished by ${}_{\text{Branch\_Name}}\text{G}_{\text{Max(Balance)}}$(Account). To find the highest balance of all accounts regardless of branch, we could simply write $\text{G}_{\text{Max(Balance)}}$(Account).

Transitive closure

Although relational algebra seems powerful enough for most practical purposes, there are some simple and natural operators on relations which cannot be expressed by relational algebra. One of them is the transitive closure of a binary relation. Given a domain *D*, let binary relation *R* be a subset of *D*×*D*. The transitive closure R^+ of *R* is the smallest subset of *D*×*D* containing *R* which satisfies the following condition:

$$\forall x \forall y \forall z \left((x, y) \in R^+ \wedge (y, z) \in R^+ \Rightarrow (x, z) \in R^+ \right)$$

There is no relational algebra expression *E*(*R*) taking *R* as a variable argument which produces R^+. This can be proved using the fact that, given a relational expression *E* for which it is claimed that *E*(*R*) = R^+, where *R* is a variable, we can always find an instance ***r*** of *R* (and a corresponding domain **d**) such that *E*(***r***) ≠ $\boldsymbol{r}^+$.

SQL however officially supports such fixpoint queries since 1999, and it had vendor-specific extensions in this direction well before that.

Use of algebraic properties for query optimization

Queries can be represented as a tree, where

- the internal nodes are operators,
- leaves are relations,
- subtrees are subexpressions.

Our primary goal is to transform expression trees into equivalent expression trees, where the average size of the relations yielded by subexpressions in the tree is smaller than it was before the optimization. Our secondary goal is to try to form common subexpressions within a single query, or if there is more than one query being evaluated at the same time, in all of those queries. The rationale behind the second goal is that it is enough to compute common subexpressions once, and the results can be used in all queries that contain that subexpression.

Here we present a set of rules that can be used in such transformations.

Selection

Rules about selection operators play the most important role in query optimization. Selection is an operator that very effectively decreases the number of rows in its operand, so if we manage to move the selections in an expression tree towards the leaves, the internal relations (yielded by subexpressions) will likely shrink.

Basic selection properties

Selection is idempotent (multiple applications of the same selection have no additional effect beyond the first one), and commutative (the order selections are applied in has no effect on the eventual result).

1. $\sigma_A(R) = \sigma_A\sigma_A(R)$
2. $\sigma_A\sigma_B(R) = \sigma_B\sigma_A(R)$

Breaking up selections with complex conditions

A selection whose condition is a conjunction of simpler conditions is equivalent to a sequence of selections with those same individual conditions, and selection whose condition is a disjunction is equivalent to a union of selections. These identities can be used to merge selections so that fewer selections need to be evaluated, or to split them so that the component selections may be moved or optimized separately.

1. $\sigma_{A \wedge B}(R) = \sigma_A(\sigma_B(R)) = \sigma_B(\sigma_A(R))$
2. $\sigma_{A \vee B}(R) = \sigma_A(R) \cup \sigma_B(R)$

Selection and cross product

Cross product is the costliest operator to evaluate. If the input relations have *N* and *M* rows, the result will contain NM rows. Therefore it is very important to do our best to decrease the size of both operands before applying the cross product operator.

This can be effectively done, if the cross product is followed by a selection operator, e.g. $\sigma_A (R \times P)$. Considering the definition of join, this is the most likely case. If the cross product is not followed by a selection operator, we can try to push down a selection from higher levels of the expression tree using the other selection rules.

In the above case we break up condition *A* into conditions *B*, *C* and *D* using the split rules about complex selection conditions, so that $A = B \wedge C \wedge D$ and *B* only contains attributes from *R*, *C* contains attributes only from *P* and *D* contains the part of *A* that contains attributes from both *R* and *P*. Note, that *B*, *C* or *D* are possibly empty. Then the following holds:

$$\sigma_A(R \times P) = \sigma_{B \wedge C \wedge D}(R \times P) = \sigma_D(\sigma_B(R) \times \sigma_C(P))$$

Selection and set operators

Selection is distributive over the setminus, intersection, and union operators. The following three rules are used to push selection below set operations in the expression tree. Note, that in the setminus and the intersection operators it is possible to apply the selection operator to only one of the operands after the transformation. This can make sense in cases, where one of the operands is small, and the overhead of evaluating the selection operator outweighs the benefits of using a smaller relation as an operand.

1. $\sigma_A(R \setminus P) = \sigma_A(R) \setminus \sigma_A(P) = \sigma_A(R) \setminus P$
2. $\sigma_A(R \cup P) = \sigma_A(R) \cup \sigma_A(P)$
3. $\sigma_A(R \cap P) = \sigma_A(R) \cap \sigma_A(P) = \sigma_A(R) \cap P = R \cap \sigma_A(P)$

Selection and projection

Selection commutes with projection if and only if the fields referenced in the selection condition are a subset of the fields in the projection. Performing selection before projection may be useful if the operand is a cross product or join. In other cases, if the selection condition is relatively expensive to compute, moving selection outside the projection may reduce the number of tuples which must be tested (since projection may produce fewer tuples due to the elimination of duplicates resulting from omitted fields).

$$\pi_{a_1,\ldots,a_n}(\sigma_A(R)) = \sigma_A(\pi_{a_1,\ldots,a_n}(R)) \text{ where fields in } A \subseteq \{a_1, \ldots, a_n\}$$

Projection

Basic projection properties

Projection is idempotent, so that a series of (valid) projections is equivalent to the outermost projection.

$$\pi_{a_1,\ldots,a_n}(\pi_{b_1,\ldots,b_m}(R)) = \pi_{a_1,\ldots,a_n}(R) \text{ where } \{a_1,\ldots,a_n\} \subseteq \{b_1,\ldots,b_m\}$$

Projection and set operators

Projection is distributive over set union.

$$\pi_{a_1,\ldots,a_n}(R \cup P) = \pi_{a_1,\ldots,a_n}(R) \cup \pi_{a_1,\ldots,a_n}(P).$$

Projection does not distribute over intersection and set difference. Counterexamples are given by:

$$\pi_A(\{\langle A=a, B=b\rangle\} \cap \{\langle A=a, B=b'\rangle\}) = \emptyset$$
$$\pi_A(\{\langle A=a, B=b\rangle\}) \cap \pi_A(\{\langle A=a, B=b'\rangle\}) = \{\langle A=a\rangle\}$$

and

$$\pi_A(\{\langle A=a, B=b\rangle\} \setminus \{\langle A=a, B=b'\rangle\}) = \{\langle A=a\}$$
$$\pi_A(\{\langle A=a, B=b\rangle\}) \setminus \pi_A(\{\langle A=a, B=b'\rangle\}) = \emptyset\,,$$

where *b* is assumed to be distinct from *b′*.

Rename

Basic rename properties

Successive renames of a variable can be collapsed into a single rename. Rename operations which have no variables in common can be arbitrarily reordered with respect to one another, which can be exploited to make successive renames adjacent so that they can be collapsed.

1. $\rho_{a/b}(\rho_{b/c}(R)) = \rho_{a/c}(R)$
2. $\rho_{a/b}(\rho_{c/d}(R)) = \rho_{c/d}(\rho_{a/b}(R))$

Rename and set operators

Rename is distributive over set difference, union, and intersection.

1. $\rho_{a/b}(R \setminus P) = \rho_{a/b}(R) \setminus \rho_{a/b}(P)$
2. $\rho_{a/b}(R \cup P) = \rho_{a/b}(R) \cup \rho_{a/b}(P)$
3. $\rho_{a/b}(R \cap P) = \rho_{a/b}(R) \cap \rho_{a/b}(P)$

Implementations

The first query language to be based on Codd's algebra was ISBL, and this pioneering work has been acclaimed by many authorities as having shown the way to make Codd's idea into a useful language. Business System 12 was a short-lived industry-strength relational DBMS that followed the ISBL example.

In 1998 Chris Date and Hugh Darwen proposed a language called **Tutorial D** intended for use in teaching relational database theory, and its query language also draws on ISBL's ideas. Rel is an implementation of **Tutorial D**.

Even the query language of SQL is loosely based on a relational algebra, though the operands in SQL (tables) are not exactly relations and several useful theorems about the relational algebra do not hold in the SQL counterpart (arguably to the detriment of optimisers and/or users). The SQL table model is a bag (multiset), rather than a set. For example, the expression (R ∪ S) − T = (R − T) ∪ (S − T) is a theorem for relational algebra on sets, but not for relational algebra on bags; for a treatment of relational algebra on bags see chapter 5 of the "Complete" textbook by Garcia-Molina, Ullman and Widom.

References

[1] Serge Abiteboul, Richard Hull, Victor Vianu, *Foundations of databases*, Addison-Wesley, 1995, ISBN 0-201-53771-0, p. 29–33
[2] Serge Abiteboul, Richard Hull, Victor Vianu, *Foundations of databases*, Addison-Wesley, 1995, ISBN 0-201-53771-0, p. 59–63 and p. 71
[3] In Unicode, the bowtie symbol is (U+22C8).
[4] In Unicode, the ltimes symbol is (U+22C9). The rtimes symbol is (U+22CA)
[5] In Unicode, the Antijoin symbol is (U+25B7).
[6] In Unicode, the Left outer join symbol is (U+27D5).
[7] In Unicode, the Right outer join symbol is (U+27D6).
[8] In Unicode, the Full Outer join symbol is (U+27D7).

Further reading

Practically any academic textbook on databases has a detailed treatment of the classic relational algebra.

- Imieliński, T.; Lipski, W. (1984). "The relational model of data and cylindric algebras". *Journal of Computer and System Sciences* **28**: 80–102. doi: 10.1016/0022-0000(84)90077-1 (http://dx.doi.org/10.1016/0022-0000(84)90077-1). (For relationship with cylindric algebras).

External links

- RAT. Software Relational Algebra Translator to SQL (http://www.slinfo.una.ac.cr/rat/rat.html)
- Lecture Notes: Relational Algebra (http://www.databasteknik.se/webbkursen/relalg-lecture/index.html) – A quick tutorial to adapt SQL queries into relational algebra
- LEAP – An implementation of the relational algebra (http://leap.sourceforge.net)
- Relational – A graphic implementation of the relational algebra (http://galileo.dmi.unict.it/wiki/relational/)
- Query Optimization (http://www-db.stanford.edu/~widom/cs346/ioannidis.pdf) This paper is an introduction into the use of the relational algebra in optimizing queries, and includes numerous citations for more in-depth study.
- bandilab.org – neat graphical illustrations of the relational operators (http://bandilab.org/bandicoot-algebra.pdf)
- Relational Algebra System for Oracle and Microsoft SQL Server (http://www.cse.fau.edu/~marty#RADownload)

Projection (relational algebra)

In relational algebra, a **projection** is a unary operation written as $\Pi_{a_1,\ldots,a_n}(R)$ where $a_1, \ldots, a_n$ is a set of attribute names. The result of such projection is defined as the set obtained when the components of the tuple R are restricted to the set $\{a_1, \ldots, a_n\}$ – it *discards* (or *excludes*) the other attributes.[1]
In practical terms, it can be roughly thought of as picking a sub-set of all available columns. For example, if the attributes are (name, age), then projection of the relation {(Alice, 5), (Bob, 8)} onto attribute list (age) yields {5,8} – we have discarded the names, and only know what ages are present.

In addition, projection can be used to modify an attribute's value: if relation R has attributes a, b, and c, and b is a number, then $\Pi_{a,\ b*0.5,\ c}(R)$ will return a relation nearly the same as R, but with all values for 'b' shrunk by half.[2]

Related concepts

The closely related concept in set theory (see: projection (set theory)) differs from that of relational algebra in that, in set theory, one projects onto ordered components, not onto attributes. For instance, projecting $(3, 7)$ onto the second component yields 7.
Projection is relational algebra's counterpart of existential quantification in predicate logic. The attributes *not* included correspond to existentially quantified variables in the predicate whose extension the operand relation represents. The example below illustrates this point.

Because of the correspondence with existential quantification, some authorities prefer to define projection in terms of the excluded attributes. In a computer language it is of course possible to provide notations for both, and that was done in ISBL and several languages that have taken their cue from ISBL.

A nearly identical concept occurs in the category of monoids, called a string projection, which consists of removing all of the letters in the string that do not belong to a given alphabet.

Example

For an example, consider the relations depicted in the following two tables which are the relation $Person$ and its projection on (some say "over") the attributes Age and $Weight$:

$Person$ $\Pi_{Age,Weight}(Person)$

Name	Age	Weight
Harry	34	180
Sally	28	164
George	29	170
Helena	54	154
Peter	34	180

Age	Weight
34	180
28	164
29	170
54	154

Suppose the predicate of Person is "*Name* is *age* years old and weighs *weight*." Then the given projection represents the predicate, "There exists *Name* such that *Name* is *age* years old and weighs *weight*."

Note that Harry and Peter have the same age and weight, but since the result is a relation, and therefore a set, this combination only appears once in the result.

More formally the semantics of projection are defined as follows:

$$\Pi_{a_1,...,a_n}(R) = \{\ t[a_1, ..., a_n] :\ t \in R\ \}$$

where $t[a_1, ..., a_n]$ is the restriction of the tuple t to the set $\{a_1, ..., a_n\}$ so that

$$t[a_1, ..., a_n] = \{\ (a', v) \mid (a', v) \in t,\ a' \in a_1, ..., a_n\ \}$$

The result of a projection $\Pi_{a_1,...,a_n}(R)$ is defined only if $\{a_1, ..., a_n\}$ is a subset of the header of R.

It is interesting to note that projection over no attributes at all is possible, yielding a relation of degree zero. In this case the cardinality of the result is zero if the operand is empty, otherwise one. The two relations of degree zero are the only ones that cannot be depicted as tables.

References

[1] http://www.cs.rochester.edu/~nelson/courses/csc_173/relations/algebra.html

[2] http://www.csee.umbc.edu/~pmundur/courses/CMSC661-02/rel-alg.pdf *See Problem 3.8.B on page 3*

Rename (relational algebra)

In relational algebra, a **rename** is a unary operation written as $\rho_{a/b}(R)$ where:

- R is a relation
- a and b are attribute names
- b is an attribute of R

The result is identical to R except that the b attribute in all tuples is renamed to a. For an example, consider the following invocation of ρ on an $Employee$ relation and the result of that invocation:

$Employee$ $\rho_{EmployeeName/Name}(Employee)$

Name	EmployeeId
Harry	3415
Sally	2241

EmployeeName	EmployeeId
Harry	3415
Sally	2241

Formally the semantics of the rename operator is defined as follows:

$$\rho_{a/b}(R) = \{\ t[a/b] : t \in R\ \}$$

where $t[a/b]$ is defined as the tuple t with the b attribute renamed to a so that:

$$t[a/b] = \{\ (c, v) \mid (c, v) \in t,\ c \neq b\ \} \cup \{\ (a,\ t(b))\ \}$$

Selection (relational algebra)

In relational algebra, a **selection** (sometimes called a **restriction** to avoid confusion with SQL's use of SELECT) is a unary operation written as $\sigma_{a\theta b}(R)$ or $\sigma_{a\theta v}(R)$ where:

- a and b are attribute names
- θ is a binary operation in the set $\{ <, \leq, =, \geq, > \}$
- v is a value constant
- R is a relation

The selection $\sigma_{a\theta b}(R)$ selects all those tuples in R for which θ holds between the a and the b attribute.

The selection $\sigma_{a\theta v}(R)$ selects all those tuples in R for which θ holds between the a attribute and the value v.

For an example, consider the following tables where the first table gives the relation $Person$, the second table gives the result of $\sigma_{Age \geq 34}(Person)$ and the third table gives the result of $\sigma_{Age=Weight}(Person)$.

$Person$

Name	Age	Weight
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

$\sigma_{Age \geq 34}(Person)$

Name	Age	Weight
Harry	34	80
Helena	54	54
Peter	34	80

$\sigma_{Age=Weight}(Person)$

Name	Age	Weight
Helena	54	54

More formally the semantics of the selection is defined as follows:

$$\sigma_{a\theta b}(R) = \{ t : t \in R,\ t(a)\ \theta\ t(b) \}$$
$$\sigma_{a\theta v}(R) = \{ t : t \in R,\ t(a)\ \theta\ v \}$$

The result of the selection is only defined if the attribute names that it mentions are in the heading of the relation that it operates upon.

In computer languages it is expected that any truth-valued expression be permitted as the selection condition rather than restricting it to be a simple comparison.

In SQL, selections are performed by using `WHERE` definitions in `SELECT`, `UPDATE`, and `DELETE` statements, but note that the selection condition can result in any of three truth values (*true*, *false* and *unknown*) instead of the usual two.

References

- http://cisnet.baruch.cuny.edu/holowczak/classes/3400/relationalalgebra/#selectionoperator

Generalized selection

In relational algebra, a **generalized selection** is a unary operation written as $\sigma_{\varphi}(R)$ where φ is a propositional formula that consists of atoms as allowed in the normal selection and the logical operators $\wedge$ (and), $\vee$ (or) and $\neg$ (negation). This selection selects all those tuples in R for which φ holds.
For an example, consider the following tables where the first table gives the relation $Person$ and the second the result of $\sigma_{Age \geq 30 \ \wedge \ Weight \leq 60}(Person)$.

$Person$

Name	Age	Weight
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

$\sigma_{Age \geq 30 \ \wedge \ Weight \leq 60}(Person)$

Name	Age	Weight
Helena	54	54

Formally the semantics of the generalized selection is defined as follows:

$$\sigma_{\varphi}(R) = \{\ t : t \in R,\ \varphi(t)\ \}$$

The result of the selection is only defined if the attribute names that it mentions are in the header of the relation that it operates upon.

The simulation of a generalized selection that is not a fundamental selection with the fundamental operators is defined by the following rules:

$$\sigma_{\varphi \wedge \psi}(R) = \sigma_{\varphi}(R) \cap \sigma_{\psi}(R)$$
$$\sigma_{\varphi \vee \psi}(R) = \sigma_{\varphi}(R) \cup \sigma_{\psi}(R)$$
$$\sigma_{\neg \varphi}(R) = R - \sigma_{\varphi}(R)$$

The generalized selection is expressible with other basic algebraic operations.

In SQL, general selections are performed by using `WHERE` definitions with `AND`, `OR`, or `NOT` operands in `SELECT`, `UPDATE`, and `DELETE` statements.

Range query

A **range query** is a common database operation that retrieves all records where some value is between an upper and lower boundary. For example, list all employees with 3 to 5 years experience. Range queries are unusual because it is not generally known in advance how many entries a range query will return, or if it will return any at all. Many other queries, such as the top ten most senior employees, or the newest employee, can be done more efficiently because there is an upper bound to the number of results they will return. A query that returns exactly one result is sometimes called a singleton.

Partial match query

Match at least one of the requested keys

Data structures for range query

Range tree

K-D tree

Monotonic query

In database theory and systems, a **monotonic query** is one whose result size does not decrease with the addition of new tuples in the database. Formally, a query q over a schema R is monotonic if and only if for every two instances I, J of R, $I \subseteq J \Rightarrow q(I) \subseteq q(J)$ (q must be a monotonic function).

An example of a monotonic query is a select-project-join query containing only conditions of equality (also known as conjunctive queries). Examples of non-monotonic queries are aggregation queries, or queries with set difference.

Identifying whether a query is monotonic can be crucial for query optimization, especially in view maintenance and data stream management. Since the answer set for a monotonic query can only grow as more tuples are added to the database, query processing may be optimized by executing only the new portions of the database and adding the new results to the existing answer set.

Applications

Data streams

A data stream is a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items. The number of items is considered to be infinite and therefore cannot feasibly be stored in its entirety. Queries over data streams are often called *continuous* or *long-running* queries, and are mostly run over a limited window of tuples in the stream. To evaluate a continuous query, one can simply reevaluate the query over newly arrived tuples, and append the new tuples to the existing result set. More formally, let $A(Q, t)$ be the answer set of a continuous query Q at time t, τ be the current time, and 0 the start time. Then, if Q is monotonic, its result set at time τ is

$$A(Q,\tau) = \bigcup_{t=1}^{\tau} (A(Q,t) - A(Q,t-1)) \cup A(Q,0)$$

In contrast, non-montonic queries have the following answer semantics:

$$A(Q,\tau) = \bigcup_{t=0}^{\tau} A(Q,t)$$

View maintenance

Recursive join

The **recursive join** is an operation used in relational databases, also sometimes called a "fixed-point join". It is a compound operation that involves repeating the join operation, typically accumulating more records each time, until a repetition makes no change to the results (as compared to the results of the previous iteration).

For example, if a database of family relationships is to be searched, and the record for each person has "mother" and "father" fields, a recursive join would be one way to retrieve all of a person's known ancestors: first the person's direct parents' records would be retrieved, then the parents' information would be used to retrieve the grandparents' records, and so on until no new records are being found.

In this example, as in many real cases, the repetition involves only a single database table, and so is more specifically a "recursive self-join".

Recursive joins can be very time-consuming unless optimized through indexing, the addition of extra key fields, or other techniques.

Recursive joins are highly characteristic of hierarchical data, and therefore become a serious issue with XML data. In XML, operations such as determining whether one element contains another are extremely common, and the recursive join is perhaps the most obvious way to implement them when the XML data is stored in a relational database.

The standard way to define recursive joins in the SQL:1999 standard is by way of recursive common table expressions. Database management systems that support recursive CTEs include Microsoft SQL Server, Oracle, PostgreSQL and others.

Relvar

In relational databases, **relvar** is a term introduced by C. J. Date and Hugh Darwen as an abbreviation for relation variable in their 1995 paper *The Third Manifesto*, to avoid the confusion sometimes arising from the use of the term **relation**, by the inventor of the relational model, E. F. Codd, for a variable to which a relation is assigned as well as for the relation itself. The term is used in Date's well-known database textbook *An Introduction to Database Systems* and in various other books authored or coauthored by him.

Relvar is not universally accepted as a term, and it is not used in the context of existing database management system products that support SQL[*citation needed*], whose counterpart concept (but not exact equivalent) is the **base table**, this being something that, like computer language variables in general, has a name and is subject to update (i.e., being assigned different values from time to time). Other database textbooks continue to use the term **relation** for both the variable and the data it contains. Similarly, texts on SQL tend to use the term *table* for both purposes, though the qualified term *base table* is used in the standard for the variable.

A closely related term often used in academic texts is relation schema, this being a set of attributes paired with a set of constraints, together defining a set of relations for the purpose of some discussion (typically, database normalization). Constraints that mention just one relvar are termed *relvar constraints*, so relation schema can be regarded as a single term encompassing a relvar and its relvar constraints.

References

- C.J. Date. *An Introduction to Database Systems*, 8th Ed. (Addison-Wesley, 2004, ISBN 0-321-19784-4), pp. 65–6.
- C.J. Date and Hugh Darwen. *Databases, Types, and The Relational Model: The Third Manifesto* (Addison-Wesley, 2007, ISBN 0-321-39942-0), p.85

Relational calculus

Relational calculus consists of two calculi, the tuple relational calculus and the domain relational calculus, that are part of the relational model for databases and provide a declarative way to specify database queries. This in contrast to the relational algebra which is also part of the relational model but provides a more procedural way for specifying queries.

The relational algebra might suggest these steps to retrieve the phone numbers and names of book stores that supply *Some Sample Book*:

1. Join book stores and titles over the BookstoreID.
2. Restrict the result of that join to tuples for the book *Some Sample Book*.
3. Project the result of that restriction over StoreName and StorePhone.

The relational calculus would formulate a descriptive, declarative way:

> Get StoreName and StorePhone for supplies such that there exists a title BK with the same BookstoreID value and with a BookTitle value of *Some Sample Book*.

The relational algebra and the relational calculus are essentially logically equivalent: for any algebraic expression, there is an equivalent expression in the calculus, and vice versa. This result is known as Codd's theorem.

References

- Date, Christopher J. (2004). *An Introduction to Database Systems* (8th ed.). Addison Wesley. ISBN 0-321-19784-4.

Tuple relational calculus

Tuple calculus is a calculus that was introduced by Edgar F. Codd as part of the relational model, in order to provide a declarative database-query language for this data model. It formed the inspiration for the database-query languages QUEL and SQL, of which the latter, although far less faithful to the original relational model and calculus, is now the de-facto-standard database-query language; a dialect of SQL is used by nearly every relational-database-management system. Lacroix and Pirotte proposed domain calculus, which is closer to first-order logic and which showed that both of these calculi (as well as relational algebra) are equivalent in expressive power. Subsequently, query languages for the relational model were called *relationally complete* if they could express at least all of these queries.

Definition of the calculus

Relational database

Since the calculus is a query language for relational databases we first have to define a relational database. The basic relational building block is the domain, or data type. A tuple is an ordered multiset of attributes, which are ordered pairs of domain and value; or just a row. A relvar (relation variable) is a set of ordered pairs of domain and name, which serves as the header for a relation. A relation is a set of tuples. Although these relational concepts are mathematically defined, those definitions map loosely to traditional database concepts. A table is an accepted visual representation of a relation; a tuple is similar to the concept of *row*.

We first assume the existence of a set C of column names, examples of which are "name", "author", "address" et cetera. We define *headers* as finite subsets of C. A *relational database schema* is defined as a tuple $S = (D, R, h)$ where D is the domain of atomic values (see relational model for more on the notions of *domain* and *atomic value*),

R is a finite set of relation names, and

$$h : R \to 2^C$$

a function that associates a header with each relation name in R. (Note that this is a simplification from the full relational model where there is more than one domain and a header is not just a set of column names but also maps these column names to a domain.) Given a domain D we define a *tuple* over D as a partial function

$$t : C \to D$$

that maps some column names to an atomic value in D. An example would be (name : "Harry", age : 25).

The set of all tuples over D is denoted as T_D. The subset of C for which a tuple t is defined is called the *domain* of t (not to be confused with the domain in the schema) and denoted as $dom(t)$.

Finally we define a *relational database* given a schema $S = (D, R, h)$ as a function

$$db : R \to 2^T D$$

that maps the relation names in R to finite subsets of T_D, such that for every relation name r in R and tuple t in $db(r)$ it holds that

$$dom(t) = h(r).$$

The latter requirement simply says that all the tuples in a relation should contain the same column names, namely those defined for it in the schema.

Atoms

For the construction of the formulae we will assume an infinite set V of tuple variables. The formulas are defined given a database schema $S = (D, R, h)$ and a partial function $type : V \to 2^C$ that defines a *type assignment* that assigns headers to some tuple variables. We then define the *set of atomic formulas* $A[S,type]$ with the following rules:

1. if v and w in V, a in $type(v)$ and b in $type(w)$ then the formula " $v.a = w.b$ " is in $A[S,type]$,
2. if v in V, a in $type(v)$ and k denotes a value in D then the formula " $v.a = k$ " is in $A[S,type]$, and
3. if v in V, r in R and $type(v) = h(r)$ then the formula " $r(v)$ " is in $A[S,type]$.

Examples of atoms are:

- (t.age = s.age) — t has an age attribute and s has an age attribute with the same value
- (t.name = "Codd") — tuple t has a name attribute and its value is "Codd"
- Book(t) — tuple t is present in relation Book.

The formal semantics of such atoms is defined given a database db over S and a tuple variable binding $val : V \to T_D$ that maps tuple variables to tuples over the domain in S:

1. " $v.a = w.b$ " is true if and only if $val(v)(a) = val(w)(b)$
2. " $v.a = k$ " is true if and only if $val(v)(a) = k$
3. " $r(v)$ " is true if and only if $val(v)$ is in $db(r)$

Formulas

The atoms can be combined into formulas, as is usual in first-order logic, with the logical operators $\wedge$ (and), $\vee$ (or) and $\neg$ (not), and we can use the existential quantifier ($\exists$) and the universal quantifier ($\forall$) to bind the variables. We define the *set of formulas* $F[S,type]$ inductively with the following rules:

1. every atom in $A[S,type]$ is also in $F[S,type]$
2. if f_1 and f_2 are in $F[S,type]$ then the formula " $f_1 \wedge f_2$ " is also in $F[S,type]$
3. if f_1 and f_2 are in $F[S,type]$ then the formula " $f_1 \vee f_2$ " is also in $F[S,type]$
4. if f is in $F[S,type]$ then the formula " $\neg f$ " is also in $F[S,type]$
5. if v in V, H a header and f a formula in $F[S,type_{[v->H]}]$ then the formula " $\exists\, v : H\,(f)$ " is also in $F[S,type]$, where $type_{[v->H]}$ denotes the function that is equal to $type$ except that it maps v to H,

6. if v in V, H a header and f a formula in $F[S,type_{[v->H]}]$ then the formula " $\forall\ v : H\ (f)$ " is also in $F[S,type]$

Examples of formulas:

- t.name = "C. J. Date" $\vee$ t.name = "H. Darwen"
- Book(t) $\vee$ Magazine(t)
- $\forall\ t$: {author, title, subject} ($\neg$ (Book(t) $\wedge$ t.author = "C. J. Date" $\wedge$ $\neg$ (t.subject = "relational model")))

Note that the last formula states that all books that are written by C. J. Date have as their subject the relational model. As usual we omit brackets if this causes no ambiguity about the semantics of the formula.

We will assume that the quantifiers quantify over the universe of all tuples over the domain in the schema. This leads to the following formal semantics for formulas given a database *db* over S and a tuple variable binding $val : V -> T_D$:

1. " $f_1 \wedge f_2$ " is true if and only if " f_1 " is true and " f_2 " is true,
2. " $f_1 \vee f_2$ " is true if and only if " f_1 " is true or " f_2 " is true or both are true,
3. " $\neg f$ " is true if and only if " f " is not true,
4. " $\exists\ v : H\ (f)$ " is true if and only if there is a tuple t over D such that $dom(t) = H$ and the formula " f " is true for $val_{[v->t]}$, and
5. " $\forall\ v : H\ (f)$ " is true if and only if for all tuples t over D such that $dom(t) = H$ the formula " f " is true for $val_{[v->t]}$.

Queries

Finally we define what a query expression looks like given a schema $S = (D, R, h)$:

$$\{\ v : H \mid f(v)\ \}$$

where v is a tuple variable, H a header and $f(v)$ a formula in $F[S,type]$ where $type = \{\ (v, H)\ \}$ and with v as its only free variable. The result of such a query for a given database *db* over S is the set of all tuples t over D with $dom(t) = H$ such that f is true for *db* and $val = \{\ (v, t)\ \}$.

Examples of query expressions are:

- { t : {name} | $\exists\ s$: {name, wage} (Employee(s) $\wedge$ s.wage = 50.000 $\wedge$ t.name = s.name) }
- { t : {supplier, article} | $\exists\ s$: {s#, sname} (Supplier(s) $\wedge$ s.sname = t.supplier $\wedge$ $\exists\ p$: {p#, pname} (Product(p) $\wedge$ p.pname = t.article $\wedge$ $\exists\ a$: {s#, p#} (Supplies(a) $\wedge$ s.s# = a.s# $\wedge$ a.p# = p.p#) }

Semantic and syntactic restriction of the calculus

Domain-independent queries

Because the semantics of the quantifiers is such that they quantify over all the tuples over the domain in the schema it can be that a query may return a different result for a certain database if another schema is presumed. For example, consider the two schemas $S_1 = (\ D_1, R, h\ )$ and $S_2 = (\ D_2, R, h\ )$ with domains $D_1 = \{\ 1\ \}$, $D_2 = \{\ 1, 2\ \}$, relation names $R = \{\ "r_1"\ \}$ and headers $h = \{\ ("r_1", \{"a"\})\ \}$. Both schemas have a common instance:

$$db = \{\ (\ "r_1", \{\ ("a", 1)\ \}\ )\ \}$$

If we consider the following query expression

$$\{\ t : \{a\} \mid t.a = t.a\ \}$$

then its result on *db* is either { (a : 1) } under S_1 or { (a : 1), (a : 2) } under S_2. It will also be clear that if we take the domain to be an infinite set, then the result of the query will also be infinite. To solve these problems we will restrict our attention to those queries that are *domain independent*, i.e., the queries that return the same result for a database under all of its schemas.

An interesting property of these queries is that if we assume that the tuple variables range over tuples over the so-called *active domain* of the database, which is the subset of the domain that occurs in at least one tuple in the

database or in the query expression, then the semantics of the query expressions does not change. In fact, in many definitions of the tuple calculus this is how the semantics of the quantifiers is defined, which makes all queries by definition domain independent.

Safe queries

In order to limit the query expressions such that they express only domain-independent queries a syntactical notion of *safe query* is usually introduced. To determine whether a query expression is safe we will derive two types of information from a query. The first is whether a variable-column pair *t.a* is *bound* to the column of a relation or a constant, and the second is whether two variable-column pairs are directly or indirectly equated (denoted $t.v == s.w$).

For deriving boundedness we introduce the following reasoning rules:

1. in " $v.a = w.b$ " no variable-column pair is bound,
2. in " $v.a = k$ " the variable-column pair $v.a$ is bound,
3. in " $r(v)$ " all pairs $v.a$ are bound for a in $type(v)$,
4. in " $f_1 \wedge f_2$ " all pairs are bound that are bound either in f_1 or in f_2,
5. in " $f_1 \vee f_2$ " all pairs are bound that are bound both in f_1 and in f_2,
6. in " $\neg f$ " no pairs are bound,
7. in " $\exists\ v : H\ (f)$ " a pair $w.a$ is bound if it is bound in f and $w <> v$, and
8. in " $\forall\ v : H\ (f)$ " a pair $w.a$ is bound if it is bound in f and $w <> v$.

For deriving equatedness we introduce the following reasoning rules (next to the usual reasoning rules for equivalence relations: reflexivity, symmetry and transitivity):

1. in " $v.a = w.b$ " it holds that $v.a == w.b$,
2. in " $v.a = k$ " no pairs are equated,
3. in " $r(v)$ " no pairs are equated,
4. in " $f_1 \wedge f_2$ " it holds that $v.a == w.b$ if it holds either in f_1 or in f_2,
5. in " $f_1 \vee f_2$ " it holds that $v.a == w.b$ if it holds both in f_1 and in f_2,
6. in " $\neg f$ " no pairs are equated,
7. in " $\exists\ v : H\ (f)$ " it holds that $w.a == x.b$ if it holds in f and $w<>v$ and $x<>v$, and
8. in " $\forall\ v : H\ (f)$ " it holds that $w.a == x.b$ if it holds in f and $w<>v$ and $x<>v$.

We then say that a query expression { v : H | f(v) } is *safe* if

- for every column name a in H we can derive that $v.a$ is equated with a bound pair in f,
- for every subexpression of f of the form " $\forall\ w : G\ (g)$ " we can derive that for every column name a in G we can derive that $w.a$ is equated with a bound pair in g, and
- for every subexpression of f of the form " $\exists\ w : G\ (g)$ " we can derive that for every column name a in G we can derive that $w.a$ is equated with a bound pair in g.

The restriction to safe query expressions does not limit the expressiveness since all domain-independent queries that could be expressed can also be expressed by a safe query expression. This can be proven by showing that for a schema $S = (D, R, h)$, a given set K of constants in the query expression, a tuple variable v and a header H we can construct a safe formula for every pair $v.a$ with a in H that states that its value is in the active domain. For example, assume that K={1,2}, R={"r"} and h = { ("r", {"a, "b"}) } then the corresponding safe formula for v.b is:

$$v.\mathrm{b} = 1 \vee v.\mathrm{b} = 2 \vee \exists\ w\ (\ \mathrm{r(w)} \wedge (\ v.\mathrm{b} = w.\mathrm{a} \vee v.\mathrm{b} = w.\mathrm{b}\ )\ )$$

This formula, then, can be used to rewrite any unsafe query expression to an equivalent safe query expression by adding such a formula for every variable v and column name a in its type where it is used in the expression. Effectively this means that we let all variables range over the active domain, which, as was already explained, does not change the semantics if the expressed query is domain independent.

References

- Edgar F. Codd: A Relational Model of Data for Large Shared Data Banks [1]. *Communications of the ACM*, 13(6):377–387, 1970.

References

[1] http://doi.acm.org/10.1145/362384.362685

Overview

Query language

Query languages are computer languages used to make queries into databases and information systems.

Broadly, query languages can be classified according to whether they are database query languages or information retrieval query languages. The difference is that a database query language attempts to give factual answers to factual questions, while an information retrieval query language attempts to find documents containing information that is relevant to an area of inquiry.

Examples include:

- .QL is a proprietary object-oriented query language for querying relational databases; successor of Datalog;
- Contextual Query Language (CQL) a formal language for representing queries to information retrieval systems such as web indexes or bibliographic catalogues.
- CQLF (CODASYL Query Language, Flat) is a query language for CODASYL-type databases;
- Concept-Oriented Query Language (COQL) is used in the concept-oriented model (COM). It is based on a novel data modeling construct, concept, and uses such operations as projection and de-projection for multi-dimensional analysis, analytical operations and inference;
- DMX is a query language for Data Mining models;
- Datalog is a query language for deductive databases;
- F-logic is a declarative object-oriented language for deductive databases and knowledge representation.
- FQL enables you to use a SQL-style interface to query the data exposed by the Graph API. It provides advanced features not available in the Graph API.[1]
- Gellish English is a language that can be used for queries in Gellish English Databases, for dialogues (requests and responses) as well as for information modeling and knowledge modeling;[2]
- HTSQL is a query language that translates HTTP queries to SQL;
- ISBL is a query language for PRTV, one of the earliest relational database management systems;
- LINQ query-expressions is a way to query various data sources from .NET languages
- LDAP is an application protocol for querying and modifying directory services running over TCP/IP;
- MQL is a cheminformatics query language for a substructure search allowing beside nominal properties also numerical properties;
- MDX is a query language for OLAP databases;
- OQL is Object Query Language;
- OCL (Object Constraint Language). Despite its name, OCL is also an object query language and an OMG standard;
- OPath, intended for use in querying WinFS *Stores*;
- OttoQL, intended for querying tables, XML, and databases;
- Poliqarp Query Language is a special query language designed to analyze annotated text. Used in the Poliqarp search engine;
- QUEL is a relational database access language, similar in most ways to SQL;
- RDQL is a RDF query language;
- SMARTS is the cheminformatics standard for a substructure search;
- SPARQL is a query language for RDF graphs;
- SPL is a search language for machine-generated big data, based upon Unix Piping and SQL.

- SQL is a well known query language and Data Manipulation Language for relational databases;
- SuprTool is a proprietary query language for SuprTool, a database access program used for accessing data in *Image/SQL* (formerly TurboIMAGE) and Oracle databases;
- TMQL Topic Map Query Language is a query language for Topic Maps;
- Tutorial D is a query language for truly relational database management systems (TRDBMS);
- XQuery is a query language for XML data sources;
- XPath is a declarative language for navigating XML documents;
- XSPARQL is an integrated query language combining XQuery with SPARQL to query both XML and RDF data sources at once;
- YQL is an SQL-like query language created by Yahoo!

References

[1] https://developers.facebook.com/docs/technical-guides/fql/
[2] http://gellish.wiki.sourceforge.net/Querying+a+Gellish+English+database

Data Definition Language

A **data definition language** or **data description language** (**DDL**) is a syntax similar to a computer programming language for defining data structures, especially database schemas.

History

The data definition language concept and name was first introduced in relation to the Codasyl database model, where the schema of the database was written in a language syntax describing the records, fields, and sets of the user data model. Later it was used to refer to a subset of Structured Query Language (SQL) for creating tables and constraints. SQL-92 introduced a schema manipulation language and schema information tables to query schemas. These information tables were specified as SQL/Schemata in SQL:2003. The term DDL is also used in a generic sense to refer to any formal language for describing data or information structures.

SQL

Many data description languages use a declarative syntax to define fields and data types. SQL, however, uses a collection of imperative verbs whose effect is to modify the schema of the database by adding, changing, or deleting definitions of tables or other objects. These statements can be freely mixed with other SQL statements, so the DDL is not truly a separate language.

CREATE statements

Create - To make a new database, table, index, or stored procedure.

A `CREATE` statement in SQL creates an object in a relational database management system (RDBMS). In the SQL 1992 specification, the types of objects that can be created are schemas, tables, views, domains, character sets, collations, translations, and assertions. Many implementations extend the syntax to allow creation of additional objects, such as indexes and user profiles. Some systems (such as PostgreSQL) allow `CREATE`, and other DDL commands, inside a transaction and thus they may be rolled back.

CREATE TABLE statement

A commonly used `CREATE` command is the `CREATE TABLE` command. The typical usage is:

`CREATE TABLE` *`[table name]`* `(` *`[column definitions]`* `)` *`[table parameters]`*.

column definitions: A comma-separated list consisting of any of the following

- Column definition: *[column name] [data type] {`NULL` | `NOT NULL`} {column options}*
- Primary key definition: *`PRIMARY KEY`* (*[comma separated column list]*)
- Constraints: *{`CONSTRAINT`} [constraint definition]*
- RDBMS specific functionality

For example, the command to create a table named **employees** with a few sample columns would be:

```
CREATE TABLE employees (
    id              INTEGER       PRIMARY KEY,
    first_name      VARCHAR(50)   null,
    last_name       VARCHAR(75)   not null,
    fname           VARCHAR(50)   not null,
    dateofbirth     DATE          null
);
```

Note that some forms of `CREATE TABLE` DDL may incorporate DML (data manipulation language)-like constructs as well, such as the `CREATE TABLE AS SELECT` (CTAS) syntax of SQL.

DROP statements

Drop - To destroy an existing database, table, index, or view.

A **`DROP`** statement in SQL removes an object from a relational database management system (RDBMS). The types of objects that can be dropped depends on which RDBMS is being used, but most support the dropping of tables, users, and databases. Some systems (such as PostgreSQL) allow `DROP` and other DDL commands to occur inside of a transaction and thus be rolled back. The typical usage is simply:

`DROP` *`objecttype objectname`*.

For example, the command to drop a table named **employees** would be:

```
DROP employees;
```

The `DROP` statement is distinct from the `DELETE` and `TRUNCATE` statements, in that `DELETE` and `TRUNCATE` do not remove the table itself. For example, a `DELETE` statement might delete some (or all) data from a table while leaving the table itself in the database, whereas a `DROP` statement would remove the entire table from the database.

ALTER statements

Alter - To modify an existing database object.

An `ALTER` statement in SQL changes the properties of an object inside of a relational database management system (RDBMS). The types of objects that can be altered depends on which RDBMS is being used. The typical usage is:

`ALTER` *`objecttype objectname parameters`*.

For example, the command to add (then remove) a column named **bubbles** for an existing table named **sink** would be:

```
ALTER TABLE sink ADD bubbles INTEGER;
ALTER TABLE sink DROP COLUMN bubbles;
```

Referential integrity statements

Finally, another kind of DDL sentence in SQL is one used to define referential integrity relationships, usually implemented as primary key and foreign key tags in some columns of the tables.

These two statements can be included inside a `CREATE TABLE` or an `ALTER TABLE` sentence.

Other languages

- XML Schema is an example of a DDL for XML.
- Simple Declarative Language

References

External links

- How to change data type of a column MS SQL - How to change data type of a column (http://aspxtutorial.com/post/2010/07/01/MS-SQL-How-to-change-the-data-type-of-a-column-using-query.aspx)

Varchar

A **varchar** or **Variable Character Field** is a set of character data of indeterminate length. The term **varchar** refers to a data type of a field (or column) in a database management system which can hold letters and numbers. Varchar fields can be of any size up to a limit, which varies by databases: an Oracle 9i database has a limit of 4000 bytes, a MySQL database has a limit of 65,535 bytes (for the entire row) and Microsoft SQL Server 2005 has a limit of 8000 bytes (unless varchar(max) is used, which has a maximum storage capacity of 2 gigabytes).

References

- CHAR and VARCHAR documentation [1] for Microsoft SQL Server 2008
- VARCHAR documentation [2] for Apache Derby
- CHAR and VARCHAR documentation [3] for MySQL 5.1 [4]
- CHAR and VARCHAR documentation [5] for IBM Informix

References

[1] http://msdn.microsoft.com/en-us/library/ms176089.aspx
[2] http://db.apache.org/derby/docs/10.9/ref/rrefsqlj41207.html
[3] http://dev.mysql.com/doc/refman/5.1/en/char.html
[4] http://dev.mysql.com/
[5] http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.sqls.doc/sqls984.htm

Data Manipulation Language

A **data manipulation language** (**DML**) is a family of syntax elements similar to a computer programming language used for inserting, deleting and updating data in a database. Performing read-only queries of data is sometimes also considered a component of DML.

A popular data manipulation language is that of Structured Query Language (SQL), which is used to retrieve and manipulate data in a relational database.[1] Other forms of DML are those used by IMS/DLI, CODASYL databases, such as IDMS and others.

Data manipulation language comprises the SQL data change statements, which modify stored data but not the schema or database objects. Manipulation of persistent database objects, e.g., tables or stored procedures, via the SQL schema statements,[] rather than the data stored within them, is considered to be part of a separate data definition language. In SQL these two categories are similar in their detailed syntax, data types, expressions etc., but distinct in their overall function.

Data manipulation languages have their functional capability organized by the initial word in a statement, which is almost always a verb. In the case of SQL, these verbs are:

- `SELECT ... FROM ... WHERE ...`
- `INSERT INTO ... VALUES ...`
- `UPDATE ... SET ... WHERE ...`
- `DELETE FROM ... WHERE ...`

The purely read-only `SELECT` query statement is classed with the 'SQL-data' statements and so is considered by the standard to be outside of DML. The `SELECT ... INTO` form is considered to be DML because it manipulates (i.e. modifies) data. In common practice though, this distinction is not made and `SELECT` is widely considered to be part of DML.

Most SQL database implementations extend their SQL capabilities by providing imperative, i.e. procedural languages. Examples of these are Oracle's PL/SQL and DB2's SQL PL.

Data manipulation languages tend to have many different flavors and capabilities between database vendors. There have been a number of standards established for SQL by ANSI, but vendors still provide their own extensions to the standard while not implementing the entire standard.

Data manipulation languages are divided into two types, procedural programming and declarative programming.

Data manipulation languages were initially only used within computer programs, but with the advent of SQL have come to be used interactively by database administrators.

References

- "The SQL92 standard" [2].

[1] SQL92

[2] http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

Create, read, update and delete

In computer programming, **create, read, update and delete** (**CRUD**) (Sometimes called SCRUD with an "S" for Search) are the four basic functions of persistent storage. Sometimes *CRUD* is expanded with the words *retrieve* instead of *read*, *modify* instead of *update*, or *destroy* instead of *delete*. It is also sometimes used to describe user interface conventions that facilitate viewing, searching, and changing information; often using computer-based forms and reports. The term was likely first popularized by James Martin in his 1983 book *Managing the Data-base Environment*. The acronym may be extended to CRUDL to cover *listing* of large data sets which bring additional complexity such as pagination when the data sets are too large to hold easily in memory.

Another variation of CRUD is BREAD, an acronym for "Browse, Read, Edit, Add, Delete".

Database applications

The acronym CRUD refers to all of the major functions that are implemented in relational database applications. Each letter in the acronym can map to a standard SQL statement and HTTP method:

Operation	SQL	HTTP
Create	INSERT	PUT / POST
Read (Retrieve)	SELECT	GET
Update (Modify)	UPDATE	PUT / PATCH
Delete (Destroy)	DELETE	DELETE

Although a relational database provides a common persistence layer in software applications, numerous other persistence layers exist. CRUD functionality can be implemented with an object database, an XML database, flat text files, custom file formats, tape, or card, for example.

User interface

CRUD is also relevant at the user interface level of most applications. For example, in address book software, the basic storage unit is an individual *contact entry*. As a bare minimum, the software must allow the user to:

- Create or add new entries
- Read, retrieve, search, or view existing entries
- Update or edit existing entries
- Delete/deactivate existing entries

Without at least these four operations, the software cannot be considered complete. Because these operations are so fundamental, they are often documented and described under one comprehensive heading, such as "contact management", "content management" or "contact maintenance" (or "document management" in general, depending on the basic storage unit for the particular application).

Notes

SQL

SQL

Paradigm(s)	Multi-paradigm
Appeared in	1974
Designed by	Donald D. Chamberlin Raymond F. Boyce
Developer	ISO/IEC
Stable release	SQL:2011 (2011)
Typing discipline	Static, strong
Major implementations	Many
Dialects	SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2008, SQL:2011
Influenced by	Datalog
Influenced	CQL, LINQ, SOQL, Windows PowerShell, JPQL, jOOQ
OS	Cross-platform
File format details	
Filename extension	`.sql`
Internet media type	`application/sql`
Developed by	ISO/IEC
Initial release	1986
Latest release	SQL:2898 / 2011
Type of format	Database
Standard(s)	ISO/IEC 9075
Open format?	Yes

SQL (/ˈɛsHelp:IPA for English#KeykjuːHelp:IPA for English#Keyˈɛl/, or /ˈsiːkwəl/; **Structured Query Language**[1][2]) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language and a data manipulation language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks." Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then, the standard has been enhanced several times with added features. Despite these standards, code is not completely portable among different database systems, which can lead to vendor lock-in. The different makers do not perfectly adhere to the standard, for instance by adding extensions, and the standard itself is sometimes ambiguous.

History

SQL was initially developed at IBM by Donald D. Chamberlin, Donald C. Messerly, and Raymond F. Boyce in the early 1970s.[3] This version, initially called *SEQUEL* (*Structured English Query Language*), was designed to manipulate and retrieve data stored in IBM's original quasi-relational database management system, System R, which a group at IBM San Jose Research Laboratory had developed during the 1970s. The acronym SEQUEL was later changed to SQL because "SEQUEL" was a trademark of the UK-based Hawker Siddeley aircraft company.

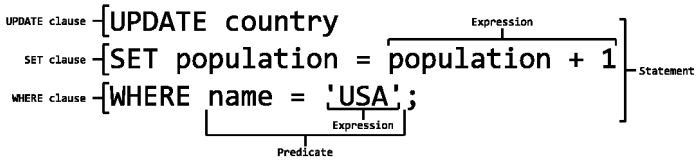
In the late 1970s, Relational Software, Inc. (now Oracle Corporation) saw the potential of the concepts described by Codd, Chamberlin, and Boyce and developed their own SQL-based RDBMS with aspirations of selling it to the U.S. Navy, Central Intelligence Agency, and other U.S. government agencies. In June 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers.

After testing SQL at customer test sites to determine the usefulness and practicality of the system, IBM began developing commercial products based on their System R prototype including System/38, SQL/DS, and DB2, which were commercially available in 1979, 1981, and 1983, respectively.

Syntax

Language elements

The SQL language is subdivided into several language elements, including:



A chart showing several of the SQL language elements that compose a single statement

- *Clauses*, which are constituent components of statements and queries. (In some cases, these are optional.)[4]
- *Expressions*, which can produce either scalar values, or tables consisting of columns and rows of data.
- *Predicates*, which specify conditions that can be evaluated to SQL three-valued logic (3VL) (true/false/unknown) or Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- *Queries*, which retrieve the data based on specific criteria. This is an important element of *SQL*.
- *Statements*, which may have a persistent effect on schemata and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
  - SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
- *Insignificant whitespace* is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

Operators

Operator	Description	Example
=	Equal to	`Author = 'Alcott'`
<>	Not equal to (most DBMS also accept != instead of <>)	`Dept <> 'Sales'`
>	Greater than	`Hire_Date > '2012-01-31'`
<	Less than	`Bonus < 50000.00`
>=	Greater than or equal	`Dependents >= 2`
<=	Less than or equal	`Rate <= 0.05`
BETWEEN	Between an inclusive range	`Cost BETWEEN 100.00 AND 500.00`
LIKE	Match a character pattern	`First_Name LIKE 'Will%'`
IN	Equal to one of multiple possible values	`DeptCode IN (101, 103, 209)`
IS *or* IS NOT	Compare to null (missing data)	`Address IS NOT NULL`
IS NOT DISTINCT FROM	Is equal to value or both are nulls (missing data)	`Debt IS NOT DISTINCT FROM Receivables`
AS	Used to change a field name when viewing results	`SELECT employee AS 'department1'`

Conditional (CASE) expressions

SQL has the `case/when/then/else/end` expression, which was introduced in SQL-92. In its most general form, which is called a "searched case" in the SQL standard, it works like else if in other programming languages:

```
CASE WHEN n > 0
          THEN 'positive'
     WHEN n < 0
          THEN 'negative'
     ELSE 'zero'
END
```

The `WHEN` conditions are tested in the order in which they appear in the source. If no `ELSE` expression is specified, it defaults to `ELSE NULL`. An abbreviated syntax exists mirroring switch statements; it is called "simple case" in the SQL standard:

```
CASE n WHEN 1
            THEN 'one'
       WHEN 2
          THEN 'two'
       ELSE 'i cannot count that high'
END
```

This syntax uses implicit equality comparisons, with the usual caveats for comparing with NULL.

For the Oracle-SQL dialect, the latter can be shortened to an equivalent `DECODE` construct:

```
SELECT DECODE(n, 1, 'one',
               2, 'two',
                 'i cannot count that high')
FROM   some_table;
```

The last value is the default; if none is specified, it also defaults to NULL. However, unlike the standard's "simple case", Oracle's DECODE considers two NULLs to be equal with each other.

Queries

The most common operation in SQL is the query, which is performed with the declarative SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO syntax that exists in some databases.

Queries allow the user to describe desired data, leaving the database management system (DBMS) responsible for planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk ("*") can also be used to specify that the query should return all columns of the queried tables. SELECT is the most complex statement in SQL, with optional keywords and clauses that include:

- The FROM clause which indicates the table(s) from which data is to be retrieved. The FROM clause can include optional JOIN subclauses to specify the rules for joining tables.
- The WHERE clause includes a comparison predicate, which restricts the rows returned by the query. The WHERE clause eliminates all rows from the result set for which the comparison predicate does not evaluate to True.
- The GROUP BY clause is used to project rows having common values into a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The WHERE clause is applied before the GROUP BY clause.
- The HAVING clause includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.
- The ORDER BY clause identifies which columns are used to sort the resulting data, and in which direction they should be sorted (options are ascending or descending). Without an ORDER BY clause, the order of rows returned by an SQL query is undefined.

The following is an example of a SELECT query that returns a list of expensive books. The query retrieves all rows from the *Book* table in which the *price* column contains a value greater than 100.00. The result is sorted in ascending order by *title*. The asterisk (*) in the *select list* indicates that all columns of the *Book* table should be included in the result set.

```
SELECT *
 FROM  Book
 WHERE price > 100.00
 ORDER BY title;
```

The example below demonstrates a query of multiple tables, grouping, and aggregation, by returning a list of books and the number of authors associated with each book.

```
SELECT Book.title AS Title,
       count(*) AS Authors
 FROM  Book
 JOIN  Book_author
   ON  Book.isbn = Book_author.isbn
 GROUP BY Book.title;
```

Example output might resemble the following:

```
Title                  Authors
---------------------- -------
```

```
SQL Examples and Guide 4
The Joy of SQL         1
An Introduction to SQL 2
Pitfalls of SQL        1
```

Under the precondition that *isbn* is the only common column name of the two tables and that a column named *title* only exists in the *Books* table, the above query could be rewritten in the following form:

```
SELECT title,
       count(*) AS Authors
 FROM  Book
 NATURAL JOIN Book_author
 GROUP BY title;
```

However, many vendors either do not support this approach, or require certain column naming conventions in order for natural joins to work effectively.

SQL includes operators and functions for calculating values on stored values. SQL allows the use of expressions in the *select list* to project data, as in the following example which returns a list of books that cost more than 100.00 with an additional *sales_tax* column containing a sales tax figure calculated at 6% of the *price*.

```
SELECT isbn,
       title,
       price,
       price * 0.06 AS sales_tax
 FROM  Book
 WHERE price > 100.00
 ORDER BY title;
```

Subqueries

Queries can be nested so that the results of one query can be used in another query via a relational operator or aggregation function. A nested query is also known as a *subquery*. While joins and other table operations provide computationally superior (i.e. faster) alternatives in many cases, the use of subqueries introduces a hierarchy in execution which can be useful or necessary. In the following example, the aggregation function `AVG` receives as input the result of a subquery:

```
SELECT isbn,
       title,
       price
 FROM  Book
 WHERE price < (SELECT AVG(price) FROM Book)
 ORDER BY title;
```

A subquery can use values from the outer query, in which case it is known as a correlated subquery.

Since 1999 the SQL standard allows named subqueries called common table expression (named and designed after the IBM DB2 version 2 implementation; Oracle calls these subquery factoring). CTEs can be also be recursive by referring to themselves; the resulting mechanism allows tree or graph traversals (when represented as relations), and more generally fixpoint computations.

Null and three-valued logic (3VL)

The concept of Null was introduced into SQL to handle missing information in the relational model. The word `NULL` is a reserved keyword in SQL, used to identify the Null special marker. Comparisons with Null, for instance equality (=) in WHERE clauses, results in an Unknown truth value. In SELECT statements SQL returns only results for which the WHERE clause returns a value of True; i.e. it excludes results with values of False and also excludes those whose value is Unknown.

Along with True and False, the Unknown resulting from direct comparisons with Null thus brings a fragment of three-valued logic to SQL. The truth tables SQL uses for AND, OR, and NOT correspond to a common fragment of the Kleene and Lukasiewicz three-valued logic (which differ in their definition of implication, however SQL defines no such operation).

p AND q		p		
		True	**False**	**Unknown**
q	**True**	True	False	Unknown
	False	False	False	False
	Unknown	Unknown	False	Unknown

p OR q		p		
		True	**False**	**Unknown**
q	**True**	True	True	True
	False	True	False	Unknown
	Unknown	True	Unknown	Unknown

q	NOT q
True	False
False	True
Unknown	Unknown

p = q		p		
		True	**False**	**Unknown**
q	**True**	True	False	Unknown
	False	False	True	Unknown
	Unknown	Unknown	Unknown	Unknown

There are however disputes about the semantic interpretation of Nulls in SQL because of its treatment outside direct comparisons. As seen in the table above direct equality comparisons between two NULLs in SQL (e.g. `NULL = NULL`) returns a truth value of Unknown. This is in line with the interpretation that Null does not have a value (and is not a member of any data domain) but is rather a placeholder or "mark" for missing information. However, the principle that two Nulls aren't equal to each other is effectively violated in the SQL specification for the `UNION` and `INTERSECT` operators, which do identify nulls with each other. Consequently, these set operations in SQL may produce results not representing sure information, unlike operations involving explicit comparisons with NULL (e.g. those in a `WHERE` clause discussed above). In Codd's 1979 proposal (which was basically adopted by SQL92) this semantic inconsistency is rationalized by arguing that removal of duplicates in set operations happens "at a lower level of detail than equality testing in the evaluation of retrieval operations." However, computer science professor Ron van der Meyden concluded that "The inconsistencies in the SQL standard mean that it is not possible to ascribe any intuitive logical semantics to the treatment of nulls in SQL."[]

Additionally, since SQL operators return Unknown when comparing anything with Null directly, SQL provides two Null-specific comparison predicates: `IS NULL` and `IS NOT NULL` test whether data is or is not Null. Universal quantification is not explicitly supported by SQL, and must be worked out as a negated existential quantification.[5][6][7] There is also the "<row value expression> IS DISTINCT FROM <row value expression>" infixed comparison operator which returns TRUE unless both operands are equal or both are NULL. Likewise, IS NOT DISTINCT FROM is defined as "NOT (<row value expression> IS DISTINCT FROM <row value expression>)". SQL:1999 also introduced `BOOLEAN` type variables, which according to the standard can also hold Unknown values. In practice, a number of systems (e.g. PostgreSQL) implement the BOOLEAN Unknown as a

BOOLEAN NULL.

Data manipulation

The Data Manipulation Language (DML) is the subset of SQL used to add, update and delete data:

- `INSERT` adds rows (formally tuples) to an existing table, e.g.:

```
INSERT INTO example
 (field1, field2, field3)
 VALUES
 ('test', 'N', NULL);
```

- `UPDATE` modifies a set of existing table rows, e.g.:

```
UPDATE example
 SET field1 = 'updated value'
 WHERE field2 = 'N';
```

- `DELETE` removes existing rows from a table, e.g.:

```
DELETE FROM example
 WHERE field2 = 'N';
```

- `MERGE` is used to combine the data of multiple tables. It combines the `INSERT` and `UPDATE` elements. It is defined in the SQL:2003 standard; prior to that, some databases provided similar functionality via different syntax, sometimes called "upsert".

```
 MERGE INTO table_name USING table_reference ON (condition)
 WHEN MATCHED THEN
 UPDATE SET column1 = value1 [, column2 = value2 ...]
 WHEN NOT MATCHED THEN
 INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...
```

Transaction controls

Transactions, if available, wrap DML operations:

- `START TRANSACTION` (or `BEGIN WORK`, or `BEGIN TRANSACTION`, depending on SQL dialect) marks the start of a database transaction, which either completes entirely or not at all.
- `SAVE TRANSACTION` (or `SAVEPOINT`) saves the state of the database at the current point in transaction

```
CREATE TABLE tbl_1(id int);
 INSERT INTO tbl_1(id) VALUES(1);
 INSERT INTO tbl_1(id) VALUES(2);
COMMIT;
 UPDATE tbl_1 SET id=200 WHERE id=1;
SAVEPOINT id_1upd;
 UPDATE tbl_1 SET id=1000 WHERE id=2;
ROLLBACK to id_1upd;
 SELECT id from tbl_1;
```

- `COMMIT` causes all data changes in a transaction to be made permanent.
- `ROLLBACK` causes all data changes since the last `COMMIT` or `ROLLBACK` to be discarded, leaving the state of the data as it was prior to those changes.

Once the `COMMIT` statement completes, the transaction's changes cannot be rolled back.

`COMMIT` and `ROLLBACK` terminate the current transaction and release data locks. In the absence of a `START TRANSACTION` or similar statement, the semantics of SQL are implementation-dependent. The following example shows a classic transfer of funds transaction, where money is removed from one account and added to another. If either the removal or the addition fails, the entire transaction is rolled back.

```
START TRANSACTION;
 UPDATE Account SET amount=amount-200 WHERE account_number=1234;
 UPDATE Account SET amount=amount+200 WHERE account_number=2345;

IF ERRORS=0 COMMIT;
IF ERRORS<>0 ROLLBACK;
```

Data definition

The Data Definition Language (DDL) manages table and index structure. The most basic items of DDL are the `CREATE`, `ALTER`, `RENAME`, `DROP` and `TRUNCATE` statements:

- `CREATE` creates an object (a table, for example) in the database, e.g.:

```
CREATE TABLE example(
 field1 INTEGER,
 field2 VARCHAR(50),
 field3 DATE NOT NULL,
 PRIMARY KEY (field1, field2)
);
```

- `ALTER` modifies the structure of an existing object in various ways, for example, adding a column to an existing table or a constraint, e.g.:

```
ALTER TABLE example ADD field4 NUMBER(3) NOT NULL;
```

- `TRUNCATE` deletes all data from a table in a very fast way, deleting the data inside the table and not the table itself. It usually implies a subsequent COMMIT operation, i.e., it cannot be rolled back (data is not written to the logs for rollback later, unlike DELETE).

```
TRUNCATE TABLE example;
```

- `DROP` deletes an object in the database, usually irretrievably, i.e., it cannot be rolled back, e.g.:

```
DROP TABLE example;
```

Data types

Each column in an SQL table declares the type(s) that column may contain. ANSI SQL includes the following data types.[8]

Character strings

- `CHARACTER(n)` or `CHAR(n)`: fixed-width `n`-character string, padded with spaces as needed
- `CHARACTER VARYING(n)` or `VARCHAR(n)`: variable-width string with a maximum size of `n` characters
- `NATIONAL CHARACTER(n)` or `NCHAR(n)`: fixed width string supporting an international character set
- `NATIONAL CHARACTER VARYING(n)` or `NVARCHAR(n)`: variable-width `NCHAR` string

Bit strings

- `BIT(n)`: an array of `n` bits
- `BIT VARYING(n)`: an array of up to `n` bits

Numbers

- `INTEGER` and `SMALLINT`
- `FLOAT`, `REAL` and `DOUBLE PRECISION`
- `NUMERIC(precision, scale)` or `DECIMAL(precision, scale)`

For example, the number 123.45 has a precision of 5 and a scale of 2. The `precision` is a positive integer that determines the number of significant digits in a particular radix (binary or decimal). The `scale` is a non-negative integer. A scale of 0 indicates that the number is an integer. For a decimal number with scale S, the exact numeric value is the integer value of the significant digits divided by 10^S.

SQL provides a function to round numerics or dates, called `TRUNC` (in Informix, DB2, PostgreSQL, Oracle and MySQL) or `ROUND` (in Informix, SQLite, Sybase, Oracle, PostgreSQL and Microsoft SQL Server)[9]

Date and time

- `DATE`: for date values (e.g. `2011-05-03`)
- `TIME`: for time values (e.g. `15:51:36`). The granularity of the time value is usually a *tick* (100 nanoseconds).
- `TIME WITH TIME ZONE` or `TIMETZ`: the same as `TIME`, but including details about the time zone in question.
- `TIMESTAMP`: This is a `DATE` and a `TIME` put together in one variable (e.g. `2011-05-03 15:51:36`).
- `TIMESTAMP WITH TIME ZONE` or `TIMESTAMPTZ`: the same as `TIMESTAMP`, but including details about the time zone in question.

SQL provides several functions for generating a date / time variable out of a date / time string (`TO_DATE`, `TO_TIME`, `TO_TIMESTAMP`), as well as for extracting the respective members (seconds, for instance) of such variables. The current system date / time of the database server can be called by using functions like `NOW`.

Data control

The Data Control Language (DCL) authorizes users to access and manipulate data. Its two main statements are:

- `GRANT` authorizes one or more users to perform an operation or a set of operations on an object.
- `REVOKE` eliminates a grant, which may be the default grant.

Example:

```
GRANT SELECT, UPDATE
 ON example
 TO some_user, another_user;

REVOKE SELECT, UPDATE
 ON example
 FROM some_user, another_user;
```

Procedural extensions

SQL is designed for a specific purpose: to query data contained in a relational database. SQL is a set-based, declarative query language, not an imperative language like C or BASIC. However, there are extensions to Standard SQL which add procedural programming language functionality, such as control-of-flow constructs. These include:

Source	Common name	Full name
ANSI/ISO Standard	SQL/PSM	SQL/Persistent Stored Modules
Interbase / Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL	SQL Procedural Language (implements SQL/PSM)
IBM Informix	SPL	Stored Procedural Language
IBM Netezza	NZPLSQL[10]	(based on Postgres PL/pgSQL)
Microsoft / Sybase	T-SQL	Transact-SQL
Mimer SQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MySQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
Oracle	PL/SQL	Procedural Language/SQL (based on Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL (based on Oracle PL/SQL)
PostgreSQL	PL/PSM	Procedural Language/Persistent Stored Modules (implements SQL/PSM)
Sybase	Watcom-SQL	SQL Anywhere Watcom-SQL Dialect
Teradata	SPL	Stored Procedural Language

In addition to the standard SQL/PSM extensions and proprietary SQL extensions, procedural and object-oriented programmability is available on many SQL platforms via DBMS integration with other languages. The SQL standard defines SQL/JRT extensions (SQL Routines and Types for the Java Programming Language) to support Java code in SQL databases. SQL Server 2005 uses the SQLCLR (SQL Server Common Language Runtime) to host managed .NET assemblies in the database, while prior versions of SQL Server were restricted to using unmanaged extended stored procedures that were primarily written in C. PostgreSQL allows functions to be written in a wide variety of languages including Perl, Python, Tcl, and C.

Criticism

SQL deviates in several ways from its theoretical foundation, the relational model and its tuple calculus. In that model, a table is a set of tuples, while in SQL, tables and query results are lists of rows: the same row may occur multiple times, and the order of rows can be employed in queries (e.g. in the LIMIT clause). Furthermore, additional features (such as NULL and views) were introduced without founding them directly on the relational model, which makes them more difficult to interpret.

Critics argue that SQL should be replaced with a language that strictly returns to the original foundation: for example, see *The Third Manifesto*. Other critics suggest that Datalog has two advantages over SQL: it has a cleaner semantics which facilitates program understanding and maintenance, and it is more expressive, in particular for recursive queries.[11]

Another criticism is that SQL implementations are incompatible between vendors. In particular date and time syntax, string concatenation, `NULL`s, and comparison case sensitivity vary from vendor to vendor. A particular exception is PostgreSQL, which strives for compliance.

Popular implementations of SQL commonly omit support for basic features of Standard SQL, such as the `DATE` or `TIME` data types. The most obvious such examples, and incidentally the most popular commercial and proprietary

SQL DBMSs, are Oracle (whose `DATE` behaves as `DATETIME`, and lacks a `TIME` type) and MS SQL Server (before the 2008 version). As a result, SQL code can rarely be ported between database systems without modifications.

There are several reasons for this lack of portability between database systems:

- The complexity and size of the SQL standard means that most implementors do not support the entire standard.
- The standard does not specify database behavior in several important areas (e.g. indexes, file storage...), leaving implementations to decide how to behave.
- The SQL standard precisely specifies the syntax that a conforming database system must implement. However, the standard's specification of the semantics of language constructs is less well-defined, leading to ambiguity.
- Many database vendors have large existing customer bases; where the SQL standard conflicts with the prior behavior of the vendor's database, the vendor may be unwilling to break backward compatibility.
- There is little commercial incentive for vendors to make it easier for users to change database suppliers (see vendor lock-in).
- Users evaluating database software tend to place other factors such as performance higher in their priorities than standards conformance.

Standardization

SQL was adopted as a standard by the American National Standards Institute (ANSI) in 1986 as SQL-86 and the International Organization for Standardization (ISO) in 1987. Nowadays the standard is subject to continuous improvement by the Joint Technical Committee *ISO/IEC JTC 1, Information technology, Subcommittee SC 32, Data management and interchange* which affiliate to ISO as well as IEC. It is commonly denoted by the pattern: *ISO/IEC 9075-n:yyyy Part n: title*, or, as a shortcut, *ISO/IEC 9075*.

ISO/IEC 9075 is complemented by *ISO/IEC 13249: SQL Multimedia and Application Packages* (SQL/MM) which defines SQL based interfaces and packages to widely spread applications like video, audio and spatial data.

Until 1996, the National Institute of Standards and Technology (NIST) data management standards program certified SQL DBMS compliance with the SQL standard. Vendors now self-certify the compliance of their products.

The original SQL standard declared that the official pronunciation for SQL is "es queue el". Many English-speaking database professionals still use the original pronunciation /ˈsiːkwəl/ (like the word "sequel"), including Donald Chamberlin himself.

The SQL standard has gone through a number of revisions:

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision, in which the major addition were integrity constraints. Adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice-versa (SQL/JRT).
2003	SQL:2003	SQL 2003	Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).

2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it enables applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.
2008	SQL:2008	SQL 2008	Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement.
2011	SQL:2011		

Interested parties may purchase SQL standards documents from ISO, IEC or ANSI. A draft of SQL:2008 is freely available as a zip archive.

The SQL standard is divided into nine parts.

- ISO/IEC 9075-1:2011 Part 1: *Framework* (SQL/Framework). It provides logical concepts.
- ISO/IEC 9075-2:2011 Part 2: *Foundation* (SQL/Foundation). It contains the most central elements of the language and consists of both *mandatory and optional* features.
- ISO/IEC 9075-3:2008 Part 3: *Call-Level Interface* (SQL/CLI). It defines interfacing components (structures, procedures, variable bindings) that can be used to execute SQL statements from applications written in Ada, C respectively C++, COBOL, Fortran, MUMPS, Pascal or PL/I. (For Java see part 10.) SQL/CLI is defined in such a way that SQL statements and SQL/CLI procedure calls are treated as separate from the calling application's source code. Open Database Connectivity is a well-known superset of SQL/CLI. This part of the standard consists solely of *mandatory* features.
- ISO/IEC 9075-4:2011 Part 4: *Persistent Stored Modules* (SQL/PSM) It standardizes procedural extensions for SQL, including flow of control, condition handling, statement condition signals and resignals, cursors and local variables, and assignment of expressions to variables and parameters. In addition, SQL/PSM formalizes declaration and maintenance of persistent database language routines (e.g., "stored procedures"). This part of the standard consists solely of *optional* features.
- ISO/IEC 9075-9:2008 Part 9: *Management of External Data* (SQL/MED). It provides extensions to SQL that define foreign-data wrappers and datalink types to allow SQL to manage external data. External data is data that is accessible to, but not managed by, an SQL-based DBMS. This part of the standard consists solely of *optional* features.
- ISO/IEC 9075-10:2008 Part 10: *Object Language Bindings* (SQL/OLB). It defines the syntax and semantics of SQLJ, which is SQL embedded in Java (see also part 3). The standard also describes mechanisms to ensure binary portability of SQLJ applications, and specifies various Java packages and their contained classes. This part of the standard consists solely of optional features, as opposed to SQL/OLB JDBC, which is not part of the SQL standard, which defines an API.[*citation needed*]
- ISO/IEC 9075-11:2011 Part 11: *Information and Definition Schemas* (SQL/Schemata). It defines the Information Schema and Definition Schema, providing a common set of tools to make SQL databases and objects self-describing. These tools include the SQL object identifier, structure and integrity constraints, security and authorization specifications, features and packages of ISO/IEC 9075, support of features provided by SQL-based DBMS implementations, SQL-based DBMS implementation information and sizing items, and the values supported by the DBMS implementations. This part of the standard contains both *mandatory and optional* features.
- ISO/IEC 9075-13:2008 Part 13: *SQL Routines and Types Using the Java Programming Language* (SQL/JRT). It specifies the ability to invoke static Java methods as routines from within SQL applications ('Java-in-the-database'). It also calls for the ability to use Java classes as SQL structured user-defined types. This part of the standard consists solely of *optional* features.
- ISO/IEC 9075-14:2011 Part 14: *XML-Related Specifications* (SQL/XML). It specifies SQL-based extensions for using XML in conjunction with SQL. The *XMLType* data type is introduced, as well as several routines,

functions, and XML-to-SQL data type mappings to support manipulation and storage of XML in an SQL database. This part of the standard consists solely of *optional* features.[*citation needed*]

ISO/IEC 9075 is complemented by ISO/IEC 13249 *SQL Multimedia and Application Packages*. This closely related but separate standard is developed by the same committee. It defines interfaces and packages which are based on SQL. The aim is a unified access to typical database applications like text, pictures, data mining or spatial data.

- ISO/IEC 13249-1:2007 Part 1: *Framework*
- ISO/IEC 13249-2:2003 Part 2: *Full-Text*
- ISO/IEC 13249-3:2011 Part 3: *Spatial*
- ISO/IEC 13249-5:2003 Part 5: *Still image*
- ISO/IEC 13249-6:2006 Part 6: *Data mining*
- ISO/IEC 13249-8:xxxx Part 8: *Metadata registries (MDR)* (work in progress)

Alternatives

A distinction should be made between alternatives to SQL as a language, and alternatives to the relational model itself. Below are proposed relational alternatives to the SQL language. See navigational database and NoSQL for alternatives to the relational model.

- .QL: object-oriented Datalog
- 4D Query Language (4D QL)
- Datalog
- HTSQL: URL based query method
- IBM Business System 12 (IBM BS12): one of the first fully relational database management systems, introduced in 1982
- ISBL
- jOOQ: SQL implemented in Java as an internal internal domain-specific language
- Java Persistence Query Language (JPQL): The query language used by the Java Persistence API and Hibernate persistence library
- LINQ: Runs SQL statements written like language constructs to query collections directly from inside .Net code.
- Object Query Language
- OttoQL
- QBE (Query By Example) created by Moshè Zloof, IBM 1977
- Quel introduced in 1974 by the U.C. Berkeley Ingres project.
- Tutorial D
- XQuery

Notes

[1] From Oxford Dictionaries: "Definition of SQL - abbreviation, Structured Query Language, an international standard for database manipulation."

[2] From Microsoft: "Structured Query Language, invented at IBM in the 1970s. It is more commonly known by its acronym, SQL .."

[3] http://www.kkhsou.in/main/EVidya2/computer_science/intro_SQL.html

[4] ANSI/ISO/IEC International Standard (IS). Database Language SQL—Part 2: Foundation (SQL/Foundation). 1999.

[5] M. Negri, G. Pelagatti, L. Sbattella (1989) *Semantics and problems of universal quantification in SQL* (http://portal.acm.org/citation.cfm?id=63224.68822&coll=GUIDE&dl=GUIDE).

[6] Fratarcangeli, Claudio (1991). *Technique for universal quantification in SQL*. Retrieved from ACM.org. (http://portal.acm.org/citation.cfm?id=126482.126484&coll=GUIDE&dl=GUIDE&CFID=5934371&CFTOKEN=55309005)

[7] Kawash, Jalal (2004) *Complex quantification in Structured Query Language (SQL): a tutorial using relational calculus* - Journal of Computers in Mathematics and Science Teaching ISSN 0731-9258 Volume 23, Issue 2, 2004 AACE Norfolk, Virginia. Retrieved from Thefreelibrary.com (http://www.thefreelibrary.com/Complex+quantification+in+Structured+Query+Language+(SQL):+a+tutorial...-a0119901477).

[8] (proposed revised text of DIS 9075)].

[9] Arie Jones, Ryan K. Stephens, Ronald R. Plew, Alex Kriegel, Robert F. Garrett (2005), *SQL Functions Programmer's Reference*. Wiley, 127 pages.

[10] http://pic.dhe.ibm.com/infocenter/ntz/v7r0m3/index.jsp?topic=%2Fcom.ibm.nz.sproc.doc%2Fc_sproc_stored_procs.html

[11] http://lbd.udc.es/jornadas2011/actas/PROLE/PROLE/S5/13_article.pdf

References

- Codd, Edgar F (June 1970). "A Relational Model of Data for Large Shared Data Banks" (http://www.acm.org/classics/nov95/toc.html). *Communications of the ACM* **13** (6): 377–87. doi: 10.1145/362384.362685 (http://dx.doi.org/10.1145/362384.362685).
- Discussion on alleged SQL flaws (C2 wiki)
- C. J. Date with Hugh Darwen: *A Guide to the SQL standard : a users guide to the standard database language SQL, 4th ed.*, Addison Wesley, USA 1997, ISBN 978-0-201-96426-4

External links

- *1995 SQL Reunion: People, Projects, and Politics*, by Paul McJones (ed.) (http://www.mcjones.org/System_R/SQL_Reunion_95/sqlr95.html): transcript of a reunion meeting devoted to the personal history of relational databases and SQL.
- American National Standards Institute. X3H2 Records, 1978–1995 (http://special.lib.umn.edu/findaid/xml/cbi00168.xml) Charles Babbage Institute Collection documents the H2 committee's development of the NDL and SQL standards.
- Oral history interview with Donald D. Chamberlin (http://purl.umn.edu/107215) Charles Babbage Institute In this oral history Chamberlin recounts his early life, his education at Harvey Mudd College and Stanford University, and his work on relational database technology. Chamberlin was a member of the System R research team and, with Raymond F. Boyce, developed the SQL database language. Chamberlin also briefly discusses his more recent research on XML query languages.
- Comparison of Different SQL Implementations (http://troels.arvin.dk/db/rdbms/) This comparison of various SQL implementations is intended to serve as a guide to those interested in porting SQL code between various RDBMS products, and includes comparisons between SQL:2008, PostgreSQL, DB2, MS SQL Server, MySQL, Oracle, and Informix.

SQL-92

SQL-92 was the third revision of the SQL database query language. Unlike SQL-89, it was a major revision of the standard. For all but a few minor incompatibilities, the SQL-89 standard is forward compatible with SQL-92.

The standard specification itself grew about five times compared to SQL-89. Much of it was due more precise specifications of existing features; the increase due to new features was only by a factor of 1.5–2. Many of the new features had already been implemented by vendors before the new standard was adopted. However, most of the new features were added to the "intermediate" and "full" tiers of the specification, meaning that conformance with SQL-92 entry level was scarcely any more demanding than conformance with SQL-89.

Later revisions of the standard include SQL:1999 (SQL3), SQL:2003, SQL:2008, and SQL:2011.

New Features

Significant new features include:[1]

- New data types defined: `DATE`, `TIME`, `TIMESTAMP`, `INTERVAL`, `BIT` string, `VARCHAR` strings, and `NATIONAL CHARACTER` strings.
- Support for additional character sets beyond the base requirement for representing SQL statements.
- New scalar operations such as string concatenation and substring extraction, date and time mathematics, and conditional statements.
- New set operations such as `UNION JOIN`, `NATURAL JOIN`, set differences, and set intersections.
- Conditional expressions with `CASE`. For an example, see Case (SQL).
- Support for alterations of schema definitions via `ALTER` and `DROP`.
- Bindings for C, Ada, and MUMPS.
- New features for user privileges.
- New integrity-checking functionality such as within a `CHECK` constraint.
- A new *information schema*—read-only views about database metadata like what tables it contains, etc. For example, `SELECT * FROM INFORMATION_SCHEMA.TABLES;`.
- Dynamic execution of queries (as opposed to prepared).
- Better support for remote database access.
- Temporary tables; `CREATE TEMP TABLE` etc.
- Transaction isolation levels.
- New operations for changing data types on the fly via `CAST (expr AS type)`.
- Scrolled cursors.
- Compatibility flagging for backwards and forwards compatibility with other SQL standards.

Extensions

Two significant extension were published after standard (but before the next major iteration.)

- SQL/CLI (Call Level Interface) in 1995
- SQL/PSM (stored procedures) in 1996

References

[1] C. J. Date with Hugh Darwen: *A Guide to the SQL standard : a users guide to the standard database language SQL, 4th ed.*, Addison Wesley, USA 1997, ISBN 978-0-201-96426-4

External links

- The SQL-92 standard (http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt)
- BNF Grammar for ISO/IEC 9075:1992 - Database Language SQL (SQL-92) (http://savage.net.au/SQL/sql-92.bnf.html)
- Presentation of SQL:1999 (http://dbis-informatik.uibk.ac.at/files/ext/lehre/ss11/vo-ndbm/lit/ORel-SQL1999-IBM-Nelson-Mattos.pdf); covers history and features of SQL-92 as well.

SQL:1999

SQL:1999 (also called SQL 3) was the fourth revision of the SQL database query language. It introduced a large number of new features, many of which required clarifications in the subsequent SQL:2003. The latest revision of the standard is SQL:2011.

Summary

The ISO standard documents were published between 1999 and 2002 in several installments, the first one consisting of multiple parts. Unlike previous editions, the standard's name used a colon instead of a hyphen for consistency with the names of other ISO standards. The first installment of SQL:1999 had five parts:

- SQL/Framework ISO/IEC 9075-1:1999 [1]
- SQL/Foundation ISO/IEC 9075-2:1999 [2]
- SQL/CLI : an updated definition of the extension Call Level Interface, originally published in 1995, also known as CLI-95 ISO/IEC 9075-3:1999 [3]
- SQL/PSM : an updated definition of the extension Persistent Stored Modules, originally published in 1996, also known as PSM-96 ISO/IEC 9075-4:1999 [4]
- SQL/Bindings ISO/IEC 9075-5:1999 [5]

Three more parts, also considered part of SQL:1999 were published subsequently:

- SQL/MED Management of External Data (SQL:1999 part 9) ISO/IEC 9075-9:2001 [6]
- SQL/OLB Object Language Bindings (SQL:1999 part 10) ISO/IEC 9075-10:2000 [7]
- SQL/JRT SQL Routines and Types using the Java Programming Language (SQL:1999 part 13) ISO/IEC 9075-13:2002 [8]

New features

Data types

Boolean data types

The SQL:1999 standard calls for a Boolean type,[9] but many commercial SQL Servers (Microsoft SQL Server 2005, Oracle 9i, IBM DB2) do not support it as a column type, variable type or allow it in the results set. MySQL interprets "BOOLEAN" as a synonym for TINYINT (8-bit signed integer).

Distinct user-defined types of power

Sometimes called just *distinct types*, these were introduced as an optional feature (S011) to allow existing atomic types to be extended with a distinctive meaning to create a new type and thereby enabling the type checking mechanism to detect some logical errors, e.g. accidentally adding an age to a salary. For example:

```
create type age as integer FINAL;
create type salary as integer FINAL;
```

creates two different and incompatible types. The SQL distinct types use name equivalence not structural equivalence like typedefs in C. It's still possible to perform compatible operations on (columns or data) of distinct types by using an explicit type `CAST`.

Few SQL systems support these. IBM DB2 is one those supporting them. Oracle database does not currently support them, recommending instead to emulate them by a one-place structured type.

Structured user-defined types

These are the backbone of the object-relational database extension in SQL:1999. They are analogous to classes in objected-oriented programming languages. SQL:1999 allows only single inheritance.

Common table expressions and recursive queries

SQL:1999 added a WITH [RECURSIVE] construct allowing recursive queries, like transitive closure, to be specified in the query language itself; see common table expressions.

Some OLAP capabilities

GROUP BY was extended with ROLLUP, CUBE, and GROUPING SETS.

Role-based access control

Full support for RBAC via CREATE ROLE.

References

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26196
[2] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26197
[3] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=30609
[4] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29864
[5] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26198
[6] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=31370
[7] http://www.iso.org/iso/catalogue_detail.htm?csnumber=30613
[8] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=35340
[9] ISO/IEC 9075-2:1999 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26197) section 4.6 Boolean types

Further reading

- Jim Melton; Alan R. Simon (2002). *SQL:1999: Understanding Relational Language Components*. Morgan Kaufmann. ISBN 978-1-55860-456-8.
- Jim Melton (2003). *Advanced SQL, 1999: Understanding Object-Relational and Other Advanced Features*. Morgan Kaufmann. ISBN 978-1-55860-677-7.

External links

- Extensive presentation of SQL:1999 (http://dbis-informatik.uibk.ac.at/files/ext/lehre/ss11/vo-ndbm/lit/ORel-SQL1999-IBM-Nelson-Mattos.pdf)

SQL:2003

SQL:2003 is the fifth revision of the SQL database query language. The latest revision of the standard is SQL:2011.

Summary

The SQL:2003 standard makes minor modifications to all parts of SQL:1999 (also known as SQL3), and officially introduces a few new features such as:

- XML-related features (SQL/XML)
- Window functions
- the sequence generator, which allows standardized sequences
- two new column types: auto-generated values and identity-columns
- the new MERGE statement
- extensions to the CREATE TABLE statement, to allow "CREATE TABLE AS" and "CREATE TABLE LIKE"
- removal of the poorly implemented "BIT" and "BIT VARYING" data types
- OLAP capabilities (initially added in SQL:1999) were extended with a window function.

Documentation availability

The SQL standard is not freely available. SQL:2003 may be purchased from ISO [1] or ANSI [2]. A late draft is available as a zip archive [3] from Whitemarsh Information Systems Corporation [4]. The zip archive contains a number of PDF files that define the parts of the SQL:2003 specification.

- ISO/IEC 9075(1-4,9-11,13,14):2003 CD-ROM (352 CHF, or approximately 225 EUR, to order the CD)
  - ISO/IEC 9075-1:2003 [5] – Framework (SQL/Framework)
  - ISO/IEC 9075-2:2003 [6] – Foundation (SQL/Foundation)
  - ISO/IEC 9075-3:2003 [7] – Call-Level Interface (SQL/CLI)
  - ISO/IEC 9075-4:2003 [8] – Persistent Stored Modules (SQL/PSM)
  - ISO/IEC 9075-9:2003 [9] – Management of External Data (SQL/MED)
  - ISO/IEC 9075-10:2003 [10] – Object Language Bindings (SQL/OLB)
  - ISO/IEC 9075-11:2003 [11] – Information and Definition Schemas (SQL/Schemata)
  - ISO/IEC 9075-13:2003 [12] – SQL Routines and Types Using the Java Programming Language (SQL/JRT)
  - ISO/IEC 9075-14:2003 [13] – XML-Related Specifications (SQL/XML)

References

[1] http://www.iso.org/
[2] http://webstore.ansi.org/
[3] http://www.wiscorp.com/sql_2003_standard.zip
[4] http://www.wiscorp.com/
[5] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34132
[6] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34133
[7] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34134
[8] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34135
[9] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34136
[10] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34137
[11] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34917
[12] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=37102
[13] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35341

External links

- BNF Grammar for ISO/IEC 9075-1:2003 (http://savage.net.au/SQL/sql-2003-1.bnf.html) – **SQL/Framework**
- BNF Grammar for ISO/IEC 9075-2:2003 (http://savage.net.au/SQL/sql-2003-2.bnf.html) – SQL/Foundation

SQL:2008

SQL:2008 is the sixth revision of the ISO and ANSI standard for the SQL database query language. It was formally adopted in July 2008.[1]

Summary

The SQL:2008 standard is split into several parts, covering the Framework, the Foundation, the Call-Level Interface, Persistent Stored Modules, Management of External Data, Object Language Bindings, Information and Definition Schemas, Routines and Types Using Java, and various "Related Specifications."

Additions to the Foundation include

- enhanced MERGE and DIAGNOSTIC statements,
- the TRUNCATE TABLE statement,
- comma-separated WHEN clauses in a CASE expression,
- INSTEAD OF database triggers
- partitioned JOIN tables,
- support for various XQuery regular expression/pattern-matching features, and
- enhancements to derived column names.

The Related Specifications for XML defines ways in which SQL can be used in conjunction with XML, including importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form.[2]

Documentation

The SQL standard is not freely available. The whole standard may be purchased from the ISO as *ISO/IEC 9075(1-4,9-11,13,14):2008*. The standard consists of the following parts:

- ISO/IEC 9075-1:2008 [3] Framework (SQL/Framework)
- ISO/IEC 9075-2:2008 [4] Foundation (SQL/Foundation)
- ISO/IEC 9075-3:2008 [5] Call-Level Interface (SQL/CLI)
- ISO/IEC 9075-4:2008 [6] Persistent Stored Modules (SQL/PSM)
- ISO/IEC 9075-9:2008 [7] Management of External Data (SQL/MED)
- ISO/IEC 9075-10:2008 [8] Object Language Bindings (SQL/OLB)
- ISO/IEC 9075-11:2008 [9] Information and Definition Schemas (SQL/Schemata)
- ISO/IEC 9075-13:2008 [10] SQL Routines and Types Using the Java TM Programming Language (SQL/JRT)
- ISO/IEC 9075-14:2008 [11] XML-Related Specifications (SQL/XML)

Claims of conformance

The minimum level of conformance to SQL:2008 that a product can claim is called "Core SQL:2008" and is limited to definitions specified in two parts of the standard: the Foundation and the Information and Definition Schemas.[12]

References

[1] SQL:2008 now an approved ISO International Standard (http://iablog.sybase.com/paulley/2008/07/sql2008-now-an-approved-iso-international-standard/), a July 29th, 2008 blog post from a Sybase blog

[2] International Organization for Standardization: "ISO/IEC 9075-14:2008"

[3] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45498

[4] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38640

[5] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38641

[6] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38642

[7] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38643

[8] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38644

[9] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38645

[10] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38646

[11] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45499

[12] Oracle Compliance To Core SQL:2008 (http://docs.oracle.com/cd/E11882_01/server.112/e26088/ap_standard_sql.htm) from Oracle Corporation

External links

- Freely downloadable drafts of this standard (http://www.wiscorp.com/SQLStandards.html)

SQL:2011

SQL:2011 or **ISO/IEC 9075:2011** (under the general title "Information technology – Database languages – SQL") is the seventh revision of the ISO (1987) and ANSI (1986) standard for the SQL database query language. It was formally adopted in December 2011.

Parts

The standard is split into these parts:

- Part 1: Framework (SQL/Framework)
- Part 2: Foundation (SQL/Foundation)
- Part 3: Call-Level Interface (SQL/CLI)
- Part 4: Persistent Stored Modules (SQL/PSM)
- Part 9: Management of External Data (SQL/MED)
- Part 10: Object Language Bindings (SQL/OLB)
- Part 11: Information and Definition Schemas (SQL/Schemata)
- Part 13: SQL Routines and Types Using the Java™ Programming Language (SQL/JRT)
- Part 14: XML-Related Specifications (SQL/XML)

New features

Temporal support

One of the main new features is improved support for temporal databases.[1][2] Language enhancements for temporal data definition and manipulation include:

- **Time Period definitions** use two standard table columns as the start and end of a named time period, with closed-open semantics. This provides compatibility with existing data models, application code, and tools
- Definition of **application time period tables** (elsewhere called valid time tables), using the `PERIOD FOR` annotation
- Update and deletion of application time rows with **automatic time period splitting**
- **Temporal primary keys** incorporating application time periods with optional non-overlapping constraints via the `WITHOUT OVERLAPS` clause
- **Temporal referential integrity** constraints for application time tables
- Application time tables are queried using regular query syntax or using new **temporal predicates** for time periods including `CONTAINS`, `OVERLAPS`, `EQUALS`, `PRECEDES`, `SUCCEEDS`, `IMMEDIATELY PRECEDES`, and `IMMEDIATELY SUCCEEDS` (which are modified versions of Allen's interval relations)
- Definition of **system-versioned tables** (elsewhere called transaction time tables), using the `PERIOD FOR SYSTEM_TIME` annotation and `WITH SYSTEM VERSIONING` modifier. System time periods are maintained automatically. Constraints for system-versioned tables are not required to be temporal and are only enforced on current rows
- Syntax for **time-sliced** and **sequenced** queries on system time tables via the `AS OF SYSTEM TIME` and `VERSIONS BETWEEN SYSTEM TIME ... AND ...` clauses
- Application time and system versioning can be used together to provide **bitemporal tables**

IBM DB2 version 10 claims to be the first database to have a conforming implementation of this feature in what they call "Time Travel Queries".,[3][4] although they use the alternative syntax `FOR SYSTEM_TIME AS OF`.

Oracle (version 10 and above) has similar functionality in what they call **Flashback Queries**, using the alternative syntax `AS OF TIMESTAMP`.[5]

References

[1] Zemke, Fred. " What's new in SQL:2011 (http://www.sigmod.org/publications/sigmod-record/1203/pdfs/10.industry.zemke.pdf)". ACM SIGMOD Record 41.1 (2012): 67-73.

[2] Kulkarni, Krishna, and Jan-Eike Michels. " Temporal features in SQL: 2011 (http://www.sigmod.org/publications/sigmod-record/1209/pdfs/07.industry.kulkarni.pdf)". ACM SIGMOD Record 41.3 (2012): 34-43.

[3] http://www.ibm.com/developerworks/data/library/techarticle/dm-1204whatsnewdb210/index.html

[4] http://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/

[5] http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10002.htm

External links

- "SQL:2011" (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53681) (webshop), *Catalogue*, ISO.
- *Part 1: Framework (SQL/Framework)* (http://www.jtc1sc32.org/doc/N2151-2200/32N2153T-text_for_ballot-FDIS_9075-1.pdf) (draft), JTC1SC32, 2011-08-06.
- *Part 14: XML-Related Specifications (SQL/XML)* (http://www.jtc1sc32.org/doc/N2151-2200/32N2157T-text_for_ballot-FDIS_9075-14.pdf) (draft), JTC1SC32, 2011-08-06.
- *List of further freely available Final Committee Drafts* (https://www.google.com/search?q=intitle:"final+committee+draft+"+intitle:9075+site:jtc1sc32.org+2011) (search), Google.

Data

Data (/ˈdeɪtə/ ***DAY***-*tə* or /ˈdætə/ ***DA***-*tə*, also /ˈdɑːtə/ ***DAH***-*tə*) is a set of values of qualitative or quantitative variables; restated, data are individual pieces of information. Data in computing (or data processing) are represented in a structure that is often tabular (represented by rows and columns), a tree (a set of nodes with parent-children relationship), or a graph (a set of connected nodes). Data are typically the results of measurements and can be visualised using graphs or images.

Data as an abstract concept can be viewed as the lowest level of abstraction, from which information and then knowledge are derived.

Raw data, i.e., unprocessed data, refers to a collection of numbers, characters and is a relative term; data processing commonly occurs by stages, and the "processed data" from one stage may be considered the "raw data" of the next. Field data refers to raw data that is collected in an uncontrolled in situ environment. Experimental data refers to data that is generated within the context of a scientific investigation by observation and recording.

The word *data* is the traditional plural form of the now-archaic *datum*, neuter past participle of the Latin *dare*, "to give", hence "something given". In discussions of problems in geometry, mathematics, engineering, and so on, the terms *givens* and *data* are used interchangeably. This usage is the origin of *data* as a concept in computer science or data processing: data are accepted numbers, words, images, etc.

Data is also increasingly used in humanities (particularly in the growing digital humanities) the highly interpretive nature whereof might oppose the ethos of data as "given". Peter Checkland introduced the term *capta* (from the Latin *capere*, “to take”) to distinguish between an immense number of possible data and a sub-set of them, to which attention is oriented. Johanna Drucker has argued that the humanities affirm knowledge production as “situated, partial, and constitutive” and that using *data* may therefore introduce assumptions that are counterproductive, for example that phenomena are discrete or observer-independent. The term *capta*, which emphasizes the act of observation as constitutive, is offered as an alternative to *data* for visual representations in the humanities.

Usage in English

Datum means "an item given". In cartography, geography, nuclear magnetic resonance and technical drawing it often refers to a reference datum wherefrom distances to all other data are measured. Any measurement or result is a *datum*, though *data point* is now far more common.

In one sense, *datum* is a count noun with the plural *datums* (see usage in datum article) that can be used with cardinal numbers (e.g. "80 datums"); *data* (originally a Latin plural) is not used like a normal count noun with cardinal numbers and can be plural with such plural determiners as *these* and *many* or as a singular abstract mass noun with a verb in the singular form. Even when a very small quantity of data is referenced (one number, for example) the phrase *piece of data* is often used, as opposed to *datum*. The debate over appropriate usage continues.

The IEEE Computer Society allows usage of *data* as either a mass noun or plural based on author preference. Some professional organizations and style guidesWikipedia:Link rot require that authors treat *data* as a plural noun. For example, the Air Force Flight Test Center specifically states that the word *data* is always plural, never singular.

Data is most often used as a singular mass noun in educated everyday usage.[1][2] Some major newspapers such as *The New York Times* use it either in the singular or plural. In the *New York Times* the phrases "the survey data are still being analyzed" and "the first year for which data is available" have appeared within one day. The *Wall Street Journal* explicitly allows this usage in its style guide. The Associated Press style guide classifies *data* as a collective noun that is singular when a unit and plural when referring to individual items ("The data is sound.", and "The data have been carefully collected.").

In scientific writing *data* is often treated as a plural, as in *These data do not support the conclusions*, and as a singular mass entity like *information*, for instance in computing and related disciplines. British usage now widely accepts treating *data* as singular in standard English, including everyday newspaper usage at least in non-scientific use. UK scientific publishing still prefers treating it as a plural. Some UK university style guides recommend using *data* for both singular and plural use and some recommend treating it only as a singular in connection with computers.

Meaning of data, information and knowledge

Data, information and knowledge frequently overlap, mainly differing in abstraction. Data is least abstract, information next least, and knowledge most. Data becomes information by interpretation; e.g., the height of Mt. Everest is generally considered as "data", a book on Mt. Everest geological characteristics may be considered as "information", and a report containing practical information on the best way to reach Mt. Everest's peak may be considered as "knowledge".

'Information' bears a diversity of meanings that ranges from everyday to technical. Generally speaking, the concept of information is closely related to notions of constraint, communication, control, data, form, instruction, knowledge, meaning, mental stimulus, pattern, perception, and representation.

Beynon-Davies uses the concept of a sign to distinguish between data and information; data are symbols while information occurs when the symbols are used to refer to something.

It is people and computers who collect data and impose patterns on it. These patterns are seen as information which can be used to enhance knowledge. These patterns can be interpreted as truth, and are authorized as aesthetic and ethical criteria. Events that leave behind perceivable physical or virtual remains can be traced back through data. Marks are no longer considered data once the link between the mark and observation is broken.

Mechanical computing devices are classified according to the means by which they represent data. An analog computer represents a datum as a voltage, distance, position, or other physical quantity. A digital computer represents a datum as a sequence of symbols drawn from a fixed alphabet. The most common digital computers use a binary alphabet, that is, an alphabet of two characters, typically denoted "0" and "1". More familiar representations, such as numbers or letters, are then constructed from the binary alphabet.

Some special forms of data are distinguished. A computer program is a collection of data, which can be interpreted as instructions. Most computer languages make a distinction between programs and the other data on which programs operate, but in some languages, notably Lisp and similar languages, programs are essentially indistinguishable from other data. It is also useful to distinguish metadata, that is, a description of other data. A similar yet earlier term for metadata is "ancillary data." The prototypical example of metadata is the library catalog, which is a description of the contents of books.

References

This article is based on material taken from the Free On-line Dictionary of Computing prior to 1 November 2008 and incorporated under the "relicensing" terms of the GFDL, version 1.3 or later.

[1] New Oxford Dictionary of English, 1999

[2] "...in educated everyday usage as represented by the Guardian newspaper, it is nowadays most often used as a singular." http://www.lexically.net/TimJohns/Kibbitzer/revis006.htm

External links

- Data is a singular noun (http://purl.org/nxg/note/singular-data) (a detailed assessment)

Metadata

Metadata is "data about data". The term is ambiguous, as it is used for two fundamentally different concepts (types). **Structural metadata** is about the design and specification of data structures and is more properly called "data about the containers of data"; **descriptive metadata**, on the other hand, is about individual instances of application data, the data content.

Metadata are traditionally found in the card catalogs of libraries. As information has become increasingly digital, metadata are also used to describe digital data using metadata standards specific to a particular discipline. By describing the contents and context of data files, the quality of the original data/files is greatly increased. For example, a webpage may include metadata specifying what language it is written in, what tools were used to create it, and where to go for more on the subject, allowing browsers to automatically improve the experience of users.

Definition

Metadata (metacontent) are defined as the data providing information about one or more aspects of the data, such as:

- Means of creation of the data
- Purpose of the data
- Time and date of creation
- Creator or author of the data
- Location on a computer network where the data were created
- Standards used

For example, a digital image may include metadata that describe how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document's metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document.

Metadata are data. As such, metadata can be stored and managed in a database, often called a metadata registry or metadata repository.[1] However, without context and a point of reference, it might be impossible to identify metadata just by looking at them. For example: by itself, a database containing several numbers, all 13 digits long could be the results of calculations or a list of numbers to plug into an equation - without any other context, the

numbers themselves can be perceived as the data. But if given the context that this database is a log of a book collection, those 13-digit numbers may now be identified as ISBNs - information that refers to the book, but is not itself the information within the book.

The term "metadata" was coined in 1968 by Philip Bagley, in his book "Extension of programming language concepts" where it is clear that he uses the term in the ISO 11179 "traditional" sense, which is "structural metadata" i.e. "data about the containers of data"; rather than the alternate sense "content about individual instances of data content" or metacontent, the type of data usually found in library catalogues.[2] Since then the fields of information management, information science, information technology, librarianship and GIS have widely adopted the term. In these fields the word *metadata* is defined as "data about data". While this is the generally accepted definition, various disciplines have adopted their own more specific explanation and uses of the term.

Libraries

Metadata have been used in various forms as a means of cataloging archived information. The Dewey Decimal System employed by libraries for the classification of library materials is an early example of metadata usage. Library catalogues used 3x5 inch cards to display a book's title, author, subject matter, and a brief plot synopsis along with an abbreviated alpha-numeric identification system which indicated the physical location of the book within the library's shelves. Such data help classify, aggregate, identify, and locate a particular book. Another form of older metadata collection is the use by US Census Bureau of what is known as the "Long Form." The Long Form asks questions that are used to create demographic data to find patterns of distribution.

Photographs

Metadata may be written into a digital photo file that will identify who owns it, copyright and contact information, what camera created the file, along with exposure information and descriptive information such as keywords about the photo, making the file searchable on the computer and/or the Internet. Some metadata are written by the camera and some is input by the photographer and/or software after downloading to a computer. However, not all digital cameras enable you to edit metadata; this functionality has been available on most Nikon DSLRs since the Nikon D3 and on most new Canon cameras since the Canon EOS 7D.

Photographic Metadata Standards are governed by organizations that develop the following standards. They include, but are not limited to:

- IPTC Information Interchange Model IIM (International Press Telecommunications Council),
- IPTC Core Schema for XMP
- XMP – Extensible Metadata Platform (an ISO standard)
- Exif – Exchangeable image file format, Maintained by CIPA (Camera & Imaging Products Association) and published by JEITA (Japan Electronics and Information Technology Industries Association)
- Dublin Core (Dublin Core Metadata Initiative – DCMI)
- PLUS (Picture Licensing Universal System).

Video

Metadata are particularly useful in video, where information about its contents (such as transcripts of conversations and text descriptions of its scenes) are not directly understandable by a computer, but where efficient search is desirable.

Web pages

Web pages often include metadata in the form of meta tags. Description and keywords meta tags are commonly used to describe the Web page's content. Most search engines use these data when adding pages to their search index.

Creation of metadata

Metadata can be created either by automated information processing or by manual work. Elementary metadata captured by computers can include information about when an object was created, who created it, when it was last updated, file size and file extension.

For the purposes of this article, an "object" refers to any of the following:

- A physical item such as a book, CD, DVD, map, chair, table, flower pot, etc.
- An electronic file such as a digital image, digital photo, document, program file, database table, etc.

Metadata types

The metadata application is manyfold covering a large variety of fields of application there are nothing but specialised and well accepted models to specify types of metadata. Bretheron & Singley (1994) distinguish between two distinct classes: structural/control metadata and guide metadata. **Structural metadata** are used to describe the structure of database objects such as tables, columns, keys and indexes. **Guide metadata** are used to help humans find specific items and are usually expressed as a set of keywords in a natural language. According to Ralph Kimball metadata can be divided into 2 similar categories: technical metadata and business metadata. **Technical metadata** correspond to internal metadata, *business metadata* - to external metadata. Kimball adds a third category named **process metadata**. On the other hand, NISO distinguishes among three types of metadata: descriptive, structural and administrative. **Descriptive metadata** are the information used to search and locate an object such as title, author, subjects, keywords, publisher; **structural metadata** give a description of how the components of the object are organised; and **administrative metadata** refer to the technical information including file type. Two sub-types of administrative metadata are rights management metadata and preservation metadata.

Metadata structures

Metadata (metacontent), or more correctly, the vocabularies used to assemble metadata (metacontent) statements, are typically structured according to a standardized concept using a well-defined metadata scheme, including: metadata standards and metadata models. Tools such as controlled vocabularies, taxonomies, thesauri, data dictionaries and metadata registries can be used to apply further standardization to the metadata. Structural metadata commonality is also of paramount importance in data model development and in database design.

Metadata syntax

Metadata (metacontent) syntax refers to the rules created to structure the fields or elements of metadata (metacontent). A single metadata scheme may be expressed in a number of different markup or programming languages, each of which requires a different syntax. For example, Dublin Core may be expressed in plain text, HTML, XML and RDF.

A common example of (guide) metacontent is the bibliographic classification, the subject, the Dewey Decimal class number. There is always an implied statement in any "classification" of some object. To classify an object as, for example, Dewey class number 514 (Topology) (i.e. books having the number 514 on their spine) the implied statement is: "<book><subject heading><514>. This is a subject-predicate-object triple, or more importantly, a class-attribute-value triple. The first two elements of the triple (class, attribute) are pieces of some structural metadata having a defined semantic. The third element is a value, preferably from some controlled vocabulary, some reference (master) data. The combination of the metadata and master data elements results in a statement which is a metacontent statement i.e. "metacontent = metadata + master data". All these elements can be thought of as "vocabulary". Both metadata and master data are vocabularies which can be assembled into metacontent statements. There are many sources of these vocabularies, both meta and master data: UML, EDIFACT, XSD, Dewey/UDC/LoC, SKOS, ISO-25964, Pantone, Linnaean Binomial Nomenclature etc. Using controlled vocabularies for the components of metacontent statements, whether for indexing or finding, is endorsed by ISO-25964 [3]: "If both the indexer and the searcher are guided to choose the same term for the same concept, then relevant documents will be retrieved." This is particularly relevant when considering the behemoth of the internet, Google. It simply indexes pages then matches text strings using its complex algorithm, there is no intelligence or "inferencing" occurring. Just the illusion thereof.

Hierarchical, linear and planar schemata

Metadata schema can be hierarchical in nature where relationships exist between metadata elements and elements are nested so that parent-child relationships exist between the elements. An example of a hierarchical metadata schema is the IEEE LOM schema where metadata elements may belong to a parent metadata element. Metadata schema can also be one-dimensional, or linear, where each element is completely discrete from other elements and classified according to one dimension only. An example of a linear metadata schema is Dublin Core schema which is one dimensional. Metadata schema are often two dimensional, or planar, where each element is completely discrete from other elements but classified according to two orthogonal dimensions.

Metadata hypermapping

In all cases where the metadata schemata exceed the planar depiction, some type of hypermapping is required to enable display and view of metadata according to chosen aspect and to serve special views. Hypermapping frequently applies to layering of geographical and geological information overlays.

Granularity

The degree to which the data or metadata are structured is referred to as their granularity. Metadata with a high granularity allow for deeper structured information and enable greater levels of technical manipulation however, a lower level of granularity means that metadata can be created for considerably lower costs but will not provide as detailed information. The major impact of granularity is not only on creation and capture, but moreover on maintenance. As soon as the metadata structures get outdated, the access to the referred data will get outdated. Hence granularity shall take into account the effort to create as well as the effort to maintain.

Metadata standards

International standards apply to metadata. Much work is being accomplished in the national and international standards communities, especially ANSI (American National Standards Institute) and ISO (International Organization for Standardization) to reach consensus on standardizing metadata and registries.

The core standard is ISO/IEC 11179-1:2004 and subsequent standards (see ISO/IEC 11179). All yet published registrations according to this standard cover just the definition of metadata and do not serve the structuring of metadata storage or retrieval neither any administrative standardisation. It is important to note that this standard refers to metadata as the data about containers of the data and not to metadata (metacontent) as the data about the data contents. It should also be noted that this standard describes itself originally as a "data element" registry, describing disembodied data elements, and explicitly disavows the capability of containing complex structures. Thus the original term "data element" is more applicable than the later applied buzzword "metadata".

The Dublin Core metadata terms are a set of vocabulary terms which can be used to describe resources for the purposes of discovery. The original set of 15 classic metadata terms, known as the Dublin Core Metadata Element Set are endorsed in the following standards documents:

- IETF RFC 5013
- ISO Standard 15836-2009
- NISO Standard Z39.85.

Although not a standard, Microformat (also mentioned in the section metadata on the internet below) is a web-based approach to semantic markup which seeks to re-use existing HTML/XHTML tags to convey metadata. Microformat follows XHTML and HTML standards but is not a standard in itself. One advocate of microformats, Tantek Çelik, characterized a problem with alternative approaches:

> Here's a new language we want you to learn, and now you need to output these additional files on your server. It's a hassle. (Microformats) lower the barrier to entry.

Metadata usage

Data virtualization

Data virtualization has emerged as the new software technology to complete the virtualization stack in the enterprise. Metadata are used in data virtualization servers which are enterprise infrastructure components, alongside database and application servers. Metadata in these servers are saved as persistent repository and describe business objects in various enterprise systems and applications. Structural metadata commonality is also important to support data virtualization.

Statistics and census services

Standardization work has had a large impact on efforts to build metadata systems in the statistical community[*citation needed*]. Several metadata standardsWikipedia:Avoid weasel words are described, and their importance to statistical agencies is discussed. Applications of the standardsWikipedia:Avoid weasel words at the Census Bureau, Environmental Protection Agency, Bureau of Labor Statistics, Statistics Canada, and many others are described[*citation needed*]. Emphasis is on the impact a metadata registry can have in a statistical agency.

Library and information science

Libraries employ metadata in library catalogues, most commonly as part of an Integrated Library Management System. Metadata are obtained by cataloguing resources such as books, periodicals, DVDs, web pages or digital images. These data are stored in the integrated library management system, ILMS, using the MARC metadata standard. The purpose is to direct patrons to the physical or electronic location of items or areas they seek as well as to provide a description of the item/s in question.

More recent and specialized instances of library metadata include the establishment of digital libraries including e-print repositories and digital image libraries. While often based on library principles, the focus on non-librarian use, especially in providing metadata, means they do not follow traditional or common cataloging approaches. Given the custom nature of included materials, metadata fields are often specially created e.g. taxonomic classification fields, location fields, keywords or copyright statement. Standard file information such as file size and format are usually automatically included.

Standardization for library operation has been a key topic in international standardization (ISO) for decades. Standards for metadata in digital libraries include Dublin Core, METS, MODS, DDI, ISO standard Digital Object Identifier (DOI), ISO standard Uniform Resource Name (URN), PREMIS schema, Ecological Metadata Language, and OAI-PMH. Leading libraries in the world give hints on their metadata standards strategies.

Metadata and the law

United States

Problems involving metadata in litigation in the United States are becoming widespread.Wikipedia:Manual of Style/Dates and numbers#Chronological items Courts have looked at various questions involving metadata, including the discoverability of metadata by parties. Although the Federal Rules of Civil Procedure have only specified rules about electronic documents, subsequent case law has elaborated on the requirement of parties to reveal metadata. In October 2009, the Arizona Supreme Court has ruled that metadata records are public record.

Document metadata have proven particularly important in legal environments in which litigation has requested metadata, which can include sensitive information detrimental to a party in court.

Using metadata removal tools to "clean" documents can mitigate the risks of unwittingly sending sensitive data. This process partially (see Data remanence) protects law firms from potentially damaging leaking of sensitive data through electronic discovery.

Metadata in healthcare

Australian researches in medicine started a lot of metadata definition for applications in health care. That approach offers the first recognized attempt to adhere to international standards in medical sciences instead of defining a proprietary standard under the WHO umbrella first.

The medical community yet did not approve the need to follow metadata standards despite respective research.[4]

Metadata and data warehousing

Data warehouse (DW) is a repository of an organization's electronically stored data. Data warehouses are designed to manage and store the data whereas the business intelligence (BI) focuses on the usage of the data to facilitate reporting and analysis.[5]

The purpose of a data warehouse is to house standardized, structured, consistent, integrated, correct, cleansed and timely data, extracted from various operational systems in an organization. The extracted data are integrated in the data warehouse environment in order to provide an enterprise wide perspective, one version of the truth. Data are structured in a way to specifically address the reporting and analytic requirements. The design of structural metadata

commonality using a data modeling method such as entity relationship model diagramming is very important in any data warehouse development effort.

An essential component of a data warehouse/business intelligence system is the metadata and tools to manage and retrieve the metadata. Ralph Kimball describes metadata as the DNA of the data warehouse as metadata defines the elements of the data warehouse and how they work together.

Kimball et al. refers to three main categories of metadata: Technical metadata, business metadata and process metadata. Technical metadata are primarily definitional, while business metadata and process metadata are primarily descriptive. Keep in mind that the categories sometimes overlap.

- **Technical metadata** define the objects and processes in a DW/BI system, as seen from a technical point of view. The technical metadata include the system metadata which define the data structures such as: tables, fields, data types, indexes and partitions in the relational engine, and databases, dimensions, measures, and data mining models. Technical metadata define the data model and the way it is displayed for the users, with the reports, schedules, distribution lists and user security rights.
- **Business metadata** are a content from the data warehouse described in more user-friendly terms. The business metadata tell you what data you have, where they come from, what they mean and what their relationship is to other data in the data warehouse. Business metadata may also serve as a documentation for the DW/BI system. Users who browse the data warehouse are primarily viewing the business metadata.
- **Process metadata** are used to describe the results of various operations in the data warehouse. Within the ETL process, all key data from tasks are logged on execution. This includes start time, end time, CPU seconds used, disk reads, disk writes and rows processed. When troubleshooting the ETL or query process, this sort of data becomes valuable. Process metadata are the fact measurement when building and using a DW/BI system. Some organizations make a living out of collecting and selling this sort of data to companies - in that case the process metadata becomes the business metadata for the fact and dimension tables. Collecting process metadata is in the interest of business people who can use the data to identify the users of their products, which products they are using and what level of service they are receiving.

Metadata on the Internet

The HTML format used to define web pages allows for the inclusion of a variety of types of metadata, from basic descriptive text, dates and keywords to further advanced metadata schemes such as the Dublin Core, e-GMS, and AGLS[6] standards. Pages can also be geotagged with coordinates. Metadata may be included in the page's header or in a separate file. Microformats allow metadata to be added to on-page data in a way that users do not see, but computers can readily access.

Interestingly, many search engines are cautious about using metadata in their ranking algorithms due to exploitation of metadata and the practice of search engine optimization, SEO, to improve rankings. See Meta element article for further discussion. Studies show that search engines respond to web pages with metadata implementations.[7]

Metadata in the broadcast industry

In broadcast industry, metadata are linked to audio and video Broadcast media to:

- *identify* the media: clip or playlist names, duration, timecode, etc.
- *describe* the content: notes regarding the quality of video content, rating, description (for example, during a sport event, keywords like *goal*, *red card* will be associated to some clips)
- *classify* media: metadata allow to sort the media or to easily and quickly find a video content (a TV news could urgently need some archive content for a subject). For example, the BBC have a large subject classification system, Lonclass, a customized version of the more general-purpose Universal Decimal Classification.

These metadata can be linked to the video media thanks to the video servers. All latest broadcasted sport events like FIFA World Cup or Olympic Games use these metadata to distribute their video content to TV stations through keywords. It's often the host broadcaster who is in charge of organizing metadata through its *International Broadcast Centre* and its video servers. Those metadata are recorded with the images and are entered by metadata operators (*loggers*) who associate in live metadata available in *metadata grids* through software (such as Multicam(LSM) or IPDirector used during FIFA World Cup or Olympic Games).

Geospatial metadata

Metadata that describe geographic objects (such as datasets, maps, features, or simply documents with a geospatial component) have a history dating back to at least 1994 (refer MIT Library page on FGDC Metadata [8]). This class of metadata is described more fully on the Geospatial metadata page.

Ecological and environmental metadata

Ecological and environmental metadata are intended to document the who, what, when, where, why, and how of data collection for a particular study. Metadata should be generated in a format commonly used by the most relevant science community, such as Darwin Core, Ecological Metadata Language, or Dublin Core. Metadata editing tools exist to facilitate metadata generation (e.g. Metavist, Mercury: Metadata Search System, Morpho). Metadata should describe provenance of the data (where they originated, as well as any transformations the data underwent) and how to give credit for (cite) the data products.

Digital music

CDs such as recordings of music will carry a layer of metadata about the recordings such as dates, artist, genre, copyright owner, etc. The metadata, not normally displayed by CD players, can be accessed and displayed by specialized music playback and/or editing applications.

The metadata for compressed and uncompressed digital music is often encoded in the ID3 tag. Common editors such as TagLib support MP3, Ogg Vorbis, FLAC, MPC, Speex, WavPack TrueAudio, WAV, AIFF, MP4 and ASF file formats.

Cloud applications

With the availability of Cloud applications, which include those to add metadata to content, metadata is increasingly available over the Internet.

Metadata administration and management

Metadata storage

Metadata can be stored either *internally*, in the same file as the data, or *externally*, in a separate file. Metadata that are embedded with content is called *embedded metadata*. A data repository typically stores the metadata *detached* from the data. Both ways have advantages and disadvantages:

- Internal storage allows transferring metadata together with the data they describe; thus, metadata are always at hand and can be manipulated easily. This method creates high redundancy and does not allow holding metadata together.
- External storage allows bundling metadata, for example in a database, for more efficient searching. There is no redundancy and metadata can be transferred simultaneously when using streaming. However, as most formats use URIs for that purpose, the method of how the metadata are linked to their data should be treated with care. What if a resource does not have a URI (resources on a local hard disk or web pages that are created on-the-fly using a content management system)? What if the metadata can only be evaluated if there is a connection to the Web, especially when using RDF? How to realize that a resource is replaced by another with the same name but different content?

Moreover, there is the question of data format: storing metadata in a human-readable format such as XML can be useful because users can understand and edit it without specialized tools. On the other hand, these formats are not optimized for storage capacity; it may be useful to store metadata in a binary, non-human-readable format instead to speed up transfer and save memory.

Metadata management

Metadata management is the end-to-end process and governance framework for creating, controlling, enhancing, attributing, defining and managing a metadata schema, model or other structured aggregation, either independently or within a repository and the associated supporting processes (often to enable the management of content). The world Wide Web Consortium (W3C) has identified Governance as a key challenge in the advancement of third generation Web Technologies (Web 3.0, Semantic Web), and a number of research prototypes, such as S3DB, explore the use of semantic modeling to identify practical solutions.

Database management

Each relational database system has its own mechanisms for storing metadata. Examples of relational-database metadata include:

- Tables of all tables in a database, their names, sizes and number of rows in each table.
- Tables of columns in each database, what tables they are used in, and the type of data stored in each column.

In database terminology, this set of metadata is referred to as the catalog. The SQL standard specifies a uniform means to access the catalog, called the information schema, but not all databases implement it, even if they implement other aspects of the SQL standard. For an example of database-specific metadata access methods, see Oracle metadata. Programmatic access to metadata is possible using APIs such as JDBC, or SchemaCrawler.

- Agris: International Information System for the Agricultural Sciences and Technology
- Classification scheme
- Crosswalk (metadata)
- DataONE
- Data Dictionary (aka metadata repository)
- Dublin Core
- Folksonomy
- GEOMS – Generic Earth Observation Metadata Standard
- IPDirector
- ISO/IEC 11179
- Knowledge tag
- Mercury: Metadata Search System
- Meta element
- Metadata Access Point Interface
- Metadata discovery
- Metadata facility for Java
- Metadata from Wikiversity
- Metadata publishing
- Metadata registry
- Metamathematics
- METAFOR Common Metadata for Climate Modelling Digital Repositories
- Microcontent
- Microformat
- Multicam(LSM)
- Ontology (computer science)
- Official statistics
- Paratext
- Preservation Metadata
- SDMX
- Semantic Web
- SGML
- The Metadata Company
- Universal Data Element Framework
- Vocabulary OneSource
- XSD

References

[1] Hüner, K.; Otto, B.; Österle, H.: Collaborative management of business metadata, in: International Journal of Information Management, 2011

[2] "The notion of "metadata" introduced by Bagley".

[3] http://www-personal.umich.edu/~kdow/ISO_CD_25964-1(E).pdf

[4] M. Löbe, M. Knuth, R. Mücke TIM: A Semantic Web Application for the Specification of Metadata Items in Clinical Research (http://ceur-ws.org/Vol-559/Paper1.pdf), CEUR-WS.org, urn:nbn:de:0074-559-9

[5] Inmon, W.H. Tech Topic: What is a Data Warehouse? Prism Solutions. Volume 1. 1995.

[6] National Archives of Australia, AGLS Metadata Standard, accessed 7 January 2010, (http://www.naa.gov.au/records-management/create-capture-describe/describe/AGLS/index.aspx)

[7] The impact of webpage content characteristics on webpage visibility in search engine results http://web.simmons.edu/~braun/467/part_1.pdf

[8] http://libraries.mit.edu/guides/subjects/metadata/standards/fgdc.html

External links

- Mercury: Metadata Management, Data Discovery and Access (http://mercury.ornl.gov/ornldaac), managed by Oak Ridge National Laboratory Distributed Active Archive Center
- Guardian US interactive team. " A Guardian guide to your Metadata (http://www.guardian.co.uk/technology/interactive/2013/jun/12/what-is-metadata-nsa-surveillance#meta=0000000)." *The Guardian*. Wednesday 12 June 2013.
- Metacrap: Putting the torch to seven straw-men of the meta-utopia (http://www.well.com/~doctorow/metacrap.htm) – Cory Doctorow's opinion on the limitations of metadata on the Internet, 2001
- Retrieving Meta Data from Documents and Pictures Online (http://www.anonwatch.com/?p=9) - AnonWatch
- Understanding Metadata (http://www.niso.org/publications/press/UnderstandingMetadata.pdf) - NISO, 2004
- DataONE (http://www.dataone.org) Investigator Toolkit
- Journal of Library Metadata (http://www.informaworld.com/openurl?genre=journal&issn=1938-6389), Routledge, Taylor & Francis Group, ISSN 1937-5034
- International Journal of Metadata, Semantics and Ontologies (IJMSO) (http://www.inderscience.com/ijmso), Inderscience Publishers, ISSN 1744-263X

- AFC2IC Vocabulary OneSource Tool (https://gcic.af.mil/onesource)
- On metadata and metacontent (http://www.metalounge.org/_literature_52579/Stephen_Machin_â_ON_METADATA_AND_METACONTENT) archiv.org (http://web.archive.org/web/*/http://www.metalounge.org/_literature_52579/Stephen_Machin_â_ON_METADATA_AND_METACONTENT)
- Managing Metadata (http://library.caltech.edu/laura/) blog

Database objects

Table

In relational databases and flat file databases, a **table** is an organized set of data elements (values) using a model of vertical columns (which are identified by their name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a unique key index.

Table is another term for relation; although there is the difference in that a table is usually a multiset (bag) of rows whereas a relation is a set and does not allow duplicates. Besides the actual data rows, tables generally have associated with them some metadata, such as constraints on the table or on the values within particular columns.Wikipedia:Disputed statement|

The data in a table does not have to be physically stored in the database. Views are also relational tables, but their data are calculated at query time. Another example are nicknames[*clarify*], which represent a pointer to a table in another database.

Comparisons

In non-relational systems, hierarchical databases, the distant counterpart of a table is a structured file, representing the rows of a table in each record of the file and each column in a record.This structure implies that a record can have repeating information, Generally in the child data segments.Data are stored in sequence of records which are equivalent to table term of a relational database.with each record having equivalent rows.

Unlike a spreadsheet, the datatype of field is ordinarily defined by the schema describing the table. Some SQL systems, such as SQLite, are less strict about field datatype definitions.

Tables versus relations

In terms of the relational model of databases, a table can be considered a convenient representation of a relation, but the two are not strictly equivalent. For instance, an SQL table can potentially contain duplicate rows, whereas a true relation cannot contain duplicate tuples. Similarly, representation as a table implies a particular ordering to the rows and columns, whereas a relation is explicitly unordered. However, the database system does not guarantee any ordering of the rows unless an `ORDER BY` clause is specified in the `SELECT` statement that queries the table.

An equally valid representations of a relation is as an *n*-dimensional chart, where *n* is the number of attributes (a table's columns). For example, a relation with two attributes and three values can be represented as a table with two columns and three rows, or as a two-dimensional graph with three points. The table and graph representations are only equivalent if the ordering of rows is not significant, and the table has no duplicate rows.

Table types

Two types of tables exist:

- A relational table, which is the basic structure to hold user data in a relational database.
- An object table, which is a table that uses an object type to define a column. It is defined to hold instances of objects of a defined type.

In SQL, the `CREATE TABLE` statement creates these tables.

References

Column

In the context of a relational database table, a **column** is a set of data values of a particular simple type, one for each row of the table.[1] The columns provide the structure according to which the rows are composed.

The term *field* is often used interchangeably with *column*, although many consider it more correct to use *field* (or *field value*) to refer specifically to the single item that exists at the intersection between one row and one column.

In relational database terminology, column's equivalent is called **attribute**.

For example, a table that represents companies might have the following columns:

- ID (integer identifier, unique to each row)
- Name (text)
- Address line 1 (text)
- Address line 2 (text)
- City (integer identifier, drawn from a separate table of cities, from which any state or country information would be drawn)
- Postal code (text)
- Industry (integer identifier, drawn from a separate table of industries)
- etc.

Each row would provide a data value for each column and would then be understood as a single structured data value, in this case representing a company. More formally, each row can be interpreted as a relvar, composed of a set of tuples, with each tuple consisting of the two items: the name of the relevant column and the value this row provides for that column.

	Column 1	**Column 2**
Row 1	Row 1, Column 1	Row 1, Column 2
Row 2	Row 2, Column 1	Row 2, Column 2
Row 3	Row 3, Column 1	Row 3, Column 2

Examples of database: PostgreSQL, MySQL, SQL Server, Access, Oracle, Sybase, DB2.

Coding involved: SQL [Structured Query Language]

See more at SQL.

References

[1] The term "column" also has equivalent application in other, more generic contexts. See e.g., Flat file database, Table (information).

Field

In computer science, data that has several parts can be divided into **fields**. Relational databases arrange data as sets of database records, also called rows. Each record consists of several *fields*; the fields of all records form the columns.

In object-oriented programming, *field* (also called *data member* or *member variable*) is the data encapsulated within a class or object. In the case of a regular field (also called *instance variable*), for each instance of the object there is an instance variable: for example, an `Employee` class has a `Name` field and there is one distinct name per employee. A static field (also called *class variable*) is one variable, which is shared by all instances.

Fixed length

Fields that contain a fixed number of bits are known as fixed length fields. A four byte field for example may contain a 31 bit binary integer plus a sign bit (32 bits in all). A 30 byte name field may contain a persons name typically padded with blanks at the end. The disadvantage of using fixed length fields is that some part of the field may be wasted but space is still required for the maximum length case. Also, where fields are omitted, padding for the missing fields is still required to maintain fixed start positions within a record for instance.

Variable length

A variable length field is not always the same physical size. Such fields are nearly always used for text fields that can be large, or fields that vary greatly in length. For example, a bibliographical database like PubMed has many small fields such as publication date and author name, but also has abstracts, which vary greatly in length. Reserving a fixed-length field of some length would be inefficient because it would enforce a maximum length on abstracts, and because space would be wasted in most records (particularly if many articles lacked abstracts entirely).

Database implementations commonly store varying-length fields in special ways, in order to make all the records of a given type have a uniform small size. Doing so can help performance. On the other hand, data in serialized forms such as stored in typical file systems, transmitted across networks, and so on usually uses quite different performance strategies. The choice depends on factors such as the total size of records, performance characteristics of the storage medium, and the expected patterns of access.

Database implementations typically store variable length fields in ways such as

- a sequence of characters or bytes, followed by an *end-marker* that is prohibited within the string itself. This makes it slower to access later fields in the same record because the later fields are not always at the same physical distance from the start of the record.
- a *pointer* to data in some other location, such as a URI, a file offset (and perhaps length), or a key identifying a record in some special place. This typically speeds up processes that don't need the contents of the variable length field(s), but slows processes that do.
- a *length prefix* followed by the specified number of characters or bytes. This avoids searches for an end-marker as in the first method, and avoids the loss of locality of reference as in the second method. On the other hand, it imposes a maximum length: the biggest number that can be represented using the (generally fixed length) prefix. In addition, records still vary in length, and must be traversed in order to reach later fields.

If a varying-length field is often empty, additional optimizations come into play.

References

Row

In the context of a relational database, a **row**—also called a **record** —represents a single, implicitly structured data item in a table. In simple terms, a database table can be thought of as consisting of *rows* and columns or fields. Each row in a table represents a set of related data, and every row in the table has the same structure.

For example, in a table that represents companies, each row would represent a single company. Columns might represent things like company name, company street address, whether the company is publicly held, its VAT number, etc.. In a table that represents *the association* of employees with departments, each row would associate one employee with one department.

In a less formal usage, e.g. for a database which is not formally relational, a record is equivalent to a row as described above, but is not usually referred to as a row.

The implicit structure of a row, and the meaning of the data values in a row, requires that the row be understood as providing a succession of data values, one in each column of the table. The row is then interpreted as a relvar composed of a set of tuples, with each tuple consisting of the two items: the name of the relevant column and the value this row provides for that column.

Each column expects a data value of a particular type. For example, one column might require a unique identifier, another might require text representing a person's name, another might require an integer representing hourly pay in cents.

-	**Column 1**	**Column 2**
Row (Record) 1	Row 1, Column (Field)1	Row 1, Column 2
Row 2	Row 2, Column 1	Row 2, Column 2
Row 3	Row 3, Column 1	Row 3, Column 2

Data type

In computer science and computer programming, a **data type** or simply **type** is a classification identifying one of various types of data, such as real, integer or Boolean, that determines the possible values for that type; the operations that can be done on values of that type; the meaning of the data; and the way values of that type can be stored.[1][2]

Overview

Data types are used within type systems, which offer various ways of defining, implementing and using them. Different type systems ensure varying degrees of type safety. Formally, a type can be defined as "any property of a programme we can determine without executing the program".[3]

Almost all programming languages explicitly include the notion of data type, though different languages may use different terminology. Common data types may include:

- integers,
- booleans,
- characters,
- floating-point numbers,
- alphanumeric strings.

For example, in the Java programming language, the "int" type represents the set of 32-bit integers ranging in value from -2,147,483,648 to 2,147,483,647, as well as the operations that can be performed on integers, such as addition, subtraction, and multiplication. Colors, on the other hand, are represented by three bytes denoting the amounts each of red, green, and blue, and one string representing that color's name; allowable operations include addition and subtraction, but not multiplication.

Most programming languages also allow the programmer to define additional data types, usually by combining multiple elements of other types and defining the valid operations of the new data type. For example, a programmer might create a new data type named "complex number" that would include real and imaginary parts. A data type also represents a constraint placed upon the interpretation of data in a type system, describing representation, interpretation and structure of values or objects stored in computer memory. The type system uses data type information to check correctness of computer programs that access or manipulate the data.

Most data types in statistics have comparable types in computer programming, and vice-versa, as shown in the following table:

Statistics	Computer programming
real-valued (interval scale)	floating-point
real-valued (ratio scale)	
count data (usually non-negative)	integer
binary data	Boolean
categorical data	enumerated type
random vector	list or array
random matrix	two-dimensional array
random tree	tree

Definition of a "type"

(Parnas, Shore & Weiss 1976) identified five definitions of a "type" that were used—sometimes implicitly—in the literature:

Syntactic

: A type is a purely syntactic label associated with a variable when it is declared. Such definitions of "type" do not give any semantic meaning to types.

Representation

: A type is defined in terms of its composition of more primitive types—often machine types.

Representation and behaviour

: A type is defined as its representation and a set of operators manipulating these representations.

Value space

: A type is a set of possible values which a variable can possess. Such definitions make it possible to speak about (disjoint) unions or Cartesian products of types.

Value space and behaviour

: A type is a set of values which a variable can possess and a set of functions that one can apply to these values.

The definition in terms of a representation was often done in imperative languages such as ALGOL and Pascal, while the definition in terms of a value space and behaviour was used in higher-level languages such as Simula and CLU.

Classes of data types

Primitive data types

Machine data types

All data in computers based on digital electronics is represented as bits (alternatives 0 and 1) on the lowest level. The smallest addressable unit of data is usually a group of bits called a byte (usually an octet, which is 8 bits). The unit processed by machine code instructions is called a word (as of 2011, typically 32 or 64 bits). Most instructions interpret the word as a binary number, such that a 32-bit word can represent unsigned integer values from 0 to $2^{32} - 1$ or signed integer values from -2^{31} to $2^{31} - 1$. Because of two's complement, the machine language and machine doesn't need to distinguish between these unsigned and signed data types for the most part.

There is a specific set of arithmetic instructions that use a different interpretation of the bits in word as a floating-point number.

Machine data types need to be *exposed* or made available in systems or low-level programming languages, allowing fine-grained control over hardware. The C programming language, for instance, supplies integer types of various widths, such as `short` and `long`. If a corresponding native type does not exist on the target platform, the compiler will break them down into code using types that do exist. For instance, if a 32-bit integer is requested on a 16 bit platform, the compiler will tacitly treat it as an array of two 16 bit integers.

Several languages allow binary and hexadecimal literals, for convenient manipulation of machine data.

In higher level programming, machine data types are often hidden or *abstracted* as an implementation detail that would render code less portable if exposed. For instance, a generic `numeric` type might be supplied instead of integers of some specific bit-width.

Boolean type

The Boolean type represents the values: true and false. Although only two values are possible, they are rarely implemented as a single binary digit for efficiency reasons. Many programming languages do not have an explicit boolean type, instead interpreting (for instance) 0 as false and other values as true.

Numeric types

Such as:

- The integer data types, or "whole numbers". May be subtyped according to their ability to contain negative values (e.g. `unsigned` in C and C++). May also have a small number of predefined subtypes (such as `short` and `long` in C/C++); or allow users to freely define subranges such as 1..12 (e.g. Pascal/Ada).
- Floating point data types, sometimes misleadingly called reals, contain fractional values. They usually have predefined limits on both their maximum values and their precision. These are often represented as decimal numbers.
- Fixed point data types are convenient for representing monetary values. They are often implemented internally as integers, leading to predefined limits.
- Bignum or arbitrary precision numeric types lack predefined limits. They are not primitive types, and are used sparingly for efficiency reasons.

Composite types

Composite types are derived from more than one primitive type. This can be done in a number of ways. The ways they are combined are called data structures. Composing a primitive type into a compound type generally results in a new type, e.g. *array-of-integer* is a different type to *integer*.

- An array stores a number of elements of the same type in a specific order. They are accessed using an integer to specify which element is required (although the elements may be of almost any type). Arrays may be fixed-length or expandable.
- Record (also called tuple or struct) Records are among the simplest data structures. A record is a value that contains other values, typically in fixed number and sequence and typically indexed by names. The elements of records are usually called *fields* or *members*.
- Union. A union type definition will specify which of a number of permitted primitive types may be stored in its instances, e.g. "float or long integer". Contrast with a record, which could be defined to contain a float *and* an integer; whereas, in a union, there is only one value at a time.
- A tagged union (also called a variant, variant record, discriminated union, or disjoint union) contains an additional field indicating its current type, for enhanced type safety.
- A set is an abstract data structure that can store certain values, without any particular order, and no repeated values. Values themselves are not retrieved from sets, rather one tests a value for membership to obtain a boolean "in" or "not in".
- An object contains a number of data fields, like a record, and also a number of program code fragments for accessing or modifying them. Data structures not containing code, like those above, are called plain old data structure.

Many others are possible, but they tend to be further variations and compounds of the above.

Enumerations

The enumerated type. This has values which are different from each other, and which can be compared and assigned, but which do not necessarily have any particular concrete representation in the computer's memory; compilers and interpreters can represent them arbitrarily. For example, the four suits in a deck of playing cards may be four enumerators named *CLUB*, *DIAMOND*, *HEART*, *SPADE*, belonging to an enumerated type named *suit*. If a variable *V* is declared having *suit* as its data type, one can assign any of those four values to it. Some implementations allow programmers to assign integer values to the enumeration values, or even treat them as type-equivalent to integers.

String and text types

Such as:

- Alphanumeric character. A letter of the alphabet, digit, blank space, punctuation mark, etc.
- Alphanumeric strings, a sequence of characters. They are typically used to represent words and text.

Character and string types can store sequences of characters from a character set such as ASCII. Since most character sets include the digits, it is possible to have a numeric string, such as `"1234"`. However, many languages would still treat these as belonging to a different type to the numeric value `1234`.

Character and string types can have different subtypes according to the required character "width". The original 7-bit wide ASCII was found to be limited, and superseded by 8 and 16-bit sets, which can encode a wide variety of non-Latin alphabets (Hebrew, Chinese) and other symbols. Strings may be either stretch-to-fit or of fixed size, even in the same programming language. They may also be subtyped by their maximum size.

Note: strings are not primitive in all languages, for instance C: they may be composed from arrays of characters.

Other types

Types can be based on, or derived from, the basic types explained above. In some languages, such as C, functions have a type derived from the type of their return value.

Pointers and references

The main non-composite, derived type is the pointer, a data type whose value refers directly to (or "points to") another value stored elsewhere in the computer memory using its address. It is a primitive kind of reference. (In everyday terms, a page number in a book could be considered a piece of data that refers to another one). Pointers are often stored in a format similar to an integer; however, attempting to dereference or "look up" a pointer whose value was never a valid memory address would cause a program to crash. To ameliorate this potential problem, pointers are considered a separate type to the type of data they point to, even if the underlying representation is the same.

Abstract data types

Any type that does not specify an implementation is an abstract data type. For instance, a stack (which is an abstract type) can be implemented as an array (a contiguous block of memory containing multiple values), or as a linked list (a set of non-contiguous memory blocks linked by pointers).

Abstract types can be handled by code that does not know or "care" what underlying types are contained in them. Programming that is agnostic about concrete data types is called generic programming. Arrays and records can also contain underlying types, but are considered concrete because they specify how their contents or elements are laid out in memory.

Examples include:

- A queue is a first-in first-out list. Variations are Deque and Priority queue.
- A set can store certain values, without any particular order, and with no repeated values.
- A stack is a last-in, first out.

- A tree is a hierarchical structure.
- A graph.
- A hash or dictionary or map or Map/Associative array/Dictionary is a more flexible variation on a record, in which name-value pairs can be added and deleted freely.
- A smart pointer is the abstract counterpart to a pointer. Both are kinds of reference

Utility types

For convenience, high-level languages may supply ready-made "real world" data types, for instance *times*, *dates* and *monetary values* and *memory*, even where the language allows them to be built from primitive types.

Type systems

A type system associates types with each computed value. By examining the flow of these values, a type system attempts to prove that no *type errors* can occur. The type system in question determines what constitutes a type error, but a type system generally seeks to guarantee that operations expecting a certain kind of value are not used with values for which that operation does not make sense.

A compiler may use the static type of a value to optimize the storage it needs and the choice of algorithms for operations on the value. In many C compilers the `float` data type, for example, is represented in 32 bits, in accord with the IEEE specification for single-precision floating point numbers. They will thus use floating-point-specific microprocessor operations on those values (floating-point addition, multiplication, etc.).

The depth of type constraints and the manner of their evaluation affect the *typing* of the language. A programming language may further associate an operation with varying concrete algorithms on each type in the case of type polymorphism. Type theory is the study of type systems, although the concrete type systems of programming languages originate from practical issues of computer architecture, compiler implementation, and language design.

Type systems may be variously static or dynamic, strong or weak typing, and so forth.

References

[1] http://foldoc.org/data+type

[2] Shaffer, C.A. *Data Structures and Algorthms*, 1.2

[3] *Programming Languages: Application and Interpretation*, Shriram Krishnamurthi, Brown University

Further reading

- Parnas, David L.; Shore, John E.; Weiss, David (1976). "Abstract types defined as classes of variables". *Proceedings of the 1976 conference on Data : Abstraction, definition and structure*: 149–154. doi: 10.1145/800237.807133 (http://dx.doi.org/10.1145/800237.807133).
- Cardelli, Luca; Wegner, Peter (December 1985). "On Understanding Types, Data Abstraction, and Polymorphism" (http://lucacardelli.name/Papers/OnUnderstanding.A4.pdf). *ACM Computing Surveys* (New York, NY, USA: ACM) **17** (4): 471–523. doi: 10.1145/6041.6042 (http://dx.doi.org/10.1145/6041.6042). ISSN 0360-0300 (http://www.worldcat.org/issn/0360-0300).
- Cleaveland, J. Craig (1986). *An Introduction to Data Types*. Addison-Wesley. ISBN 0201119404.

Statements

Select

The SQL **SELECT** statement returns a result set of records from one or more tables.

A **SELECT** statement retrieves zero or more rows from one or more database tables or database views. In most applications, `SELECT` is the most commonly used Data Manipulation Language (DML) command. As SQL is a declarative programming language, `SELECT` queries specify a result set, but do not specify how to calculate it. The database translates the query into a "query plan" which may vary between executions, database versions and database software. This functionality is called the "query optimizer" as it is responsible for finding the best possible execution plan for the query, within applicable constraints.

The SELECT statement has many optional clauses:

- `WHERE` specifies which rows to retrieve.
- `GROUP BY` groups rows sharing a property so that an aggregate function can be applied to each group.
- `HAVING` selects among the groups defined by the GROUP BY clause.
- `ORDER BY` specifies an order in which to return the rows.
- `AS` provides an alias which can be used to temporarily rename tables or columns.

Examples

Table "T"	Query	Result
C1 | **C2** 1 | a 2 | b	`SELECT * FROM T;`	**C1** | **C2** 1 | a 2 | b
C1 | **C2** 1 | a 2 | b	`SELECT C1 FROM T;`	**C1** 1 2
C1 | **C2** 1 | a 2 | b	`SELECT * FROM T WHERE C1 = 1;`	**C1** | **C2** 1 | a
C1 | **C2** 1 | a 2 | b	`SELECT * FROM T ORDER BY C1 DESC;`	**C1** | **C2** 2 | b 1 | a

Given a table T, the *query* `SELECT * FROM T` will result in all the elements of all the rows of the table being shown.

With the same table, the query `SELECT C1 FROM T` will result in the elements from the column C1 of all the rows of the table being shown. This is similar to a *projection* in Relational algebra, except that in the general case, the result may contain duplicate rows. This is also known as a Vertical Partition in some database terms, restricting query output to view only specified fields or columns.

With the same table, the query `SELECT * FROM T WHERE C1 = 1` will result in all the elements of all the rows where the value of column C1 is '1' being shown — in Relational algebra terms, a *selection* will be performed, because of the WHERE clause. This is also known as a Horizontal Partition, restricting rows output by a query according to specified conditions.

With more than one table, the result set will be every combination of rows. So if two tables are T1 and T2, `SELECT * FROM T1, T2` will result in every combination of T1 rows with every T2 rows. E.g., if T1 has 3 rows and T2 has 5 rows, then 15 rows will result.

Limiting result rows

Often it is convenient to indicate a maximum number of rows that are returned. This can be used for testing or to prevent consuming excessive resources if the query returns more information than expected. The approach to do this often varies per vendor.

In ISO SQL:2003, result sets may be limited by using

- cursors, or
- By introducing *SQL window function* to the SELECT-statement

ISO SQL:2008 introduced the `FETCH FIRST` clause.

ROW_NUMBER() window function

`ROW_NUMBER() OVER` may be used for a *simple table* on the returned rows, e.g. to return no more than ten rows:

```
SELECT * FROM
( SELECT
    ROW_NUMBER() OVER (ORDER BY sort_key ASC) AS row_number,
    columns
  FROM tablename
) AS foo
WHERE row_number <= 11
```

ROW_NUMBER can be non-deterministic: if *sort_key* is not unique, each time you run the query it is possible to get different row numbers assigned to any rows where *sort_key* is the same. When *sort_key* is unique, each row will always get a unique row number.

RANK() window function

The `RANK() OVER` window function acts like ROW_NUMBER, but may return more than *n* rows in case of tie conditions, e.g. to return the top-10 youngest persons:

```
SELECT * FROM (
  SELECT
    RANK() OVER (ORDER BY age ASC) AS ranking,
    person_id,
    person_name,
    age
  FROM person
```

```
)AS foo
WHERE ranking <= 10
```

The above code could return more than ten rows, e.g. if there are two people of the same age, it could return eleven rows.

FETCH FIRST clause

Since ISO SQL:2008 results limits can be specified as in the following example using the `FETCH FIRST` clause.

```
SELECT * FROM T FETCH FIRST 10 ROWS ONLY
```

This clause currently is supported by CA DATACOM/DB 11,IBM DB2, Sybase SQL Anywhere, PostgreSQL, EffiProz, H2, HSQLDB version 2.0, Microsoft SQL Server 2012 and Oracle 12c.

Result limits

Not all DBMSs support the mentioned window functions, and non-standard syntax has to be used. Below, variants of the *simple limit* query for different DBMSes are listed:

`SET ROWCOUNT 10` `SELECT * FROM T`	MS SQL Server (This also works on Microsoft SQL Server 6.5 while the **Select top 10 * from T** does not)
`SELECT * FROM T LIMIT 10 OFFSET 20`	Netezza, MySQL, Sybase SQL Anywhere, PostgreSQL (also supports the standard, since version 8.4), SQLite, HSQLDB, H2, Vertica, Polyhedra
`SELECT * from T WHERE ROWNUM <= 10`	Oracle
`SELECT FIRST 10 * from T`	Ingres
`SELECT FIRST 10 * FROM T order by a`	Informix
`SELECT SKIP 20 FIRST 10 * FROM T order by c, d`	Informix (row numbers are filtered after order by is evaluated. SKIP clause was introduced in a v10.00.xC4 fixpack)
`SELECT TOP 10 * FROM T`	MS SQL Server, Sybase ASE, MS Access, Sybase IQ, Teradata
`SELECT TOP 10 START AT 20 * FROM T`	Sybase SQL Anywhere (also supports the standard, since version 9.0.1)
`SELECT FIRST 10 SKIP 20 * FROM T`	Interbase, Firebird
`SELECT * FROM T ROWS 20 TO 30`	Firebird (since version 2.1)
`SELECT * FROM T` `WHERE ID_T > 10 FETCH FIRST 10 ROWS ONLY`	DB2
`SELECT * FROM T` `WHERE ID_T > 20 FETCH FIRST 10 ROWS ONLY`	DB2 (new rows are filtered after comparing with key column of table T)

Hierarchical query

Some databases provide specialised syntax for hierarchical data.

Window function

A window function in SQL:2003 is an aggregate function applied to a partition of the result set.

For example,

```
sum(population) OVER( PARTITION BY city )
```

calculates the sum of the populations of all rows having the same *city* value as the current row.

Partitions are specified using the **OVER** clause which modifies the aggregate. Syntax:

```
<OVER_CLAUSE> :: =
   OVER ( [ PARTITION BY <expr>, ... ]
          [ ORDER BY <expression> ] )
```

The OVER clause can partition and order the result set. Ordering is used for order-relative functions such as row_number.

Query evaluation ANSI

The processing of a SELECT statement according to ANSI SQL would be the following:[1]

```
select g.*
from users u inner join groups g on g.Userid = u.Userid
where u.LastName = 'Smith'
and u.FirstName = 'John'
```

2. the FROM clause is evaluated, a cross join or Cartesian product is produced for the first two tables in the FROM clause resulting in a virtual table as Vtable1
3. the ON clause is evaluated for vtable1; only records which meet the join condition g.Userid = u.Userid are inserted into Vtable2
4. If an outer join is specified, records which were dropped from vTable2 are added into VTable 3, for instance if the above query were:

```
select u.*
from users u left join groups g on g.Userid = u.Userid
where u.LastName = 'Smith'
and u.FirstName = 'John'
```

   all users who did not belong to any groups would be added back into Vtable3
5. the WHERE clause is evaluated, in this case only group information for user John Smith would be added to vTable4
6. the GROUP BY is evaluated; if the above query were:

```
select g.GroupName, count(g.*) as NumberOfMembers
from users u inner join groups g on g.Userid = u.Userid
group by GroupName
```

   vTable5 would consist of members returned from vTable4 arranged by the grouping, in this case the GroupName

7. the HAVING clause is evaluated for groups for which the HAVING clause is true and inserted into vTable6. For example:

```
select g.GroupName, count(g.*) as NumberOfMembers
from users u inner join groups g on g.Userid = u.Userid
group by GroupName
having count(g.*) > 5
```

8. the SELECT list is evaluated and returned as Vtable 7
9. the DISTINCT clause is evaluated; duplicate rows are removed and returned as Vtable 8
10. the ORDER BY clause is evaluated, ordering the rows and returning VCursor9. This is a cursor and not a table because ANSI defines a cursor as an ordered set of rows (not relational).

Generating Data in T-SQL

Method to generate data based on the union all

```
select 1 a, 1 b union all
select 1, 2 union all
select 1, 3 union all
select 2, 1 union all
select 5, 1
```

Enhancement made in SQL Server 2008 release

```
select *
from (values (1, 1), (1, 2), (1, 3), (2, 1), (5, 1)) as x(a, b)
```

References

[1] Inside Microsoft SQL Server 2005: T-SQL Querying by Itzik Ben-Gan, Lubor Kollar, and Dejan Karka

Sources

- Horizontal & Vertical Partitioning, Microsoft SQL Server 2000 Books Online

External links

- Windowed Tables and Window function in SQL (http://wwwlgis.informatik.uni-kl.de/cms/fileadmin/courses/SS2008/NEDM/RDDM.Chapter.06.Windows_and_Query_Functions_in_SQL.pdf), Stefan Deßloch
- Oracle SELECT Syntax. (http://download.oracle.com/docs/cd/B14117_01/server.101/b10759/statements_10002.htm)
- Firebird SELECT Syntax. (http://www.firebirdsql.org/rlsnotesh/rlsnotes210.html#rnfb20x-dml-select-syntax)
- Mysql SELECT Syntax. (http://dev.mysql.com/doc/refman/5.1/en/select.html)
- Postgres SELECT Syntax. (http://www.postgresql.org/docs/current/static/sql-select.html)
- SQLite SELECT Syntax. (http://www.sqlite.org/lang_select.html)

Result set

An SQL **result set** is a set of rows from a database, as well as metadata about the query such as the column names, and the types and sizes of each column. Depending on the database system, the number of rows in the result set may or may not be known. Usually, this number is not known up front because the result set is built on-the-fly. Precomputations often impose undesired performance impacts.Wikipedia:Disputed statement

A result set is effectively a table. The `ORDER BY` clause can be used in a query to impose a certain sort condition on the rows. Without that clause, there is no guarantee whatsoever on the order in which the rows are returned.

Synonym (database)

A **synonym** is an alias or alternate name for a table, view, sequence, or other schema object. They are used mainly to make it easy for users to access database objects owned by other users. They hide the underlying object's identity and make it harder for a malicious program or user to target the underlying object. Because a synonym is just an alternate name for an object, it requires no storage other than its definition. When an application uses a synonym, the DBMS forwards the request to the synonym's underlying base object. By coding your programs to use synonyms instead of database object names, you insulate yourself from any changes in the name, ownership, or object locations. If you frequently refer to a database object that has a long name, you might appreciate being able to refer to it with a shorter name without having to rename it and alter the code referring to it.

Synonyms are very powerful from the point of view of allowing users access to objects that do not lie within their schema. All synonyms have to be created explicitly with the CREATE SYNONYM command and the underlying objects can be located in the same database or in other databases that are connected by database links[*clarify*].

There are two major uses of synonyms:

- **Object invisibility**: Synonyms can be created to keep the original object hidden from the user.
- **Location invisibility**: Synonyms can be created as aliases for tables and other objects that are not part of the local database.

When you create a table or a procedure, it is created in your schema, and other users can access it only by using your schema name as a prefix to the object's name. The way around for this is for the schema owner creates a synonym with the same name as the table name.

Public synonyms

Public synonyms are owned by special schema in the Oracle Database called PUBLIC. As mentioned earlier, public synonyms can be referenced by all users in the database. Public synonyms are usually created by the application owner for the tables and other objects such as procedures and packages so the users of the application can see the objects.

The following code shows how to create a public synonym for the employee table:

```
CREATE PUBLIC SYNONYM employees for hr.employees;
```

Now any user can see the table by just typing the original table name. If you wish, you could provide a different table name for that table in the CREATE SYNONYM statement. Remember that the DBA must explicitly grant the CREATE PUBLIC SYNONYM privilege to the user HR before HR can create any public synonyms. Just because you can see a table through public (or private) synonym doesn't mean that you can also perform SELECT, INSERT, UPDATE or DELETE operations on the table. To be able to perform those operations, a user needs specific privileges for the underlying object, either directly or through roles from the application owner.

Private synonyms

A private synonym is a synonym within a database schema that a developer typically uses to mask the true name of a table, view stored procedure, or other database object in an application schema.

Private synonyms, unlike public synonyms, can be referenced only by the schema that owns the table or object. You may want to create private synonyms when you want to refer to the same table by different contexts. Private synonym overrides public synonym definitions. You create private synonyms the same way you create public synonyms, but you omit the PUBLIC keyword in the CREATE statement.

The following example shows how to create a private synonym called addresses for the locations table. Note that once you create the private synonym, you can refer to the synonym exactly as you would the original table name.

```
CREATE SYNONYM addresses FOR hr.locations;
```

Drop a synonym

Synonyms, both private and public, are dropped in the same manner by using the DROP SYNONYM command, but there is one important difference. If you are dropping a public synonym; you need to add the keyword PUBLIC after the keyword DROP.

```
DROP SYNONYM addresses;
```

The ALL_SYNONYMS (or DBA_SYNONYMS) view provides information on all synonyms in your database.

References

- Palinski, John Adolph (2002). *Oracle SQL and PL/SQL Handbook: A Guide for Data Administrators, Developers, and Business Analysts*. Addison–Wesley. ISBN 978-0-201-75294-6.
- Gennick, Jonathan (2004). *Oracle SQL*Plus: the definitive guide*. O'Reilly Media. ISBN 978-0-596-00746-1.
- Alapati, Sam R (2005). *Expert Oracle Database 10g Administration*. Apress. ISBN 978-1-59059-451-3.
- Bobrowski, Steve. *Hands-on Oracle Database 10g Express Edition for Windows*. McGraw-Hill. ISBN 978-0-07-226331-2.

Alias (SQL)

An **alias** is a feature of SQL that is supported by most, if not all, relational database management systems (RDBMSs). Aliases provide database administrators, as well as other database users, with the ability to reduce the amount of code required for a query, and to make queries generally simpler to understand. In addition, aliasing can be used as an obfuscation technique to protect the real names of database fields.

In SQL, you can alias both tables themselves, which is called a Correlation Name,[1] or columns. A programmer can temporarily assign another name to a table or column (for the duration of the SELECT query) by using an alias. In other words, it does not actually rename the column or table. This is often useful when either tables or their columns have very long or complex names. An alias name could be anything, but usually it is kept short. For example, it might be common to use a table alias such as "pi" for a table named "price_information".

The general syntax of an alias is `SELECT * FROM table_name [AS] alias_name`. Note that the AS keyword is completely optional and is usually kept for readability purposes. Here is some sample data that the queries below will be referencing:

Department Table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Using a table alias:

```
SELECT D.DepartmentName FROM Department AS D
```

We can also write the same query like this (Note that the AS clause is omitted this time):

```
SELECT D.DepartmentName FROM Department D
```

A column alias is similar:

```
SELECT d.DepartmentId AS Id, d.DepartmentName AS Name FROM Department d
```

In the returned result sets, the data shown above would be returned, with the only exception being "DepartmentID" would show up as "Id", and "DepartmentName" would show up as "Name".

Also, if only one table is being selected and the query is not using table joins, it is permissible to omit the table name or table alias from the column name in the SELECT statement. Example as follows:

```
SELECT DepartmentId AS Id, DepartmentName AS Name FROM Department d
```

References

[1] ANSI Standard SQL - Foundation Document - Date: 2010-10-14

Insert

An SQL **INSERT** statement adds one or more records to any single table in a relational database.

Basic form

Insert statements have the following form:

- INSERT INTO *table* (*column1* [, *column2*, *column3* ...]) VALUES (*value1* [, *value2*, *value3* ...])

The number of columns and values must be the same. If a column is not specified, the default value for the column is used. The values specified (or implied) by the `INSERT` statement must satisfy all the applicable constraints (such as primary keys, `CHECK` constraints, and `NOT NULL` constraints). If a syntax error occurs or if any constraints are violated, the new row is not added to the table and an error returned instead.

Example:

```
INSERT INTO phone_book (name, number) VALUES ('John Doe', '555-1212');
```

Shorthand may also be used, taking advantage of the order of the columns when the table was created. It is not required to specify all columns in the table since any other columns will take their default value or remain null:

- INSERT INTO *table* VALUES (*value1*, [*value2*, ...])

Example for inserting data into 2 columns in the phone_book table and ignoring any other columns which may be after the first 2 in the table.

```
INSERT INTO phone_book VALUES ('John Doe', '555-1212');
```

Advanced forms

Multirow inserts

A SQL feature (since SQL-92) is the use of *row value constructors* to insert multiple rows at a time in a single SQL statement:

```
INSERT INTO table (column-a, [column-b, ...])
VALUES ('value-1a', ['value-1b', ...]),
       ('value-2a', ['value-2b', ...]),
       ...
```

This feature is supported by DB2, SQL Server (since version 10.0 - i.e. 2008), PostgreSQL (since version 8.2), MySQL, sqlite (since version 3.7.11) and H2.

Example (assuming that 'name' and 'number' are the only columns in the 'phone_book' table):

```
INSERT INTO phone_book VALUES ('John Doe', '555-1212'), ('Peter Doe',
'555-2323');
```

which may be seen as a shorthand for the two statements

```
INSERT INTO phone_book VALUES ('John Doe', '555-1212');
INSERT INTO phone_book VALUES ('Peter Doe', '555-2323');
```

Note that the two separate statements may have different semantics (especially with respect to statement triggers) and may not provide the same performance as a single multi-row insert.

To insert multiple rows in MS SQL you can use such a construction:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212'
UNION ALL
SELECT 'Peter Doe', '555-2323';
```

Note that this is not a valid SQL statement according to the SQL standard (SQL:2003) due to the incomplete subselect clause.

To do the same in Oracle use the DUAL table, which always consists of a single row only:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM DUAL
UNION ALL
SELECT 'Peter Doe','555-2323' FROM DUAL
```

A standard-conforming implementation of this logic shows the following example, or as shown above:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM LATERAL ( VALUES (1) ) AS t(c)
UNION ALL
SELECT 'Peter Doe','555-2323' FROM LATERAL ( VALUES (1) ) AS t(c)
```

Oracle PL/SQL supports the "INSERT ALL" statement, where multiple insert statements are terminated by a SELECT:

```
INSERT ALL
INTO phone_book VALUES ('John Doe', '555-1212')
INTO phone_book VALUES ('Peter Doe', '555-2323')
SELECT * FROM DUAL;
```

In Firebird inserting multiple rows can be achieved like this:

```
INSERT INTO phone_book ("name", "number")
SELECT 'John Doe', '555-1212' FROM RDB$DATABASE
UNION ALL
SELECT 'Peter Doe', '555-2323' FROM RDB$DATABASE;
```

Firebird however restricts the number of rows than can be inserted in this way, since there is a limit to the number contexts that can be used in a single query.

Copying rows from other tables

An INSERT statement can also be used to retrieve data from other tables, modify it if necessary and insert it directly into the table. All this is done in a single SQL statement that does not involve any intermediary processing in the client application. A subselect is used instead of the VALUES clause. The subselect can contain joins, function calls, and it can even query the same table into which the data is inserted. Logically, the select is evaluated before the actual insert operation is started. An example is given below.

```
INSERT INTO phone_book2
SELECT *
FROM   phone_book
WHERE  name IN ('John Doe', 'Peter Doe')
```

A variation is needed when some of the data from the source table is being inserted into the new table, but not the whole record. (Or when the tables' schemas are not the same.)

```
INSERT INTO phone_book2 ( [name], [phoneNumber] )
SELECT [name], [phoneNumber]
FROM   phone_book
WHERE  name IN ('John Doe', 'Peter Doe')
```

The `SELECT` statement produces a (temporary) table, and the schema of that temporary table must match with the schema of the table where the data is inserted into.

Retrieving the key

Database designers that use a surrogate key as the primary key for every table will run into the occasional scenario where they need to automatically retrieve the database generated primary key from an SQL INSERT statement for use in another SQL statements. Most systems do not allow SQL INSERT statements to return row data. Therefore, it becomes necessary to implement a workaround in such scenarios. Common implementations include:

- Using a database-specific stored procedure that generates the surrogate key, performs the INSERT operation, and finally returns the generated key. For example, in Microsoft SQL Server, the key is retrieved via the SCOPE_IDENTITY() special function, while in SQLite the function is named last_insert_rowid().
- Using a database-specific SELECT statement on a temporary table containing last inserted row(s). DB2 implements this feature in the following way:

```
SELECT *
FROM NEW TABLE (
    INSERT INTO phone_book
    VALUES ( 'Peter Doe','555-2323' )
) AS t
```

DB2 for z/OS implements this feature in the following way.

```
SELECT EMPNO, HIRETYPE, HIREDATE
FROM FINAL TABLE (
    INSERT INTO EMPSAMP (NAME, SALARY, DEPTNO, LEVEL)
    VALUES('Mary Smith', 35000.00, 11, 'Associate')
);
```

- Using a SELECT statement after the INSERT statement with a database-specific function that returns the generated primary key for the most recently inserted row. For example, LAST_INSERT_ID() for MySQL.
- Using a unique combination of elements from the original SQL INSERT in a subsequent SELECT statement.
- Using a GUID in the SQL INSERT statement and retrieving it in a SELECT statement.
- Using the OUTPUT clause in the SQL INSERT statement for MS-SQL Server 2005 and MS-SQL Server 2008.
- Using an INSERT statement with RETURNING clause for Oracle.

```
INSERT INTO phone_book VALUES ( 'Peter Doe','555-2323' )
RETURNING phone_book_id INTO v_pb_id
```

- Using an INSERT statement with RETURNING clause for PostgreSQL (since 8.2). The returned list is identical to the result of a SELECT.

Firebird has the same syntax in Data Modification Language statements (DSQL); the statement may add at most one row. In stored procedures, triggers and execution blocks (PSQL) the aforementioned Oracle syntax is used.

```
INSERT INTO phone_book VALUES ( 'Peter Doe','555-2323' )
RETURNING phone_book_id
```

- Using the IDENTITY() function in H2 returns the last identity inserted.

```
SELECT IDENTITY();
```

Triggers

If triggers are defined on the table on which the INSERT statement operates, those triggers are evaluated in the context of the operation. BEFORE INSERT triggers allow the modification of the values that shall be inserted into the table. AFTER INSERT triggers cannot modify the data anymore, but can be used to initiate actions on other tables, for example- to implement auditing mechanism.

References

External links

- Oracle SQL INSERT statement (Oracle Database SQL Language Reference, 11g Release 2 (11.2) on oracle.com) (http://download.oracle.com/docs/cd/E11882_01/server.112/e10592/statements_9014.htm)
- Microsoft Access Append Query Examples and SQL INSERT Query Syntax (http://www.fmsinc.com/MicrosoftAccess/query/snytax/append-query.html)

Update

An SQL **UPDATE** statement changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a condition.

The `UPDATE` statement has the following form:[1]

`UPDATE` *table_name* **`SET`** *column_name* = *value* [, *column_name* = *value* ...] [**`WHERE`** *condition*]

For the `UPDATE` to be successful, the user must have data manipulation privileges (`UPDATE` privilege) on the table or column and the updated value must not conflict with all the applicable constraints (such as primary keys, unique indexes, `CHECK` constraints, and `NOT NULL` constraints).

In some databases, such as PostgreSQL, when a FROM clause is present, what essentially happens is that the target table is joined to the tables mentioned in the fromlist, and each output row of the join represents an update operation for the target table. When using FROM, one should ensure that the join produces at most one output row for each row to be modified. In other words, a target row shouldn't join to more than one row from the other table(s). If it does, then only one of the join rows will be used to update the target row, but which one will be used is not readily predictable.[2]

Because of this indeterminacy, referencing other tables only within sub-selects is safer, though often harder to read and slower than using a join.

Examples

Set the value of column *C1* in table *T* to 1, only in those rows where the value of column *C2* is "a".

```
UPDATE T
   SET C1 = 1
 WHERE C2 = 'a'
```

In table *T*, set the value of column *C1* to 9 and the value of *C3* to 4 for all rows for which the value of column *C2* is "a".

```
UPDATE T
   SET C1 = 9,
       C3 = 4
 WHERE C2 = 'a'
```

Increase value of column *C1* by 1 if the value in column *C2* is "a".

```
UPDATE T
   SET C1 = C1 + 1
 WHERE C2 = 'a'
```

Prepend the value in column *C1* with the string "text" if the value in column *C2* is "a".

```
UPDATE T
   SET C1 = 'text' || C1
 WHERE C2 = 'a'
```

Set the value of column *C1* in table *T1* to 2, only if the value of column *C2* is found in the sublist of values in column *C3* in table *T2* having the column *C4* equal to 0.

```
UPDATE T1
   SET C1 = 2
 WHERE C2 IN ( SELECT C3
                 FROM T2
                WHERE C4 = 0)
```

One may also update multiple columns in a single update statement:

```
UPDATE T
   SET C1 = 1,
       C2 = 2
```

Complex conditions and JOINs are also possible:

```
UPDATE T
   SET A = 1
 WHERE C1 = 1
   AND C2 = 2
```

Some databases allow the non-standard use of the FROM clause:

```
UPDATE a
   SET a.[updated_column] = updatevalue
  FROM articles a
```

```
       JOIN classification c
         ON a.articleID = c.articleID
 WHERE c.classID = 1
```

Or on Oracle systems (assuming there is an index on classification.articleID):

```
UPDATE
(
  SELECT *
    FROM articles
    JOIN classification
      ON articles.articleID = classification.articleID
   WHERE classification.classID = 1
)
SET [updated_column] = updatevalue
```

Potential issues

See Halloween Problem. It is possible for certain kinds of UPDATE statements to become an infinite loop when the WHERE clause and one or more SET clauses may utilize an intertwined index.

References

[1] http://dev.mysql.com/doc/refman/5.0/en/update.html simplified from this page
[2] http://www.postgresql.org/docs/8.1/static/sql-update.html

Merge

A relational database management system uses SQL `MERGE` (also called *upsert*) statements to `INSERT` new records or `UPDATE` existing records depending on whether or not a condition matches. It was officially introduced in the SQL:2003 standard, and expanded in the SQL:2008 standard.

Usage

```
MERGE INTO tablename USING table_reference ON (condition)
  WHEN MATCHED THEN
  UPDATE SET column1 = value1 [, column2 = value2 ...]
  WHEN NOT MATCHED THEN
  INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...
```

Right join is employed over the Target (the INTO table) and the Source (the USING table / view / sub-query). That is:

- If rows present in the Source but missing from the Target do run the action then specifically the NOT MATCHED action
- If rows missing from the Source and present in Target are ignored then no action is performed on the Target.

If multiple Source rows match a given Target row, an error is mandated by SQL:2003 standards. You cannot update a Target row multiple times with a MERGE statement

Implementations

Database management systems Oracle Database, DB2, Teradata, EXASOL and MS SQL support the standard syntax. Some also add non-standard SQL extensions.

Synonymous

Some database implementations adopted the term "**Upsert**" (a portmanteau of *update* and *insert*) to a database statement, or combination of statements, that inserts a record to a table in a database if the record does not exist or, if the record already exists, updates the existing record. It is also used to abbreviate the "MERGE" equivalent pseudo-code.

It is used in Microsoft SQL Azure and MongoDB.

MongoDB provides an atomic upsert operation, which creates a new document by combining the criteria for the update with the fields to change.

Example

Suppose a collection is used to track the number of times each page of a website is viewed. Upserts can be used to avoid "seeding" the collection with all possible pages in advance. The collection starts off empty:

```
 > db.pages.find()
```

On each page view, the page's document is created if it doesn't exist yet and its views are incremented if it does.

```
 > db.pages.update({"_id" : "http://www.example.com"}, {"$inc" :
{"views" : 1}}, {upsert : true})
 > db.pages.find()
 { "_id" : "http://www.example.com", "views" : 1 }
 > db.pages.update({"_id" : "http://www.example.com"}, {"$inc" :
{"views" : 1}}, {upsert : true})
 { "_id" : "http://www.example.com", "views" : 2 }
```

Other non-standard implementations

Some other database management systems support this, or very similar behavior, through their own, non-standard SQL extensions.

MySQL, for example, supports the use of `INSERT ... ON DUPLICATE KEY UPDATE` syntax[1] which can be used to achieve a similar effect with the limitation that the join between target and source has to be made only on PRIMARY KEY or UNIQUE constraints, which is not required in the ANSI/ISO standard. It also supports `REPLACE INTO` syntax,[2] which first attempts an insert, and if that fails, deletes the row, if exists, and then inserts the new one. There is also an `IGNORE` clause for the `INSERT` statement, which tells the server to ignore "duplicate key" errors and go on (existing rows will not be inserted or updated, but all new rows will be inserted).

SQLite's `INSERT OR REPLACE INTO` works similarly. It also supports `REPLACE INTO` as an alias for compatibility with MySQL.

Firebird supports `MERGE INTO` though fails at throwing error when multiple Source data. Additionally a single-row version, `UPDATE OR INSERT INTO tablename (columns) VALUES (values) [MATCHING (columns)]`, but the latter does not give you the option to take different actions on insert vs. update (e.g. setting a new sequence value only for new rows, not for existing ones.)

IBM DB2 extends syntax with multiple `WHEN MATCHED` and `WHEN NOT MATCHED` clauses, distinguishing them with `... AND some-condition` guards.

Microsoft SQL extends with supporting guards and also with supporting Left Join via `WHEN NOT MATCHED BY SOURCE` clauses.

References

[1] MySQL :: MySQL 5.1 Reference Manual :: 12.2.4.3 INSERT ... ON DUPLICATE KEY UPDATE Syntax (http://dev.mysql.com/doc/refman/5.1/en/insert-on-duplicate.html)

[2] MySQL 5.1 Reference Manual: 11.2.6 REPLACE Syntax (http://dev.mysql.com/doc/refman/5.1/en/replace.html)

- Hsu, Leo; Obe, Regina (May 18, 2008). "Cross Compare of SQL Server, MySQL, and PostgreSQL" (http://www.postgresonline.com/journal/archives/51-Cross-Compare-of-SQL-Server,-MySQL,-and-PostgreSQL.html). *Postgres OnLine Journal*. Retrieved 8 October 2010.
- Chodorow, Kristina; Mike Dirolf (September 2010). *MongoDB: The Definitive Guide*. O'Reilly. ISBN 978-1-449-38156-1.

External links

- Oracle 11g Release 2 documentation (http://docs.oracle.com/cd/E18283_01/timesten.112/e13070/state.htm#insertedID46) on **`MERGE`**
- Firebird 2.1 documentation (http://www.firebirdsql.org/refdocs/langrefupd21-merge.html) on **`MERGE`**
- DB2 v9 MERGE statement (http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0010873.htm)
- SQL Server 2008 documentation (http://msdn.microsoft.com/en-us/library/bb510625.aspx)
- H2 (1.2) SQL Syntax page (http://www.h2database.com/html/grammar.html#merge)

Delete

In the database structured query language (SQL), the **DELETE** statement removes one or more records from a table. A subset may be defined for deletion using a condition, otherwise all records are removed. Some DBMSs, like MySQL, allow to delete rows from multiple tables with one DELETE statement (this is sometimes called multi-table DELETE).

Usage

The `DELETE` statement follows the syntax:

> **`DELETE`** `FROM` *table_name* [**`WHERE`** condition];

Any rows that match the `WHERE` condition will be removed from the table. If the `WHERE` clause is omitted, all rows in the table are removed. The `DELETE` statement should thus be used with caution.

The `DELETE` statement does not return any rows; that is, it will not generate a result set.

Executing a `DELETE` statement can cause triggers to run that can cause deletes in other tables. For example, if two tables are linked by a foreign key and rows in the referenced table are deleted, then it is common that rows in the referencing table would also have to be deleted to maintain referential integrity.

Examples

Delete rows from table *pies* where the column *flavour* equals *Lemon Meringue*:

```
DELETE FROM pies
 WHERE flavour='Lemon Meringue';
```

Delete rows in *trees*, if the value of *height* is smaller than 80.

```
DELETE FROM trees
 WHERE height < 80;
```

Delete all rows from *mytable*:

```
DELETE FROM mytable;
```

Delete rows from *mytable* using a subquery in the where condition:

```
DELETE FROM mytable
 WHERE id IN (
       SELECT id
         FROM mytable2
      );
```

Delete rows from *mytable* using a list of values:

```
DELETE FROM mytable
 WHERE id IN (
       value1,
       value2,
       value3,
       value4,
       value5
      );
```

Example with related tables

Suppose there is a simple database that lists people and addresses. More than one person can live at a particular address and a person can live at more than one address (this is an example of a many-to-many relationship). The database only has three tables, *person*, *address*, and *pa*, with the following data:

person

pid	name
1	Joe
2	Bob
3	Ann

address

aid	description
100	2001 Main St.
200	35 Pico Blvd.

pa

pid	aid
1	100
2	100
3	100
1	200

The *pa* table relates the person and address tables, showing that Joe, Bob and Ann all live at 2001 Main Street, but Joe also takes up residence on Pico Boulevard.

In order to remove joe from the database, two deletes must be executed:

```
DELETE FROM person WHERE pid=1;
DELETE FROM pa WHERE pid=1;
```

To maintain referential integrity, Joe's records must be removed from both *person* and *pa*. The means by which integrity is sustained can happen differently in varying relational database management systems [*citation needed*]. It could be that beyond just having three tables, the database also has been set up with a trigger so that whenever a row is deleted from *person* any linked rows would be deleted from *pa*. Then the first statement:

```
DELETE FROM person WHERE pid=1;
```

would *automatically* trigger the second:

```
DELETE FROM pa WHERE pid=1;
```

Related commands

Deleting all rows from a table can be very time consuming. Some DBMSWikipedia:Please clarify offer a TRUNCATE TABLE command that works a lot quicker, as it only alters metadata and typically does not spend time enforcing constraints or firing triggers.

DELETE only deletes the rows. For deleting a table entirely the DROP command can be used.

References

Join

An SQL **join** clause combines records from two or more tables in a database. It creates a set that can be saved as a table or used as it is. A `JOIN` is a means for combining fields from two tables by using values common to each. ANSI-standard SQL specifies five types of `JOIN`: `INNER`, `LEFT OUTER`, `RIGHT OUTER`, `FULL OUTER` and `CROSS`. As a special case, a table (base table, view, or joined table) can `JOIN` to itself in a *self-join*.

A programmer writes a `JOIN` statement to identify the records for joining. If the evaluated predicate is true, the combined record is then produced in the expected format, a record set or a temporary table.

Sample tables

Relational databases are often normalized to eliminate duplication of information when objects may have one-to-many relationships. For example, a Department may be associated with many different Employees. Joining two tables effectively creates another table which combines information from both tables. This is at some expense in terms of the time it takes to compute the join. While it is also possible to simply maintain a denormalized table if speed is important, duplicate information may take extra space, and add the expense and complexity of maintaining data integrity if data which is duplicated later changes.

All subsequent explanations on join types in this article make use of the following two tables. The rows in these tables serve to illustrate the effect of different types of joins and join-predicates. In the following tables the `DepartmentID` column of the `Department` table (which can be designated as `Department.DepartmentID`) is the primary key, while `Employee.DepartmentID` is a foreign key.

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Heisenberg	33
Robinson	34
Smith	34
John	

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Note: In the Employee table above, the employee "John" has not been assigned to any department yet. Also, note that no employees are assigned to the "Marketing" department.

This is the SQL to create the aforementioned tables.

```
CREATE TABLE department
(
```

```
 DepartmentID INT,
 DepartmentName VARCHAR(20)
);

CREATE TABLE employee
(
 LastName VARCHAR(20),
 DepartmentID INT
);

INSERT INTO department(DepartmentID, DepartmentName) VALUES(31,
'Sales');
INSERT INTO department(DepartmentID, DepartmentName) VALUES(33,
'Engineering');
INSERT INTO department(DepartmentID, DepartmentName) VALUES(34,
'Clerical');
INSERT INTO department(DepartmentID, DepartmentName) VALUES(35,
'Marketing');

INSERT INTO employee(LastName, DepartmentID) VALUES('Rafferty', 31);
INSERT INTO employee(LastName, DepartmentID) VALUES('Jones', 33);
INSERT INTO employee(LastName, DepartmentID) VALUES('Heisenberg', 33);
INSERT INTO employee(LastName, DepartmentID) VALUES('Robinson', 34);
INSERT INTO employee(LastName, DepartmentID) VALUES('Smith', 34);
INSERT INTO employee(LastName, DepartmentID) VALUES('John', NULL);
```

Cross join

CROSS JOIN returns the Cartesian product of rows from tables in the join. In other words, it will produce rows which combine each row from the first table with each row from the second table.[1]

Example of an explicit cross join:

```
SELECT *
FROM employee CROSS JOIN department;
```

Example of an implicit cross join:

```
SELECT *
FROM employee, department;
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Rafferty	31	Sales	31
Jones	33	Sales	31
Heisenberg	33	Sales	31
Smith	34	Sales	31
Robinson	34	Sales	31
John		Sales	31
Rafferty	31	Engineering	33
Jones	33	Engineering	33
Heisenberg	33	Engineering	33
Smith	34	Engineering	33
Robinson	34	Engineering	33
John		Engineering	33
Rafferty	31	Clerical	34
Jones	33	Clerical	34
Heisenberg	33	Clerical	34
Smith	34	Clerical	34
Robinson	34	Clerical	34
John		Clerical	34
Rafferty	31	Marketing	35
Jones	33	Marketing	35
Heisenberg	33	Marketing	35
Smith	34	Marketing	35
Robinson	34	Marketing	35
John		Marketing	35

The cross join does not apply any predicate to filter records from the joined table. Programmers can further filter the results of a cross join by using a `WHERE` clause.

In the SQL:2011 standard, cross joins are part of the optional F401, "Extended joined table", package.

Inner join

An ***'inner join'*** is a commonly used join operation used in applications. It can only be safely used in a database that enforces referential integrity or where the join fields are guaranteed not to be NULL. Many transaction processing relational databases rely on Atomicity, Consistency, Isolation, Durability (ACID) data update standards to ensure data integrity, making inner joins an appropriate choice. Many reporting relational database and data warehouses use high volume Extract, Transform, Load (ETL) batch updates which make referential integrity difficult or impossible to enforce, resulting in potentially NULL join fields that a SQL query author cannot modify and which cause inner joins to omit data with no indication of an error. The choice to use an inner join depends on the database design and data characteristics. A left outer join can usually be substituted for an inner join when the join field in one table may contain NULL values. A commitment to an inner join assumes NULL join fields will not be introduced by future changes, including vendor updates, design changes and bulk processing outside of the application's data validation

rules such as data conversions.

Inner join creates a new result table by combining column values of two tables (A and B) based upon the join-predicate. The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. The result of the join can be defined as the outcome of first taking the Cartesian product (or Cross join) of all records in the tables (combining every record in table A with every record in table B) and then returning all records which satisfy the join predicate. Actual SQL implementations normally use other approaches, such as hash joins or sort-merge joins, since computing the Cartesian product is very inefficient.

SQL specifies two different syntactical ways to express joins: "explicit join notation" and "implicit join notation".

The "explicit join notation" uses the **`JOIN`** keyword, optionally preceded by the **`INNER`** keyword, to specify the table to join, and the **`ON`** keyword to specify the predicates for the join, as in the following example:

```
SELECT *
FROM employee INNER JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```

The "implicit join notation" simply lists the tables for joining, in the `FROM` clause of the `SELECT` statement, using commas to separate them. Thus it specifies a cross join, and the `WHERE` clause may apply additional filter-predicates (which function comparably to the join-predicates in the explicit notation).

The following example is equivalent to the previous one, but this time using implicit join notation:

```
SELECT *
FROM employee, department
WHERE employee.DepartmentID = department.DepartmentID;
```

The queries given in the examples above will join the Employee and Department tables using the DepartmentID column of both tables. Where the DepartmentID of these tables match (i.e. the join-predicate is satisfied), the query will combine the *LastName*, *DepartmentID* and *DepartmentName* columns from the two tables into a result row. Where the DepartmentID does not match, no result row is generated.

Thus the result of the execution of either of the two queries above will be:

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Robinson	34	Clerical	34
Jones	33	Engineering	33
Smith	34	Clerical	34
Heisenberg	33	Engineering	33
Rafferty	31	Sales	31

Note: Programmers should take special care when joining tables on columns that can contain NULL values, since NULL will never match any other value (not even NULL itself), unless the join condition explicitly uses the `IS NULL` or `IS NOT NULL` predicates.

Notice that the employee "John" and the department "Marketing" do not appear in the query execution results. Neither of these has any matching records in the other respective table: "John" has no associated department, and no employee has the department ID 35 ("Marketing"). Depending on the desired results, this behavior may be a subtle bug, which can be avoided with an outer join.

One can further classify inner joins as equi-joins, as natural joins, or as cross-joins.

Equi-join

An **equi-join** is a specific type of comparator-based join, that uses only equality comparisons in the join-predicate. Using other comparison operators (such as <) disqualifies a join as an equi-join. The query shown above has already provided an example of an equi-join:

```
SELECT *
FROM employee JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```

We can write equi-join as below,

```
SELECT *
FROM employee, department
WHERE employee.DepartmentID = department.DepartmentID;
```

If columns in an equi-join have the same name, SQL-92 provides an optional shorthand notation for expressing equi-joins, by way of the `USING` construct:[2]

```
SELECT *
FROM employee INNER JOIN department USING (DepartmentID);
```

The `USING` construct is more than mere syntactic sugar, however, since the result set differs from the result set of the version with the explicit predicate. Specifically, any columns mentioned in the `USING` list will appear only once, with an unqualified name, rather than once for each table in the join. In the above case, there will be a single `DepartmentID` column and no `employee.DepartmentID` or `department.DepartmentID`.

The `USING` clause is not supported by MS SQL Server and Sybase.

Natural join

A natural join is a type of equi-join where the join predicate arises implicitly by comparing all columns in both tables that have the same column-names in the joined tables. The resulting joined table contains only one column for each pair of equally named columns. In the case that no columns with the same names are found, a cross join is performed.

Most experts agree that NATURAL JOINs are dangerous and therefore strongly discourage their use.[3] The danger comes from inadvertently adding a new column, named the same as another column in the other table. An existing natural join might then "naturally" use the new column for comparisons, making comparisons/matches using different criteria (from different columns) than before. Thus an existing query could produce different results, even though the data in the tables have not been changed, but only augmented. The use of column names to automatically determine table links is not an option in large databases with hundreds or thousands of tables where it would place an unrealistic constraint on naming conventions. Real world databases are commonly designed with foreign key data that is not consistently populated (NULL values are allowed), due to business rules and context. It is common practice to modify column names of similar data in different tables and this lack of rigid consistency relegates natural joins to a theoretical concept for discussion.

The above sample query for inner joins can be expressed as a natural join in the following way:

```
SELECT *
FROM employee NATURAL JOIN department;
```

As with the explicit `USING` clause, only one DepartmentID column occurs in the joined table, with no qualifier:

DepartmentID	Employee.LastName	Department.DepartmentName
34	Smith	Clerical
33	Jones	Engineering
34	Robinson	Clerical
33	Heisenberg	Engineering
31	Rafferty	Sales

PostgreSQL, MySQL and Oracle support natural joins; Microsoft T-SQL and IBM DB2 do not. The columns used in the join are implicit so the join code does not show which columns are expected, and a change in column names may change the results. In the SQL:2011 standard, natural joins are part of the optional F401, "Extended joined table", package.

In many database environments the column names are controlled by an outside vendor, not the query developer. A natural join assumes stability and consistency in column names which can change during vendor mandated version upgrades.

Outer join

An **outer join** does not require each record in the two joined tables to have a matching record. The joined table retains each record—even if no other matching record exists. Outer joins subdivide further into left outer joins, right outer joins, and full outer joins, depending on which table's rows are retained (left, right, or both).

(In this case *left* and *right* refer to the two sides of the `JOIN` keyword.)

No implicit join-notation for outer joins exists in standard SQL.

Left outer join

The result of a *left outer join* (or simply **left join**) for tables A and B always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B). This means that if the `ON` clause matches 0 (zero) records in B (for a given record in A), the join will still return a row in the result (for that record)—but with NULL in each column from B. A **left outer join** returns all the values from an inner join plus all values in the left table that do not match to the right table.

For example, this allows us to find an employee's department, but still shows the employee(s) even when they have not been assigned to a department (contrary to the inner-join example above, where unassigned employees were excluded from the result).

Example of a left outer join (the **`OUTER`** keyword is optional), with the additional result row (compared with the inner join) italicized:

```
SELECT *
FROM employee LEFT OUTER JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
John			
Heisenberg	33	Engineering	33

Alternative syntaxes

Oracle supports the deprecated[4] syntax:

```
SELECT *
FROM employee, department
WHERE employee.DepartmentID = department.DepartmentID(+)
```

Sybase supports the syntax:

```
SELECT *
FROM employee, department
WHERE employee.DepartmentID *= department.DepartmentID
```

IBM Informix supports the syntax:

```
SELECT *
FROM employee, OUTER department
WHERE employee.DepartmentID = department.DepartmentID
```

Right outer join

A **right outer join** (or **right join**) closely resembles a left outer join, except with the treatment of the tables reversed. Every row from the "right" table (B) will appear in the joined table at least once. If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in B.

A right outer join returns all the values from the right table and matched values from the left table (NULL in the case of no matching join predicate). For example, this allows us to find each employee and his or her department, but still show departments that have no employees.

Below is an example of a right outer join (the **`OUTER`** keyword is optional), with the additional result row italicized:

```
SELECT *
FROM employee RIGHT OUTER JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Heisenberg	33	Engineering	33
Rafferty	31	Sales	31
		Marketing	*35*

Right and left outer joins are functionally equivalent. Neither provides any functionality that the other does not, so right and left outer joins may replace each other as long as the table order is switched.

Alternate syntaxes

Oracle supports the deprecated syntax:

```
SELECT *
FROM employee, department
WHERE employee.DepartmentID(+) = department.DepartmentID
```

Full outer join

Conceptually, a **full outer join** combines the effect of applying both left and right outer joins. Where records in the FULL OUTER JOINed tables do not match, the result set will have NULL values for every column of the table that lacks a matching row. For those records that do match, a single row will be produced in the result set (containing fields populated from both tables).

For example, this allows us to see each employee who is in a department and each department that has an employee, but also see each employee who is not part of a department and each department which doesn't have an employee.

Example of a full outer join (the **`OUTER`** keyword is optional):

```
SELECT *
FROM employee FULL OUTER JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
John			
Heisenberg	33	Engineering	33
Rafferty	31	Sales	31
		Marketing	*35*

Some database systems do not support the full outer join functionality directly, but they can emulate it through the use of an inner join and UNION ALL selects of the "single table rows" from left and right tables respectively. The same example can appear as follows:

```
SELECT employee.LastName, employee.DepartmentID,
       department.DepartmentName, department.DepartmentID
```

```
FROM employee
INNER JOIN department ON employee.DepartmentID = department.DepartmentID

UNION ALL

SELECT employee.LastName, employee.DepartmentID,
       cast(NULL as varchar(20)), cast(NULL as integer)
FROM employee
WHERE NOT EXISTS (
    SELECT * FROM department
             WHERE employee.DepartmentID = department.DepartmentID)

UNION ALL

SELECT cast(NULL as varchar(20)), cast(NULL as integer),
       department.DepartmentName, department.DepartmentID
FROM department
WHERE NOT EXISTS (
    SELECT * FROM employee
             WHERE employee.DepartmentID = department.DepartmentID)
```

Self-join

A self-join is joining a table to itself.

Example

A query to find all pairings of two employees in the same country is desired. If there were two separate tables for employees and a query which requested employees in the first table having the same country as employees in the second table, a normal join operation could be used to find the answer table. However, all the employee information is contained within a single large table.[5]

Consider a modified `Employee` table such as the following:

Employee Table

EmployeeID	LastName	Country	DepartmentID
123	Rafferty	Australia	31
124	Jones	Australia	33
145	Heisenberg	Australia	33
201	Robinson	United States	34
305	Smith	Germany	34
306	John	Germany	

An example solution query could be as follows:

```
SELECT F.EmployeeID, F.LastName, S.EmployeeID, S.LastName, F.Country
FROM Employee F INNER JOIN Employee S ON F.Country = S.Country
WHERE F.EmployeeID < S.EmployeeID
```

```
ORDER BY F.EmployeeID, S.EmployeeID;
```

Which results in the following table being generated.

Employee Table after Self-join by Country

EmployeeID	LastName	EmployeeID	LastName	Country
123	Rafferty	124	Jones	Australia
123	Rafferty	145	Heisenberg	Australia
124	Jones	145	Heisenberg	Australia
305	Smith	306	John	Germany

For this example:

- `F` and `S` are aliases for the first and second copies of the employee table.
- The condition `F.Country = S.Country` excludes pairings between employees in different countries. The example question only wanted pairs of employees in the same country.
- The condition `F.EmployeeID < S.EmployeeID` excludes pairings where the `EmployeeID` of the first employee is greater than or equal to the `EmployeeID` of the second employee. In other words, the effect of this condition is to exclude duplicate pairings and self-pairings. Without it, the following less useful table would be generated (the table below displays only the "Germany" portion of the result):

EmployeeID	LastName	EmployeeID	LastName	Country
305	Smith	305	Smith	Germany
305	Smith	306	John	Germany
306	John	305	Smith	Germany
306	John	306	John	Germany

Only one of the two middle pairings is needed to satisfy the original question, and the topmost and bottommost are of no interest at all in this example.

Alternatives

The effect of an outer join can also be obtained using a UNION ALL between an INNER JOIN and a SELECT of the rows in the "main" table that do not fulfill the join condition. For example

```
SELECT employee.LastName, employee.DepartmentID,
department.DepartmentName
FROM employee
LEFT OUTER JOIN department ON employee.DepartmentID =
department.DepartmentID;
```

can also be written as

```
SELECT employee.LastName, employee.DepartmentID,
department.DepartmentName
FROM employee
INNER JOIN department ON employee.DepartmentID =
department.DepartmentID
```

```
UNION ALL

SELECT employee.LastName, employee.DepartmentID, cast(NULL as
varchar(20))
FROM employee
WHERE NOT EXISTS (
    SELECT * FROM department
             WHERE employee.DepartmentID = department.DepartmentID)
```

Implementation

Much work in database-systems has aimed at efficient implementation of joins, because relational systems commonly call for joins, yet face difficulties in optimising their efficient execution. The problem arises because inner joins operate both commutatively and associatively. In practice, this means that the user merely supplies the list of tables for joining and the join conditions to use, and the database system has the task of determining the most efficient way to perform the operation. A query optimizer determines how to execute a query containing joins. A query optimizer has two basic freedoms:

1. **Join order**: Because it joins functions commutatively and associatively, the order in which the system joins tables does not change the final result set of the query. However, join-order **could** have an enormous impact on the cost of the join operation, so choosing the best join order becomes very important.
2. **Join method**: Given two tables and a join condition, multiple algorithms can produce the result set of the join. Which algorithm runs most efficiently depends on the sizes of the input tables, the number of rows from each table that match the join condition, and the operations required by the rest of the query.

Many join-algorithms treat their inputs differently. One can refer to the inputs to a join as the "outer" and "inner" join operands, or "left" and "right", respectively. In the case of nested loops, for example, the database system will scan the entire inner relation for each row of the outer relation.

One can classify query-plans involving joins as follows:

left-deep
: using a base table (rather than another join) as the inner operand of each join in the plan

right-deep
: using a base table as the outer operand of each join in the plan

bushy
: neither left-deep nor right-deep; both inputs to a join may themselves result from joins

These names derive from the appearance of the query plan if drawn as a tree, with the outer join relation on the left and the inner relation on the right (as convention dictates).

Join algorithms

Three fundamental algorithms for performing a join operation exist: nested loop join, sort-merge join and hash join.

Join Indexes

Join indexes are database indexes that facilitate the processing of join queries in data warehouses: they are currently (2012) available in implementations by Oracle[6] and Teradata.[7]

In the Teradata implementation, specified columns, aggregate functions on columns, or components of date columns from one or more tables are specified using a syntax similar to the definition of a database view: up to 64 columns/column expressions can be specified in a single join index. Optionally, a column that defines the primary key of the composite data may also be specified: on parallel hardware, the column values are used to partition the index's contents across multiple disks. When the source tables are updated interactively by users, the contents of the join index are automatically updated. Any query whose WHERE clause specifies any combination of columns or column expressions that are an exact subset of those defined in a join index (a so-called "covering query") will cause the join index, rather than the original tables and their indexes, to be consulted during query execution.

The Oracle implementation limits itself to using bitmap indexes. A *bitmap join index* is used for low-cardinality columns (i.e., columns containing fewer than 300 distinct values, according to the Oracle documentation): it combines low-cardinality columns from multiple related tables. The example Oracle uses is that of an inventory system, where different suppliers provide different parts. The schema has three linked tables: two "master tables", Part and Supplier, and a "detail table", Inventory. The last is a many-to-many table linking Supplier to Part, and contains the most rows. Every part has a Part Type, and every supplier is based in the USA, and has a State column. There are not more than 60 states+territories in the USA, and not more than 300 Part Types. The bitmap join index is defined using a standard three-table join on the above three tables, and specifying the Part_Type and Supplier_State columns for the index. However, it is defined on the Inventory table, even though the columns Part_Type and Supplier_State are "borrowed" from Supplier and Part respectively.

As for Teradata, an Oracle bitmap join index is only utilized to answer a query when the query's WHERE clause specifies columns limited to those that are included in the join index.

Notes

[1] SQL CROSS JOIN (http://www.sqlguides.com/sql_cross_join.php)

[2] Simplifying Joins with the USING Keyword (http://www.java2s.com/Tutorial/Oracle/0140__Table-Joins/SimplifyingJoinswiththeUSINGKeyword.htm)

[3] Ask Tom "Oracle support of ANSI joins." (http://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:13430766143199) Back to basics: inner joins » Eddie Awad's Blog (http://awads.net/wp/2006/03/20/back-to-basics-inner-joins/#comment-2837)

[4] Oracle Left Outer Join (http://www.dba-oracle.com/tips_oracle_left_outer_join.htm)

[5] Adapted from

[6] Oracle Bitmap Join Index. URL: http://www.dba-oracle.com/art_builder_bitmap_join_idx.htm

[7] Teradata Join Indexes. http://www.coffingdw.com/sql/tdsqlutp/join_index.htm

References

- Pratt, Phillip J (2005), *A Guide To SQL, Seventh Edition*, Thomson Course Technology, ISBN 978-0-619-21674-0
- Shah, Nilesh (2005) [2002], *Database Systems Using Oracle − A Simplified Guide to SQL and PL/SQL Second Edition* (International ed.), Pearson Education International, ISBN 0-13-191180-5
- Yu, Clement T.; Meng, Weiyi (1998), *Principles of Database Query Processing for Advanced Applications* (http://books.google.com/?id=aBHRDhrrehYC), Morgan Kaufmann, ISBN 978-1-55860-434-6, retrieved 2009-03-03

External links

- Specific to products
  - Sybase ASE 15 Joins (http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.sqlug/html/sqlug/sqlug138.htm)
  - MySQL 5.5 Joins (http://dev.mysql.com/doc/refman/5.5/en/join.html)
  - PostgreSQL 9.3 Joins (http://www.postgresql.org/docs/9.3/static/tutorial-join.html)
  - Joins in Microsoft SQL Server (http://msdn2.microsoft.com/en-us/library/ms191517.aspx)
  - Joins in MaxDB 7.6 (http://maxdb.sap.com/currentdoc/45/f31c38e95511d5995d00508b5d5211/content.htm)
  - Joins in Oracle 11g (http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/queries006.htm)
- General
  - A Visual Explanation of SQL Joins (http://www.codinghorror.com/blog/archives/000976.html)
  - Another visual explanation of SQL joins, along with some set theory (http://www.halfgaar.net/sql-joins-are-easy)
  - SQL join types classified with examples (http://www.gplivna.eu/papers/sql_join_types.htm)
  - An alternative strategy to using FULL OUTER JOIN (http://weblogs.sqlteam.com/jeffs/archive/2007/04/19/Full-Outer-Joins.aspx)
  - Latest article explaining Join in simple diagrams and relevant code (http://blog.sqlauthority.com/2009/04/13/sql-server-introduction-to-joins-basic-of-joins/)
  - Visual representation of 7 possible SQL joins between two sets (http://www.flickr.com/photos/mattimattila/8190148857/)

Set operations

UNION operator

In SQL the **`UNION`** clause combines the results of two SQL queries into a single table of all matching rows. The two queries must result in the same number of columns and compatible data types in order to unite. Any duplicate records are automatically removed unless `UNION ALL` is used.

`UNION` can be useful in data warehouse applications where tables aren't perfectly normalized.[1] A simple example would be a database having tables `sales2005` and `sales2006` that have identical structures but are separated because of performance considerations. A `UNION` query could combine results from both tables.

Note that `UNION` does not guarantee the order of rows. Rows from the second operand may appear before, after, or mixed with rows from the first operand. In situations where a specific order is desired, `ORDER BY` must be used.

Note that `UNION ALL` may be much faster than plain `UNION`.

Examples

Given these two tables:

sales2005

person	amount
Joe	1000
Alex	2000
Bob	5000

sales2006

person	amount
Joe	2000
Alex	2000
Zach	35000

Executing this statement:

```
SELECT * FROM sales2005
UNION
SELECT * FROM sales2006;
```

yields this result set, though the order of the rows can vary because no `ORDER BY` clause was supplied:

person	amount
Joe	1000
Alex	2000
Bob	5000
Joe	2000
Zach	35000

Note that there are two rows for Joe because those rows are distinct across their columns. There is only one row for Alex because those rows are not distinct for both columns.

`UNION ALL` gives different results, because it will not eliminate duplicates. Executing this statement:

```
SELECT * FROM sales2005
UNION ALL
SELECT * FROM sales2006;
```

would give these results, again allowing variance for the lack of an `ORDER BY` statement:

person	amount
Joe	1000
Joe	2000
Alex	2000
Alex	2000
Bob	5000
Zach	35000

The discussion of full outer joins also has an example that uses `UNION`.

INTERSECT operator

The SQL `INTERSECT` operator takes the results of two queries and returns only rows that appear in both result sets. For purposes of duplicate removal the `INTERSECT` operator does not distinguish between `NULL`s. The `INTERSECT` operator removes duplicate rows from the final result set. The `INTERSECT ALL` operator does not remove duplicate rows from the final result set.

Example

The following example `INTERSECT` query returns all rows from the Orders table where Quantity is between 50 and 100.

```
SELECT *
FROM   Orders
WHERE  Quantity BETWEEN 1 AND 100

INTERSECT

SELECT *
FROM   Orders
WHERE  Quantity BETWEEN 50 AND 200;
```

EXCEPT operator

The SQL `EXCEPT` operator takes the distinct rows of one query and returns the rows that do not appear in a second result set. The `EXCEPT ALL` operator (not supported in MSSQL) does not remove duplicates. For purposes of row elimination and duplicate removal, the `EXCEPT` operator does not distinguish between `NULL`s.

Notably, the Oracle platform provides a `MINUS` operator which is functionally equivalent to the SQL standard `EXCEPT DISTINCT` operator [2].

Example

The following example `EXCEPT` query returns all rows from the Orders table where Quantity is between 1 and 49, and those with a Quantity between 76 and 100.

Worded another way; the query returns all rows where the Quantity is between 1 and 100, apart from rows where the quantity is between 50 and 75.

```
SELECT *
FROM   Orders
WHERE  Quantity BETWEEN 1 AND 100

EXCEPT

SELECT *
FROM   Orders
WHERE  Quantity BETWEEN 50 AND 75;
```

Alternatively, in implementations of the SQL language without the `EXCEPT` operator, the equivalent form of a `LEFT JOIN` where the right hand values are `NULL` can be used instead.

Example

The following example is equivalent to the above example but without using the `EXCEPT` operator.

```
SELECT o1.*
FROM (
    SELECT *
    FROM Orders
    WHERE Quantity BETWEEN 1 AND 100) o1
LEFT JOIN (
    SELECT *
    FROM Orders
    WHERE Quantity BETWEEN 50 AND 75) o2
ON o1.id = o2.id
WHERE o2.id IS NULL
```

References

[1] "a `UNION ALL` views technique for managing maintenance and performance in your large data warehouse environment ... This `UNION ALL` technique has saved many of my clients with issues related to time-sensitive database designs. These databases usually have an extremely volatile current timeframe, month, or day portion and the older data is rarely updated. Using different container DASD allocations, tablespaces, tables, and index definitions, the settings can be tuned for the specific performance considerations for these different volatility levels and update frequency situations." Terabyte Data Warehouse Table Design Choices - Part 2 (http://www.dbazine.com/datawarehouse/dw-articles/beulke4) (URL accessed on July 25, 2006)

[2] http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14200/ap_standard_sql003.htm#g14847

External links

- MSDN documentation on UNION in Transact-SQL for SQL Server (http://msdn2.microsoft.com/en-us/library/ms180026.aspx)
- Naming of select list items in set operations (http://www.sqlexpert.co.uk/2007/11/order-by-clause-in-statements-with.html)
- UNION in MySQL with Examples (http://www.mysqltutorial.org/sql-union-mysql.aspx)
- UNION in MySQL (http://dev.mysql.com/doc/refman/5.0/en/union.html)
- UNION Clause in PostgreSQL (http://www.postgresql.org/docs/current/static/sql-select.html#SQL-UNION)
- SQL UNION and UNION ALL (http://www.w3schools.com/sql/sql_union.asp)
- Sort order within UNION statement (http://www.sqlexpert.co.uk/2007/11/order-by-clause-in-statements-with.html)
- Designing a data flow that loads a warehouse table (http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.dwe.tutorial.doc/tutetlmod1_lesson2.htm)
- Oracle 11g documentation for UNION (ALL), INTERSECT and MINUS (http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/queries004.htm)
- SQL Set Operators (http://www.gplivna.eu/papers/sql_set_operators.htm)

Commit

In computer science and data management, a **commit** is the making of a set of tentative changes permanent. A popular usage is at the end of a transaction. A *commit* is an act of committing.

Data management

A `COMMIT` statement in SQL ends a transaction within a relational database management system (RDBMS) and makes all changes visible to other users. The general format is to issue a `BEGIN WORK` statement, one or more SQL statements, and then the `COMMIT` statement. Alternatively, a `ROLLBACK` statement can be issued, which undoes all the work performed since `BEGIN WORK` was issued. A `COMMIT` statement will also release any existing savepoints that may be in use.

In terms of transactions, the opposite of commit is to discard the tentative changes of a transaction, a rollback.

Revision control

Commits are also done for revision control systems for source code such as Subversion, Git or CVS. A commit in the context of these version control systems refers to submitting the latest changes of the source code to the repository, and making these changes part of the head revision of the repository. Thus, when other users do an `UPDATE` or a `checkout` from the repository, they will receive the latest committed version, unless they specify they wish to retrieve a previous version of the source code in the repository. Version control systems also have similar functionality to SQL databases in that they allow rolling back to previous versions easily. In this context, a commit with version control systems is not as dangerous as it allows easy rollback, even after the commit has been done.

Notes

Rollback

In database technologies, a **rollback** is an operation which returns the database to some previous state. Rollbacks are important for database integrity, because they mean that the database can be restored to a clean copy even after erroneous operations are performed. They are crucial for recovering from database server crashes; by rolling back any transaction which was active at the time of the crash, the database is restored to a consistent state.

The rollback feature is usually implemented with a transaction log, but can also be implemented via multiversion concurrency control.

Cascading rollback

A *cascading rollback* occurs in database systems when a transaction (T1) causes a failure and a rollback must be performed. Other transactions dependent on T1's actions must also be rollbacked due to T1's failure, thus causing a cascading effect. That is, one transaction's failure causes many to fail.

Practical database recovery techniques guarantee cascadeless rollback, therefore a cascading rollback is not a desirable result.

SQL

In SQL, `ROLLBACK` is a command that causes all data changes since the last `BEGIN WORK`, or `START TRANSACTION` to be discarded by the relational database management systems (RDBMS), so that the state of the data is "rolled back" to the way it was before those changes were made.

A `ROLLBACK` statement will also release any existing savepoints that may be in use.

In most SQL dialects, `ROLLBACK`s are connection specific. This means that if two connections are made to the same database, a `ROLLBACK` made in one connection will not affect any other connections. This is vital for proper concurrency.

References

- Ramez Elmasri (2007). *Fundamentals of Database Systems*. Pearson Addison Wesley. ISBN 0-321-36957-2.
- "ROLLBACK Transaction" [1], Microsoft SQL Server.
- "Sql Commands" [2], MySQL.

References

[1] http://msdn2.microsoft.com/en-us/library/ms181299.aspx

[2] http://www.pantz.org/software/mysql/mysqlcommands.html

Truncate

In SQL, the **`TRUNCATE TABLE`** statement is a Data Definition Language (DDL) operation that marks the extents of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing a number of integrity enforcing mechanisms. It was officially introduced in the SQL:2008 standard.

The `TRUNCATE TABLE mytable` statement is logically (though not physically) equivalent to the `DELETE FROM mytable` statement (without a `WHERE` clause). The following characteristics distinguish `TRUNCATE TABLE` from `DELETE`:

- In the Oracle Database, `TRUNCATE` is implicitly preceded and followed by a commit operation. (This may also be the case in MySQL, when using a transactional storage engine.)
- Typically, `TRUNCATE TABLE` quickly deletes all records in a table by deallocating the data pages used by the table. This reduces the resource overhead of logging the deletions, as well as the number of locks acquired. Records removed this way cannot be restored in a rollback operation. Two notable exceptions to this rule are the implementations found in PostgreSQL and Microsoft SQL Server, both of which allow `TRUNCATE TABLE` statements to be committed or rolled back transactionally.
- You cannot specify a `WHERE` clause in a `TRUNCATE TABLE` statement—it is all or nothing.
- `TRUNCATE TABLE` cannot be used when a foreign key references the table to be truncated, since `TRUNCATE TABLE` statements do not fire triggers. This could result in inconsistent data because `ON DELETE`/`ON UPDATE` triggers would not fire.
- In some database systems, `TRUNCATE TABLE` resets the count of an Identity column back to the identity's *seed*.
- In Microsoft SQL Server 2000 and beyond in full recovery mode, every change to the database is logged, so `TRUNCATE TABLE` statements can be used for tables involved in log shipping.

References

Views

View (database)

In database theory, a **view** is the result set of a *stored* query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary. Unlike ordinary *base tables* in a relational database, a view does not form part of the physical schema: as a result set, it is a virtual table computed or collated from data in the database, dynamically when access to that view is requested. Changes applied to the data in a relevant *underlying table* are reflected in the data shown in subsequent invocations of the view. In some NoSQL databases, views are the only way to query data.

Views can provide advantages over tables:

- Views can represent a subset of the data contained in a table; consequently, a view can limit the degree of exposure of the underlying tables to the outer world: a given user may have permission to query the view, while denied access to the rest of the base table.
- Views can join and simplify multiple tables into a single virtual table
- Views can act as aggregated tables, where the database engine aggregates data (sum, average etc.) and presents the calculated results as part of the data
- Views can hide the complexity of data; for example a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data which it presents
- Depending on the SQL engine used, views can provide extra security

Just as a function (in programming) can provide abstraction, so can a database view. In another parallel with functions, database users can manipulate nested views, thus one view can aggregate data from other views. Without the use of views, the normalization of databases above second normal form would become much more difficult. Views can make it easier to create lossless join decomposition.

Just as rows in a base table lack any defined ordering, rows available through a view do not appear with any default sorting. A view is a relational table, and the relational model defines a table as a set of rows. Since sets are not ordered - by definition - nor are the rows of a view. Therefore, an ORDER BY clause in the view definition is meaningless; the SQL standard (SQL:2003) does not allow an ORDER BY clause in the subquery of a CREATE VIEW command, just as it is refused in a CREATE TABLE statement. However, sorted data can be obtained from a view, in the same way as any other table — as part of a query statement on that view. Nevertheless, some DBMS (such as Oracle Database) do not abide by this SQL standard restriction.

Read-only vs. updatable views

Database practitioners can define views as read-only or updatable. If the database system can determine the reverse mapping from the view schema to the schema of the underlying base tables, then the view is updatable. INSERT, UPDATE, and DELETE operations can be performed on updatable views. Read-only views do not support such operations because the DBMS cannot map the changes to the underlying base tables. A view update is done by key preservation.

Some systems support the definition of INSTEAD OF triggers on views. This technique allows the definition of other logic for execution in place of an insert, update, or delete operation on the views. Thus database systems can implement data modifications based on read-only views. However, an INSTEAD OF trigger does not change the

read-only or updatable property of the view itself.

Advanced view features

Various database management systems have extended the views from read-only subsets of data.

Oracle Database introduced the concept of materialized views: pre-executed, non-virtual views commonly used in data warehousing. They give a static snapshot of the data and may include data from remote sources. The accuracy of a materialized view depends on the frequency of trigger mechanisms behind its updates. IBM DB2 provides so-called "materialized query tables" (MQTs) for the same purpose. Microsoft SQL Server introduced in its 2000 version indexed views which only store a separate index from the table, but not the entire data. PostgreSQL implemented materialized views in its 9.3 release.

Equivalence

A view is equivalent to its source query. When queries are run against views, the query is modified. For example, if there exists a view named accounts_view with the content as follows:

```
accounts_view:
--------------
SELECT name,
       money_received,
       money_sent,
       (money_received - money_sent) AS balance,
       address,
         ...
  FROM table_customers c
  JOIN accounts_table a
    ON a.customer_id = c.customer_id
```

then the application could run a simple query such as:

```
Simple query
------------
SELECT name,
       balance
  FROM accounts_view
```

The RDBMS then takes the simple query, replaces the equivalent view, then sends the following to the query optimizer:

```
Preprocessed query:
-------------------
SELECT name,
       balance
  FROM (SELECT name,
               money_received,
               money_sent,
               (money_received - money_sent) AS balance,
               address,
                      ...
          FROM table_customers c JOIN accounts_table a
```

```
                ON a.customer_id = c.customer_id        )
```

The optimizer then removes unnecessary fields and complexity (for example: it is not necessary to read the address, since the parent invocation does not make use of it) and then sends the query to the SQL engine for processing.

External links

- Materialized query tables in DB2 [1]
- Views in Microsoft SQL Server [2]
- Views in MySQL [3]
- Views in PostgreSQL [4]
- Views in SQLite [5]
- Views in Oracle 11.2 [6]
- Views in CouchDB [7]
- Materialized Views in Oracle 11.2 [8]

References

[1] http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z10.doc.intro/src/tpc/db2z_typesoftables.htm
[2] http://msdn.microsoft.com/en-us/library/ms187956.aspx
[3] http://dev.mysql.com/doc/refman/5.1/en/views.html
[4] http://www.postgresql.org/docs/current/interactive/tutorial-views.html
[5] http://www.sqlite.org/lang_createview.html
[6] http://download.oracle.com/docs/cd/E11882_01/server.112/e17118/statements_8004.htm#SQLRF01504
[7] http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views
[8] http://download.oracle.com/docs/cd/E11882_01/server.112/e17118/statements_6002.htm#SQLRF01302

Materialized view

A **materialized view** is a database object that contains the results of a query. For example, it may be a local copy of data located remotely, or may be a subset of the rows and/or columns of a table or join result, or may be a summary based on aggregations of a table's data. Materialized views, which store data based on remote tables, are also known as snapshots. A snapshot can be redefined as a materialized view.

Introduction

In any database management system following the relational model, a view is a virtual table representing the result of a database query. Whenever a query or an update addresses an ordinary view's virtual table, the DBMS converts these into queries or updates against the underlying base tables. A materialized view takes a different approach in which the query result is cached as a concrete table that may be updated from the original base tables from time to time. This enables much more efficient access, at the cost of some data being potentially out-of-date. It is most useful in data warehousing scenarios, where frequent queries of the actual base tables can be extremely expensive.

In addition, because the materialized view is manifested as a real table, anything that can be done to a real table can be done to it, most importantly building indexes on any column, enabling drastic speedups in query time. In a normal view, it's typically only possible to exploit indexes on columns that come directly from (or have a mapping to) indexed columns in the base tables; often this functionality is not offered at all.

Materialized views were implemented first by the Oracle Database: the Query rewrite feature was added from version 8i.[1] They are also supported in Sybase SQL Anywhere.[2] In IBM DB2, they are called "materialized query tables"; Microsoft SQL Server has a similar feature called "indexed views".[3] MySQL doesn't support materialized views natively, but workarounds can be implemented by using triggers or stored procedures [4] or by using the open-source application Flexviews.[5] It is also possible to implement materialized views in PostgreSQL; version 9.3 natively supports materialized views.[6] In the current version 9.3 the materialized views are not auto-refreshed, and are populated only at time of creation (unless WITH NO DATA is used) and may be refreshed later manually using REFRESH MATERIALIZED VIEW command.[7]

Example syntax to create a materialized view in Oracle:

```
 CREATE MATERIALIZED VIEW MV_MY_VIEW
REFRESH FAST START WITH SYSDATE
   NEXT SYSDATE + 1
     AS SELECT * FROM <table_name>;
```

References

[1] Oracle8i Tuning Release 8.1.5 (http://www.ecst.csuchico.edu/~melody/courses/Fall2001CSCI379/DOC/server.815/a67775/ch2.htm). Ecst.csuchico.edu. Retrieved on 2012-02-09.

[2] Materialized Views – Sybase SQL Anywhere (http://www.ianywhere.com/developer/product_manuals/sqlanywhere/1000/en/html/dbugen10/ug-workingwdb-s-3142433.html). Ianywhere.com. Retrieved on 2012-02-09.

[3] Improving Performance with SQL Server 2005 Indexed Views (http://www.microsoft.com/technet/prodtechnol/sql/2005/impprfiv.mspx). Microsoft.com. Retrieved on 2012-02-09.

[4] Implementing materialized views in MySQL (http://www.shinguz.ch/MySQL/mysql_mv.html). Shinguz.ch (2006-11-06). Retrieved on 2012-02-09.

[5] Flexviews for MySQL – incrementally refreshable materialized views w/ MySQL (http://flexviews.sourceforge.net/index.html). Flexviews.sourceforge.net. Retrieved on 2012-02-09.

[6] PostgreSQL: Materialized Views (http://wiki.postgresql.org/wiki/Materialized_Views). Wiki.postgresql.org (2010-05-07). Retrieved on 2013-09-25.

[7] PostgreSQL: Documentation: 9.3: CREATE MATERIALIZED VIEW (http://www.postgresql.org/docs/9.3/static/sql-creatematerializedview.html). PostgreSQL.com. Retrieved on 2014-01-25.

External links

- Materialized View Concepts and Architecture – Oracle (http://download.oracle.com/docs/cd/B10501_01/server.920/a96567/repmview.htm)
- SQL Snippets: SQL Features Tutorials – Materialized Views – Oracle (http://www.sqlsnippets.com/en/topic-12868.html)
- Oracle9i Replication Management API Reference Release 2 (9.2) (http://download.oracle.com/docs/cd/B10501_01/server.920/a96568/rarmviea.htm#94135)

Advanced Issues

Hierarchical query

A **hierarchical query** is a type of SQL query that handles hierarchical model data. They are special case of more general recursive fixpoint queries, which compute transitive closures.

In standard SQL:1999 hierarchical queries are implemented by way of recursive *common table expressions* (CTEs). Unlike the Oracle extension described below, the recursive CTEs were designed with fixpoint semantics from the beginning. The recursive CTEs from the standard were relatively close to the existing implementation in IBM DB2 version 2. Recursive CTEs are also supported by Microsoft SQL Server, Firebird 2.1 , PostgreSQL 8.4+,[1] Oracle 11g Release 2 and CUBRID.

An alternative syntax is the non-standard `CONNECT BY` construct; it was introduced by Oracle in the 1980s. Prior to Oracle 10g, the construct was only useful for traversing acyclic graphs because it returned an error on detecting any cycles; in version 10g Oracle introduced the NOCYCLE feature (and keyword), making the traversal work in the presence of cycles as well.

CONNECT BY

`CONNECT BY` is supported by EnterpriseDB,[2] Oracle database,[3] CUBRID, and DB2 although only if it is enabled as a compatibility mode. The syntax is as follows:

```
 SELECT select_list
 FROM table_expression
 [ WHERE ... ]
 [ START WITH start_expression ]
 CONNECT BY [NOCYCLE] { PRIOR parent_expr = child_expr | child_expr =
PRIOR parent_expr }
 [ ORDER SIBLINGS BY column1 [ ASC | DESC ] [, column2 [ ASC | DESC ] ]
 ...
 [ GROUP BY ... ]
 [ HAVING ... ]
 ...
```

For example

```
 SELECT LEVEL, LPAD (' ', 2 * (LEVEL - 1)) || ename "employee", empno,
mgr "manager"
 FROM emp START WITH mgr IS NULL
 CONNECT BY PRIOR empno = mgr;
```

The output from the above query would look like:

```
 level |  employee   | empno | manager
-------+-------------+-------+---------
     1 | KING        |  7839 |
     2 |   JONES     |  7566 |    7839
     3 |     SCOTT   |  7788 |    7566
```

```
     4 |       ADAMS |  7876 |    7788
     3 |     FORD    |  7902 |    7566
     4 |       SMITH |  7369 |    7902
     2 |   BLAKE     |  7698 |    7839
     3 |     ALLEN   |  7499 |    7698
     3 |     WARD    |  7521 |    7698
     3 |     MARTIN  |  7654 |    7698
     3 |     TURNER  |  7844 |    7698
     3 |     JAMES   |  7900 |    7698
     2 |   CLARK     |  7782 |    7839
     3 |     MILLER  |  7934 |    7782
(14 rows)
```

Pseudo-columns

- `LEVEL`
- `CONNECT_BY_ISLEAF`
- `CONNECT_BY_ISCYCLE`
- `CONNECT_BY_ROOT`

Unary operators

The following example returns the last name of each employee in department 10, each manager above that employee in the hierarchy, the number of levels between manager and employee, and the path between the two:

```
   SELECT ename "Employee", CONNECT_BY_ROOT ename "Manager",
   LEVEL-1 "Pathlen", SYS_CONNECT_BY_PATH(ename, '/') "Path"
   FROM emp
   WHERE LEVEL > 1 and deptno = 10
   CONNECT BY PRIOR empno = mgr
   ORDER BY "Employee", "Manager", "Pathlen", "Path";
```

Functions

- `SYS_CONNECT_BY_PATH`

Common table expression

A Common Table Expression, or CTE, (in SQL) is a temporary named result set, derived from a simple query and defined within the execution scope of a `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement.

CTEs can be thought of as alternatives to derived tables (subquery), views, and inline user-defined functions.

Common table expressions are supported by DB2, Firebird,[4] Microsoft SQL Server, Oracle (with recursion since 11g release 2), PostgreSQL (since 8.4), SQLite (since 3.8.3), HyperSQL and H2 (experimental).[5] Oracle calls CTEs "subquery factoring". The syntax is as follows:

```
WITH [RECURSIVE] with_query [, ...]
SELECT...
```

where `with_query`'s syntax is:

```
with query_name [ (column_name [,...]) ] AS (SELECT ...)
```

Recursive CTEs (or "recursive subquery factoring" in Oracle lingo) can be used to traverse relations (as graphs or trees) although the syntax is much more involved because there are no automatic pseudo-columns created (like `LEVEL` above); if these are desired, they have to be created in the code. See MSDN documentation or IBM documentation[6][7] for tutorial examples.

The `RECURSIVE` keyword is not usually needed after WITH in systems other than PostgreSQL.

In SQL:1999 a recursive (CTE) query may appear anywhere a query is allowed. It's possible, for example, to name the result using `CREATE [RECURSIVE] VIEW`. Using a CTE inside an `INSERT INTO`, one can populate a table with data generated from a recursive query; random data generation is possible using this technique without using any procedural statements.

An example of a recursive query computing the factorial of numbers from 0 to 9 is the following:

```
WITH RECURSIVE temp (n, fact) AS
(SELECT 0, 1 -- Initial Subquery
  UNION ALL
 SELECT n+1, (n+1)*fact FROM temp -- Recursive Subquery
        WHERE n < 9)
SELECT * FROM temp;
```

References

[1] PostgreSQL

[2] Hierarchical Queries (http://www.enterprisedb.com/documentation/hierarchical-queries.html), EnterpriseDB

[3] Hierarchical Queries (http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/queries003.htm), Oracle

[4] Comparison of relational database management systems#Database capabilities

[5] http://www.h2database.com/html/advanced.html#recursive_queries

[6] http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z9.doc.apsg/src/tpc/db2z_xmprecursivecte.htm

[7] http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=%2Fsqlp%2Frbafyrecursivequeries.htm

Further reading

- C. J. Date (2011). *SQL and Relational Theory: How to Write Accurate SQL Code* (2nd ed.). O'Reilly Media. pp. 159–163. ISBN 978-1-4493-1640-2.

Academic textbooks. Note that these cover only the SQL:1999 standard (and Datalog), but not the Oracle extension.

- Abraham Silberschatz; Henry Korth; S. Sudarshan (2010). *Database System Concepts* (6th ed.). McGraw-Hill. pp. 187–192. ISBN 978-0-07-352332-3.
- Raghu Ramakrishnan; Johannes Gehrke (2003). *Database management systems* (3rd ed.). McGraw-Hill. ISBN 978-0-07-246563-1. Chapter 24.
- Hector Garcia-Molina; Jeffrey D. Ullman; Jennifer Widom (2009). *Database systems: the complete book* (2nd ed.). Pearson Prentice Hall. pp. 437–445. ISBN 978-0-13-187325-4.

External links

- http://stackoverflow.com/questions/1731889/cycle-detection-with-recursive-subquery-factoring
- http://explainextended.com/2009/11/18/sql-server-are-the-recursive-ctes-really-set-based/
- http://gennick.com/with.html
- http://www.cs.duke.edu/courses/fall04/cps116/lectures/11-recursion.pdf

Hint (SQL)

In database query operations, various **SQL** implementations use **hints** as additions to the SQL standard that instruct the database engine on how to execute the query. For example, a hint may tell the engine to use as little memory as possible (even if the query will run slowly), or to use or not to use an index (even if the query optimizer would decide otherwise).

Implementation

Different database engines use different approaches in implementing hints.

- MySQL uses its own extension to the SQL standard, where a table name may be followed by `USE INDEX`, `FORCE INDEX` or `IGNORE INDEX` keywords.[1]
- Oracle implements hints by using specially-crafted comments in the query that begin with a `+` symbol, thus not affecting SQL compatibility.[2]
- Postgres Plus® Advanced Server (a proprietary version of PostgreSQL from EnterpriseDB) offers hints compatible with those of Oracle.[3][4]

References

[1] MySQL 5.5 Reference Manual: 12.2.9.3 Index Hint Syntax (http://dev.mysql.com/doc/refman/5.5/en/index-hints.html)

[2] Mike Ault: Oracle SQL Hints Tuning (http://www.dba-oracle.com/t_sql_hints_tuning.htm)

[3] Postgres Plus Advanced Server Performance and Scalability Guide: Query Optimization Hints (http://www.enterprisedb.com/docs/en/9.2/perffeat/Postgres_Plus_Advanced_Server_Performance_Guide-33.htm)

[4] Postgres Plus Advanced Server Oracle Compatibility Developer's Guide: Optimizer Hints (http://www.enterprisedb.com/docs/en/9.2/oracompat/Postgres_Plus_Advanced_Server_Oracle_Compatibility_Guide-75.htm)

Extensions

SQL/CLI

The **SQL/CLI**, or *Call-Level Interface*, is an extension to the SQL standard is defined in SQL:1999 (based on CLI-95), but also available in later editions such as ISO/IEC 9075-3:2003. This extension defines common interfacing components (structures and procedures) that can be used to execute SQL statements from applications written in other programming languages. The SQL/CLI extension is defined in such a way that SQL statements and SQL/CLI procedure calls are treated as separate from the calling application's source code.

SQL/JRT

SQL/JRT, or *SQL Routines and Types for the Java Programming Language*, is an extension to the SQL standard first published as ISO/IEC 9075-13:2002 (part 13 of SQL:1999). SQL/JRT specifies the ability to invoke static Java methods as routines from within SQL applications, commonly referred to as "Java stored procedures". SQL/JRT also calls for the ability to use Java classes as SQL structured user-defined types. The two parts of the extension originate from the earlier ANSI SQLJ part 1 and 2 standards (not to be confused with SQLJ part 0, which defined an embedding of SQL into Java, later standardized by ISO as SQL/OLB.)

Example

SQL/JRT allows a Java function to be called from SQL code like this:

```
CREATE FUNCTION sinh(v DOUBLE) RETURNS DOUBLE
  LANGUAGE JAVA DETERMINISTIC NO SQL
  EXTERNAL NAME 'CLASSPATH:java.lang.Math.sinh'

SELECT sinh(doublecolumn) FROM mytable
```

SQL/JRT also allows Java code to dynamically generate tables using a `java.sql.ResultSet` object. The result sets returned are converted to SQL tables and can be used anywhere a table or view can be used.

Implementations

SQL/JRT stored procedures are implemented in HSQLDB.[] Java stored procedures have also been implemented in Oracle's JServer (or Aurora JVM), which was introduced in the Oracle Database version 8i in 1999;[1] it is now called Oracle JVM. IBM DB2 also supported Java stored procedures since about 1998, although using an external JVM (at that time).

References

[1] The Aurora JVM and Its Components (http://docs.oracle.com/cd/F49540_01/DOC/java.815/a64686/01_intr4.htm), Oracle Corp.

External links

- SQL:2003 SQL/JRT draft (http://www.podgoretsky.com/ftp/docs/DB/SQL2003/5WD-13-JRT-2003-09.pdf)
- SQL:2003 SQL Standard User Defined Types and Routines (http://farrago.sourceforge.net/design/UserDefinedTypesAndRoutines.html) from the Farrago documentation

SQL/MED

The **SQL/MED**, or *Management of External Data*, extension to the SQL standard is defined by ISO/IEC 9075-9:2003. SQL/MED provides extensions to SQL that define foreign-data wrappers and datalink types to allow SQL to manage external data. External data is data that is accessible to, but not managed by, an SQL-based DBMS. This standard can be used in the development of federated database systems.

Implementations

- PostgreSQL has support for some SQL/MED since version 9.1. 9.2 supports more.[1]
- LucidDB has support for SQL/MED.[2]

References

[1] PostgreSQL Wiki: SQL/MED (http://wiki.postgresql.org/wiki/SQL/MED)
[2] LucidDB Architecture (http://www.luciddb.org/html/arch.html)

External links

- SQL/MED - A Status Report (http://www.sigmod.org/publications/sigmod-record/0209/jimmelton.pdf/view)

SQL/OLB

SQL/OLB, or *Object Language Bindings*, is a standard for embedding SQL in Java, commonly known by its prior name as SQLJ (part 0). Besides describing the syntax and semantics of SQLJ, which are typically given relative to JDBC, the standard also describes mechanisms to ensure binary portability of SQLJ applications, and specifies various Java packages and their contained classes.

SQL/OLB was informally known as "SQLJ part 0" before standardization, which first occurred under the aegis of ANSI in 1998 and then ISO in 2000. Although the latter was published after the bulk of SQL:1999, officially it was "part 10" of that standard—a convention that was maintained for subsequent ISO SQL standards, including the current one, SQL:2011.

Examples

For some (possibly outdated) examples, see the article on SQLJ.

Implementations

Both Oracle 8i and IBM DB2 introduced support around 1999. Oracle 12c claims conformance with SQL/OLB:1999, but not with the newer SQL/OLB:2008.[1]

References

[1] http://docs.oracle.com/cd/E16655_01/server.121/e17209/ap_standard_sql008.htm

SQL/PSM

SQL/PSM (SQL/Persistent Stored Modules) is an ISO standard mainly defining an extension of SQL with a procedural language for use in stored procedures. Initially published in 1996 as an extension of SQL-92 (ISO/IEC 9075-4:1996, a version sometimes called PSM-96 or even SQL-92/PSM), SQL/PSM was later incorporated into the multi-part SQL:1999 standard, and has been part 4 of that standard since then, most recently in SQL:2011. The SQL:1999 part 4 covered less than the original PSM-96 because the SQL statements for defining, managing, and invoking routines were actually incorporated into part 2 SQL/Foundation, leaving only the procedural language itself as SQL/PSM. The SQL/PSM facilities are still optional as far as the SQL standard is concerned; most of them are grouped in Features P001-P008.

SQL/PSM standardizes syntax and semantics for control flow, exception handling (called "condition handling" in SQL/PSM), local variables, assignment of expressions to variables and parameters, and (procedural) use of cursors. It also defines an information schema (metadata) for stored procedures. SQL/PSM is one language in which methods for the SQL:1999 structured types can be defined. (The other is Java, via SQL/JRT.)

In practice MySQL's procedural language and IBM's SQL PL (used in DB2) are closest to the SQL/PSM standard.

SQL/PSM resembles and inspired by PL/SQL, as well as PL/pgSQL, so they are similar languages. With PostgreSQL v9.X some SQL/PSM features, like overloading of SQL-invoked functions and procedures[1] are now supported. A PostgreSQL addon implements SQL/PSM[2] (alongside its own procedural language), although it is not part of the core product.[3]

References

[1] feature T322 (http://www.postgresql.org/docs/9.0/static/features-sql-standard.html)

[2] See git's repository of plpsm0 (https://github.com/okbob/plpsm0), 2011-05's announce (http://www.postgresql.org/message-id/1305291347.14548.13.camel@jara.office.nic.cz), 2012-02's Proposal PL/pgPSM announce (http://www.postgresql.org/message-id/CAFj8pRDWFdcjNSnwQB_3j1-rMO6b8=TmLTNBvDCSpRrOW2Dfeg@mail.gmail.com) and 2008's Guide (http://postgres.cz/wiki/SQL/PSM_Manual).

[3] PostgreSQL: Documentation: 9.2: SQL Conformance (http://www.postgresql.org/docs/9.2/static/features.html)

Further reading

- Jim Melton, *Understanding SQL's Stored Procedures: A Complete Guide to SQL/PSM*, Morgan Kaufmann Publishers, 1998, ISBN 1-55860-461-8

SQL/Schemata

The **SQL/Schemata**, or *Information and Definition Schemas*, part to the SQL standard is defined by ISO/IEC 9075-11:2008. SQL/Schemata defines the information schema and definition schema, providing a common set of tools to make SQL databases and objects self-describing. These tools include the SQL object identifier[*clarify*], structure[*clarify*] and integrity constraints, security and authorization specifications[*clarify*], features and packages[*clarify*] of ISO/IEC 9075, support of features provided by SQL-based DBMS implementations, SQL-based DBMS implementation information and sizing items[*clarify*], and the values supported[*clarify*] by the DBMS implementations. SQL/Schemata defines a number of features, some of which are mandatory.

References

StreamSQL

StreamSQL is a query language that extends SQL with the ability to process real-time data streams. SQL is primarily intended for manipulating relations (also known as tables), which are finite bags of tuples (rows). StreamSQL adds the ability to manipulate streams, which are infinite sequences of tuples that are not all available at the same time. Because streams are infinite, operations over streams must be monotonic. Queries over streams are generally "continuous", executing for long periods of time and returning incremental results.

The StreamSQL language is typically used in the context of a Data Stream Management System (DSMS), for applications including algorithmic trading, market data analytics, network monitoring, surveillance, e-fraud detection and prevention, clickstream analytics and real-time compliance (anti-money laundering, RegNMS, MiFID).

Technical details

StreamSQL extends the type system of SQL to support streams in addition to tables. Several new operations are introduced to manipulate streams.

Selecting from a stream - A standard `SELECT` statement can be issued against a stream to calculate functions (using the target list) or filter out unwanted tuples (using a `WHERE` clause). The result will be a new stream.

Stream-Relation Join - A stream can be joined with a relation to produce a new stream. Each tuple on the stream is joined with the current value of the relation based on a predicate to produce 0 or more tuples.

Union and Merge - Two or more streams can be combined by unioning or merging them. Unioning combines tuples in strict FIFO order. Merging is more deterministic, combining streams according to a sort key.

Windowing and Aggregation - A stream can be windowed to create finite sets of tuples. For example, a window of size 5 minutes would contain all the tuples in a given 5 minute period. Window definitions can allow complex selections of messages, based on tuple field values. Once a finite batch of tuples is created, analytics such as count, average, max, etc., can be applied.

Windowing and Joining - A pair of streams can also be windowed and then joined together. Tuples within the join windows will combine to create resulting tuples if they fulfill the predicate.

History

StreamSQL is derived from academic research into Event Stream Processing, closely related to complex event processing. Led by Dr. Michael Stonebraker, a team of 30 professors and students on project Aurora worked collaboratively from 2001 through 2003 to develop the core principles behind StreamSQL.

The Aurora project has since been superseded by the Borealis project. Borealis is a distributed multi-processor version of Aurora.

External links

- Documentation, technical resources, and blog describing StreamSQL usage for complex event processing and event stream processing [1]

References

[1] http://www.sqlstream.com

dBase and derived languages

DBase

dBASE

Paradigm(s)	Imperative, Declarative
Appeared in	1979
Developer	C. Wayne Ratliff
Influenced	Clipper, WordTech products, Harbour. FoxBASE+, FoxPro, Visual FoxPro
Website	www.dbase.com [1]

dBASE was one of the first and in its day the most successful database management systems for microcomputers. The dBASE system includes the core database engine, a query system, a forms engine, and a programming language that ties all of these components together. dBASE's underlying file format, the `.dbf` file, is widely used in applications needing a simple format to store structured data.

dBASE was originally published by Ashton-Tate for microcomputer operating system CP/M in 1980, and later ported to Apple II and IBM PC computers running DOS. On the PC platform, in particular, dBASE became one of the best-selling software titles for a number of years. A major upgrade was released as dBASE III, and ported to a wider variety of platforms, adding UNIX, and VMS. By the mid-1980s, Ashton-Tate was one of the "big three" software publishers in the early business software market, the others being Lotus Development and WordPerfect.

Starting in the mid-1980s, several companies produced their own variations on the dBASE product and especially the dBASE programming language. These included FoxBASE+(later renamed FoxPro), Clipper, and other so-called xBase products. Many of these were technically stronger than dBASE, but could not push it aside in the market. This changed with the disastrous introduction of dBASE IV, whose design and stability were so poor many users switched to other products.[*citation needed*] At the same time, there was growing use of IBM-invented SQL (Structured Query Language) in database products. Another factor was user adoption of Microsoft Windows on desktop computers. The shift toward SQL and Windows put pressure on the makers of xBase products to invest in major redesign to provide new capabilities.

In spite of growing pressure to evolve, in the early 1990s xBase products constituted the leading database platform for implementing business applications. The size and impact of the xBase market did not go unnoticed, and within one year, the three top xBase firms were acquired by larger software companies. Borland purchased Ashton-Tate, Microsoft bought Fox Software, and Computer Associates acquired Nantucket. However, by the following decade most of the original xBase products had faded from prominence and several disappeared. Products known as dBASE still exist, owned by dBase LLC.

History

Origins

In the late 1960s, Fred Thompson at the Jet Propulsion Laboratory (JPL) was using a Tymshare product named RETRIEVE to manage a database of electronic calculators, which were at that time *very* expensive products. In 1971 Fred collaborated with Jack Hatfield, a programmer at JPL, to write an enhanced version of RETRIEVE which became the JPLDIS project. JPLDIS was written in FORTRAN on the UNIVAC 1108 mainframe, and was presented publicly in 1973. When Hatfield left JPL in 1974, Jeb Long took over his role.[2]

While working at JPL as a contractor, C. Wayne Ratliff entered the office football pool. He had no interest in the game, but felt he could win the pool by processing the post-game statistics found in newspapers. In order to do this, he turned his attention to a database system and, by chance, came across the documentation for JPLDIS. He used this as the basis for a port to PTDOS on his kit-built IMSAI 8080 microcomputer, and called the resulting system **Vulcan** (after Mr. Spock on *Star Trek*).[3]

Ashton-Tate

George Tate and Hal Lashlee had built two successful start-up companies - Discount Software was one of the first to sell PC software programs through the mail to consumers, and Software Distributors which was one of the first wholesale distributors of PC software in the world. They entered into an agreement with Ratliff to market Vulcan, and formed Ashton-Tate to do so. Ratliff ported Vulcan from PTDOS to CP/M. Hal Pawluk, who handled marketing for the nascent company, decided to change the name to the more business-like "dBASE". Pawluk devised the use of lower case "d" and all-caps "BASE" to create a distinctive name (it is an error to write it any other way). Pawluk suggested calling the new product version two ("II") to suggest it was less buggy than an initial release. **dBASE II** was the result and became a standard CP/M application along with WordStar and SuperCalc.[4]

In 1981, IBM commissioned a port of dBASE for the then-in-development PC. The resultant program was one of the initial pieces of software available when the IBM PC went on sale the fall of 1981. dBASE was one of a very few "professional" programs on the platform at that time, and became a huge success. The customer base included not only end-users, but an increasing number of "value added resellers", or VARs, who purchased dBASE, wrote applications with it, and sold the completed systems to their customers. The May 1983 release of **dBASE II RunTime** further entrenched dBASE in the VAR market by allowing the VARs to deploy their products using the lower-cost RunTime system.

The success of dBASE created many opportunities for third-parties. Soon, hundreds of companies (ranging from one-person shops to large firms) began offering dBASE-related application development, libraries of code to add functionality, applications with source code that could be customized and re-sold, consulting, training, and how-to books. A new company in San Diego (today known as **Advisor Media** http://www.Advisor.com) premiered a magazine devoted to professional use of dBASE, **Data Based Advisor**. All of these activities fueled the rapid rise of dBASE as the leading product of its type.

dBASE III

As platforms and operating systems proliferated in the early 1980s, the company found it difficult to port the assembly language-based dBASE to target systems. This led to a re-write of the platform in the C programming language, using automated code conversion tools. The resulting code worked, but was essentially undocumented and inhuman in syntax, a problem that would prove to be serious in the future.[*citation needed*]

The resulting dBASE III was released in May 1984. Although reviewers widely panned its lowered performance, the product was otherwise well reviewed. After a few rapid upgrades the system stabilized and was once again a best-seller throughout the 1980s, and formed the famous "application trio" of PC compatibles (dBASE, Lotus 123, and WordPerfect). By the fall of 1984, the company had over 500 employees and was taking in $40 million a year in sales, the vast majority from dBASE products.

Recent history

dBASE has evolved into a modern object oriented language that runs on 32 bit Windows. It can be used to build a wide variety of applications including web apps hosted on a Windows server, Windows rich client applications, and middleware applications. dBASE can access most modern database engines via ODBC drivers.

dBASE features an IDE with a Command Window and Navigator, a just in time compiler, a preprocessor, a virtual machine interpreter, a linker for creating dBASE application .exe's, a freely available runtime engine, and numerous two-way GUI design tools including a Form Designer, Report Designer, Menu Designer, Label Designer, Datamodule Designer, SQL Query Designer, and Table Designer. Two-way Tools refers to the ability to switch back and forth between using a GUI design tool and the source code editor. Other tools include a Source Code Editor, a Project Manager that simplifies building and deploying a dBASE application, and an integrated Debugger. dBASE features structured exception handling and has many built-in classes that can be subclassed via single inheritance. There are visual classes, data classes, and many other supporting classes. Visual classes include Form, SubForm, Notebook, Container, Entryfield, RadioButton, SpinBox, ComboBox, ListBox, PushButton, Image, Grid, ScrollBar, ActiveX, Report, ReportViewer, Text, TextLabel and many others. Database classes include Session, Database, Query, Rowset, Field, StoredProc and Datamodule classes. Other classes include File, String, Math, Array, Date, Exception, Object and others. dBASE objects can be dynamically subclassed by adding new properties to them at runtime.

The current version, dBASE PLUS 8, was announced on March 19, 2013. The product not only supports the existing BDE connectivity, but it also adds support for ADO and ODBC either through the new ADO approach or with the existing BDE connections. This update of dBASE PLUS 8 has been graphically enhanced, has been modernized for continued support on today's more recent operating systems including support for running applications on Microsoft's Windows 8 and Windows Server 2012 product lines. The product is fully compatible with earlier 32 bit versions of dBASE including Visual dBASE 7.x and dB2K. It is partly compatible with 16 bit Windows versions (dBASE for Windows 5.x and Visual dBASE 5.x) and with older DOS versions (dBASE II, dBASE III, dBASE IV, and dBASE 5). In May of 2012, dBase introduced a new product to allow users of the DOS based dBASE products to continue to use the product on the latest Windows OSs and hardware. The new product was called dbDOS(TM)dbdos The latest dbDOS is called dbDOS(TM) PRO 2 and continues to be enhanced.

dBASE / xBase programming language

For handling data, dBASE provided detailed procedural commands and functions to open and traverse records in data files (e.g., USE, SKIP, GO TOP, GO BOTTOM, and GO recno), manipulate field values (REPLACE and STORE), and manipulate text strings (e.g., STR() and SUBSTR()), numbers, and dates. Its ability to simultaneously open and manipulate multiple files containing related data led Ashton-Tate to label dBASE a "relational database" although it did not meet the criteria defined by Dr. Edgar F. Codd's relational model; it could more accurately be called an application development language and integrated navigational database management system that is

influenced by relational concepts.

The dBASE product used a runtime interpreter architecture, which allowed the user to execute commands by typing them in a command line "dot prompt." Upon typing a command or function and pressing the return key, the interpreter would immediately execute or evaluate it. Similarly, program scripts (text files with PRG extensions) ran in the interpreter (with the DO command), where each command and variable was evaluated at runtime. This made dBASE programs quick and easy to write and test because programmers didn't have to first compile and link them before running them. (For other languages, these steps were tedious in the days of single- and double-digit megahertz CPUs.) The interpreter also handled automatically and dynamically all memory management (i.e., no preallocating memory and no hexadecimal notation), which more than any other feature made it possible for a business person with no programming experience to develop applications.

Conversely, the ease and simplicity of dBASE presented a challenge as its users became more expert and as professional programmers were drawn to it. More complex and more critical applications demanded professional programming features for greater reliability and performance, as well as greater developer productivity.

Over time, Ashton-Tate's competitors introduced so-called clone products and compilers that had more robust programming features such as user-defined functions (UDFs) to supplement the built-in function set, scoped variables for writing routines and functions that were less likely to be affected by external processes, arrays for complex data handling, packaging features for delivering applications as executable files without external runtime interpreters, object-oriented syntax, and interfaces for accessing data in remote database management systems. Ashton-Tate also implemented many of these features with varying degrees of success. Ashton-Tate and its competitors also began to incorporate SQL, the ANSI/ISO standard language for creating, modifying, and retrieving data stored in relational database management systems.

Eventually, it became clear that the dBASE world had expanded far beyond Ashton-Tate, which was considered to be retarding innovation, growth and the impact of the technology. A "third-party" community had formed, consisting of Fox Software, Nantucket, Alpha Software, Data Based Advisor Magazine, SBT and other application development firms, and major developer groups. They sought to create a dBASE language standard, supported by IEEE committee X3J19 and initiative IEEE 1192. They began using "xBase" to generically refer to the language and database design, to distinguish it from the Ashton-Tate product.

Ashton-Tate was invited to participate, but instead it saw the rise of xBase as an illegal threat to its proprietary technology. In 1988 Ashton-Tate filed suit against Fox Software and Santa Cruz Operation (SCO) for copying dBASE's "structure and sequence" in FoxBase+ (SCO marketed XENIX and UNIX versions of the Fox products). In December 1990, U.S. District judge Terry Hatter, Jr. dismissed Ashton-Tate's lawsuit and invalidated Ashton-Tate's copyrights for not disclosing that dBASE had been based, in part, on the public domain JPLDIS. In October 1991, while the case was still under appeal, Borland International acquired Ashton-Tate, and as one of the merger's provisions the U.S. Justice Department required Borland to end the lawsuit against Fox and allow other companies to use the dBASE/xBase language without the threat of legal action.

By the end of 1992, major software companies raised the stakes by acquiring the leading xBase products. Borland acquired Ashton-Tate's dBASE products (and later WordTech's xBase products), Microsoft acquired Fox Software's FoxBASE+ and FoxPro products, and Computer Associates acquired Nantucket's Clipper products. Advisor Media built on its Data Based Advisor magazine by launching FoxPro Advisor and Clipper Advisor (and other) developer magazines and journals, and live conferences for developers. However, a planned dBASE Advisor Magazine was aborted due the market failure of dBASE IV.

By the year 2000 the xBase market had faded as developers shifted to new database systems and programming languages. Computer Associates (later known as CA) eventually dropped Clipper. Borland restructured and sold dBASE. Of the major acquirers, Microsoft stuck with xBase the longest, evolving FoxPro into Visual FoxPro, but the product is no longer offered. In 2006 Advisor Media stopped its last-surviving xBase magazine, FoxPro Advisor. The era of xBase dominance has ended, but there are still xBase products. The dBASE product line is now owned by

dBase LLC.

Programming examples

Today, implementations of the dBASE language have expanded to include many features targeted for business applications, including object-oriented programming, manipulation of remote and distributed data via SQL, Internet functionality, and interaction with modern devices.

The following example opens an employee table ("empl"), gives every manager who supervises 1 or more employees a 10-percent raise, and then prints the names and salaries.

```
USE empl
REPLACE ALL salary WITH salary * 1.1 FOR supervisors > 0
LIST ALL fname, lname, salary TO PRINT
* (comment: reserved words shown in CAPITALS for illustration purposes)
```

Note how one does not have to keep mentioning the table name. The assumed ("current") table stays the same until told otherwise. This is in contrast to SQL which almost always needs explicit tables. Because of its origins as an interpreted interactive language, dBASE used a variety of contextual techniques to reduce the amount of typing needed. This facilitated incremental, interactive development but also made larger-scale modular programming difficult. A tenet of modular programming is that the correct execution of a program module must not be affected by external factors such as the state of memory variables or tables being manipulated in other program modules. Because dBASE was not designed with this in mind, developers had to be careful about porting (borrowing) programming code that assumed a certain context and it would make writing larger-scale modular code difficult. Work-area-specific references were still possible using the arrow notation ("B->customer") so that multiple tables could be manipulated at the same time. In addition, if the developer had the foresight to name their tables appropriately, they could clearly refer to a large number of tables open at the same time by notation such as ("employee->salary") and ("vacation->start_date"). Alternatively, the alias command could be appended to the initial opening of a table statement which made referencing a table field unambiguous and simple. For example. one can open a table and assign an alias to it in this fashion, "use EMP alias Employee", and henceforth, refer to table variables as "Employee->Name".

Another notable feature is the re-use of the same clauses for different commands. For example, the FOR clause limits the scope of a given command. (It is somewhat comparable to SQL's WHERE clause). Different commands such as LIST, DELETE, REPLACE, BROWSE, etc. could all accept a FOR clause to limit (filter) the scope of their activity. This simplifies the learning of the language.

dBASE was also one of the first business-oriented languages to implement string evaluation.

```
i = 2
myMacro = "i + 10"
i = &myMacro
* comment: i now has the value 12
```

Here the "&" tells the interpreter to evaluate the string stored in "myMacro" as if it were programming code. This is an example of a feature that made dBASE programming flexible and dynamic, sometimes called "meta ability" in the profession. This could allow programming expressions to be placed inside tables, somewhat reminiscent of formulas in spreadsheet software.

However, it could also be problematic for pre-compiling and for making programming code secure from hacking. But, dBASE tended to be used for custom internal applications for small and medium companies where the lack of protection against copying, as compared to compiled software, was often less of an issue.

Interactivity

In addition to the dot-prompt, dBASE III, III+ and dBASE IV came packaged with an ASSIST application to manipulate data and queries, as well as an APPSGEN application which allowed the user to generate applications without resorting to code writing like a 4GL. The dBASE IV APPSGEN tool was based largely on portions of an early CP/M product named Personal Pearl.

Niches

Although the language has fallen out of favor as a primary business language, some find dBASE an excellent interactive ad-hoc data manipulation tool. Whereas SQL retrieves data sets from a relational database (RDBMS), with dBASE one can more easily manipulate, format, analyze and perform calculations on individual records, strings, numbers, and so on in a step-by-step imperative (procedural) way instead of trying to figure out how to use SQL's declarative operations.

Its granularity of operations is generally smaller than SQL, making it easier to split querying and table processing into easy-to-understand and easy-to-test parts. For example, one could insert a BROWSE operation between the filtering and the aggregation step to study the intermediate table or view (applied filter) before the aggregation step is applied.

As an application development platform, dBASE fills a gap between lower level languages such as C, C++, and Java and high-level proprietary 4GLs (fourth generation languages) and purely visual tools, providing relative ease-of-use for business people with less formal programming skill and high productivity for professional developers willing to trade off the low-level control.

dBASE remained a popular teaching tool even after sales slowed because the text-oriented commands were easier to present in printed training material than the mouse-oriented competitors. (Mouse-oriented commands were added to the product over time, but the command language remained a popular de facto standard while mousing commands tended to be vendor-specific.)

File formats

A major legacy of dBASE is its `.dbf` file format, which has been adopted in a number of other applications. For example, the shapefile format developed by ESRI for spatial data in its PC ArcInfo geographic information system, uses .dbf files to store feature attribute data.

Microsoft recommends saving a Microsoft Works database file in the dBASE file format so that it can be read by Microsoft Excel.[5]

dBASE's database system was one of the first to provide a header section for describing the structure of the data in the file.[*citation needed*] This meant that the program no longer required advance knowledge of the data structure, but rather could ask the data file how it was structured. There are several variations on the .dbf file structure, and not all dBASE-related products and .dbf file structures are compatible.

A second filetype is the `.dbt` file format for memo fields. While character fields are limited to 254 characters each, a memo field is a 10-byte pointer into a `.dbt` file which can include a much larger text field. dBASE was very limited in its ability to process memo fields, but some other xBase languages such as Clipper treated memo fields as strings just like character fields for all purposes except permanent storage.

dBASE uses `.ndx` files for single indexes, and `.mdx` *(multiple-index)* files for holding between 1 and 48 indexes. Some xBase languages include compatibility with `.ndx` files while others use different file formats such as `.ntx` used by Clipper and `.idx/.cdx` used by FoxPro or FlagShip. Later iterations of Clipper included drivers for `.ndx, .mdx, .idx and .cdx` indexes.

References

[1] http://www.dbase.com/
[2] Susan Lammers, "How it Started - JPLDIS: How Came The Idea" (http://www.foxprohistory.org/jeblong_jpldis.htm), The History of FoxPro
[3] Susan Lammers, "Interview with Wayne Ratliff" (http://www.foxprohistory.org/interview_wayne_ratliff.htm), The History of FoxPro
[4] "Ashton-Tate People" (http://www.foxprohistory.org/people_ashton.htm), The History of FoxPro
[5] Troubleshoot converting file formats - Excel - Office.com (http://office.microsoft.com/en-us/excel-help/troubleshoot-converting-file-formats-HP005203437.aspx)

External links

- Official website (http://www.dbase.com/)
- xBase (and dBASE) File Format Description (http://www.clicketyclick.dk/databases/xbase/format/index.html)

Clip (compiler)

clip

Developer(s)	ITK / Uri Hnykin
Stable release	1.2.0 / November 1, 2006
Operating system	Unix-like
License	GPL or Commercial License [1]
Website	http://www.itk.ru/ [2]

Clip compiler is a multi-platform (Linux and Windows (Cygwin)) Clipper programming language compiler with many additional features and libraries (for gtk, fivewin, netto, MySQL, ODBC, cti, tcp, gzip, Interbase, Oracle, Postgres), which is quite fast, has support for Hyper-Six and FoxPro RDD's, and can compile existing Clipper source code with very minor changes.

It has support for all the features in the original compiler, can access multiple types of databases such as Oracle, Informix, Interbase, MySQL, Postgres, all Xbase dialects (tables: Foxpro, Visual FoxPro, COMIX, indexes: NDX,NTX,CDX,)

It supports object-oriented programming, preprocessor, dynamic and static libraries, several functions for math, string management, arrays or vectors.

Clip is licensed under a "GPL type" License and uses the GNU CC compiler.

External links

- Clip compiler [2]
- Spanish user group of Clip Clip-castellano [3]
- Brazilian User Group of Clip [4]

References

[1] http://www.itk.ru/english/clip/cliplicense_comm.shtml
[2] http://www.itk.ru/english/index.shtml
[3] http://www.clip-castellano.com.ar/
[4] http://clip-br.web.br.com/

Clipper (programming language)

Clipper

Appeared in	1985
Stable release	CA Clipper 5.3b (May 20, 1997)
OS	DOS
Website	[1]

Clipper is a computer programming language that is used to create software programs that originally operated primarily under DOS. Although it is a powerful general-purpose programming language, it was primarily used to create database/business programs.

History

Clipper was originally created in 1985 as a compiler for *dBASE III*, a very popular database language at the time. Compiling dBASE code changes it from interpreted code, which must be interpreted every time each line of code is executed, to p-code, which uses a Virtual Machine to process the compiled p-code. p-code is considerably faster, but still not as fast as the machine code generated by native compilers. As a technical marketing ploy, the p-code was wrapped into object code (linkable .obj files) which gave the impression that it was compiled to native code. Clipper was created by Nantucket Corporation led by Barry ReBell (management) and Brian Russell (technical), and later sold to Computer Associates. GrafX Software licensed CA-Clipper in 2002 from CA for ongoing marketing and distribution.

As the product matured, it remained a DOS tool for many years, but added elements of the C programming language and Pascal programming language, as well as OOP, and the code-block data-type (hybridizing the concepts of dBase macros, or string-evaluation, and function pointers), to become far more powerful than the original. Nantucket's Aspen project later matured into the Windows native-code Visual Objects compiler.

The Clipper language is being actively implemented and extended by multiple organizations/vendors, like *XBase++* from Alaska Software and *FlagShip*, as well as free (GPL-licensed) projects like *Harbour* and *xHarbour*.

Many of the current implementations are portable (DOS, Windows, Linux (32- and 64-bit), Unix (32- and 64-bit), and Mac OS X), supporting many language extensions [2], and have greatly extended runtime libraries, as well as various Replaceable Database Drivers (RDD) supporting many popular database formats, like DBF, DBTNTX, DBFCDX (FoxPro, Apollo, Comix, and Advantage Database Server), MachSix (SIx Driver and Apollo), SQL, and more. These newer implementations all strive for full compatibility with the standard dBase/xBase syntax, while also offering OOP approaches and target-based syntax such as `SQLExecute()`.

The Clipper Usenet newsgroups are `comp.lang.clipper` [3] and `comp.lang.clipper.visual-objects` [4].

Programming in Clipper

A simple hello world - application:

```
? "Hello World!"
```

A simple data base input mask:

```
USE Customer SHARED NEW
clear
@  1, 0 SAY "CustNum" GET Customer->CustNum PICT "999999" VALID Customer->CustNum > 0
@  3, 0 SAY "Contact" GET Customer->Contact VALID !empty(Customer->Contact)
@  4, 0 SAY "Address" GET Customer->Address
READ
```

Version history

The various versions of Clipper were

From Nantucket Corporation; the "seasonal versions", billed as "dBase compilers"

- Nantucket Clipper Winter'84 - released May 25, 1985
- Nantucket Clipper Summer'85 - released 1985
- Nantucket Clipper Winter'85 - released January 29, 1986
- Nantucket Clipper Autumn'86 - released October 31, 1986
- Nantucket Clipper Summer'87 - released December 21, 1987

From Nantucket Corporation; Clipper 5

- Nantucket Clipper 5.00 - released 1990
- Nantucket Clipper 5.01 - released April 15, 1991
- Nantucket Clipper 5.01 Rev.129 - released March 31, 1992

and from Computer Associates; CA-Clipper 5

- CA Clipper 5.01a -
- CA Clipper 5.20 - released February 15, 1993
- CA-Clipper 5.2a - released March 15, 1993
- CA Clipper 5.2b - released June 25, 1993
- CA-Clipper 5.2c - released August 6, 1993
- CA Clipper 5.2d - released March 25, 1994
- CA-Clipper 5.2e - released February 7, 1995
- CA Clipper 5.30 - released June 26, 1995
- CA Clipper 5.3a - released May 20, 1996
- CA Clipper 5.3b - released May 20, 1997

External links

- Alaska Software [5] vendor of XBase++
- Apollo database engine supports CA-Clipper and FoxPro data files [6]
- Free Open Source Graphic,GUI & Form Designer for CA-Clipper [7]
- mini Clipper FAQ [8]
- Print from Clipper to newest Windows printers [9] article
- The Oasis [10] is the largest file archive for CA-Clipper and xBase on the web
- Visual FlagShip Clipper compatible compiler for Linux, Unix and Windows [11]
- Xailer [12] Integrated development environment for Windows
- Harbour Project [13] A 32/64 bit multiplatform Clipper compiler
- From CA-Clipper to Windows in 5 minutes [14] How to install Harbour MiniGUI and compile a windows-exe.
- DBFree - Xbase/Clipper for the web [15] :Active web pages using server-side Xbase scripts, embedded DBF data engine, freeware, based on MaxScript Xbase interpreter (adds xBase/Clipper server-side scripting to IIS / Apache / Xitami web servers).
- DBMax - MaxScript Command Processor [16] :the Xbase interpreter for desktop and web applications.
- The NTK Project [17], WIN32 Gui Framework for (x)Harbour, backward compatible with Clipper and Clip4Win.

References

[1] http://www.grafxsoft.com/clipper.htm
[2] http://www.xharbour.org/index.asp?page=product/extensions
[3] http://groups.google.com/group/comp.lang.clipper
[4] http://groups.google.com/group/comp.lang.clipper.visual-objects
[5] http://www.alaska-software.com/
[6] http://www.vistasoftware.com/
[7] http://www.sourceforge.net/projects/fglib
[8] http://www.davep.org/clipper/
[9] http://www.printfil.com/article/clipper-print-windows-printer.htm
[10] http://www.the-oasis.net/
[11] http://www.fship.com/
[12] http://www.xailer.com
[13] http://www.harbour-project.org
[14] http://raumi75.jimdo.com/2010/02/27/from-ca-clipper-to-windows-in-5-minutes/
[15] http://www.dbfree.org/
[16] http://www.maxsis.it/
[17] http://www.ntkproject.com/

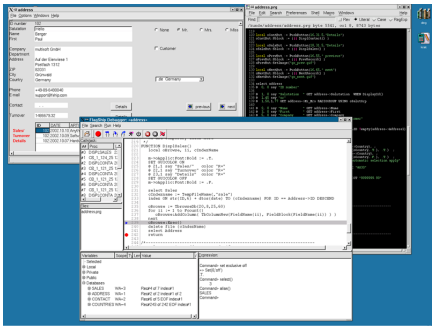
Flagship compiler

FlagShip is both object oriented and procedural programming language, based on the xBase language dialect and conventions. FlagShip is available for and is cross-compatible to different computer platforms, such as Linux, Unix and Microsoft Windows. As a true compiler, it translates the very popular database 4GL xBase source code to native 32-bit or 64-bit executables, using the same source-code and databases.

Recent history

The first FlagShip version was introduced by *multisoft Datentechnik GmbH* in 1992 to port Clipper, dBASE III+, FoxBase and FoxPro applications to different operating systems, i.e. SCO Unix, IBM AIX, Sun Solaris, HP-UX, Siemens SINIX and many other Unix systems. In 1995 also Linux ports became available. In 2002, **Visual FlagShip** *(abbreviated as VFS)* was announced for Linux, and in 2004 additionally for 32/64-bit based Microsoft Windows operating systems. The current VFS product line covers all common 32-bit and 64-bit operating systems (Windows NT, 2000, XP, Vista, 7, Server 2008).

Programming



FlagShip environment
Executing an application, including the embedded source-code debugger

FlagShip is a programming and development tool. As with other compilers, it is designed mainly for professional software developers. But because of its simple interface, it is also perfectly suitable for semi-professionals and advanced computer users, who need to create database applications with minimal programming effort. Thanks to its full Clipper compatibility, it is also best suited for porting already available DOS applications to nearly any Unix, Linux or MS-Windows systems.

There is no learning curve if you are already familiar with any xBase dialect, like dBase, FoxBase, FoxPro, Clipper, Visual Objects etc. There are millions of well trained programmers who have been using this easy but powerful 4GL syntax for years. All of them can directly switch to FlagShip. If you don't have any programming experience with xBase, but are familiar with any other language (C, C++, Java, Pascal, Delphi, Basic, Perl etc.), the learning curve is very short. With about 10 commands, you will be able to create your first application.

Visual FlagShip makes an GUI based application from your available textual xBase code automatically. Of course, because object oriented, you can modify the behavior by yourself too, using either classes or corresponding functions in procedural programming. The same source and the same application supports GUI, textual and stream mode (e.g. for Web or background). The i/o mode is either detected automatically from the current environment (heterogenal application), or can be specified at compile time or at run-time using command-line switch.

For example, these few statements, stored in text file *address.prg*

```
USE address ALIAS adr SHARED NEW
SET COLOR TO "W+/B,GR+/R,W/B,W/B,GR+/BG"
SET GUICOLOR OFF
cls
```

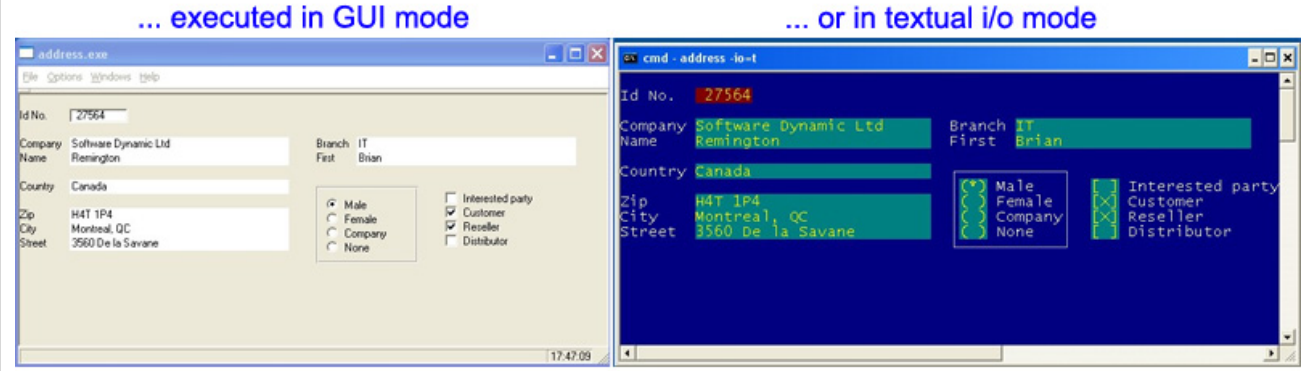
```
@  1, 0 SAY "Id No. " GET adr->IdNum   PICT "999999" VALID IdNum > 0
@  3, 0 SAY "Company" GET adr->Company
@  3,35 SAY "Branch"  GET adr->Branch  WHEN  !empty(adr->Company)
@  4, 0 SAY "Name   " GET adr->Name    VALID !empty(adr->Name)
@  4,35 SAY "First "  GET adr->First
@  6, 0 SAY "Country" GET adr->Country PICTURE "!" + repli("x",24)
@  8, 0 SAY "Zip    " GET adr->Zip     PICT "@!" VALID !empty(adr->Zip)
@  9, 0 SAY "City   " GET adr->City
@ 10, 0 SAY "Street " GET adr->Street

@  6,35,11.4,47 GET adr->Type RADIOGROUP {"Male","Female","Company","None"}
@  7,50   GET  adr->Interest  CHECKBOX  CAPTION "Interested party"
@  8,50   GET  adr->Customer  CHECKBOX  CAPTION "Customer"
@  9,50   GET  adr->Reseller  CHECKBOX  CAPTION "Reseller"
@ 10,50   GET  adr->Distrib   CHECKBOX  CAPTION "Distributor"
READ
```

... and compiled by simply

```
>FlagShip address.prg -o address
```

... creates self-containing executable (i.e. address.exe in Windows)



Additional examples and screenshots are available via the *External links* below.

External links

- FlagShip (multisoft) home page [1]
- VFS screenshots and specs [2]

References

[1] http://www.fship.com
[2] http://www.fship.com/vfs.html

FoxPro 2

FoxPro was a text-based procedurally oriented programming language and DBMS, originally published by Fox Software and later by Microsoft, for MS-DOS, Windows, Macintosh, and UNIX. The final published release of FoxPro was 2.6, after which the development has continued under Visual FoxPro.

Although FoxPro is a Database Management System (DBMS) and it does support relationships between tables, it is not considered a Relational Database Management System (RDBMS), lacking transactional processing.

Sold and supported by Microsoft, there is an active worldwide community of FoxPro users and programmers. FoxPro 2.6 for UNIX (FPU26) has even been successfully installed on Linux and FreeBSD using the Intel Binary Compatibility Standard (ibcs2) support library.

Version information

Operating system compatibility

Extant Versions by OS

Version	FP 2.0	FP 2.5	FP 2.6
MS-DOS	Yes	Yes	Yes
Windows 3.1 to XP	Yes	Yes	Yes
Macintosh	Yes	Yes	Yes
SCO UNIX	No	No	Yes
Linux & FreeBSD	No	No	Yes[1]
Windows 2000	No	No	Yes

Technical aspects

FoxPro 2 included "Rushmore" optimizing engine, which used indices to accelerate data retrieval and updating. Rushmore technology examined every data-related statement, and looked for filter expressions. If one was used, it looked for an index matching the same expression.

In addition, FoxPro2 was originally built on WatCOM C++, which had its own memory extensor - at that time state-of-the-art. FoxPro2 could access expanded and extended memory, using almost all available RAM (DOS). It used some interrupts in absence of the extended memory driver: if no HIMEM.SYS was loaded, FoxPro enabled that mechanism.

Version Timeline

Version	VERSION() returns	EXE Size	EXE Date
FPW 2.6a	FoxPro 2.6a for Windows	2,444 kb	28 September 1994
FPM 2.6a	FoxPro 2.6a for Macintosh	? kb	August 1994
FPD 2.6a	FoxPro 2.6a for DOS	1,788 kb	August 1994
FPW 2.6	FoxPro 2.6 for Windows	2.38 Mb	9 March 1994
FPM 2.6	FoxPro 2.6 for Macintosh	? kb	1993
FPD 2.6	FoxPro 2.6 for DOS	? kb	March 1994
FPU 2.6	FoxPro 2.6 for Unix	2.3 Mb	1993
FPW 2.5	FoxPro 2.5 for Windows	1.63 Mb	January 1993
FPD 2.5	FoxPro 2.5 for DOS	509,013 bytes	February 1993
FPD 2.0	FoxPro 2.0 for DOS	465.86 kb	1991
FPD 1.0	FoxPro 1.0 for DOS	?	1989

References

[1] using the ibcs files from the

External links

- History of FoxPro - Timeline (http://www.foxprohistory.org/foxprotimeline.htm)
- A site devoted to the history of FoxPro (http://www.foxprohistory.org/)
- Virtual FoxPro User Group (http://www.vfug.org/)

Harbour (software)

Harbour Project

Paradigm(s)	multi-paradigm: imperative, functional, object-oriented, reflective
Appeared in	1999
Designed by	Antonio Linares
Developer	Viktor Szakáts and community
Stable release	3.0.0 (17 July 2011)
Preview release	3.2.0dev [1]
Typing discipline	optionally duck, dynamic, safe, partially strong
Dialects	Clipper, Xbase++, FlagShip, FoxPro, xHarbour
Influenced by	dBase, Clipper
Influenced	xHarbour
OS	Cross-platform
License	Open source GPL Compatible
Usual filename extensions	.prg, .ch, .hb, .hbp
Website	http://harbour-project.org/

Harbour is a modern computer programming language, primarily used to create database/business programs. It is a modernized, open sourced and cross-platform version of the older and largely DOS-only Clipper system, which in turn developed from the dBase database market of the 1980s and 90s.

Harbour code using the same databases can be compiled under a wide variety of platforms, including DOS, Microsoft Windows, Linux, Unix variants, several BSD descendants, Mac OS X, MINIX 3, Windows CE, Pocket PC, Symbian, iOS, Android, QNX, VxWorks, OS/2/eComStation, BeOS/Haiku, AIX.

History

The idea of a free software Clipper compiler had been floating around for a long time and the subject has often cropped up in discussion on comp.lang.clipper. Antonio Linares founded the Harbour project and the implementation was started in March, 1999. The name "Harbour" was proposed by Linares, it is a play on a Clipper as a type of ship. Harbour is a synonym for port (where ships dock), and Harbour is a port of the Clipper language.

In 2009 Harbour was substantially redesigned, mainly by Viktor Szakáts and Przemyslaw Czerpak.

Database support

Harbour extends the Clipper Replaceable Database Drivers (RDD) approach. It offers multiple RDDs such as DBF, DBFNTX, DBFCDX, DBFDBT and DBFFPT. In Harbour multiple RDDs can be used in a single application, and new logical RDDs can be defined from combination of other RDDs. The RDD architecture allows for inheritance, so that a given RDD may extend the functionality of other existing RDD(s). Third-party RDDs, like RDDSQL, RDDSIX, RMDBFCDX, Advantage Database Server, and Mediator exemplify some of the RDD architecture features. DBFNTX implementation has almost same functionality of DBFCDX and RDDSIX. NETIO and LetoDB provide remote access over TCP protocol.

Harbour also offers ODBC support by means of an OOP syntax, and ADO support by means of OLE. MySQL, PostgreSQL, SQLite, Firebird, Oracle are examples of databases which Harbour can connect.

xBase technologies often is confused with a RDBMS software. Although this is true, xBase is more than a simple database system as the same time xBase languages using purely DBF can not provide full concept of a real RDBMS.[*citation needed*]

Programming philosophy

Unlike Java which is intended to be write once, run anywhere, Harbour aims to be **write once, compile anywhere**. As the same compiler is available for all of the above operating systems, there is no need for recoding to produce identical products for different platforms, except when operating system dependent features are used. Cross-compiling is supported with MinGW. Under Microsoft Windows, Harbour is more stable but less well-documented than Clipper, but has multi-platform capability and is more transparent, customizable and can run from a USB flash drive.

Under Linux and Windows Mobile, Clipper source code can be compiled with Harbour with very little adaptation. Most software originally written to run on Xbase++, Flagship, FoxPro, xHarbour and others dialects can be compiled with Harbor with some adaptation. As 2010 many efforts have been made to turn the transition from other xBase dialects easier.

Harbour can use the following C compilers, among others: GCC, MinGW, Clang, ICC, Microsoft Visual C++ (6.0+), Borland C++, Watcom C, Pelles C and Sun Studio.

Harbour can make use of multiple Graphic Terminal emulations, including console drivers, and Hybrid Console/GUIs, such as GTWvt, and GTWvg.

Harbour supports external GUIs, free (e.g. HBQt, HWGui, MiniGUI (latest version based on Qt) and QtContribs [2]) and commercial (e.g. FiveWin, Xailer). HBQt is a library provinding bindings to Qt. HBIDE application is a sample of HBQt potencial.

Harbour is 100% Clipper-compatible and supports many language syntax extensions including greatly extended run-time libraries such as OLE, Blat, OpenSSL, FreeImage, GD, hbtip, hbtpathy, PCRE, hbmzip (zlib), hbbz2 (bzip2), cURL, Cairo, its own implementation of CA-Tools, updated NanFor libraries and many others. Harbour has an active development community and extensive third party support.

Any xBase language provides a very productive way to build business and data intensive applications. Harbour is not an exception.

Macro Operator (runtime compiler)

One of the most powerful features of xBase languages is the Macro Operator '&'. Harbour's implementation of the Macro Operator allows for runtime compilation of any valid Harbour expression. Such a compiled expression may be used as a VALUE, i.e. the right side of an assignment (rvalue), but more interestingly, such a compiled expression may be used to resolve the left side (lvalue) of an assignment, i.e. PRIVATE, or PUBLIC variables, or a database FIELD.

Additionally, the Macro Operator may compile and execute function calls, complete assignments, or even list of arguments, and the result of the macro may be used to resolve any of the above contexts in the compiled application. In other words, any Harbour application may be extended and modified at runtime to compile and execute additional code on-demand.

Latest Macro compiler can compile any valid Harbour code including code to pre-process before compile.

Syntax:

```
&( ... )
```

The text value of the expression '...' will be compiled, and the value resulting from the execution of the compiled code is the result.

```
&SomeId
```

is the short form for &(SomeId).

```
&SomeId.postfix
```

is the short form of &(SomeId + "postfix").

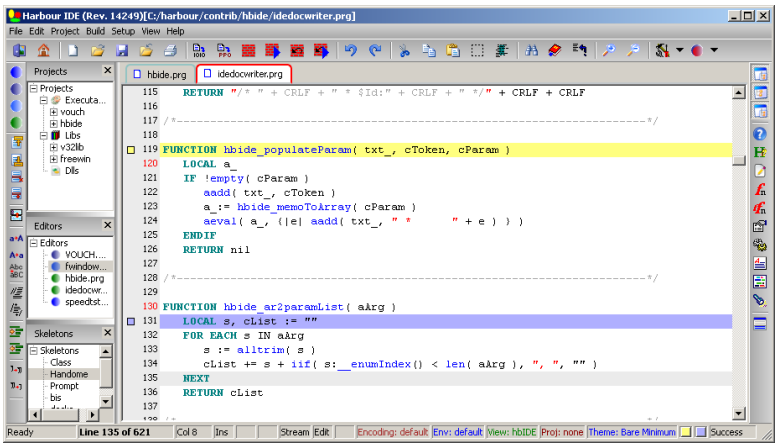
Object Oriented Programming

Programming in an OOP style is a broader issue than a specific library or a specific interface, but OOP programming is something many Clipper programmers have come to expect. CA-Clipper 5.2 and especially 5.3 added a number of base classes, and a matching OOP syntax. Libraries such as Class(y) [3], Fivewin, Clip4Win, and TopClass provide additional OOP functionality.

Harbour has OOP extensions with full support for classes including inheritance, based on Class(y) syntax. OOP syntax in Harbour is very similar to that of earlier Clipper class libraries so it should be possible to maintain legacy Clipper code with minimal changes.

Syntax and semantics

Harbour as every xBase language is case insensitive and can optionally accept keywords written just by first four characters.



Harbour code on HBIDE.

Built-in data types

Harbour has 6 scalar types : Nil, String, Date, Logical, Number, Pointer, and 4 complex types: Array, Object, CodeBlock, and Hash. A scalar holds a single value, such as a string, number, or reference to any other type. Arrays are ordered lists of scalars or complex types, indexed by number, starting at 1. Hashes, or associative arrays, are unordered collections of any type values indexed by their associated key, which may be of any scalar or complex type.

Literal (static) representation of scalar types:

- Nil: *NIL*
- String: *"hello", 'hello', [hello]*
- Date: *0d20100405*
- Logical: *.T., .F.*
- Number: *1, 1.1, –1, 0xFF*

Complex Types may also be represent as literal values:

- Array:

```
{ "String", 1, { "Nested Array" }, .T., FunctionCall(), @FunctionPointer() }
```

- CodeBlock:

```
{ |Arg1, ArgN| Arg1 := ArgN + OuterVar + FunctionCall() }
```

- Hash:

```
{ "Name" => "John", 1 => "Numeric key", { "Nested" => "Hash" } }
```

Hashes may use *any* type including other Hashes as the *Key* for any element. Hashes and Arrays may contain *any* type as the *Value* of any member, including nesting arrays, and Hashes.

Codeblocks may have references to Variables of the Procedure/Function>method in which it was defined. Such Codeblocks may be returned as a value, or by means of an argument passed BY REFERENCE, in such case the Codeblock will "outlive" the routine in which it was defined, and any variables it references, will be a *DETACHED* variable.

Detached variables will maintain their value for as long as a Codeblock referencing them still exists. Such values will be shared with any other Codeblock which may have access to those same variables. If the Codeblock did not outlive its containing routine, and will be evaluated within the lifetime of the routine in which it is defined, changes to its *Detached Variables*(s) by means of its evaluation, will be reflected back at its parent routine.

Codeblocks can be evaluated any number of times, by means of the Eval(*BlockExp*) function.

Variables

All types can be assigned to named variables. Named variable identifiers are 1 to 63 characters long, start with [A-Z|_] and further consist of the characters [A-Z|0–9|_] up to a maximum of 63 characters. Named variables are not case sensitive.

Variables have one of the following scopes:

- *LOCAL*: Visible only within the routine which declared it. Value is lost upon exit of the routine.
- *STATIC*: Visible only within the routine which declared it. Value is preserved for subsequent invocations of the routine. If a STATIC variable is declared before any Procedure/Function/Method is defined, it has a MODULE scope, and is visible within any routine defined within that same source file, it will maintain its life for the duration of the application lifetime.
- *PRIVATE*: Visible within the routine which declared it, and all routines *called* by that routine.
- *PUBLIC*: Visible by *all* routines in the same application.

LOCAL and *STATIC* are resolved at compile time, and thus are much faster than *PRIVATE* and *PUBLIC* variables which are dynamic entities accessed by means of a runtime Symbol table. For this same reason, *LOCAL* and *STATIC* variables are *not* exposed to the Macro compiler, and any macro code which attempts to reference them will generate a runtime error.

Due to the dynamic nature of *PRIVATE* and *PUBLIC* variables, they can be created and destroyed at runtime, can be accessed and modified by means of runtime macros, and can be accessed and modified by Codeblocks created on the fly.

Control structures

The basic control structures include all of the standard dBase, and Clipper control structures as well as additional ones inspired by the C or Java programming languages:

Loops

```
[DO] WHILE ConditionExp
   ...
   [LOOP]
   [EXIT]
END[DO]
```

```
FOR Var := InitExp TO EndExp [STEP StepExp]
   ...
   [LOOP]
   [EXIT]
NEXT
```

```
FOR EACH Var IN CollectionExp
   ...
   [Var:__enumIndex()]
   [LOOP]
   [EXIT]
NEXT
```

- The ... is a sequence of one of more Harbour statements, and square bracketes `[]` denote optional syntax.
- The *Var*:__enumIndex() may be optionally used to retrieve the current iteration index (1 based).
- The *LOOP* statement restarts the current iteration of the enclosing loop structure, and if the enclosing loop is a *FOR* or *FOR EACH* loop, it increases the iterator, moving to the next iteration of the loop.
- The *EXIT* statement immediately terminates execution of the enclosing loop structure.
- The *NEXT* statement closes the control structure and moves to the next iteration of loop structure.

In the *FOR* statement, the *assignment* expression is evaluated prior to the first loop iteration. The *TO* expression is evaluated and compared against the value of the control variable, prior to each iteration, and the loop is terminated if it evaluates to a numeric value greater than the numeric value of the control variable. The optional *STEP* expression is evaluated after each iteration, prior to deciding whether to perform the next iteration.

In *FOR EACH*, the *Var* variable will have the value (scalar, or complex) of the respective element in the collection value. The collection expression, may be an Array (of any type or combinations of types), an Hash Table, or an Object type.

IF statements

```
IF CondExp
   ...
[ELSEIF] CondExp
   ...
[ELSE]
   ...
END[IF]
```

... represents 0 or more *statement(s)*.

The condition expression(s) has to evaluate to a *LOGICAL* value.

SWITCH statements

Harbour supports a SWITCH construct inspired by the C implementation of switch().

```
SWITCH SwitchExp
CASE LiteralExp
   ...
   [EXIT]
[CASE LiteralExp]
   ...
   [EXIT]
[OTHERWISE]
   ...
END[SWITCH]
```

- The *LiteralExp* must be a compiled time resolvable numeric expression, and may involve operators, as long as such operators involve compile time static value.
- The *EXIT* optional statement is the equivalent of the C statement *break*, and if present, execution of the SWITCH structure will end when the EXIT statement is reached, otherwise it will continue with the first statement below the next CASE statement (fall through).

BEGIN SEQUENCE statements

```
BEGIN SEQUENCE
   ...
   [BREAK]
   [Break( [Exp] )]
RECOVER [USING Var]
   ...
END[SEQUENCE]
```

or:

```
BEGIN SEQUENCE
   ...
   [BREAK]
   [Break()]
END[SEQUENCE]
```

The BEGIN SEQUENCE structure allows for a well behaved abortion of any sequence, even when crossing nested procedures/functions. This means that a called procedure/function, may issue a BREAK statement, or a Break() expression, to force unfolding of any nested procedure/functions, all the way back to the first outer BEGIN SEQUENCE structure, either after its respective END statement, or a RECOVER clause if present. The Break statement may optionally pass any type of expression, which may be accepted by the RECOVER statement to allow further recovery handing.

Additionally the Harbour *Error Object* supports *canDefault*, *canRetry* and *canSubstitute* properties, which allows error handlers to perform some preparations, and then request a *Retry Operation*, a *Resume*, or return a Value to replace the expression triggering the error condition.

Alternatively TRY [CATCH] [FINALLY] statements are available on *xhb* library working like the SEQUENCE construct.

Procedures/Functions

```
[STATIC] PROCEDURE SomeProcedureName
[STATIC] PROCEDURE SomeProcedureName()
[STATIC] PROCEDURE SomeProcedureName( Param1 [, ParamsN] )
```

```
INIT PROCEDURE SomeProcedureName
EXIT PROCEDURE SomeProcedureName
```

```
[STATIC] FUNCTION SomeProcedureName
[STATIC] FUNCTION SomeProcedureName()
[STATIC] FUNCTION SomeProcedureName( Param1 [, ParamsN] )
```

Procedures/Functions in Harbour can be specified with the keywords `PROCEDURE`, or `FUNCTION`. Naming rules are same as those for *Variables* (up to 63 characters non case sensitive). Both Procedures and Functions may be qualified by the scope qualifier *STATIC* to restrict their usage to the scope of the module where defined.

The *INIT* or *EXIT* optional qualifiers, will flag the procedure to be automatically invoked just before calling the application startup procedure, or just after quitting the application, respectively. Parameters passed to a procedure/function appear in the subroutine as local variables, and may accept any type, including references.

Changes to argument variables are not reflected in respective variables passed by the calling procedure/function/method unless explicitly passed BY REFERENCE using the @ prefix.

PROCEDURE have no return value, and if used in an Expression context will produce a *NIL* value.

FUNCTION may return any type by means of the RETURN statement, anywhere in the body of its definition.

An example procedure definition and a function call follows:

```
x := Cube( 2 )

FUNCTION Cube( n )
   RETURN n ** 3
```

Sample code

The typical "hello world" program would be:

```
  ? "Hello, world!"
```

Or:

```
  QOut( "Hello, world!" )
```

Or:

```
  Alert( "Hello, world!" )
```

Or, enclosed in an explicit procedure:

```
PROCEDURE Main()

   ? "Hello, world!"
```

```
    RETURN
```

OOP examples

```
#include "hbclass.ch"

PROCEDURE Main()

   LOCAL oPerson

   CLS

   oPerson := Person():New( "Dave" )

   oPerson:Eyes := "Invalid"

   oPerson:Eyes := "Blue"

   Alert( oPerson:Describe() )

   RETURN

CREATE CLASS Person

   VAR Name INIT ""

   METHOD New( cName )
   METHOD Describe()

   ACCESS Eyes INLINE ::pvtEyes
   ASSIGN Eyes( x ) INLINE iif( HB_ISSTRING( x ) .AND. x $ "Blue,Brown,Green", ::pvtEyes := x, Alert( "Invalid value" ) )

   PROTECTED:

   VAR pvtEyes

END CLASS

// Sample of normal Method definition
METHOD New( cName ) CLASS Person

   ::Name := cName

   RETURN Self

METHOD Describe() CLASS Person

   LOCAL cDescription
```

```
IF Empty( ::Name )
   cDescription := "I have no name yet."
ELSE
   cDescription := "My name is: " + ::Name + ";"
ENDIF

IF ! Empty( ::Eyes )
   cDescription += "my eyes' color is: " + ::Eyes
ENDIF

RETURN cDescription
```

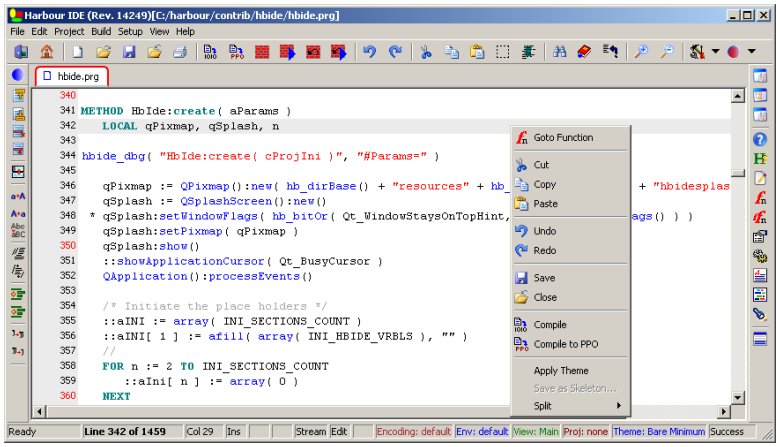
Tools

- hbmk2 – Powerful build tool like make
- hbrun – Shell interpreter for Harbour. Macro compiling allows to run any valid Harbour code as it's being compiled
- hbformat – Formats source code written on Harbour or another dialect according defined rules
- hbpp – Pre-processor, a powerful tool which avoids typical problems found on C language pre-processor
- hbi18n – Tools to localizing text on applications
- hbdoc – Creates documentation for Harbour

All tools are multiplatform.

Development

Today Habour development is leading by Viktor Szakáts with huge collaborations and leading many components of core and contribs by Przemysław Czerpak. HBIDE and some components, specially HBQt, are developed by Pritpal Bedi. Others members send minor changes to the GitHub source repository. As of 2010 Harbour development is keeping vibrant activity.



HBIDE look.

Popularity

Although there is no way to measure popularity of Harbour or xBase, the *TIOBE Programming Community Index* As of June 2006[4] ranked Microsoft Visual FoxPro, a high profile dialect of xBase, on 12th position on programming languages popularity ranking. FoxPro/xBase ranked on 25th position As of August 2010[4]. As of September 2010[4], the Clipper Usenet newsgroups `comp.lang.clipper` [3] is still active. As of August 2010[4] Harbour figured on 16th position on weekly downloads in compiler category and 132nd position on global rank.

xHarbour comparison

xHarbour is a fork of the earlier Harbour project. xHarbour takes a more aggressive approach to implementing new features in the language, while Harbour is more conservative in its approach, aiming first of all for an exact replication of Clipper behaviour and then implementing new features and extensions as a secondary consideration. It should also be noted that Harbour is supported on a wide variety of operating systems while xHarbour only really supports MS Windows and Linux 32-bit.

The Harbour developers have attempted to document all hidden behaviour in the Clipper language and test Harbour-compiled code alongside the same code compiled with Clipper to maintain compatibility.

The Harbour developers explicitly reject extensions to the language where those extensions would break Clipper compatibility. These rejections were soften recently since the new Harbour architecture allow extensions out of the core compiler.

A detailed comparison between extensions implemented in Harbour and xHarbour can be found in source repository of the project on GitHub.

As of 2009–2010, Harbour has seen a huge increase in its adoption while xHarbour decline as can be seen on its mailing list.

GUI libraries and tools

- **hbide** – Integrated Development Environment to help Harbour development and various xBase dialects
- **HwGui** [5] – Open Source cross-platform GUI library for Harbour
- **HMG** [6] – Free / Open Source xBase Win32 / GUI Development System for Harbour
- **MiniGUI** [7] – Free / Open Source xBase Win32 / GUI Development System (a Fork (software development) of both HMG and Harbour)
- **ooHG** [8] – Object Oriented Harbour GUI - a fork "class based and oop programming" of HMG

References

[1] http://sourceforge.net/projects/harbour-project/files/binaries-windows/nightly/
[2] http://sourceforge.net/projects/qtcontribs/
[3] http://www.appsolutions.com/Classy/
[4] http://en.wikipedia.org/w/index.php?title=Harbour_(software)&action=edit
[5] http://sourceforge.net/projects/hwgui/
[6] https://sites.google.com/site/hmgweb/
[7] http://hmgextended.com/home.html
[8] http://www.oohg.org

External links

- Harbour project official site (http://harbour-project.org)
- The Oasis (http://www.the-oasis.net/) Clipper, FoxPro and Xbase++ community repository
- HBIDE (http://hbide.vouch.info/)
- Harbour Developers Mailing List (http://groups.google.com/group/harbour-devel/)
- Harbour Users Mailing List (http://groups.google.com/group/harbour-users/)
- Extensive Harbour documentation, libraries, tools site (http://www.kresin.ru/en/harbour.html)

Visual FoxPro

Visual FoxPro

Visual FoxPro v9 running on Windows XP	
Developer(s)	Microsoft
Final release	v9.0 SP2 / October 16, 2007
Development status	Discontinued
Operating system	Windows 2000, Windows XP and Windows Server 2003
Platform	IA-32
Available in	**IDE:** English, German, Spanish **Runtime:** Above plus French, Chinese, Russian, Czech, Korean
Type	Integrated development environment, programming language
License	Commercial proprietary software
Website	msdn.microsoft.com/vfoxpro [1]

Visual FoxPro is a data-centric object-oriented procedural programming language produced by Microsoft. It is derived from FoxPro (originally known as **FoxBASE**) which was developed by Fox Software beginning in 1984. Fox Technologies merged with Microsoft in 1992, after which the software acquired further features and the prefix "Visual". The last version of FoxPro (2.6) worked under Mac OS, DOS, Windows, and Unix: Visual FoxPro 3.0, the first "Visual" version, reduced platform support to only Mac and Windows, and later versions were Windows-only. The current version of Visual FoxPro is COM-based and Microsoft has stated that they do not intend to create a Microsoft .NET version.

Version 9.0, released in 2004 and updated in 2007, is the final version of the product.

History

FoxPro originated as a member of the class of languages commonly referred to as "xBase" languages, which have syntax based on the dBase programming language. Other members of the xBase language family include Clipper and Recital. (A history of the early years of xBase can be found in the dBase article.)

Visual FoxPro, commonly abbreviated as VFP, is tightly integrated with its own relational database engine, which extends FoxPro's xBase capabilities to support SQL query and data manipulation. Unlike most database management systems, Visual FoxPro is a full-featured, dynamic programming language that does not require the use of an additional general-purpose programming environment. It can be used to write not just traditional "fat client" applications, but also middleware and web applications.

In late 2002, it was demonstrated that Visual FoxPro can run on Linux under the Wine Windows compatibility suite. In 2003, this led to complaints by Microsoft: it was claimed that the deployment of runtime FoxPro code on non-Windows machines violates the End User License Agreement.[2]

Visual FoxPro had a rapid rise and fall in popularity as measured by the TIOBE Programming Community Index.[3] In December 2005, VFP broke into the top 20 for the first time. In June 2006 it peaked at position 12, making it (at the time) a "B" language. By September 2010, FoxPro and its variants had fallen out of the top 50, where it has remained ever since.

In March 2007, Microsoft announced that there will be no VFP 10,[4] thus making VFP9 (released to manufacturing on December 17, 2004) the last commercial VFP release from Microsoft. The support of Version 9 is ongoing with

service packs that were released December 8, 2005 and October 11, 2007.

At the time of the end of life announcement, work on the next release codenamed Sedna (named after a recently discovered dwarf planet) which was built on top of the VFP9 codebase had already begun. "Sedna" is a set of add-ons to VFP 9.0 of xBase components to support a number of interoperability scenarios with various Microsoft technologies including SQL Server 2005, .NET Framework, Windows Vista, Office 2007, Windows Search and Team Foundation Server (TFS). Microsoft released Sedna under the Shared source license on the CodePlex site. Microsoft has clarified that the VFP core will still remain closed source. Sedna was released on January 25, 2008.[5] As of March 2008, all xBase components of the VFP 9 SP2 (including Sedna) were available for community-development on CodePlex.

In late March 2007 a grassroots campaign was started by the Spanish-speaking FoxPro community at MasFoxPro[6] ("MoreFoxPro" in English) to sign a petition to Microsoft to continue updating Visual FoxPro or release it to the community as Open Source. On April 3, 2007 the movement was noted by the technical press[7]

Also on April 3, 2007 Microsoft responded to the petitioner's requests with this statement from Alan Griver:

```
   "We're very aware of the FoxPro community and that played a large
part in what we announced on March 13th. It's never an easy decision to
 announce that we're not going to release another version of a product
and it's one that we consider very carefully.
```

```
   "We're not announcing the end of FoxPro: Obviously, FoxPro
applications will continue to work.  By some of our internal estimates,
 there are more applications running in FoxPro 2.6 than there are in
VFP and FoxPro 2.6 hasn't been supported in many years.  Visual FoxPro
9 will be supported by Microsoft through 2015.
```

```
   "For Microsoft to continue to evolve the FoxPro base, we would need
to look at creating a 64-bit development environment and that would
involve an almost complete rewrite of the core product.  We've also
invested in creating a scalable database with SQL Server, including the
 freely available SQL Server Express Edition. As far as forming a
partnership with a third-party is concerned, we've heard from a number
of large FoxPro customers that this would make it impossible for them
to continue to use FoxPro since it would no longer be from an approved
vendor. We felt that putting the environment into open source on
CodePlex, which balances the needs of both the community and the large
customers, was the best path forward."
```

Code samples

The FoxPro language contains commands quite similar to other programming languages such as Basic. Loops include do, if, while, for, else commands in a usage easily understood by anyone familiar with other programming languages. Commands take the form of "command" and "endcommand"

Some basic syntax samples:

```
FOR i = 1 to 10
   x = x + 6.5
ENDFOR

IF i = 25
```

```
   i = i + 1
ELSE
   i = i + 3
ENDIF

x = 1
DO WHILE x < 50
   x =  x + 1
ENDDO

x = 1
DO WHILE .T.
  x = x + 1
  IF x < 50
   LOOP
  ELSE
   EXIT
  ENDIF
ENDDO

nMonth = MONTH(DATE())
DO CASE
      CASE nMonth <= 3
               MESSAGEBOX("Q1")

      CASE nMonth <= 6
               MESSAGEBOX("Q2")

      CASE nMonth <= 9
               MESSAGEBOX("Q3")

      OTHERWISE
               MESSAGEBOX("Q4")
ENDCASE

FOR EACH oControl IN THISFORM.Controls
  MESSAGEBOX(oControl.Name)
ENDFOR

f = Factorial(10)

FUNCTION Factorial(n)
 LOCAL i,r
 r = 1
 FOR i = n TO 1 STEP -1
  r = r * n
 ENDFOR
```

```
 RETURN r
ENDFUNC
```

Hello World example:

```
MESSAGEBOX("Hello World")
```

Object

```
loForm = CREATEOBJECT("HiForm")
loForm.Show(1)

DEFINE CLASS HiForm AS Form
  AutoCenter = .T.
  Caption = "Hello, World"

  ADD OBJECT lblHi as Label WITH ;
    Caption = "Hello, World!"
ENDDEFINE
```

```
loMine = CREATEOBJECT("MyClass")
? loMine.cProp1   && This will work. (Double-ampersand marks an end-of-line comment)
? loMine.cProp2   && Program Error: Property CPROP2 is not found.

? loMine.MyMethod1()  && This will work.
? loMine.MyMethod2()  && Program Error: Property MYMETHOD2 is not found.

DEFINE CLASS MyClass AS Custom
  cProp1 = "My Property"    && This is a public property
  HIDDEN cProp2     && This is a private (hidden) property
  dProp3 = {}     && Another public property

  PROCEDURE Init()    && Class constructor
    This.cProp2 = "This is a hidden property."
  ENDPROC

  PROCEDURE dProp3_Access    && Property Getter
   RETURN DATE()
  ENDPROC
  PROCEDURE dProp3_Assign(vNewVal)   && Property Setter
    IF VARTYPE(vNewVal) = "D"
     THIS.dProp3 = vNewVal
    ENDIF
  ENDPROC

  PROCEDURE MyMethod1()
    * This is a public method, calling a hidden method that returns
    * the value of a hidden property.
    RETURN This.MyMethod2()
```

```
  ENDPROC

  HIDDEN PROCEDURE MyMethod2()  && This is a private (hidden) method
    RETURN This.cProp2
  ENDPROC
ENDDEFINE
```

Data handling

The language also has extensive database manipulation and indexing commands. The "help" index of commands in VFP 9 has several hundred commands and functions described. The examples below show how to code the creation and indexing of tables, however VFP has table and database builder screens which create the tables and indexes without making you write code.

```
&& Create a table
CREATE TABLE randData (iData I)

&& Populate with random data using xBase and SQL DML commands
FOR i = 1 TO 50
    APPEND BLANK
    REPLACE iData WITH (RAND() * 100)

    INSERT INTO randData (iData) VALUES (RAND() * 100)
ENDFOR

&& Place a structural index on the data
INDEX ON iData TAG iData
CLOSE ALL

&& Display ordered data using xBase-style commands
USE randData
SET ORDER TO iData
GO TOP
LIST NEXT 10  && First 10
GO BOTTOM
SKIP -10
LIST REST     && Last 10
CLOSE ALL

&& Browse ordered data using SQL DML commands
SELECT * ;
  FROM randData ;
  ORDER BY iData DESCENDING
```

ODBC access using SQL passthrough

```
&& Connect to an ODBC data source
LOCAL nHnd
nHnd = SQLCONNECT ("ODBCDSN", "user", "pwd")

&& Execute a SQL command
LOCAL nResult
nResult = SQLEXEC (nHnd, "USE master")
IF nResult < 0
  MESSAGEBOX ("MASTER database does not exist!")
  RETURN
ENDIF

&& Retrieve data from the remote server and stores it in
&& a local data cursor
nResult = SQLEXEC (nHnd, "SELECT * FROM authors", "QAUTHORS")

&& Update a record in a remote table using parameters
PRIVATE cAuthorID, cAuthorName
cAuthorID = "1001"
cAuthorName = "New name"
nResult = SQLEXEC (nHnd, "UPDATE authors SET auth_name = ?cAuthorName WHERE auth_id = ?cAuthorID")

&& Close the connection
SQLDISCONNECT(nHnd)
```

References

[1] http://msdn.microsoft.com/vfoxpro

[2] Visual FoxPro for Linux: A Violation of the EULA? (http://www.linuxjournal.com/article/6869), May 13, 2003, By Ed Leafe, Linux Journal

[3] Tiobe Index History for FoxPro (http://www.tiobe.com/index.php/paperinfo/tpci/FoxPro_xBase.html)

[4] A Message to the Community (http://msdn2.microsoft.com/en-us/vfoxpro/bb308952.aspx)

[5] Microsoft SEDNA download (http://www.microsoft.com/downloads/details.aspx?FamilyId=C04BCF8C-0944-49F0-AC2B-563518CE1D70&displaylang=en)

[6] MasFoxPro (http://www.masfoxpro.com)

[7] Developers petition Microsoft to reconsider FoxPro phase out (http://blogs.zdnet.com/microsoft/?p=361) Posted by Mary Jo Foley (April 3rd, 2007) - All about Microsoft - ZDNet.com

External links

Microsoft pages

- Main Visual FoxPro Microsoft page (http://msdn.microsoft.com/vfoxpro/)
- MSDN FoxPro support board (http://forums.microsoft.com/msdn/showforum.aspx?forumid=60&siteid=1)
- VFP's online help (http://msdn.microsoft.com/en-us/library/ms950411.aspx)
- Visual FoxPro Downloads page (http://msdn.microsoft.com/en-us/vfoxpro/bb190230.aspx)

Other pages

- Visual FoxPro Wiki (http://fox.wikis.com) A repository of FoxPro information (written in VFP)
- A site devoted to the history of FoxPro (http://www.foxprohistory.org)
- VFPx (http://vfpx.codeplex.com) A Visual FoxPro Community effort to create open source add-ons for VFP 9.0
- PortalFox (http://www.portalfox.com) A Hispanoamerican Community Portal for VFP developers

Visual Objects

Visual Objects is an object-oriented computer programming language that is used to create software programs that operate primarily under Windows. Although it can be used as a general-purpose programming tool, it is almost exclusively used to create database programs.

The original Visual Objects project (code-named **Aspen**) was started as part of Nantucket's attempts to bring the Clipper language to Windows, and move from the procedural to the object-oriented style. It also converted Clipper from a p-code system to being a true native compiler and introduced more elements of the C language (such as typed variables), while including Windows extensions (such as COM, ODBC, and later ADO). With its symbol datatype, it offers the ability to form name-based linkages, which may be used to connect menu events to object methods or form direct linkages between server columns and controls.

The Windows version was finally brought to market by Computer Associates. Unfortunately it was released before it was market-ready and in almost head-to-head competition with the first release of Borland's Delphi product. The language is still in use however the last release by GrafX Software was in 2007. The next incarnation of the Visual Objects language is Vulcan.NET, written by GrafX from scratch to be both Visual Objects compatible and be a true CLS compliant .NET language, taking full advantage of the .NET framework.

External links

- GrafXSoft [1]
- Current Visual Objects home [2]
- Vulcan.NET... VO like language for Microsoft .NET [3]

References

[1] http://www.grafxsoft.com/
[2] http://www.cavo.com
[3] http://www.govulcan.net/portal/

XBase

xBase is the generic term for all programming languages that derive from the original dBASE (Ashton-Tate) programming language and database formats. These are sometimes informally known as dBASE "clones". While there was a non-commercial predecessor to the Ashton-Tate product (Vulcan written by Wayne Ratliff), most clones are based on Ashton-Tate's 1986 dBASE III+ release — scripts written in the dBASE III+ dialect are most likely to run on all the clones.

History of the X

Ashton-Tate always maintained that everything relating to dBASE was proprietary, and as a result, filed lawsuits against several of the "clone" software vendors. One effect of this action was to cause the clone vendors to avoid using the term "dBASE": a trademark term held by Ashton-Tate. This gave rise to the creation of the generic term "xBase" meaning "dBASE or dBASE-like." A suggested name that narrowly failed was "*base" (pronounced "star base" and an homage to Vulcan and Star Trek), and some wanted it spelled "X-base" to further differentiate it from the trademark.

Standards effort

By 1987 there were an increasing number of "clone" software products that mimicked dBASE. Each of these products had its own unique set of supported language features and syntax. As such, it was often very difficult to move code developed with one dBASE-like product to run in another one. (This was in contrast to older programming languages such as C or COBOL where due to published official standards, carefully developed code could possibly be run in a wide range of software environments.) While there were many cries for a standard for the dBASE programming language syntax, nothing would happen as long as Ashton-Tate asserted ownership of all-things dBASE.

Once Borland acquired Ashton-Tate in mid-1991 (and was apparently required to drop the lawsuits as an anti-trust related condition of the merger), such standardization efforts were given new life. An ANSI committee (ANSI/X3J19) was officially formed, and began regular meetings in 1992. Marc Schnapp was the first chairman, and the first meeting was held at the Jet Propulsion Laboratory in Pasadena, California which was essentially the birthplace of Vulcan and dBASE II. The group met on a regular basis in a variety of locations over the next few years, and representatives from most major vendors participated. But despite lip service from all the vendors on the need for a standard, no one seemed willing to change their product syntax to match that of a competitor.

Influences over time

In 1989, Microtrend Books published the first "Xbase" cross-reference book (before the term was coined), *The dBASE Language Handbook*, by David M. Kalman, which covered Quicksilver, Clipper, dbxl, dBASE III, dBASE III Plus, dBASE IV, and FoxBase+. At more than 1,000 pages, it compared the execution of commands and functions to enable developers to build and maintain portable applications.

In 1993, Sybex, Inc. (computer books) published the *Xbase Cross Reference Handbook*, by Sheldon M. Dunn, another cross reference of the most commonly used xBase languages at that time—dBASE III+, dBASE IV, FoxPro for DOS, FoxPro for Windows, FoxPro for Macintosh and Clipper 5.1. At 1352 pages and 5.1 pounds shipping weight, the Cross Reference was hardly a handbook, but it provided the xBase community with an up-to-date, all-in-one reference manual, and addressed one of the major documentation problems that the community was facing. The software companies had decided to break their manuals into sections, separating commands from functions, etc., and splitting the (previous) manual into two or three different manuals, and the community was left trying to figure what-was-what and which manual to keep close at hand. 1993 was pivotal for the xBase community

because, as previously noted, Ashton-Tate had earlier sold dBASE as well as the rest of their product line to Borland and Microsoft had purchased FoxPro from Fox Software. Borland had also purchased QuickSilver to get a foot up the development ladder for a dBASE version for Windows (then 3.1). In 1994, Borland launched dBase V for Windows and dBASE V for DOS before selling the dBASE name and product line to dBASE Inc. [1]

In recent years there seems to be a renewed interest in xBase, mostly because of a number of open source, portable, xBase implementations (listed below), and the scripting applicability of the language. While newer desk-top database tools are optimized for mouse usage, xBase has always been "keyboard friendly", which helps make scripting and meta-programming (automating the automation) easier. Meta-programming generally does not work as well with mouse-oriented techniques because automating mouse movements can require calculating and processing of screen coordinates, something most developers find tedious and difficult to debug. xBase is one of the few table-oriented scripting languages still available.

xBase products

As of 2010, xBase is available and still expanding in terms of platform support (operating system), HTML clients, ASP Servers, Windows Scripting Host, and self-contained interpreters. Its current usage tends to be wider in Europe and Asia than in the United States.

The commercially available products:

- Apollo database engine [1] for managing CA-Clipper and FoxPro from Vista Software
- Clipper, Visual Objects (Windows 32) and Vulcan.NET [3] from GrafX Software
- dBase from dBASE Inc.
- DBFView, DBF editor/viewer/converter from Apycom Software
- FlagShip from multisoft GmbH
- Recital from Recital Corp.
- Visual FoxPro is available from Microsoft
- XBase++ from Alaska Software
- xHarbour Builder from xHarbour.com Inc.
- DBF Viewer 2000 from HiBase Group
- DBF Database management tools [2] from Astersoft Co. Ltd.
- DBF Commander Professional [3]
- Visual DBU [4] visual administration of any database/table from DS-Datasoft

Some free versions are also available, including:

- ActiveVFP – Free and open source project for creating web applications with Foxpro
- CLIP – GNU, object oriented, CA-Clipper compatible compiler
- Harbour Project – from active community
- xHarbour – Open Source alternative
- FlagShip free test version
- DBF Commander (free version) [5]
- DBFree [15] – script interpreter for developing xBase applications for the web

Defunct products:

- Vulcan (dBase predecessor)
- DBIII Compiler
- QuickSilver
- dbXL
- Dialog
- Joinner (from Brazilian company Tuxon)

- VP-Info
- Force
- dbFast
- CodeBase
- Multibase
- FoxBase
- Cule.Net from Software Perspectives

Interpreted versus compiled

xBase products generally split into an interpreted camp and the compiler camp. The original product was interpreted, but the "clones", led by Clipper, began creating compiler versions of the product. Compiling improved overall run-time speed and source-code security, but at the expense of an interpreted mode for interactive development or ad-hoc projects.

External links

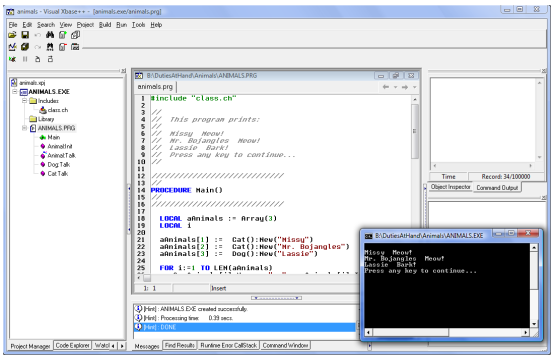
- [news:comp.lang.clipper Clipper Newsgroup]
- The History of FoxPro: People Who Helped FoxPro Become a Legend [6]
- Vulcan.NET Xbase language for Microsoft .NET [7]
- The NTK Project [17], WIN32 Gui Framework for (x)Harbour, backward compatible with Clipper and Clip4Win.
- Xbase (& dBASE) File Format Description [8]
- Active web pages using server-side Xbase scripts, freeware MaxScript Xbase interpreter, embedded DBF data engine [9]
- MaxScript Xbase interpreter for desktop and web applications, [10]

References

[1] http://www.apollodb.com
[2] http://astersoft.com
[3] http://dbf-software.com
[4] http://www.ds-datasoft.de/vdbu_e.html/
[5] http://dbf-software.com/download.html
[6] http://www.foxprohistory.org/people_legend.htm
[7] http://www.GoVulcan.Net/
[8] http://www.clicketyclick.dk/databases/xbase/format/index.html
[9] http://www.dbfree.org
[10] http://www.maxsis.it

XBase++

Xbase++ is an object oriented programming language which has multiple inheritance and polymorphism. It is based on the XBase language dialect and conventions. It is 100% Clipper compatible language supporting multiple inheritance, polymorphism, object oriented programming. It supports the xBase data types, including Codeblocks. With Xbase++ it is possible to generate applications for Windows NT, 95, 98, Me, 2000, XP, VISTA and Windows 7.



Visual Xbase++ with sample *Animals*

Clipper Support

XBase++ supports the old commands @SAY/GET to define data entry forms as well as a graphic editor to create data entry forms similar to Visual FoxPro. It also has a visual development environment [1], support for OEM files (DOS format) and ANSI (Windows), an integrated debugger and a resource compiler to add icons and graphics to the application. It can generate EXE or DLL files.

RDD

Xbase++ supports the Replaceable Database Drivers (RDD, which provide access to multiple database formats) of Clipper through the DatabaseEngines (DBEs). The basic package includes support for DBF, FOX, NTX, CDX, SDF and DEL(delimited). It also supports CORBA 2.0, Visual FoxPro 3.0 to 5.0 database formats, and access to SQL servers.

Birth

XBase++ was born after the decision of Computer Associates to abandon Clipper to develop Visual Objects. The failure of Visual Objects as Clipper substitute empowered the creation of third party libraries and the creation of Clipper syntax compilers.

Source Code samples

```
#include "class.ch"

//
//  This program prints:
//
//  Missy  Meow!
//  Mr. Bojangles  Meow!
//  Lassie  Bark!
//  Press any key to continue...
//

////////////////////////////
//
PROCEDURE Main()
//
```

```
////////////////////////////

  LOCAL aAnimals := Array(3)
  LOCAL i

  aAnimals[1] :=  Cat():New("Missy")
  aAnimals[2] :=  Cat():New("Mr. Bojangles")
  aAnimals[3] :=  Dog():New("Lassie")

  FOR i:=1 TO LEN(aAnimals)
     ? aAnimals[i]:Name + "  " + aAnimals[i]:Talk()
  NEXT i

  WAIT

RETURN

////////////////////////////
//
CLASS Animal
//
////////////////////////////

   EXPORTED:
      VAR Name   READONLY

      METHOD Init
      DEFERRED CLASS METHOD Talk
ENDCLASS

METHOD Animal:Init( cName )
   ::Name := cName
RETURN Self

////////////////////////////
//
CLASS Dog FROM Animal
//
////////////////////////////
   EXPORTED:
   METHOD Talk
ENDCLASS

METHOD Dog:Talk()
RETURN "Bark!"

////////////////////////////
```

```
//
CLASS Cat FROM Animal
//
////////////////////////////
   EXPORTED:
   METHOD Talk
ENDCLASS

METHOD Cat:Talk()
RETURN "Meow!"
```

External links

- Xbase++ web page [2]
- SQLExpress for Xbase++ [3] Object-Oriented ODBC and SQL interface for Xbase++ | www.sqlexpress.net
- Xb2.NET [4] Xbase++ web server & Internet development tool (TCP/IP, HTTP, SOAP, FTP, SSL, XML) | www.xb2.net
- ot4xb [5] Open Source Tools for Xbase++ (ot4xb.dll) www.xbwin.com
- The Oasis [10] Clipper, FoxPro and xBase++ community repository
- xBase Dialects [6]
- ODBC DBE [7]
- Diario Clarín [8] Xbase++: a bridge towards Windows
- DS-Datasoft [9] manufacturer of Xbase++ tools for developer: XClass++, AdsClass++, AFX++, Visual DBU

References

[1] http://www.alaska-software.com/products/vx/intro.shtm
[2] http://www.alaska-software.com/products/xpp/xpp.shtm
[3] http://www.sqlexpress.net/sqlxpp/index.htm
[4] http://www.xb2.net/xb2net/index.htm
[5] http://www.xbwin.com
[6] http://www.masfoxpro.com/index.php?title=XBase&printable=yes
[7] http://www.alaska-software.com/products/subscription.shtm#ODBCDBE
[8] http://www.clarin.com/suplementos/informatica/1998/11/04/t-00301d.htm
[9] http://www.ds-datasoft.de

XHarbour

xHarbour is a free multi-platform extended Clipper compiler, offering multiple graphic terminals (GTs), including console drivers, GUIs, and hybrid console/GUIs. xHarbour is backward-compatible with Clipper and supports many language syntax extensions, greatly extended run-time libraries, and extensive third party support.

Like most dynamic languages, xHarbour is also available as a scripting language (standalone application, linkable library, MS ActiveScript engine [Windows Script Host, HTML, ASP]) utilizing an interpreter written in the xHarbour language.

The xHarbour Usenet newsgroup `[news:comp.lang.xharbour comp.lang.xharbour]` is an active community for discussing xHarbour related questions.

Built-in data types

xHarbour has 6 scalar types : Nil, String, Date, Logical, Number, Pointer, and 4 complex types: Array, Object, CodeBlock, and Hash. A scalar holds a single value, such as a string, number, or reference to any other type. Arrays are ordered lists of scalars or complex types, indexed by number, starting at 1. Hashes, or associative arrays, are unordered collections of any type values indexed by their associated key, which may be of any scalar or complex type.

Literal (static) representation of scalar types:

- Nil: *NIL*
- String: *"hello", 'hello', [hello], or E"hello\n"*
- Date: *ctod("2005-03-17")*
- Logical: *.T., .F.*
- Number: *1, 1.1, -1, 0xFF*

Complex Types may also be represent as literal values:

- Array:

```
{ "String", 1, { "Nested Array" }, .T., FunctionCall(), @FunctionPointer() }
```

- CodeBlock:

```
{ |Arg1, ArgN| Arg1 := ArgN + OuterVar + FunctionCall() }
```

- Hash:

```
{ "Name" => "John", 1 => "Numeric key", { "Nested" => "Hash" } }
```

Hashes may use *any* type including other Hashes as the *Key* for any element. Hashes and Arrays may contain *any* type as the *Value* of any member, including nesting arrays, and Hashes.

Codeblocks may have references to Variables of the Procedure/Function>method in which it was defined. Such Codeblocks may be returned as a value, or by means of an argument passed BY REFERENCE, in such case the Codeblock will "outlive" the routine in which it was defined, and any variables it references, will be a *DETACHED* variable.

Detached variables will maintain their value for as long as a Codeblock referencing them still exists. Such values will be shared with any other Codeblock which may have access to those same variables. If the Codeblock did not outlive its containing routine, and will be evaluated within the lifetime of the routine in which it is defined, changes to its *Detached Variables*(s) by means of its evaluation, will be reflected back at its parent routine.

Codeblocks can be evaluated any number of times, by means of the Eval(*BlockExp*) function.

Variables

All types can be assigned to named variables. Named variable identifiers are 1 to 63 characters long, start with [A-Z|_] and further consist of the characters [A-Z|0-9|_] up to a maximum of 63 characters. Named variables are not case sensitive.

Variables have one of the following scopes:

- *LOCAL*: Visible only within the routine which declared it. Value is lost upon exit of the routine.
- *STATIC*: Visible only within the routine which declared it. Value is preserved for subsequent invocations of the routine. If a STATIC variable is declared before any Procedure/Function/Method is defined, it has a MODULE scope, and is visible within any routine defined within that same source file, it will maintain its life for the duration of the application lifetime.
- *GLOBAL* Visible within any routine defined in the same source module where the GLOBAL variable is declared, as well as any routine of any other source module, which explicitly declares it, by means of the *GLOBAL EXTERNAL* declaration. Both GLOBAL and GLOBAL EXTERNAL declarations must be declared before any Procedure/Function/Method is defined.
- *PRIVATE*: Visible within the routine which declared it, and all routines *called* by that routine.
- *PUBLIC*: Visible by *all* routines in the same application.

LOCAL, *STATIC*, and *GLOBAL* are resolved at compile time, and thus are much faster than *PRIVATE* and *PUBLIC* variables which are dynamic entities accessed by means of a runtime Symbol table. For this same reason, *LOCAL*, *STATIC* and *GLOBAL* variables are *not* exposed to the Macro compiler, and any macro code which attempts to reference them will generate a runtime error.

Due to the dynamic nature of *PRIVATE* and *PUBLIC* variables, they can be created and destroyed at runtime, can be accessed and modified by means of runtime macros, and can be accessed and modified by Codeblocks created on the fly.

Control structures

The basic control structures include all of the standard dBase, and Clipper control structures as well as additional ones inspired by the C or Java programming languages:

Loops

```
[DO] WHILE ConditionExp
   ...
   [LOOP]
   [EXIT]
END[DO]
```

```
FOR Var := InitExp TO EndExp [STEP StepExp]
   ...
   [LOOP]
   [EXIT]
NEXT
```

```
FOR EACH Var IN CollectionExp
   ...
   [HB_EnumIndex()]
   [LOOP]
```

```
   [EXIT]
NEXT
```

- The ... is a sequence of one of more xHarbour statements, and square bracketes `[]` denote optional syntax.
- The *HB_EnumIndex()* may be optionally used to retrieve the current iteration index (1 based).
- The *LOOP* statement restarts the current iteration of the enclosing loop structure, and if the enclosing loop is a *FOR* or *FOR EACH* loop, it increases the iterator, moving to the next iteration of the loop.
- The *EXIT* statement immediately terminates execution of the enclosing loop structure.
- The *NEXT* statement closes the control structure and moves to the next iteration of loop structure.

In the *FOR* statement, the *assignment* expression is evaluated prior to the first loop iteration. The *TO* expression is evaluated and compared against the value of the control variable, prior to each iteration, and the loop is terminated if it evaluates to a numeric value greater than the numeric value of the control variable. The optional *STEP* expression is evaluated after each iteration, prior to deciding whether to perform the next iteration.

In *FOR EACH*, the *Var* variable will have the value (scalar, or complex) of the respective element in the collection value. The collection expression, may be an Array (of any type or combinations of types), an Hash Table, or an Object type.

IF statements

```
IF CondExp
   ...
[ELSEIF] CondExp
   ...
[ELSE]
   ...
END[IF]
```

... represents 0 or more *statement(s)*.

The condition expression(s) has to evaluate to a *LOGICAL* value.

DO CASE statements

```
DO CASE
   CASE CondExp
      ...
   [CASE CondExp]
      ...
   [OTHERWISE]
      ...
END[CASE ]
```

Above construct is logically equivalent to:

```
IF CondExp
   ...
ELSEIF CondExp
   ...
[ELSEIF CondExp]
   ...
[ELSE]
```

```
   ...
END[IF]
```

SWITCH statements

xHarbour supports a SWITCH construct inspired by the C implementation of switch().

```
SWITCH SwitchExp
   CASE LiteralExp
      ...
      [EXIT]
```

```
   [CASE LiteralExp]
      ...
      [EXIT]

   [DEFAULT]
      ...
END
```

- The *LiteralExp* must be a compiled time resolvable numeric expression, and may involve operators, as long as such operators involve compile time static value.
- The *EXIT* optional statement is the equivalent of the C statement *break*, and if present, execution of the SWITCH structure will end when the EXIT statement is reached, otherwise it will continue with the first statement below the next CASE statement (fall through).

BEGIN SEQUENCE statements

```
BEGIN SEQUENCE
   ...
   [BREAK]
   [Break([Exp])]
RECOVER [USING Var]
   ...
END[SEQUENCE]
```

or:

```
BEGIN SEQUENCE
   ...
   [BREAK]
   [Break()]
END[SEQUENCE]
```

The BEGIN SEQUENCE structure allows for a well behaved abortion of any sequence, even when crossing nested procedures/functions. This means that a called procedure/function, may issue a BREAK statement, or a Break() expression, to force unfolding of any nested procedure/functions, all the way back to the first outer BEGIN SEQUENCE structure, either after its respective END statement, or a RECOVER clause if present. The Break statement may optionally pass any type of expression, which may be accepted by the RECOVER statement to allow further recovery handing.

Additionally the xHarbour Error Object supports canDefault, canRetry and canSubstitute properties, which allows error handlers to perform some preparations, and then request a Retry Operation, a Resume, or return a Value to replace the expression triggering the error condition.

TRY [CATCH] [FINALLY] statements

```
TRY
   ...
   [BREAK]
   [Break([Exp])]
   [Throw([Exp])]
CATCH [Var]
   ...
END
```

```
TRY
   ...
   [BREAK]
   [Break([Exp])]
   [Throw([Exp])]
CATCH [Var]
   ...
FINALLY
   ...
END
```

or:

```
TRY
   ...
   [BREAK]
   [Break([Exp])]
   [Throw([Exp])]
FINALLY
   ...
END
```

The TRY construct is very similar to the BEGIN SEQUENCE construct, except it automatically integrates error handling, so that any error will be intercepted, and recovered by means of the CATCH statement or forwarded to an outer CATCH handler otherwise. The FINALLY section is guaranteed to be executed before the TRY or CATCH sections forward flow control by means of RETURN, BREAK, or THROW.

Procedures/Functions

```
[STATIC] PROCEDURE SomeProcedureName
[STATIC] PROCEDURE SomeProcedureName()
[STATIC] PROCEDURE SomeProcedureName( Param1' [, ParamsN] )
```

```
INIT PROCEDURE SomeProcedureName
EXIT PROCEDURE SomeProcedureName
```

```
[STATIC] FUNCTION SomeProcedureName
[STATIC] FUNCTION SomeProcedureName()
[STATIC] FUNCTION SomeProcedureName( Param1' [, ParamsN] )
```

Procedures/Functions in xHarbour can be specified with the keywords `PROCEDURE`, or `FUNCTION`. Naming rules are same as those for *Variables* (up to 63 characters non case sensitive). Both Procedures and Functions may be qualified by the scope qualifier *STATIC* to restrict their usage to the scope of the module where defined.

The *INIT* or *EXIT* optional qualifiers, will flag the procedure to be automatically invoked just before calling the application startup procedure, or just after quitting the application, respectively. Parameters passed to a procedure/function appear in the subroutine as local variables, and may accept any type, including references.

Changes to argument variables are not reflected in respective variables passed by the calling procedure/function/method unless explicitly passed BY REFERENCE using the @ prefix.

PROCEDURE have no return value, and if used in an Expression context will produce a *NIL* value.

FUNCTION may return any type by means of the RETURN statement, anywhere in the body of its definition.

An example procedure definition and a function call follows:

```
x := Cube( 2 )

FUNCTION Cube( n )
RETURN n ** 3
```

Database support

xHarbour extends the Clipper Replaceable Database Drivers (RDD) approach. It offers multiple RDDs such as DBF, DBFNTX, DBFCDX, DBFDBT, and DBFFPT. In xHarbour multiple RDDs can be used in a single application, and new logical RDDs can be defined from combination of other RDD. The RDD architecture allows for inheritance, so that a given RDD may extend the functionality of other existing RDD(s). 3rd party RDDs, like RDDSQL, RDDSIX, RMDBFCDX, *Advantage Database Server*, and *Mediator* exemplify some of the RDD architecture features.

xHarbour also offers ODBC support by means of an OOP syntax, and ADO support by means of OLE.

Macro Operator (runtime compiler)

One of the most powerful features of the xBase languages is the MACRO Operator '&'. xHarbour's implementation of the Macro Operator allows for runtime compilation of any valid xHarbour expression. Such compiled expression may be used as a VALUE, i.e. the right side of an Assignment, but more interestingly, such compiled expression may be used to resolve the LEFT side of an assignment, i.e. PRIVATE, or PUBLIC variables, or Database FIELD.

Additionally the Macro Operator may compile and execute function calls, complete assignments, or even list of arguments, and the result of the macro may be used to resolve any of the above contexts in the compiled application. IOW, any xHarbour application may be extended, and/or modified in runtime, to compile and execute additional code on demand.

The xHarbour implementation of this feature is so complete that the xHarbour interpreter, xbScript, uses it heavily, to compile xHarbour scripts.

Syntax:

```
&( ... )
```

The text value of the expression '...' will be compiled, and the value resulting from the execution of the compiled code is the result.

```
&SomeId
```

is the short form for &(SomeId).

```
&SomeId.postfix
```

is the short form of &(SomeId + "postfix").

Sample code

The typical "hello world" program would be:

```
? "Hello, world!"
```

Or:

```
QOut( "Hello, world!" )
```

Or:

```
Alert( "Hello, world!" )
```

Or, enclosed in an explicit procedure:

```
PROCEDURE Main()

   ? "Hello, world!"

RETURN
```

OOP examples

```
#include "hbclass.ch"


PROCEDURE Main()


   LOCAl oPerson := Person( "Dave" )


   oPerson:Eyes := "Invalid"


   oPerson:Eyes := "Blue"


   Alert( oPerson:Describe() )
RETURN


CLASS Person
   DATA Name INIT ""


   METHOD New() CONSTRUCTOR


   ACCESS Eyes INLINE ::pvtEyes
   ASSIGN Eyes( x ) INLINE IIF( ValType( x ) == 'C' .AND. x IN "Blue,Brown,Green", ::pvtEyes := x, Alert( "Invalid value" ) )


   // Sample of IN-LINE Method definition
   INLINE METHOD Describe()
      LOCAL cDescription


      IF Empty( ::Name )
         cDescription := "I have no name yet."
      ELSE
         cDescription := "My name is: " + ::Name + ";"
      ENDIF


      IF ! Empty( ::Eyes )
         cDescription += "my eyes' color is: " + ::Eyes
      ENDIF
   ENDMETHOD


   PRIVATE:
      DATA pvtEyes
ENDCLASS


// Sample of normal Method definition.
METHOD New( cName ) CLASS Person


  ::Name := cName


RETURN Self
```

Scripting

xHarbour is also available as an interpreted language in few flavors of scripting engines.

- *Stand alone Interpreter*: Portable, self-contained, interpreter xBaseScript.
- *ActiveScript*: Microsoft ActiveScript compliant OLE DLL, which supports xHarbour scripting in:
  - Windows Script Host (WSH).
  - Internet Explorer, HTML client side scripting.
  - IIS, and any other ASP compliant server.

External links

- Official site [1]
- Report Generator for xHarbour [2]
- Object Oriented Harbour GUI (ooHG) [8]
- FiveWin [3]
- ViaOpen [4]
- Xailer [12]
- HbWxW (wxWidgets Bindings) [5]

References

[1] http://www.xHarbour.org
[2] http://www.paritetsoft.ru/frh.htm
[3] http://www.FiveTechSoft.com
[4] http://www.viaopen.com
[5] http://harbour.fm.interia.pl/

Other Query Languages

QUEL query languages

QUEL is a relational database query language, based on tuple relational calculus, with some similarities to SQL. It was created as a part of the Ingres DBMS effort at University of California, Berkeley, based on Codd's earlier suggested but not implemented *Data Sub-Language ALPHA*. QUEL was used for a short time in most products based on the freely available Ingres source code, most notably PostgreSQL. As Oracle and DB2 gained market share in the early 1980s, most companies then supporting QUEL moved to SQL instead. QUEL continues to be available as a part of the Ingres DBMS, although no QUEL-specific language enhancements have been added for many yearsWikipedia:Manual of Style/Dates and numbers#Chronological items.

Usage

QUEL statements are always defined by *tuple variables*, which can be used to limit queries or return result sets. Consider this example, taken from one of the first original Ingres papers:

> *Example* 1.1. Compute salary divided by age-18 for employee Jones.
>
> range of E is EMPLOYEE
>
> retrieve into W
>
> (COMP = E.Salary / (E.Age - 18))
>
> where E.Name = "Jones"
>
> Here E is a tuple variable which ranges over the EMPLOYEE relation, and all tuples in that relation are found which satisfy the qualification E.Name = “Jones.” The result of the query is a new relation W, which has a single domain COMP that has been calculated for each qualifying tuple.

An equivalent SQL statement is:

```
select (e.salary / (e.age - 18)) as comp
from employee as e
where e.name = "Jones"
```

QUEL is generally more "normalized" than SQL.[*citation needed*] Whereas every major SQL command has a format that is at least somewhat different from the others, in QUEL a single syntax is used for all commands.[*citation needed*]

For instance, here is a sample of a simple session that creates a table, inserts a row into it, and then retrieves and modifies the data inside it and finally deletes the row that was added (assuming that name is a unique field).

```
create student(name = c10, age = i4, sex = c1, state = c2)
range of s is student
append to s (name = "philip", age = 17, sex = "m", state = "FL")
retrieve (s.all) where s.state = "FL"
replace s (age=s.age+1)
retrieve (s.all)
delete s where s.name="philip"
```

Here is a similar set of SQL statements:

```
 create table student(name char(10), age int, sex char(1), state
char(2))
 insert into student (name, age, sex, state) values ("philip", 17, "m",
 "FL")
 select * from student where state = "FL"
 update student set age=age+1
 select * from student
 delete from student where name="philip"
```

Note that syntax varies significantly between commands, and that even similar commands like `insert` and `update` use different styles.

Another feature of QUEL was a built-in system for moving records en-masse into and out of the system. Consider this command:

```
copy student(name=c0, comma=d1, age=c0, comma=d1, sex=c0, comma=d1, address=c0, nl=d1)
 into "/student.txt"
```

which creates a comma-delimited file of all the records in the student table. The d1 indicates a delimiter, as opposed to a data type. Changing the `into` to a `from` reverses the process. Similar commands are available in many SQL systems, but usually as external tools, as opposed to being internal to the SQL language. This makes them unavailable to stored procedures.

QUEL has an extremely powerful aggregation capability. Aggregates can be nested, and different aggregates can have independent by-lists and/or restriction clauses. For example:

```
 retrieve (a=count(y.i by y.d where y.str = "ii*" or y.str = "foo"),b=max(count(y.i by y.d)))
```

This example illustrates one of the arguably less desirable quirks of QUEL, namely that all string comparisons are potentially pattern matches. `y.str = "ii*"` matches all y.str values starting with ii.

References

Further reading

- C. J. Date: A Critique of the SQL Database Language (https://www.cs.duke.edu/courses/spring03/cps216/papers/date-1983.pdf). SIGMOD Record 14(3): 8-54, 1984.

Query by Example

Query by Example (**QBE**) is a database query language for relational databases. It was devised by Moshé M. Zloof at IBM Research during the mid-1970s, in parallel to the development of SQL. It is the first graphical query language, using visual tables where the user would enter commands, example elements and conditions. Many graphical front-ends for databases use the ideas from QBE today. Originally limited only for the purpose of retrieving data, QBE was later extended to allow other operations, such as inserts, deletes and updates, as well as creation of temporary tables.

The motivation behind QBE is that a parser can convert the user's actions into statements expressed in a database manipulation language, such as SQL. Behind the scenes, it is this statement that is actually executed. A suitably comprehensive front-end can minimize the burden on the user to remember the finer details of SQL, and it is easier and more productive for end-users (and even programmers) to select tables and columns by selecting them rather than typing in their names,

In the context of information retrieval, QBE has a somewhat different meaning. The user can submit a document, or several documents, and ask for "similar" documents to be retrieved from a document database. Similarity search is based comparing document vectors (see Vector Space Model).

QBE is a seminal work in end-user development, frequently cited in research papers as an early example of this topic.

Currently, QBE is supported in several relational database front ends, notably Microsoft Access, which implements "Visual Query by Example", as well as Microsoft SQL Server Enterprise Manager. It is also implemented in several object-oriented databases (e.g. in db4o).

QBE is based on the logical formalism called tableau query, although QBE adds some extensions to that, much like SQL is based on the relational algebra.

Example

A simple example using the Suppliers and Parts database is given here to illustrate how QBE works.

S	S#	SNAME	OWNER	SCITY
	P.SX		J. DOE	ROME

Simple QBE Example

As a general technique

The term also refers to a general technique influenced by Zloof's work whereby only items with search values are used to "filter" the results. It provides a way for a software user to perform queries without having to know a query language (such as SQL). The software can automatically generate the queries for the user (usually behind the scenes). Here are some examples:

Example Form **B**:

```
.....Name: Bob
..Address:
.....City:
....State: TX
..Zipcode:
```

Resulting SQL:

```
SELECT * FROM Contacts WHERE Name='Bob' AND State='TX'
```

Note how blank items do not generate SQL terms. Since "Address" is blank, there is no clause generated for it.

Example Form **C**:

```
.....Name:
..Address:
.....City: Sampleton
....State:
..Zipcode: 12345
```

Resulting SQL:

```
SELECT * FROM Contacts WHERE City='Sampleton' AND Zipcode='12345'
```

More advanced versions of QBE have other comparison operator options, often via a pull-down menu, such as "Contains", "Starts With", "Greater-Than", and so forth.

References

Sources

- Zloof, Moshé M (May 1975), "Query by Example", *NCC* (proceedings) **44**, Anaheim, CA, US: AFIPS.
- Ramakrishnan, Raghu; Gehrke, Johannes, "6. QBE" (http://www.cs.wisc.edu/~dbbook/openAccess/thirdEdition/qbe.pdf) (PDF), *Database Management Systems* (3rd ed.), Wisc.
- Date, Christopher 'Chris' J (2004), "8. Relational Calculus", *An Introduction to Database Systems*, Addison-Wesley Pearson, pp. 242–7, ISBN 0-321-18956-6.
- *Oracle Definitions* (http://searchoracle.techtarget.com/sDefinition/0,,sid41_gci214554,00.html), Tech target.
- Zaiane, "5" (http://www.cs.sfu.ca/CC/354/zaiane/material/notes/Chapter5/node2.html), *CC 354 notes*, CA: SFI.

External links

- *Query by Example for PostgreSQL* (http://pgfoundry.org/projects/qbe), Pg foundry.
- "Query by Example in Java using Hibernate" (http://docs.jboss.org/hibernate/core/3.3/reference/en/html/querycriteria.html#querycriteria-examples), *Jboss*.
- "OptiqueVQS – Towards an Ontology-based Visual Query System for Big Data" (http://www.ahmetsoylu.com/wp-content/uploads/2013/10/soylu_medes2013.pdf), *Optique*.

SQR

SQR (Hyperion SQR Production Reporting, Part of OBIEE) is a programming language designed for generating reports from database management systems. The name is an acronym of **S**tructured **Q**uery **R**eporter, which suggests its relationship to SQL (Structured Query Language). Any SQL statement can be embedded in an SQR program.

History

SQ Software created SQR in the mid 1980s. It had a marketing agreement with D & N Systems, which changed its name to SQL Solutions and was later acquired by Sybase Inc. Sybase purchased SQ Software in the early 1990s. To avoid competing directly with Oracle Corporation, Sybase had a marketing and development agreement with MITI for the Oracle database versions of SQR. MITI acquired the full rights to SQR in the mid 1990s. MITI changed its name to SQRiBE Technologies in 1997. Brio Technology acquired SQRiBE in August, 1999. Brio Technology later changed its name to Brio Software. Brio licensed the compiler source code to PeopleSoft Inc. sometime around 2000. Hyperion Solutions Corporation acquired Brio Software in October, 2003. Oracle Corporation acquired PeopleSoft in December, 2004. And as of March 2007 Oracle Corporation has acquired Hyperion Solutions.

SQR-Related Products

- ORACLE: Hyperion SQR Production Reporting - System 9 (Release 9.3.1, 2008);
- ORACLE: PeopleSoft Enterprise Tools & Technology (PeopleTools, Release 8.52, 2011)

General Components

1. SQR Server
2. SQR Viewer
3. SQR Print
4. SQR Execute
5. SQR Workbench for Windows; SQR Developer

Features

SQR is notable for its powerful database and printing functions. It can embed any SQL statement almost anywhere in a program. There is a version of SQR that can use multidimensional databases like Essbase. It can combine database reads with print instructions. It flexibly formats data breaks and page breaks. It prints variable fonts, sizes, and colors. It has a graph generation command that offers dozens of parameters for adjusting content and appearance.

Syntax

SQR has four scalar data types. The first three are numeric (variables begin with “#”), character string (variables begin with “$”), and date (variables begin with “$”, same as with character string variables.). Date variables must be declared, to be distinguished from character string variables. There is the option to declare numeric variables to specify them more precisely (integer, floating point, etc.). The last data type is a database column (variables begin with “&”). The values of database columns are set only by a SQL “select” statement; no other command can change their values.

SQR has arrays like those of the C programming language or COBOL. An array has one or more fields, each field either a numeric, string, or date variable. Each field may have multiple occurrences, giving SQR the equivalent of two-dimensional arrays. SQR has special commands that manipulate multiple items within a single array. There are also many commands that cannot use an array element in place of a scalar variable.

SQR has four control structures. The first three are if-else-end, while-end, and evaluate (like case or switch). The fourth is the embedded SQL “select” statement, which allows SQR statements to be executed for each row of data in a loop.

SQR has commands to open, read, write, append, and close computer files. File input and output is sequential and record-oriented only; random access is not supported.

.QL

.QL

Paradigm(s)	multi-paradigm, logic-paradigm, object-oriented-paradigm
Appeared in	2007
Developer	Semmle
Typing discipline	static, strong
Major implementations	SemmleCode
Influenced by	Datalog

.QL (pronounced "dot-cue-el") is an object-oriented query language used to retrieve data from relational database management systems. It is reminiscent of the standard query language SQL and the object-oriented programming language Java. .QL is an object-oriented variant of a logical query language called Datalog. Hierarchical data can therefore be naturally queried in .QL in a recursive manner.

Queries written in .QL are optimised, compiled into SQL and can then be executed on any major relational database management system. .QL query language is being used in SemmleCode to query a relational representation of Java programs.

.QL is developed at Semmle Limited and is based on the company's proprietary technology.

Language Features

.QL has several language features to make queries concise, intuitive and reusable:

- Extensible type hierarchy
- Methods and predicates
- Definition before use

Example query

The sample query below illustrates use of .QL to query a Java program. This is how one would select all classes that contain more than ten public methods:

```
from Class c, int numofm
where numofm = count(Method m| m.getDeclaringType()=c
                     and m.hasModifier("public"))
      and numofm > 10
select c.getPackage(), c, numofm
```

In fact, this query selects not only all classes with more than ten public methods, but also their corresponding packages and the number of methods each class has.

References

- Hervé Gallaire and Jack Minker. Logic and Databases. Plenum Press, New York,1978.
- Serge Abiteboul and Paris C. Kanellakis. Object identity as a query language primitive. In SIGMOD Rec, pages 159–173, ACM Press, 1989.
- Oege de Moor, Elnar Hajiyev and Mathieu Verbaere. Object-oriented queries over software systems. In Proceedings of the 2007 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation (PEPM), page 91, ACM Press, 2007.

External links

- Semmle Limited [1] creators of .QL

References

[1] http://semmle.com/

Yahoo! query language

Yahoo! Query Language (**YQL**) is an SQL-like query language created by Yahoo! as part of their Developer Network. YQL is designed to retrieve and manipulate data from APIs through a single Web interface, thus allowing mashups that enable developers to create their own applications.

Initially launched in October 2008 with access to Yahoo APIs, February 2009 saw the addition of open data tables from third parties such as Google Reader, the *Guardian*, and *The New York Times*. Some of these APIs still require an API key to access them. On April 29th of 2009, Yahoo introduced the capability to execute the tables of data built through YQL using JavaScript run on the company's servers for free.

References

External links

- Official site (http://developer.yahoo.com/yql/), including the YQL console

YQL (programming language)

Yahoo! Query Language (**YQL**) is an SQL-like query language created by Yahoo! as part of their Developer Network. YQL is designed to retrieve and manipulate data from APIs through a single Web interface, thus allowing mashups that enable developers to create their own applications.

Initially launched in October 2008 with access to Yahoo APIs, February 2009 saw the addition of open data tables from third parties such as Google Reader, the *Guardian*, and *The New York Times*. Some of these APIs still require an API key to access them. On April 29th of 2009, Yahoo introduced the capability to execute the tables of data built through YQL using JavaScript run on the company's servers for free.

References

External links

- Official site (http://developer.yahoo.com/yql/), including the YQL console

YANG

In Network management systems, it is necessary to get/set parameters that reflect the state of the equipment being managed. So there is a need to represent data and there is a need to retrieve/set the data. So, any network management protocol would configure and get status/notifications from the equipment.

There are various protocols to get/set the data. Ex: SNMP, NetConf, TL1 etc. Each protocol also creates its own data format, which the protocol needs to understand/interpret. Building network management systems also becomes cumbersome, since there is an inevitable need for integration across various NMS systems into OSS systems. Companies earmark good budget for building the NMS systems - reasonable chunk of time of which goes into defining the data format. YANG, a data modeling language, cuts across all of these protocols and creates a well-defined standard that could be used by any protocol.

'YANG' [1] is a data modeling language for the NETCONF network configuration protocol. The YANG data modeling language was developed by the NETMOD working group in the IETF and was published as RFC 6020 in October 2010. The data modeling language can be used to model both configuration data as well as state data of network elements. Furthermore, YANG can be used to define the format of event notifications emitted by network elements and it allows data modelers to define the signature of remote procedure calls that can be invoked on network elements via the NETCONF protocol.

YANG is a modular language representing data structures in an XML tree format. The data modeling language comes with a number of builtin data types. Additional application specific data types can be derived from the builtin data types. More complex reusable data structures can be represented as groupings. YANG data models can use XPATH expressions to define constraints on the elements of a YANG data model.

References

[1] https://tools.ietf.org/html/rfc6020

External links

- MG-SOFT Visual YANG Designer (http://www.mg-soft.com/mgYangDesigner.html), a user friendly YANG definition file creator/editor/modeler/builder/designer, based on MG-SOFT's own YANG compiler, implemented in Java.
- ChampNMS MasterYANG (http://www.champnms.com), MasterYANG is a NETCONF YANG Data Model Designer, Visualizer and Editor.
- yangbuilder (https://bitbucket.org/novakmi/yangbuilder), a groovy (http://groovy.codehaus.org/) builder for YANG.

WQL

Windows Management Instrumentation Query Language (**WQL**) is Microsoft's implementation of the CIM Query Language (CQL), a query language for the Common Information Model (CIM) standard from the Distributed Management Task Force (DMTF). It is a subset of the standard ANSI SQL with minor semantic changes.[1] A basic WQL query remains fairly understandable for people with basic SQL knowledge.

WQL is dedicated to WMI and is designed to perform queries against the CIM repository to retrieve information or get event notifications.

Example

As an example, the following WQL query selects all the drives on a computer that have less than 2 MB of free space:[2]

```
SELECT * FROM Win32_LogicalDisk WHERE FreeSpace < 2097152
```

References

[1] WQL (SQL for WMI) (http://msdn2.microsoft.com/en-us/library/aa394606.aspx)
[2] WMI Queries (http://msdn2.microsoft.com/en-us/library/ms186146(VS.80).aspx)

External links

- Querying with WQL (http://msdn.microsoft.com/en-us/library/aa392902.aspx)
- WQL Operators (http://msdn.microsoft.com/en-us/library/aa394605.aspx)
- WQL-Supported Date Formats (http://msdn.microsoft.com/en-us/library/aa394607.aspx)
- WQL-Supported Time Formats (http://msdn.microsoft.com/en-us/library/aa394608.aspx)
- WQL (SQL for WMI) (http://msdn.microsoft.com/en-us/library/aa394606.aspx)
- Using WQL with the WMI Provider for Server Events (http://msdn.microsoft.com/en-us/library/ms180524.aspx)
- WMI Queries (http://msdn.microsoft.com/en-us/library/ms186146(VS.80).aspx)
- Learn WMI Query Language using PowerShell (http://www.ravichaganti.com/blog/?p=1845)

Versa (query language)

Versa is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. Versa differs from most other RDF query languages, which are typically based on SQL or specialized XML vocabularies. Although the design of Versa was inspired by XPath, its compact, functional syntax also somewhat resembles Lisp. As of 2006[1], the only Versa implementation is in the Python-language, open source 4Suite XML framework.

History and status

Versa's origins can be traced to 2000, when professional XML consultancy Fourthought, Inc. began developing RIL, an open, XML-based RDF Inference Language. RIL was implemented briefly in Fourthought's 4Suite Server product, which allowed for persistent storage and querying of an RDF model and associated XML document store.

In October 2001, the query portion of RIL was spun off into a separate project named Versa, with the intent that after Versa stabilized, development would resume to establish RIL as a formal language for working Versa query results. RIL development never resumed, however; inference abilities in 4Suite were easily handled by XSLT extensions and did not need a separate language.

Versa initially had limits that required the language to be redesigned. The overhaul was performed in late 2001 and early 2002 by software engineers from Sun Microsystems and Fourthought, to facilitate internal knowledge management applications at Sun and to drive applications that were in development for Fourthought's other clients. 4Suite's Versa library was used as the reference implementation, and a draft specification was published.

Versa development slowed in 2002, although Sun and Fourthought continued to work together to develop Versa based applications through 2004. In mid-2005, Chimezie Ogbuji and Daniel Krech began planning to fold the abilities of 4Suite's outdated RDF libraries (4RDF), including Versa support, into RDFLib, which offers complementary features and would allow Versa to be used independently of 4Suite. This coincided with renewed interest in refining the Versa draft specification and publishing a "1.0" version.

As of 2006, the development of Versa is being coordinated primarily by Chimezie and Uche Ogbuji of Fourthought.

Syntax examples

Get the URIs of all known resources:

```
all()
```

Get the URIs of all known instances of type `edu:Subject`:

```
type(edu:Subject)
```

Get the `rdfs:label`s of all `edu:Subject`s having one or more `rdfs:label`:

```
type(edu:Subject) - rdfs:label -> *
```

Get the URIs of all `edu:Subject`s having a `rdfs:label` matching `"Russian language"`:

```
type(edu:Subject) |- rdfs:label -> eq("Russian language")
```

Get the URIs of all "super-`edu:Subject`s" (transitively) of the `edu:Subject` identified by `"http://en.wikipedia.org/wiki/Russian_language"`:

```
traverse(@"http://en.wikipedia.org/wiki/Russian_language",
         @"http://example.com/education#subTopicOf",
```

```
          vtrav:forward, vtrav:transitive)
```

External links

- Official website [2]
- IBM developerWorks: Thinking XML: Basic XML and RDF techniques for knowledge management, Part 6 [3]
- XML.com: Versa: Path-Based RDF Query Language [4]
- irc://irc.freenode.net/versa ; there exists a Versa "IRC bot" [5], useful for learning, experimentation, and debugging
- RDF query use cases, including query language samples [6]

References

[1] http://en.wikipedia.org/w/index.php?title=Versa_(query_language)&action=edit
[2] http://wiki.xml3k.org/Versa
[3] http://www-128.ibm.com/developerworks/xml/library/x-think10/
[4] http://www.xml.com/lpt/a/2005/07/20/versa.html
[5] http://copia.ogbuji.net/blog/2005-07-18/Emeka
[6] http://rdfstore.sourceforge.net/2002/06/24/rdf-query/

SPARQL

SPARQL

Paradigm(s)	Query language
Appeared in	2008
Developer	W3C
Stable release	1.1 (2013-03-21)
Major implementations	C#, Java, C
Website	[1]

SPARQL (pronounced "sparkle", a recursive acronym for ***S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage*) is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. It was made a standard by the *RDF Data Access Working Group* (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web. On 15 January 2008, SPARQL 1.0 became an official W3C Recommendation, and SPARQL 1.1 in March, 2013.

SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

Implementations for multiple programming languages exist. "SPARQL will make a huge difference" making the web machine-readable according to Sir Tim Berners-Lee in a May 2006 interview.

There exist tools that allow one to connect and semi-automatically construct a SPARQL query for a SPARQL endpoint, for example ViziQuer. In addition, there exist tools that translate SPARQL queries to other query languages, for example to SQL and to XQuery. SPARQL City's SPARQLverse also allows queries directly against non-SPARQL databases such as MongoDB and Cassandra, representing their data as though it is RDF.

Advantages

SPARQL allows users to write queries against data that can loosely be called "key-value" data, more specifically it is data that follows the RDF specification of the W3C. The entire database is thus a set of "subject-predicate-object" triples. This is analogous to some NoSQL database's usage of the term "document-key-value", such as MongoDB.

RDF data can also be considered in SQL relational database terms as a table with three columns - the subject column, the predicate column and the object column. Unlike relational databases, the object column is heterogeneous, the per-cell data type is usually implied (or specified in the ontology) by the predicate value. Alternately, again comparing to SQL relational, all of the triples for a given subject could be represented as a row, with the subject being the primary key and each possible predicate being a column and the object is the value in the cell. However, SPARQL/RDF becomes easier and more powerful for columns that could contain multiple values (like "children"), and where the column itself could be a joinable variable in the query, rather than directly specified.

SPARQL thus provides a full set of analytic query operations such as JOIN, SORT, AGGREGATE for data whose schema is intrinsically part of the data rather than requiring a separate schema definition. Schema information (the ontology) is often provided externally, though, to allow different datasets to be joined in an unambiguous manner. In addition, SPARQL provides specific graph traversal syntax for data that can be thought of as a graph. Some implementations, such as SPARQLverse [2] also allow additional triple attributes such as timestamp and allow additional analytic functionality such as windowed aggregates.

The example below demonstrates a simple query that leverages the ontology definition "foaf", often called the "friend-of-a-friend" ontology.

Specifically, the following query returns names and emails of every person in the dataset:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

This query joins together all of the triples with a matching subject, where the type predicate, "a", is a person (foaf:Person) and the person has one or more names (foaf:name) and mailboxes (foaf:mbox).

The author of this query chose to reference the subject using the variable name "?person" for readable clarity. Since the first element of the triple is always the subject, the author could have just as easily used any variable name, such as "?subj" or "?x". Whatever name is chosen, it must be that same on each line of the query to signify that the query engine is to join triples with the same subject.

The result of the join is a set of rows - ?person, ?name, ?email. This query is returning the the ?name and ?email because ?person is often a complex URI rather than a human-friendly string. Note that in some of the ?people may have multiple mailboxes, so in the returned set, a ?name row may appear multiple times, once for each mailbox.

This query can be distributed to multiple SPARQL endpoints (services that accept SPARQL queries and return results), computed, and results gathered, a procedure known as federated query.

Whether in a federated manner or locally, additional triple definitions in the query could allow joins to different subject types, such as automobiles, to allow simple queries, for example, to return a list of names and emails for people who drive automobiles with a high MPG rating.

Query forms

In the case of queries that read data from the database, the SPARQL language specifies four different query variations for different purposes.

SELECT query

: Used to extract raw values from a SPARQL endpoint, the results are returned in a table format.

CONSTRUCT query

: Used to extract information from the SPARQL endpoint and transform the results into valid RDF.

ASK query

: Used to provide a simple True/False result for a query on a SPARQL endpoint.

DESCRIBE query

: Used to extract an RDF graph from the SPARQL endpoint, the contents of which is left to the endpoint to decide based on what the maintainer deems as useful information.

Each of these query forms takes a WHERE block to restrict the query although in the case of the DESCRIBE query the WHERE is optional.

SPARQL 1.1 specifies a language for updating the database with several new query forms.

Example

Another SPARQL query example that models the question "What are all the country capitals in Africa?":

```
PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
  ?x abc:cityname ?capital ;
     abc:isCapitalOf ?y .
  ?y abc:countryname ?country ;
     abc:isInContinent abc:Africa .
}
```

Variables are indicated by a "?" or "$" prefix. Bindings for ?capital and the ?country will be returned.

The SPARQL query processor will search for sets of triples that match these four triple patterns, binding the variables in the query to the corresponding parts of each triple. Important to note here is the "property orientation" (class matches can be conducted solely through class-attributes or properties - see Duck typing)

To make queries concise, SPARQL allows the definition of prefixes and base URIs in a fashion similar to Turtle. In this query, the prefix "abc" stands for “http://example.com/exampleOntology#”.

Extensions

GeoSPARQL defines filter functions for geographic information system (GIS) queries using well-understood OGC standards (GML, WKT, etc.).

SPARQL implementations

This list shows triplestore, APIs, and other storages that have implemented the SPARQL query language.

- 4store [3]
- Algebraix Data SPARQL Server [4]
- AllegroGraph
- Apache Jena with ARQ
- ARC2 [5]
- BigData [6]
- BrightstarDB [7]
- Corese [8]
- D2R Server [9]
- Dydra [10]
- Hercules [11]
- Intellidimension Semantics Platform 2.0
- KAON2
- Knowledge Explorer
- LUPOSDATE – Open Source (Java) Query-Engine for SPARQL and RIF available at Github[12] and as webdemo[13]
- MarkLogic [14]
- Mulgara [15]
- OntoBroker
- Ontotext OWLIM
- Open Anzo [16]

- OpenLink Virtuoso
- Oracle Oracle DB Enterprise Spatial & Graph [17]
- Oracle NoSQL KV RDF
- Parliament [18]
- Pellet [19]
- Profium Sense
- RAP RDF API for PHP [20]
- RDF-3X
- rdfQuery [21] JavaScript module
- RDF::Query [22] Perl module with complete SPARQL 1.1 implementation.
- Redland / Redstore
- SemWeb.NET [23]
- Sesame 2 [24]
- SPARQL Engine [25]
- SPARQL2XQuery [26]
- SPARQL-RW [27]
- SPARQL City SPARQLverse [2]
- Stardog [28]
- SWObjects
- TopQuadrant's TopBraid Suite [29]
- Twinql [30]
- Ultrawrap [31]
- Web Query
- IBM DB2

References

[1] http://www.w3.org/TR/sparql11-query/
[2] http://sparqlcity.com
[3] http://4store.org/
[4] http://algebraixdata.com/
[5] https://github.com/semsol/arc2/wiki
[6] http://www.systap.com/bigdata.htm
[7] http://www.brightstardb.com/
[8] http://wimmics.inria.fr/corese
[9] http://d2rq.org/
[10] http://dydra.com
[11] http://hercules.arielworks.net/
[12] Github - LuposDate (https://github.com/luposdate/luposdate)
[13] LuposDate Demo-Applet (http://www.ifis.uni-luebeck.de/index.php?id=luposdate-demo)
[14] http://www.marklogic.com
[15] http://www.mulgara.org/
[16] http://www.openanzo.org/
[17] http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/rdfsemantic-graph-1902016.html
[18] http://parliament.semwebcentral.org/
[19] http://clarkparsia.com/pellet
[20] http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/
[21] http://code.google.com/p/rdfquery/
[22] https://metacpan.org/module/RDF::Query
[23] http://razor.occams.info/code/semweb/
[24] http://www.openrdf.org/
[25] http://sparql.sourceforge.net/
[26] http://www.dblab.ntua.gr/~bikakis/SPARQL2XQuery.html

[27] http://www.dblab.ntua.gr/~bikakis/SPARQL-RW.html
[28] http://stardog.com
[29] http://www.topquadrant.com/products/TB_Suite.html
[30] http://www.holygoat.co.uk/projects/twinql/
[31] http://www.capsenta.com/product.html

External links

- W3C SPARQL Working Group (http://www.w3.org/2001/sw/DataAccess/), was RDF Data Access Working Group
- SPARQL 1.1 Recommendation (http://www.w3.org/TR/sparql11-overview/)
- SPARQL 1.0 Query language (http://www.w3.org/TR/rdf-sparql-query/) (legacy)
- SPARQL 1.0 Protocol (http://www.w3.org/TR/rdf-sparql-protocol/) (legacy)
- SPARQL 1.0 Query XML Results Format (http://www.w3.org/TR/2008/REC-rdf-sparql-XMLres-20080115/) (legacy)

RDF query language

An **RDF query language** is a computer language, specifically a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format.

SPARQL is emerging as the de facto RDF query language, and is a W3C recommendation. Released as a Candidate Recommendation in April 2006, it returned to Working Draft status in October 2006, due to open issues. It returned to Candidate Recommendation status in June 2007. On 12 November 2007 the status of SPARQL changed into Proposed Recommendation. On 15 January 2008, SPARQL was standardized.

Other RDF query languages

- DQL, XML-based, queries and results expressed in DAML+OIL
- N3QL, based on Notation 3
- R-DEVICE
- RDFQ, XML-based
- RDQ, SQL-like
- RDQL, SQL-like
- RQL/RVL, SQL-like
- SeRQL, SQL-like, similar to RQL/RVL
- Versa (query language), compact syntax (non−SQL-like), solely implemented in 4Suite (Python)
- XUL has a template [1] element in which to declare rules for matching data in RDF. XUL uses RDF extensively for databinding.
- Adenine (programming language written in RDF).

External links

- RDF Query specification [2]
- RDF query language survey [3]
  - A Comparison of (some) RDF Query Languages [4]
- RDF query use cases, including query language samples [6]
- SparQL [5]

References

[1] http://developer.mozilla.org/en/docs/XUL:Template_Guide:Introduction
[2] http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html
[3] http://www.w3.org/2001/11/13-RDF-Query-Rules/
[4] http://web.archive.org/web/20080702143156/http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/
[5] http://www.w3.org/TR/rdf-sparql-query/

Access query language

Access, the successor to ENGLISH, is an English-like query language used in the Pick operating system.

The original name ENGLISH is something of a misnomer, as PICK's flexible dictionary structure meant that file and attribute names could be given aliases in any natural language. For instance the command SORT could be given the alias TRIEZ, the file CUSTOMER the alias CLIENT, the attribute BALANCE the alias BILAN and the particle BY the alias PAR. This would allow the database to be interrogated using the French-language command string "TRIEZ CLIENT PAR BILAN", resulting in a list of customers by balance.

Nonprocedural language

NPL (for **NonProcedural Language**) was a relational database language developed by T.D. Truitt et al.[1][2] in 1980 for Apple II and MS-DOS. In general, a non-procedural language (also called a declarative language) requires the programmer to specify *what* the program should do, rather than (as with a procedural language) providing the sequential steps indicating *how* the program should perform its task(s).

Notes and references

[1] "An Introduction to Nonprocedural Languages Using NPL", T.D. Truitt et al., McGraw-Hill 1983.

[2] Truitt, T. D. "NPL: the nonprogrammer's data base language" Computer Language 4(06) June 1987 pp97-103

Facebook Query Language

Facebook Query Language

Appeared in	February 2007
Influenced by	SQL
Platform	Facebook Platform
Website	[1]

Facebook Query Language (**FQL**) is a query language that allows querying Facebook user data by using a SQL-style interface, avoiding the need to use the Facebook Platform Graph API. Data returned from an FQL query is in JSON format by default.

History

FQL was first made publicly available in February 2007.

Example

In the following query, four different types of data are retrieved from a single table (status) and for a single user ("me"):

```
SELECT status_id,message,time,source FROM `status` WHERE uid = me()
```

This query can run by querying the Facebook graph endpoint */fql* with the parameters set to *q=[FQL]*

References

[1] http://developers.facebook.com/

Databases Programming

Active database

An **active database** is a database that includes an event-driven architecture (often in the form of ECA rules) which can respond to conditions both inside and outside the database. Possible uses include security monitoring, alerting, statistics gathering and authorization.

Most modern relational databases include active database features in the form of database triggers.

References

Database trigger

A **database trigger** is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database. For example, when a new record (representing a new worker) is added to the employees table, new records should also be created in the tables of the taxes, vacations and salaries.

The need and the usage

Triggers are commonly used to:

- audit changes (e.g. keep a log of the users and roles involved in changes)
- enhance changes (e.g. ensure that every change to a record is time-stamped by the server's clock)
- enforce business rules (e.g. require that every invoice have at least one line item)
- execute business rules (e.g. notify a manager every time an employee's bank account number changes)
- replicate data (e.g. store a record of every change, to be shipped to another database later)
- enhance performance (e.g. update the account balance after every detail transaction, for faster queries)

The examples above are called Data Manipulation Language (DML) triggers because the triggers are defined as part of the Data Manipulation Language and are executed at the time the data is manipulated. Some systems, for example Microsoft SQL Server, also support non-data triggers, which fire in response to Data Definition Language (DDL) events such as creating tables, or runtime events such as logon, commit and rollback. Such DDL triggers can be used for database auditing purposes.

The following are major features of database triggers and their effects:

- triggers do not accept parameters or arguments (but may store affected-data in temporary tables)
- triggers cannot perform commit or rollback operations because they are part of the triggering SQL statement (only through autonomous transactions)

Triggers in DBMS

Below follows a series of descriptions of how some popular DBMS support triggers.

Oracle

In addition to triggers that fire when data is modified, Oracle 9i supports triggers that fire when schema level objects (that is, tables) are modified and when user logon or logoff events occur. These trigger types are referred to as "Schema-level triggers".

Schema-level triggers

- After Creation
- Before Alter
- After Alter
- Before Drop
- After Drop
- Before Logoff
- After Logon

The four main types of triggers are:

1. Row Level Trigger: This gets executed before or after *any column value of a row* changes
2. Column Level Trigger: This gets executed before or after the *specified column* changes
3. For Each Row Type: This trigger gets executed once for each row of the result set caused by insert/update/delete
4. For Each Statement Type: This trigger gets executed only once for the entire result set, but fires each time the statement is executed.

Mutating tables

When a single SQL statement modifies several rows of a table at once, the order of the operations is not well-defined; there is no "order by" clause on "update" statements, for example. Row-level triggers are executed as each row is modified, so the order in which trigger code is run is also not well-defined. Oracle protects the programmer from this uncertainty by preventing row-level triggers from modifying other rows in the same table – this is the "mutating table" in the error message. Side-effects on other tables are allowed, however.

One solution is to have row-level triggers place information into a temporary table indicating what further changes need to be made, and then have a statement-level trigger fire just once, at the end, to perform the requested changes and clean up the temporary table.

Because a foreign key's referential actions are implemented via implied triggers, they are similarly restricted. This may become a problem when defining a self-referential foreign key, or a cyclical set of such constraints, or some other combination of triggers and CASCADE rules, e.g. user deletes a record from table A, CASCADE rule on table A deletes a record from table B, trigger on table B attempts to SELECT from table A, error occurs.

Microsoft SQL Server

Microsoft SQL Server supports triggers either before, after, or instead of an insert, update or delete operation. They can be set on tables and views with the constraint that a view can be referenced only by an INSTEAD OF trigger.

Microsoft SQL Server 2005 introduced support for Data Definition Language (DDL) triggers, which can fire in reaction to a very wide range of events, including:

- Drop table
- Create table
- Alter table
- Login events

A full list [1] is available on MSDN.

Performing conditional actions in triggers (or testing data following modification) is done through accessing the temporary *Inserted* and *Deleted* tables.

PostgreSQL

PostgreSQL introduced support for triggers in 1997. The following functionality in SQL:2003 was previously not implemented in PostgreSQL:

- SQL allows triggers to fire on updates to specific columns; As of version 9.0 of PostgreSQL this feature is also implemented in PostgreSQL.
- The standard allows the execution of a number of SQL statements other than SELECT, INSERT, UPDATE, such as CREATE TABLE as the triggered action. This can be done through creating a stored procedure or function to call CREATE TABLE.[2]

Synopsis:

```
CREATE TRIGGER name { BEFORE | AFTER } { event [ OR ... ] }
    ON TABLE [ FOR [ EACH ] { ROW | STATEMENT } ]
    EXECUTE PROCEDURE funcname ( arguments )
```

Firebird

Firebird supports multiple row-level, BEFORE or AFTER, INSERT, UPDATE, DELETE (or any combination thereof) triggers per table, where they are always "in addition to" the default table changes, and the order of the triggers relative to each other can be specified where it would otherwise be ambiguous (POSITION clause.) Triggers may also exist on views, where they are always "instead of" triggers, replacing the default updatable view logic. (Before version 2.1, triggers on views deemed updatable would run in addition to the default logic.)

Firebird does not raise mutating table exceptions (like Oracle), and triggers will by default both nest and recurse as required (SQL Server allows nesting but not recursion, by default.) Firebird's triggers use NEW and OLD context variables (not Inserted and Deleted tables,) and provide UPDATING, INSERTING, and DELETING flags to indicate the current usage of the trigger.

```
{CREATE | RECREATE | CREATE OR ALTER} TRIGGER name FOR {table name |
view name}
 [ACTIVE | INACTIVE]
 {BEFORE | AFTER}
 {INSERT [OR UPDATE] [OR DELETE] | UPDATE [OR INSERT] [OR DELETE] |
DELETE [OR UPDATE] [OR INSERT] }
 [POSITION n] AS
BEGIN
```

```
 .....
END
```

As of version 2.1, Firebird additionally supports the following database-level triggers:

- CONNECT (exceptions raised here prevent the connection from completing)
- DISCONNECT
- TRANSACTION START
- TRANSACTION COMMIT (exceptions raised here prevent the transaction from committing, or preparing if a two-phase commit is involved)
- TRANSACTION ROLLBACK

Database-level triggers can help enforce multi-table constraints, or emulate materialized views. If an exception is raised in a TRANSACTION COMMIT trigger, the changes made by the trigger so far are rolled back and the client application is notified, but the transaction remains active as if COMMIT had never been requested; the client application can continue to make changes and re-request COMMIT.

Syntax for database triggers:

```
{CREATE | RECREATE | CREATE OR ALTER} TRIGGER name
 [ACTIVE | INACTIVE] ON
 {CONNECT | DISCONNECT | TRANSACTION START | TRANSACTION COMMIT |
TRANSACTION ROLLBACK}
 [POSITION n] AS
BEGIN
 .....
END
```

MySQL

MySQL 5.0.2 introduced support for triggers. MySQL supports these trigger types:

- Insert Trigger
- Update Trigger
- Delete Trigger

Note: MySQL allows only one trigger of each type on each table (i.e. one before insert, one after insert, one before update, one after update, one before delete and one after delete).

Note: MySQL does NOT fire triggers outside of a statement (i.e. API's, foreign key cascades)

The SQL:2003 standard mandates that triggers give programmers access to record variables by means of a syntax such as `REFERENCING NEW AS n`. For example, if a trigger is monitoring for changes to a salary column one could write a trigger like the following:

```
CREATE TRIGGER salary_trigger
    BEFORE UPDATE ON employee_table
    REFERENCING NEW ROW AS n, OLD ROW AS o
    FOR EACH ROW
    IF n.salary <> o.salary THEN

    END IF;
;
```

Sample Mytrigger as follows:

```
-- First of all, drop any other trigger with the same name
DROP TRIGGER IF EXISTS `Mytrigger`;
-- Create New Trigger
DELIMITER $$

CREATE
    /*[DEFINER = { user | CURRENT_USER }]*/
    TRIGGER `DB`.`mytriggers` BEFORE/AFTER INSERT/UPDATE/DELETE
    ON `DB`.`<Table Name>`
    FOR EACH ROW BEGIN

    END$$

DELIMITER ;

-- Example:
DROP TRIGGER IF EXISTS `Mytrigger`;

DELIMITER $$
CREATE TRIGGER `Mytrigger`
AFTER INSERT ON Table_Current
FOR EACH ROW
BEGIN

    UPDATE Table_Record

    SET `Value` = NEW.`Value`
    WHERE `Name` = NEW.`Name`
    AND `Value` < NEW.`Value`;

END $$
DELIMITER;
```

IBM DB2 LUW

IBM DB2 for distributed systems known as DB2 for LUW (LUW means Linux Unix Windows) supports three trigger types: Before trigger, After trigger and Instead of trigger. Both statement level and row level triggers are supported. If there are more triggers for same operation on table then firing order is determined by trigger creation data. Since version 9.7 IBM DB2 supports autonomous transactions [3].

Before trigger is for checking data and deciding if operation should be permitted. If exception is thrown from before trigger then operation is aborted and no data are changed. In DB2 before triggers are read only — you can't modify data in before triggers. After triggers are designed for post processing after requested change was performed. After triggers can write data into tables and unlike someWikipedia:Avoid weasel words other databases you can write into any table including table on which trigger operates. Instead of triggers are for making views writeable.

Triggers are usually programmed in SQL PL language.

SQLite

```
CREATE [TEMP | TEMPORARY] TRIGGER [IF NOT EXISTS] [database_name .]
trigger_name
[BEFORE | AFTER | INSTEAD OF] {DELETE | INSERT | UPDATE [OF column_name
 [, column_name]...]}
ON {table_name | view_name}
   [FOR EACH ROW] [WHEN condition]
BEGIN
   ...
END
```

SQLite only supports row-level triggers, not statement-level triggers.

Updateable views, which are not supported in SQLite, can be emulated with INSTEAD OF triggers.

XML databases

An example of implementation of triggers in non-relational database can be Sedna, that provides support for triggers based on XQuery. Triggers in Sedna were designed to be analogous to SQL:2003 triggers, but natively base on XML query and update languages (XPath, XQuery and XML update language).

A trigger in Sedna is set on any nodes of an XML document stored in database. When these nodes are updated, the trigger automatically executes XQuery queries and updates specified in its body. For example, the following trigger cancels person node deletion if there are any open auctions referenced by this person:

```
CREATE TRIGGER "trigger3"
    BEFORE DELETE
    ON doc("auction")/site//person
    FOR EACH NODE
    DO
    {
       if(exists($WHERE//open_auction/bidder/personref/@person=$OLD/@id))
       then ( )
       else $OLD;
    }
```

References

[1] http://msdn2.microsoft.com/en-us/library/ms189871(SQL.90).aspx
[2] http://www.postgresql.org/docs/9.0/static/sql-createtrigger.html
[3] http://www.ibm.com/developerworks/data/library/techarticle/dm-0907autonomoustransactions/index.html

External links

- Microsoft SQL Server DROP TRIGGER (http://msdn2.microsoft.com/en-us/library/aa258846(SQL.80).aspx)
- MySQL Database triggers (http://dev.mysql.com/doc/refman/5.0/en/triggers.html)
- MySQL DB Create Triggers (http://dev.mysql.com/doc/refman/5.0/en/create-trigger.html)
- DB2 CREATE TRIGGER statement (http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0000931.htm)
- Oracle CREATE TRIGGER (http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_7004.htm#sthref7885)

- PostgreSQL CREATE TRIGGER (http://www.postgresql.org/docs/8.2/static/sql-createtrigger.html)
- Oracle Mutating Table Problems with DELETE CASCADE (http://www.akadia.com/services/ora_mutating_table_problems.html)
- SQLite Query Language: CREATE TRIGGER (http://www.sqlite.org/lang_createtrigger.html)

Stored procedure

A **stored procedure** is a subroutine available to applications that access a relational database system. A stored procedure (sometimes called a **proc**, **sproc**, **StoPro**, **StoredProc**, **sp** or **SP**) is actually stored in the database data dictionary.

Typical use for stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures can consolidate and centralize logic that was originally implemented in applications. Extensive or complex processing that requires execution of several SQL statements is moved into stored procedures, and all applications call the procedures. One can use nested stored procedures by executing one stored procedure from within another.

Stored procedures are similar to user-defined functions (UDFs). The major difference is that UDFs can be used like any other expression within SQL statements, whereas stored procedures must be invoked using the `CALL` statement.[1]

```
CALL procedure(...)
```

or

```
EXECUTE procedure(...)
```

Stored procedures may return result sets, *i.e.* the results of a `SELECT` statement. Such result sets can be processed using cursors, by other stored procedures, by associating a result set locator, or by applications. Stored procedures may also contain declared variables for processing data and cursors that allow it to loop through multiple rows in a table. Stored procedure flow control statements typically include `IF`, `WHILE`, `LOOP`, `REPEAT`, and `CASE` statements, and more. Stored procedures can receive variables, return results or modify variables and return them, depending on how and where the variable is declared.

Implementation

The exact and correct implementation of stored procedures varies from one database system to another. Most major database vendors support them in some form. Depending on the database system, stored procedures can be implemented in a variety of programming languages, for example SQL, Java, C, or C++. Stored procedures written in non-SQL programming languages may or may not execute SQL statements themselves.

The increasing adoption of stored procedures led to the introduction of procedural elements to the SQL language in the SQL:1999 and SQL:2003 standards in the part SQL/PSM. That made SQL an imperative programming language. Most database systems offer proprietary and vendor-specific extensions, exceeding SQL/PSM. A standard specification for Java stored procedures exists as well as SQL/JRT.

Database system	Implementation language
CUBRID	Java
DB2	SQL PL (close to the SQL/PSM standard) or Java
Firebird	PSQL (Fyracle also supports portions of Oracle's PL/SQL)
Informix	SPL or Java
Microsoft SQL Server	Transact-SQL and various .NET Framework languages
MySQL	own stored procedures, closely adhering to SQL/PSM standard.
Oracle	PL/SQL or Java
PostgreSQL	PL/pgSQL, can also use own function languages such as pl/perl or pl/php
Sybase ASE	Transact-SQL

Comparison with dynamic SQL

Overhead

Because stored procedure statements are stored directly in the database, they *may* remove all or part of the compilation overhead that is typically required in situations where software applications send inline (dynamic) SQL queries to a database. (However, most database systems implement "statement caches" and other mechanisms to avoid repetitive compilation of dynamic SQL statements.) In addition, while they avoid some overhead, pre-compiled SQL statements add to the complexity of creating an optimal execution plan because not all arguments of the SQL statement are supplied at compile time. Depending on the specific database implementation and configuration, mixed performance results will be seen from stored procedures versus generic queries or user defined functions.

Avoidance of network traffic

A major advantage with stored procedures is that they can run directly within the database engine. In a production system, this typically means that the procedures run entirely on a specialized database server, which has direct access to the data being accessed. The benefit here is that network communication costs can be avoided completely. This becomes particularly important for complex series of SQL statements.

Encapsulation of business logic

Stored procedures allow programmers to embed business logic as an API in the database, which can simplify data management and reduce the need to encode the logic elsewhere in client programs. This can result in a lesser likelihood of data corruption by faulty client programs. The database system can ensure data integrity and consistency with the help of stored procedures.

Delegation of access-rights

In many systems, stored procedures can be granted access rights to the database that users who execute those procedures do not directly have.

Some protection from SQL injection attacks

Stored procedures can be used to protect against injection attacks. Stored procedure parameters will be treated as data even if an attacker inserts SQL commands. Also, some DBMSs will check the parameter's type. A stored procedure that in turn generates dynamic SQL using the input is however still vulnerable to SQL injections unless proper precautions are taken.

Other uses

In some systems, stored procedures can be used to control transaction management; in others, stored procedures run inside a transaction such that transactions are effectively transparent to them. Stored procedures can also be invoked from a database trigger or a condition handler. For example, a stored procedure may be triggered by an insert on a specific table, or update of a specific field in a table, and the code inside the stored procedure would be executed. Writing stored procedures as condition handlers also allows database administrators to track errors in the system with greater detail by using stored procedures to catch the errors and record some audit information in the database or an external resource like a file.

Comparison with functions

- A function is a subprogram written to perform certain computations
- A scalar function returns only a single value (or NULL), whereas a table function returns a (relational) table comprising zero or more rows, each row with one or more columns.
- Functions must return a value (using the `RETURN` keyword), but for stored procedures this is not compulsory.
- Stored procedures can use `RETURN` keyword but without any value being passed.
- Functions could be used in `SELECT` statements, provided they don't do any data manipulation. However, procedures cannot be included in `SELECT` statements.
- A stored procedure can return multiple values using the `OUT` parameter or return no value at all.
- A stored procedure saves the query compilation time.

Comparison with prepared statements

Prepared statements take an ordinary statement or query and parameterize it so that different literal values can be used at a later time. Like stored procedures, they are stored on the server for efficiency and provide some protection from SQL injection attacks. Although simpler and more declarative, prepared statements are not ordinarily written to use procedural logic and cannot operate on variables. Because of their simple interface and client-side implementations, prepared statements are more widely reusable between DBMSs.

Disadvantages

- Stored procedure languages are quite often **vendor-specific**. Switching to another vendor's database most likely requires rewriting any existing stored procedures.
- Stored procedure languages from different vendors have different levels of sophistication.
  - For example, Oracle's PL/SQL has more language features and built-in features (via packages such as DBMS_ and UTL_ and others) than Microsoft's T-SQL.[*citation needed*]
- Tool support for writing and debugging stored procedures is often not as good as for other programming languages, but this differs between vendors and languages.
  - For example, both PL/SQL and T-SQL have dedicated IDEs and debuggers. PL/PgSQL can be debugged from various IDEs.

References

[1] Call Procedure (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/db2/rbafzmstcallstmt.htm)

External links

- Stored Procedures in MySQL FAQ (http://dev.mysql.com/doc/refman/5.7/en/faqs-stored-procs.html)
- An overview of PostgreSQL Procedural Language support (http://www.postgresql.org/docs/current/interactive/xplang.html)
- Using a stored procedure in Sybase ASE (http://www.petersap.nl/SybaseWiki/index.php/Stored_procedure)
- PL/SQL Procedures (http://infolab.stanford.edu/~ullman/fcdb/oracle/or-plsql.html#procedures)
- Oracle Database PL/SQL Language Reference (http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28370/toc.htm)

PL/SQL

PL/SQL (**Procedural Language/Structured Query Language**) is Oracle Corporation's procedural language extension for SQL and the Oracle relational database. PL/SQL is available in Oracle Database (since version 7), TimesTen in-memory database (since version 11.2.1), and IBM DB2 (since version 9.7). Oracle Corporation usually extends PL/SQL functionality with each successive release of the Oracle Database.

PL/SQL includes procedural language elements such as conditions and loops. You can declare constants and variables, procedures and functions, types and variables of those types, and triggers. You can handle exceptions (runtime errors). Arrays are supported involving the use of PL/SQL collections. Implementations from version 8 of Oracle Database onwards have included features associated with object-orientation. You can create PL/SQL units—procedures, functions, packages, types, and triggers —that are stored in the database for reuse by applications that use any of the Oracle Database programmatic interfaces.

Similar languages

PL/SQL functions analogously to the embedded procedural languages associated with other relational databases. Sybase ASE and Microsoft SQL Server have Transact-SQL, PostgreSQL has PL/pgSQL (which tries to emulate PL/SQL to an extent), and IBM DB2 includes SQL Procedural Language, which conforms to the ISO SQL's SQL/PSM standard.

The designers of PL/SQL modelled its syntax on that of Ada. Both Ada and PL/SQL have Pascal as a common ancestor, and so PL/SQL also resembles Pascal in numerous aspects. The structure of a PL/SQL package closely resembles the basic Pascal program structure or a Borland Delphi unit. Programmers can define global data-types, constants and static variables, public and private, in a PL/SQL package.

PL/SQL also allows for the definition of classes and instantiating these as objects in PL/SQL code. This resembles usages in object-oriented programming languages like Object Pascal, C++ and Java. PL/SQL refers to a class as an "Abstract Data Type" (ADT) or "User Defined Type" (UDT), and defines it as an Oracle SQL data-type as opposed to a PL/SQL user-defined type, allowing its use in both the Oracle SQL Engine and the Oracle PL/SQL engine. The constructor and methods of an Abstract Data Type are written in PL/SQL. The resulting Abstract Data Type can operate as an object class in PL/SQL. Such objects can also persist as column values in Oracle database tables.

PL/SQL is fundamentally distinct from Transact-SQL, despite superficial similarities. Porting code from one to the other usually involves non-trivial work, not only due to the differences in the feature sets of the two languages, but also due to the very significant differences in the way Oracle and SQL Server deal with concurrency and locking.

The Fyracle project aims to enable the execution of PL/SQL code in the open-source Firebird database.

The StepSqlite product is a PL/SQL compiler for the popular small database SQLite.

PL/SQL Program Unit

A PL/SQL program unit is one of the following: PL/SQL Anonymous Block, Procedure, Function, Package Specification, Package Body, Trigger, Type Specification, Type Body, Library. Program units are the PL/SQL source code that is compiled, developed and ultimately executed on the database.

PL/SQL Anonymous Block

The basic unit of a PL/SQL source program is the block, which groups related declarations and statements. A PL/SQL block is defined by the keywords DECLARE, BEGIN, EXCEPTION, and END. These keywords divide the block into a declarative part, an executable part, and an exception-handling part. The declaration section is optional and may be used to define and initialize constants and variables. If a variable is not initialized then it defaults to NULL value. The optional exception-handling part is used to handle run time errors. Only the executable part is required. A block can have a label.

For example:

```
<<label>>   -- this is optional
declare
-- this section is optional
  number1 number(2);
  number2 number1%type    := 17;             -- value default
  text1   varchar4(12) := 'Hello world';
  text2   date         := SYSDATE;        -- current date and time
begin
-- this section is mandatory, must contain at least one executable statement
  SELECT street_number
    INTO number1
    FROM address
   WHERE name = 'INU';
exception
-- this section is optional
   WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE('Error Code is ' || to_char(sqlcode )  );
     DBMS_OUTPUT.PUT_LINE('Error Message is ' || sqlerrm   );
end;
```

The symbol := functions as an assignment operator to store a value in a variable.

Blocks can be nested: Because a block is an executable statement, it can appear in another block wherever an executable statement is allowed. You can submit a block to an interactive tool (such as SQL*Plus) or embed it in an Oracle Precompiler or OCI program. The interactive tool or program runs the block one time. The block is not stored in the database, and for that reason, it is called an anonymous block (even if it has a label).

Function

The purpose of a PL/SQL function is generally to compute and return a single value. This returned value may be a single scalar value (such as a number, date or character string) or a single collection (such as a nested table or varray). User-defined functions supplement the built-in functions provided by Oracle Corporation.

The PL/SQL function has the form:

```
CREATE OR REPLACE FUNCTION <function_name> [(input/output variable declarations)] RETURN return_type
[AUTHID <CURRENT_USER | DEFINER>] <IS|AS>   -- heading part
amount number;-- declaration block]
BEGIN  -- exectutable part
      <PL/SQL block with return statement>
        RETURN <return_value>;
[Exception
      none]
        RETURN <return_value>;
END;
```

A pipelined table functions return collections and take the form:

```
CREATE OR REPLACE FUNCTION <function_name> [(input/output variable declarations)] RETURN return_type
[AUTHID <CURRENT_USER | DEFINER>] [<AGGREGATE | PIPELINED>] <IS|USING>
      [declaration block]
BEGIN
      <PL/SQL block with return statement>
        PIPE ROW <return type>;
        RETURN;
[Exception
      exception block]
        PIPE ROW <return type>;
        RETURN;
END;
```

A function should only use the default IN type of parameter. The only out value from the function should be the value it returns.

Procedure

Procedures are similar to Functions, in that they are named program units that can be invoked repeatedly. The primary difference is that **functions can be used in a SQL statement whereas procedures cannot**. Another difference is that the procedure can return multiple values whereas a function should only return a single value.

The procedure begins with a mandatory heading part to hold the procedure name and optionally the procedure parameter list. Next are the declarative, executable and exception-handling parts, as in the PL/SQL Anonymous Block. Here is an example of a simple procedure.

```
CREATE PROCEDURE create_email_address ( -- Procedure heading part begins
name1 VARCHAR2,
name2 VARCHAR2,
company VARCHAR2
) -- Procedure heading part ends
AS
```

```
-- Declarative part begins (optional)
error_message VARCHAR2(30) := 'Email address is too long.';
BEGIN -- Executable part begins (mandatory)
email := name1 || '.' || name2 || '@' || company;
EXCEPTION -- Exception-handling part begins (optional)
WHEN VALUE_ERROR THEN
DBMS_OUTPUT.PUT_LINE(error_message);
END create_email_address;
```

The example above shows a Standalone Procedure - this type of procedure is created and stored in a database schema using the CREATE PROCEDURE statement. A procedure may also be created in a PL/SQL package - this is called a Package Procedure. A procedure created in a PL/SQL anonymous block is called a Nested Procedure. The Standalone or Package procedures are stored in the database and so are referred to as Stored Procedures.

There are three types of parameter: IN, OUT and IN OUT.

1. An IN parameter is used as input only. An IN parameter is passed by reference though it can be changed by the inactive program.
2. An OUT parameter is initially NULL. The program assigns the parameter a value and that value is returned to the calling program.
3. An IN OUT parameter may or may not have an initial value. That initial value may or may not be modified by the called program. Any changes made to the parameter are returned to the calling program by default by copying but - with the NOCOPY hint - may be passed by reference.

Package

Packages are groups of conceptually linked functions, procedures, variables, PL/SQL table and record TYPE statements, constants, cursors etc. The use of packages promotes re-use of code. Packages are composed of the package specification and an optional package body. The specification is the interface to the application; it declares the types, variables, constants, exceptions, cursors, and subprograms available. The body fully defines cursors and subprograms, and so implements the spec. Two advantages of packages include:

1. Modular approach, encapsulation/hiding of business logic, security, performance improvement, re-usability. They support Object-oriented programming features like function overloading and encapsulation.
2. Using package variables one can declare session level (scoped) variables, since variables declared in the package specification have a session scope.

Trigger

A trigger is like a stored procedure that Oracle Database invokes automatically whenever a specified event occurs. It is a named PL/SQL unit that is stored in the database and can be invoked repeatedly. Unlike a stored procedure, you can enable and disable a trigger, but you cannot explicitly invoke it. While a trigger is enabled, the database automatically invokes it—that is, the trigger fires—whenever its triggering event occurs. While a trigger is disabled, it does not fire.

You create a trigger with the CREATE TRIGGER statement. You specify the triggering event in terms of triggering statements and the item on which they act. The trigger is said to be created on or defined on the item, which is either a table, a view, a schema, or the database. You also specify the timing point, which determines whether the trigger fires before or after the triggering statement runs and whether it fires for each row that the triggering statement affects.

If the trigger is created on a table or view, then the triggering event is composed of DML statements, and the trigger is called a DML trigger. If the trigger is created on a schema or the database, then the triggering event is composed

of either DDL or database operation statements, and the trigger is called a system trigger.

An INSTEAD OF trigger is either: A DML trigger created on a view or a system trigger defined on a CREATE statement. The database fires the INSTEAD OF trigger instead of running the triggering statement.

Benefits of Triggers

Triggers can be written for the following purposes:

Generating some derived column values automatically

Enforcing referential integrity

Event logging and storing information on table access

Auditing

Synchronous replication of tables

Imposing security authorizations

Preventing invalid transactions

Data Types

The major datatypes in PL/SQL include NUMBER, INTEGER, CHAR, VARCHAR2, DATE and TIMESTAMP.

Numeric variables

```
variable_name number(P[,S]) := 0;
```

To define a numeric variable, the programmer appends the variable type **NUMBER** to the name definition. To specify the (optional) precision (P) and the (optional) scale (S), one can further append these in round brackets, separated by a comma. ("Precision" in this context refers to the number of digits which the variable can hold, "scale" refers to the number of digits which can follow the decimal point.)

A selection of other datatypes for numeric variables would include: binary_float, binary_double, dec, decimal, double precision, float, integer, int, numeric, real, smallint, binary_integer.

Character variables

```
variable_name varchar2(10) := 'Text';
```

To define a character variable, the programmer normally appends the variable type VARCHAR2 to the name definition. There follows in brackets the maximum number of characters which the variable can store.

Other datatypes for character variables include: varchar, char, long, raw, long raw, nchar, nchar2, clob, blob, bfile

Date variables

```
variable_name date := to_date('01-01-2005 14:20:23','DD-MM-YYYY hh24:mi:ss');
```

Date variables can contain date and time. The time may be left out, but there is no way to define a variable that only contains the time. There is no DATETIME type. And there is no TIME type. But there is a TIMESTAMP type that can contain fine grained timestamp up to millisecond or nanosecond. Oracle Datatypes [1]

The `TO_DATE` function can be used to convert strings to date values. The function converts the first quoted string into a date, using as a definition the second quoted string, for example:

```
 to_date('31-12-2004','dd-mm-yyyy')
```

or

```
 to_date ('31-Dec-2004','dd-mon-yyyy', 'NLS_DATE_LANGUAGE = American')
```

To convert the dates to strings one uses the function `TO_CHAR (date_string, format_string).`

PL/SQL also supports the use of ANSI date and interval literals. The following clause gives an 18-month range:

```
WHERE dateField BETWEEN DATE '2004-12-30' - INTERVAL '1-6' YEAR TO MONTH
    AND DATE '2004-12-30'
```

Exceptions

Exceptions, errors which arise during the execution of the code, have one of two types:

User-defined exceptions are always raised explicitly by the programmers, using the `RAISE` or `RAISE_APPLICATION_ERROR` commands, in any situation where they have determined that it is impossible for normal execution to continue. The `RAISE` command has the syntax:

```
RAISE <exception name>;
```

Oracle Corporation has predefined several exceptions like `NO_DATA_FOUND`, `TOO_MANY_ROWS`, *etc.* Each exception has an SQL Error Number and SQL Error Message associated with it. Programmers can access these by using the `SQLCODE` and `SQLERRM` functions.

Datatypes for specific columns

Variable_name Table_name.Column_name%type;

This syntax defines a variable of the type of the referenced column on the referenced tables.

Programmers specify user-defined datatypes with the syntax:

```
type data_type is record (field_1 type_1 :=xyz, field_2 type_2 :=xyz, ..., field_n type_n :=xyz);
```

For example:

```
declare
    type t_address is  record (
        name address.name%type,
        street address.street%type,
        street_number address.street_number%type,
        postcode address.postcode%type);
    v_address t_address;
begin
    select name, street, street_number, postcode into v_address from address where rownum = 1;
end;
```

This sample program defines its own datatype, called *t_address*, which contains the fields *name, street, street_number* and *postcode*.

So according to the example, we are able to copy the data from the database to the fields in the program.

Using this datatype the programmer has defined a variable called *v_address* and loaded it with data from the ADDRESS table.

Programmers can address individual attributes in such a structure by means of the dot-notation, thus: *"v_address.street := 'High Street';"*

Conditional statements

The following code segment shows the IF-THEN-ELSIF construct. The ELSIF and ELSE parts are optional so it is possible to create simpler IF-THEN or, IF-THEN-ELSE constructs.

```
IF x = 1 THEN
   sequence_of_statements_1;
ELSIF x = 2 THEN
   sequence_of_statements_2;
ELSIF x = 3 THEN
   sequence_of_statements_3;
ELSIF x = 4 THEN
   sequence_of_statements_4;
ELSIF x = 5 THEN
   sequence_of_statements_5;
ELSE
   sequence_of_statements_N;
END IF;
```

The CASE statement simplifies some large IF-THEN-ELSE structures.

```
CASE
   WHEN x = 1 THEN sequence_of_statements_1;
   WHEN x = 2 THEN sequence_of_statements_2;
   WHEN x = 3 THEN sequence_of_statements_3;
   WHEN x = 4 THEN sequence_of_statements_4;
   WHEN x = 5 THEN sequence_of_statements_5;
   ELSE sequence_of_statements_N;
END CASE;
```

CASE statement can be used with predefined selector:

```
CASE x
   WHEN 1 THEN sequence_of_statements_1;
   WHEN 2 THEN sequence_of_statements_2;
   WHEN 3 THEN sequence_of_statements_3;
   WHEN 4 THEN sequence_of_statements_4;
   WHEN 5 THEN sequence_of_statements_5;
   ELSE sequence_of_statements_N;
END CASE;
```

Array handling

PL/SQL refers to arrays as "collections". The language offers three types of collections:

1. Associative arrays (Index-by tables)
2. Nested tables
3. Varrays (variable-size arrays)

Programmers must specify an upper limit for varrays, but need not for index-by tables or for nested tables. The language includes several collection methods used to manipulate collection elements: for example FIRST, LAST, NEXT, PRIOR, EXTEND, TRIM, DELETE, etc. Index-by tables can be used to simulate associative arrays, as in this example of a memo function for Ackermann's function in PL/SQL.

Associative arrays (Index-by tables)

With index-by tables, the array can be indexed by numbers or strings. It parallels a Java *map*, which comprises key-value pairs. There is only one dimension and it is unbounded.

Nested tables

With nested tables the programmer needs to understand what is nested. Here, a new type is created that may be composed of a number of components. That type can then be used to make a column in a table, and nested within that column will be those components.

Varrays (variable-size arrays)

With Varrays you need to understand that the word "variable" in the phrase "variable-size arrays" doesn't apply to the size of the array in the way you might think that it would. The size the array is declared with is in fact fixed. The number of elements in the array is variable up to the declared size. Arguably then, variable-sized arrays aren't that variable in size.

Looping

As a procedural language by definition, PL/SQL provides several iteration constructs, including basic LOOP statements, WHILE loops, FOR loops, and Cursor FOR loops.

LOOP statements

Syntax [2]

```
<<parent_loop>>
LOOP
      statements

      <<child_loop>>
      loop
            statements
            exit parent_loop when <condition>; -- Terminates both loops
            exit when <condition>; -- Returns control to parent_loop
      end loop child_loop;
        if <condition> then
           continue; -- continue to next iteration
        end if;
```

```
      exit when <condition>;
END LOOP parent_loop;
```

Loops can be terminated by using the `EXIT` keyword, or by raising an exception.

FOR loops

```
 declare
     var number;
 begin
     /*N.B. for loop variables in pl/sql are new declarations, with scope only inside the loop */
for var in 0 .. 10 loop
          dbms_output.put_line(var);
     end loop;

     if (var is null) then
          dbms_output.put_line('var is null');
     else
          dbms_output.put_line('var is not null');
     end if;
 end;
```

This is the output
Output:

```
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 var is null
```

Cursor FOR loops

```
FOR RecordIndex IN (SELECT person_code FROM people_table)
LOOP
  DBMS_OUTPUT.PUT_LINE(RecordIndex.person_code);
END LOOP;
```

Cursor-for loops automatically open a cursor, read in their data and close the cursor again.

As an alternative, the PL/SQL programmer can pre-define the cursor's SELECT-statement in advance in order (for example) to allow re-use or to make the code more understandable (especially useful in the case of long or complex queries).

```
DECLARE
  CURSOR cursor_person IS
    SELECT person_code FROM people_table;
BEGIN
  FOR RecordIndex IN cursor_person
  LOOP
    DBMS_OUTPUT.PUT_LINE(recordIndex.person_code);
  END LOOP;
END;
```

The concept of the person_code within the FOR-loop gets expressed with dot-notation (".")::

```
RecordIndex.person_code
```

Dynamic SQL

While programmers can readily embed Data Manipulation Language (DML) statements directly into their PL/SQL code using straightforward SQL statements, Data Definition Language (DDL) requires more complex "Dynamic SQL" statements to be written in the PL/SQL code. However, DML statements underpin the majority of PL/SQL code in typical software applications.

In the case of PL/SQL dynamic SQL, early versions of the Oracle Database required the use of a complicated Oracle `DBMS_SQL` package library. More recent versions have however introduced a simpler "Native Dynamic SQL", along with an associated `EXECUTE IMMEDIATE` syntax.

References

[1] http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/sql_elements001.htm
[2] http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261/controlstructures.htm#sthref921

Further reading

- Feuerstein, Steven; with Bill Pribyl (2005). *Oracle PL/SQL Programming* (4th ed.). O'Reilly & Associates. ISBN 0-596-00977-1.
- Naudé, Frank (June 9, 2005). "Oracle PL/SQL FAQ rev 2.08" (http://www.orafaq.com/faqplsql.htm).

External links

- Oracle FAQ: PL/SQL (http://www.orafaq.com/wiki/PL/SQL_FAQ)
- Oracle Technology Center (http://www.oracle.com/technology/tech/pl_sql/index.html)
- Oracle Tahiti Search Engine (http://tahiti.oracle.com)
- Morgan's Library (http://www.morganslibrary.org/library.html)
- ORACLE-BASE: PL/SQL Articles (http://www.oracle-base.com/articles/plsql/ArticlesPLSQL.php)

SQL PL

SQL PL stands for Structured Query Language Procedural Language and was developed by IBM as a set of commands that extend the use of SQL in the IBM DB2 (DB2 UDB Version 7) database system.[1] It provides procedural programmability in addition to the querying commands of SQL. It is a subset of the SQL Persistent Stored Modules (SQL/PSM) language standard.

As of DB2 version 9, SQL PL stored procedures can run natively inside the DB2 process (inside the DBM1 address space, more precisely) instead of being fenced in an external process. In DB2 version 9.7 IBM also added a PL/SQL front-end to this infrastructure (called "SQL Unified Runtime Engine"), meaning that procedural SQL using either the ISO standard or Oracle's syntax compile to bytecode running on the same engine in DB2.

References

[1] IBM Info Center (http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.apdv.sql.doc/doc/c0011916.htm)

External links

- SQL PL Guide for developing Stored Procedures in DB2 (http://www.sqlpl-guide.com)

SQL programming tool

In the field of software, **SQL programming tools** provide platforms for database administrators (DBAs) and application developers to perform daily tasks efficiently and accurately.

Database administrators and application developers often face constantly changing environments which they rarely completely control. Many changes result from new development projects or from modifications to existing code, which, when deployed to production, do not always produce the expected result.

For organizations to better manage development projects and the teams that develop code, suppliers of SQL programming tools normally provide more than facility to the database administrator or application developer to aid in database management and in quality code-deployment practices.

Features

SQL programming tools may include the following features:

SQL editing

SQL editors allow users to edit and execute SQL statements. They may support the following features:

- cut, copy, paste, undo, redo, find (and replace), bookmarks
- block indent, print, save file, uppercase/lowercase
- keyword highlighting
- auto-completion
- access to frequently used files
- output of query result
- editing query-results
- committing and rolling-back transactions
- inside cut paper

Object browsing

Tools may display information about database objects relevant to developers or to database administrators. Users may:

- view object descriptions
- view object definitions (DDL)
- create database objects
- enable and disable triggers and constraints
- recompile valid or invalid objects
- query or edit tables and views

Some tools also provide features to display dependencies among objects, and allow users to expand these dependent objects recursively (for example: packages may reference views, views generally reference tables, super/subtypes, and so on).

Session browsing

Database administrators and application developers can use session browsing tools to view the current activities of each user in the database. They can check the resource-usage of individual users, statistics information, locked objects and the current running SQL of each individual session.

User-security management

DBAs can create, edit, delete, disable or enable user-accounts in the database using security-management tools. DBAs can also assign roles, system privileges, object privileges, and storage-quotas to users.

Debugging

Some tools offer features for the debugging of stored procedures: Step In, Step Over, Step Out, Run Until Exception, Breakpoints, View & Set Variables, View Call Stack, and so on. Users can debug any program-unit without making any modification to it, including triggers and object types.

Performance monitoring

Monitoring tools may show the database resources — usage summary, service time summary, recent activities, top sessions, session history or top SQL — in easy-to-read graphs. Database administrators can easily monitor the health of various components in the monitoring instance. Application developers may also make use of such tools to diagnose and correct application-performance problems as well as improve SQL server performance.

User-defined function

A **User-Defined Function (UDF)** is a function provided by the user of a program or environment, in a context where the usual assumption is that functions are built into the program or environment.

BASIC language

In some old implementations of the BASIC programming language, user-defined functions are defined using the "DEF FN" syntax. More modern dialects of BASIC are influenced by the structured programming paradigm, where most or all of the code is written as user-defined functions or procedures, and the concept becomes practically redundant.

Databases

In SQL databases, a user-defined function provides a mechanism for extending the functionality of the database server by adding a function that can be evaluated in SQL statements. The SQL standard distinguishes between scalar and table functions. A scalar function returns only a single value (or NULL), whereas a table function returns a (relational) table comprising zero or more rows, each row with one or more columns.

User-defined functions in SQL are declared using the `CREATE FUNCTION` statement. For example, a function that converts Celsius to Fahrenheit might be declared like this:

```
CREATE FUNCTION dbo.CtoF(Celsius FLOAT)
  RETURNS FLOAT
  RETURN (Celsius * 1.8) + 32
```

Once created, a user-defined function may be used in expressions in SQL statements. For example, it can be invoked where most other intrinsic functions are allowed. This also includes SELECT statements, where the function can be used against data stored in tables in the database. Conceptually, the function is evaluated once per row in such usage. For example, assume a table named ELEMENTS, with a row for each known chemical element. The table has a column named BoilingPoint for the boiling point of that element, in Celsius. The query

```
SELECT Name, CtoF(BoilingPoint)
  FROM Elements
```

would retrieve the name and the boiling point from each row. It invokes the CtoF user-defined function as declared above in order to convert the value in the column to a value in Fahrenheit.

Each user-defined function carries certain properties or characteristics. The SQL standard defines the following properties:

- Language - defines the programming language in which the user-defined function is implemented; examples are SQL, C, or Java.
- Parameter style - defines the conventions that are used to pass the function parameters and results between the implementation of the function and the database system (only applicable if language is not SQL).
- Specific name - a name for the function that is unique within the database. Note that the function name does not have to be unique, considering overloaded functions. Some SQL implementations require that function names are unique within a database, and overloaded functions are not allowed.
- Determinism - specifies whether the function is deterministic or not. The determinism characteristic has an influence on the query optimizer when compiling a SQL statement.
- SQL-data access - tells the database management system whether the function contains no SQL statements (NO SQL), contains SQL statements but does not access any tables or views (CONTAINS SQL), reads data from

tables or views (READS SQL DATA), or actually modifies data in the database (MODIFIES SQL DATA).

User-defined functions should not be confused with stored procedures. Stored procedures allow the user to group a set of SQL commands. A procedure can accept parameters and execute its SQL statements depending on those parameters. A procedure is not an expression and, thus, cannot be used like user-defined functions.

Some database management systems allow the creation of user defined functions in languages other than SQL. Microsoft SQL Server, for example, allows the user to use .NET languages including C# for this purpose. DB2 and Oracle support user-defined functions written in C or Java programming languages.

SQL Server 2000

There are three types of UDF in Microsoft SQL Server 2000:

- Scalar functions.
- Inline table-valued functions.
- Multistatementtable-valued functions.

Scalar functions return a single data value (not a table) with RETURNS clause. Scalar functions can use all scalar data types, with exception of timestamp and user-defined data types. Inline table-valued functions return the result set of a single SELECT statement. Multistatement table-valued functions return a table, which was built with many TRANSACT-SQL statements.

User-defined functions can be invoked from a query like built-in functions such as OBJECT_ID, LEN, DATEDIFF, or can be executed through an EXECUTE statement like stored procedures.

Performance Notes: 1. On Microsoft SQL Server 2000 a table-valued function which "wraps" a View may be much faster than the View itself. The following MyFunction is an example of a "function-wrapper" which runs faster than the underlying view MyView:

```
CREATE FUNCTION MyFunction()
    RETURNS @Tbl TABLE
    (
        StudentID              VARCHAR(255),
        SAS_StudentInstancesID INT,
        Label                  VARCHAR(255),
        Value                  MONEY,
        CMN_PersonsID          INT
    )
AS
BEGIN
    INSERT @Tbl
    (
        StudentID,
        SAS_StudentInstancesID,
        Label,
        Value,
        CMN_PersonsID
    )
    SELECT
        StudentID,
        SAS_StudentInstancesID,
        Label,
```

```
        Value,
        CMN_PersonsID
    FROM MyView -- where MyView selects (with joins) the same columns from large table(s)

    RETURN
END
```

2. On Microsoft SQL Server 2005 the result of the same code execution is the opposite: view is executed faster than the "function-wrapper".

User-defined functions are subroutines made of one or more Transact-SQL statements that can be used to encapsulate code for reuse. It takes zero or more arguments and evaluates a return value. Has both control-flow and DML statements in its body similar to stored procedures. Does not allow changes to any Global Session State, like modifications to database or external resource, such as a file or network. Does not support output parameter. DEFAULT keyword must be specified to pass the default value of parameter. Errors in UDF cause UDF to abort which, in turn, aborts the statement that invoked the UDF.

```
CREATE FUNCTION CubicVolume
-- Input dimensions in centimeters
(
    @CubeLength decimal(4,1),
    @CubeWidth  decimal(4,1),
    @CubeHeight decimal(4,1)
)
    RETURNS decimal(12,3)
AS
BEGIN
  RETURN(@CubeLength * @CubeWidth * @CubeHeight)
END
```

Data type supported in Microsoft SQL Server 2000 Like a temporary table used to store results Mostly used to define temporary variable of type (table) and the return value of a UDF The scope is limited to function, stored procedure, or batch in which it is defined Assignment operation is not allowed between (Table) variables May be used in SELECT, INSERT, UPDATE, and DELETE CREATE FUNCTION to create UDF ALTER FUNCTION to change the characteristics of UDF DROP FUNCTION to remove UDF

External links

- Microsoft SQL Server reference for CREATE FUNCTION [1]
- MySQL manual section on UDFs [2]
- DB2 CREATE FUNCTION statement [3]

References

[1] http://msdn2.microsoft.com/en-us/library/aa258261(SQL.80).aspx
[2] http://dev.mysql.com/doc/refman/5.5/en/adding-functions.html
[3] http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/index.jsp?topic=%2Fcom.ibm.db2.luw.sql.ref.doc%2Fdoc%2Fr0000917.html

Cursor (databases)

In computer science, a database **cursor** is a control structure that enables traversal over the records in a database. Cursors facilitate subsequent processing in conjunction with the traversal, such as retrieval, addition and removal of database records. The database cursor characteristic of traversal makes cursors akin to the programming language concept of iterator.

Cursors are used by database programmers to process individual rows returned by database system queries. Cursors enable manipulation of whole result sets at once. In this scenario, a cursor enables the rows in a result set to be processed sequentially.

In SQL procedures, a cursor makes it possible to define a result set (a set of data rows) and perform complex logic on a row by row basis. By using the same mechanics, a SQL procedure can also define a result set and return it directly to the caller of the SQL procedure or to a client application.

A cursor can be viewed as a pointer to one row in a set of rows. The cursor can only reference one row at a time, but can move to other rows of the result set as needed.

Usage

To use cursors in SQL procedures, you need to do the following:

1. Declare a cursor that defines a result set.
2. Open the cursor to establish the result set.
3. Fetch the data into local variables as needed from the cursor, one row at a time.
4. Close the cursor when done.

To work with cursors you must use the following SQL statements

This section introduces the ways the SQL:2003 standard defines how to use cursors in applications in embedded SQL. Not all application bindings for relational database systems adhere to that standard, and some (such as CLI or JDBC) use a different interface.

A programmer makes a cursor known to the DBMS by using a `DECLARE ... CURSOR` statement and assigning the cursor a (compulsory) name:

```
DECLARE cursor_name CURSOR FOR SELECT ... FROM ...
```

Before code can access the data, it must open the cursor with the `OPEN` statement. Directly following a successful opening, the cursor is positioned *before* the first row in the result set.

```
OPEN cursor_name
```

Programs position cursors on a specific row in the result set with the `FETCH` statement. A fetch operation transfers the data of the row into the application.

```
FETCH cursor_name INTO ...
```

Once an application has processed all available rows or the fetch operation is to be positioned on a non-existing row (compare scrollable cursors below), the DBMS returns a SQLSTATE '02000' (usually accompanied by an SQLCODE +100) to indicate the end of the result set.

The final step involves closing the cursor using the `CLOSE` statement:

```
CLOSE cursor_name
```

After closing a cursor, a program can open it again, which implies that the DBMS re-evaluates the same query or a different query and builds a new result set.

Scrollable cursors

Programmers may declare cursors as scrollable or not scrollable. The scrollability indicates the direction in which a cursor can move.

With a **non-scrollable** (or **forward-only**) cursor, you can `FETCH` each row at most once, and the cursor automatically moves to the next row. After you fetch the last row, if you fetch again, you will put the cursor after the last row and get the following code: SQLSTATE 02000 (SQLCODE +100).

A program may position a **scrollable** cursor anywhere in the result set using the `FETCH` SQL statement. The keyword SCROLL must be specified when declaring the cursor. The default is `NO SCROLL`, although different language bindings like JDBC may apply a different default.

```
DECLARE cursor_name sensitivity SCROLL CURSOR FOR SELECT ... FROM ...
```

The target position for a scrollable cursor can be specified relatively (from the current cursor position) or absolutely (from the beginning of the result set).

```
FETCH [ NEXT | PRIOR | FIRST | LAST ] FROM cursor_name
```

```
FETCH ABSOLUTE n FROM cursor_name
```

```
FETCH RELATIVE n FROM cursor_name
```

Scrollable cursors can potentially access the same row in the result set multiple times. Thus, data modifications (insert, update, delete operations) from other transactions could have an impact on the result set. A cursor can be SENSITIVE or INSENSITIVE to such data modifications. A sensitive cursor picks up data modifications impacting the result set of the cursor, and an insensitive cursor does not. Additionally, a cursor may be ASENSITIVE, in which case the DBMS tries to apply sensitivity as much as possible.

"WITH HOLD"

Cursors are usually closed automatically at the end of a transaction, i.e. when a COMMIT or ROLLBACK (or an implicit termination of the transaction) occurs. That behavior can be changed if the cursor is declared using the WITH HOLD clause. (The default is WITHOUT HOLD.) A holdable cursor is kept open over COMMIT and closed upon ROLLBACK. (Some DBMS deviate from this standard behavior and also keep holdable cursors open over ROLLBACK.)

```
DECLARE cursor_name CURSOR WITH HOLD FOR SELECT ... FROM ...
```

When a COMMIT occurs, a holdable cursor is positioned *before* the next row. Thus, a positioned UPDATE or positioned DELETE statement will only succeed after a FETCH operation occurred first in the transaction.

Note that JDBC defines cursors as holdable per default. This is done because JDBC also activates auto-commit per default. Due to the usual overhead associated with auto-commit and holdable cursors, both features should be explicitly deactivated at the connection level.

Positioned update/delete statements

Cursors can not only be used to fetch data from the DBMS into an application but also to identify a row in a table to be updated or deleted. The SQL:2003 standard defines positioned update and positioned delete SQL statements for that purpose. Such statements do not use a regular WHERE clause with predicates. Instead, a cursor identifies the row. The cursor must be opened and already positioned on a row by means of `FETCH` statement.

```
UPDATE table_name
SET    ...
WHERE  CURRENT OF cursor_name
```

```
DELETE
FROM   table_name
WHERE  CURRENT OF cursor_name
```

The cursor must operate on an updatable result set in order to successfully execute a positioned update or delete statement. Otherwise, the DBMS would not know how to apply the data changes to the underlying tables referred to in the cursor.

Cursors in distributed transactions

Using cursors in distributed transactions (X/Open XA Environments), which are controlled using a transaction monitor, is no different from cursors in non-distributed transactions.

One has to pay attention when using holdable cursors, however. Connections can be used by different applications. Thus, once a transaction has been ended and committed, a subsequent transaction (running in a different application) could inherit existing holdable cursors. Therefore, an application developer has to be aware of that situation.

Cursors in XQuery

The XQuery language allows cursors to be created using the **subsequence()** function.

The format is:

```
let $displayed-sequence := subsequence($result, $start, $item-count)
```

Where $**result** is the result of the initial XQuery, **$start** is the item number to start and **$item-count** is the number of items to return.

Equivalently this can also be done using a predicate:

```
let $displayed-sequence := $result[$start to $end]
```

Where $end is the end sequence.

For complete examples see the XQuery Wikibook [1].

Disadvantages of cursors

The following information may vary depending on the specific database system.

Fetching a row from the cursor may result in a network round trip each time. This uses much more network bandwidth than would ordinarily be needed for the execution of a single SQL statement like DELETE. Repeated network round trips can severely impact the speed of the operation using the cursor. Some DBMSs try to reduce this impact by using block fetch. Block fetch implies that multiple rows are sent together from the server to the client. The client stores a whole block of rows in a local buffer and retrieves the rows from there until that buffer is exhausted.

Cursors allocate resources on the server, such as locks, packages, processes, and temporary storage. For example, Microsoft SQL Server implements cursors by creating a temporary table and populating it with the query's result set. If a cursor is not properly closed (*deallocated*), the resources will not be freed until the SQL session (connection) itself is closed. This wasting of resources on the server can lead to performance degradations and failures.

Example

```
EMPLOYEES TABLE
SQL> desc EMPLOYEES_DETAILS;
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------

 EMPLOYEE_ID                               NOT NULL NUMBER(6)
 FIRST_NAME                                         VARCHAR2(20)
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 EMAIL                                     NOT NULL VARCHAR2(30)
 PHONE_NUMBER                                       VARCHAR2(20)
 HIRE_DATE                                 NOT NULL DATE
 JOB_ID                                    NOT NULL VARCHAR2(10)
 SALARY                                             NUMBER(8,2)
 COMMISSION_PCT                                     NUMBER(2,2)
 MANAGER_ID                                         NUMBER(6)
 DEPARTMENT_ID                                      NUMBER(4)

SAMPLE CURSOR KNOWN AS EE

CREATE OR REPLACE
PROCEDURE EE AS
BEGIN
      DECLARE
v_employeeID EMPLOYEES_DETAILS.EMPLOYEE_ID%TYPE;
v_FirstName EMPLOYEES_DETAILS.FIRST_NAME%TYPE;
v_LASTName EMPLOYEES_DETAILS.LAST_NAME%TYPE;

v_JOB_ID EMPLOYEES_DETAILS.JOB_ID%TYPE:= 'IT_PROG';

Cursor c_EMPLOYEES_DETAILS IS
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME
FROM EMPLOYEES_DETAILS
```

```
WHERE JOB_ID ='v_JOB_ID';
BEGIN
OPEN c_EMPLOYEES_DETAILS;

LOOP

FETCH c_EMPLOYEES_DETAILS INTO v_employeeID,v_FirstName,v_LASTName;

DBMS_OUTPUT.put_line( v_employeeID);
DBMS_OUTPUT.put_line( v_FirstName);
DBMS_OUTPUT.put_line( v_LASTName);

EXIT WHEN c_EMPLOYEES_DETAILS%NOTFOUND;
END LOOP;

CLOSE c_EMPLOYEES_DETAILS;
END;

END;
```

References

- Christopher J. Date: *Database in Depth*, O'Reilly & Associates, ISBN 0-596-10012-4
- Thomas M. Connolly, Carolyn E. Begg: *Database Systems*, Addison-Wesley, ISBN 0-321-21025-5
- Ramiz Elmasri, Shamkant B. Navathe: *Fundamentals of Database Systems*, Addison-Wesley, ISBN 0-201-54263-3
- Neil Matthew, Richard Stones: *Beginning Databases with PostgreSQL: From Novice to Professional*, Apress, ISBN 1-59059-478-9
- Thomas Kyte: *Expert One-On-One: Oracle*, Apress, ISBN 1-59059-525-4
- Kevin Loney: *Oracle Database 10g: The Complete Reference*, Oracle Press, ISBN 0-07-225351-7

External links

- Cursor Optimization Tips (for MS SQL Server) [2]
- Descriptions from Portland Pattern Repository [3]
- PostgreSQL Documentation [4]
- Berkeley DB Reference Guide: Cursor operations [5]
- Java SE 7 [6]
- Q3SqlCursor Class Reference [7]
- OCI Scrollable Cursor [8]
- function oci_new_cursor [9]
- MySQL's Cursor Documentation [10]
- FirebirdSQL cursors documentation [11]
- Cursors in DB2 CLI applications [12]; Cursors in DB2 SQL stored procedures [13]
- A Simple Example of a MySQL Stored Procedure that uses a cursor [14]
- MariaDB/MySQL Cursors: a brief Tutorial [15]

References

[1] http://en.wikibooks.org/wiki/XQuery/Searching,Paging_and_Sorting#Paging
[2] http://www.mssqlcity.com/Tips/tipCursor.htm
[3] http://c2.com/cgi/wiki?DistributedCursor
[4] http://www.postgresql.org/docs/8.3/interactive/plpgsql-cursors.html
[5] http://sleepycat.com/docs/ref/am/cursor.html
[6] http://download.oracle.com/javase/7/docs/
[7] http://doc.trolltech.com/4.0/q3sqlcursor.html
[8] http://www.oracle.com/technology/products/oracle9i/daily/mar15.html
[9] http://de2.php.net/manual/en/function.oci-new-cursor.php
[10] http://dev.mysql.com/doc/refman/5.5/en/cursors.html
[11] http://www.firebirdsql.org/refdocs/langrefupd20-psql-declare.html
[12] http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.apdv.cli.doc/doc/c0007645.htm
[13] http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.apdv.sql.doc/doc/c0024361.htm
[14] http://www.kbedell.com/2009/03/02/a-simple-example-of-a-mysql-stored-procedure-that-uses-a-cursor/
[15] http://falseisnotnull.wordpress.com/2013/06/05/mariadbmysql-cursors-a-brief-tutorial/

SQL Problems Requiring Cursors

A cursor is a construct available in most implementations of SQL that allows the programmer to handle data in a row-by-row manner rather than as a group. Parallelizing row-by-row processing is much more complex than serial processing, which is another reason to make use of non-procedural SQL wherever possible. Database vendors typically handle parallel processing without requiring special handling by application developers.

Parallel processing can be orders of magnitude faster than serial processing.

Constraints

In this article, the following constraints apply:

- **The term "cursor" includes all cursor-like behavior.** For example, using a loop in a shell script that loops across single-row SQL queries or the output of multi-row SQL queries is cursor-like behavior and does not achieve the goal of true set-based processing within the database.
- **All set-based SQL must be ANSI SQL.** A number of vendors provide some extremely powerful proprietary extensions. The goal is to avoid such extensions in favor of ANSI SQL.
- **The solution must be generalizable.** In one or more examples below, specific values may be used for demonstration purposes, but any solution must scale to any number feasible within the power of the database software and machine resources.

Example: Insert rows based on a count in the table itself

The table below represents types of marbles. The four text columns represent four marble characteristics. Each characteristic has two values for a total of 16 types of marbles.

The "quantity" column represents how many marbles of that type we have. The task is to create a second table holding one row for each marble of that type. Thus, the target table would have the four text columns, and a total of 40 + 20 + 20 + 10 + ... + 10 + 5 = 270 rows.

Source table:

```
  QUANTITY TEXTURE    APPEARANCE SHAPE      COLOR
---------- ---------- ---------- ---------- -----
        40 smooth     shiny      round      blue
        20 smooth     shiny      warped     blue
```

```
        20 smooth     dull       round      blue
        10 smooth     dull       warped     blue
        20 rough      shiny      round      blue
        10 rough      shiny      warped     blue
        10 rough      dull       round      blue
         5 rough      dull       warped     blue
        40 rough      dull       warped     red
        20 rough      dull       round      red
        20 rough      shiny      warped     red
        10 rough      shiny      round      red
        20 smooth     dull       warped     red
        10 smooth     dull       round      red
        10 smooth     shiny      warped     red
         5 smooth     shiny      round      red
```

Table to generate:

```
TEXTURE    APPEARANCE SHAPE      COLOR
---------- ---------- ---------- -----
smooth     shiny      round      blue   -- 1
smooth     shiny      round      blue   -- 2
...                                     -- and so on
smooth     shiny      round      blue   -- 40
smooth     shiny      warped     blue   -- 1
smooth     shiny      warped     blue   -- 2
...                                     -- and so on
smooth     shiny      warped     blue   -- 20
...                                     -- and so on
smooth     shiny      round      red    -- 1
smooth     shiny      round      red    -- 2
smooth     shiny      round      red    -- 3
smooth     shiny      round      red    -- 4
smooth     shiny      round      red    -- 5
```

Solution in cursor form

Generating the target table with a cursor is fairly simple.

```
declare
  cursor c is select * from marbles_seed;

begin
  for r in c loop
    for i in 1..r.quantity loop
      insert into marbles values (
        r.texture,
        r.appearance,
        r.shape,
        r.color_actual
      );
```

```
    end loop;
  end loop;
end;
```

Solution in SQL

Solving the problem with SQL is a bit more code and requires a bit more creative thought than the nested loop approach of cursors.

Number table

The solution requires an intermediate table. The table has one column of type NUMBER that has the values 0 to whatever number of rows is needed. For this discussion, we'll limit it to one million rows. The code is as follows: Setup:

```
create table numbers_seed ( n number(1) );
create table numbers ( n number(7));
insert into numbers_seed values ( 0 );
insert into numbers_seed values ( 1 );
insert into numbers_seed values ( 2 );
insert into numbers_seed values ( 3 );
insert into numbers_seed values ( 4 );
insert into numbers_seed values ( 5 );
insert into numbers_seed values ( 6 );
insert into numbers_seed values ( 7 );
insert into numbers_seed values ( 8 );
insert into numbers_seed values ( 9 );
insert into numbers
select n6.n * 100000 +
       n5.n * 10000 +
       n4.n * 1000 +
       n3.n * 100 +
       n2.n * 10 +
       n1.n * 1 n
from numbers_seed n1,
       numbers_seed n2,
       numbers_seed n3,
       numbers_seed n4,
       numbers_seed n5,
       numbers_seed n6
```

The numbers table can be created in parallel.

Solution core

Assume the marble type table above is named marbles_seed and the target table is named marbles. The code that generates the needed 270 rows is:

```
insert into marbles
(m.texture, m.appearance, m.shape, m.color_actual)

select m.texture,
       m.appearance,
       m.shape,
       m.color_actual
  from marbles_seed m,
       numbers n
 where m.quantity > n.n
```

The database can process this insert in parallel without the programmer's involvement.

WxSQLite3

wxSQLite3

Developer(s)	Ulrich Telle
Stable release	3.0.0 / January 2012
Development status	Active
Written in	C++
Operating system	Cross-platform
Type	Development Library
License	wxWindows Library Licence [1]
Website	wxSQLite3 [2]

wxSQLite3 is a C++ wrapper around the public domain SQLite 3.x database and is specifically designed for use in programs based on the wxWidgets library.

wxSQLite3 does not try to hide the underlying database, in contrary almost all special features of the current SQLite version 3.7.10 are supported, like for example the creation of user defined scalar or aggregate functions. Since SQLite stores strings in UTF-8 encoding, the wxSQLite3 methods provide automatic conversion between wxStrings and UTF-8 strings. This works best for the Unicode builds of wxWidgets. In ANSI builds the current locale conversion object (wxConvCurrent) is used for conversion to/from UTF-8. Special care has to be taken if external administration tools are used to modify the database contents, since not all of these tools operate in Unicode resp. UTF-8 mode.

Since version 1.7.0 optional support for key based database encryption (128 bit AES) is also included. Starting with version 1.9.6 of wxSQLite3 the encryption extension is compatible with the SQLite amalgamation source and includes the extension functions module. Support for 256 bit AES encryption has been added in version 1.9.8.

External links

- wxSQLite3 home page [3]
- Lua binding for wxSQLite3 [4]

References

[1] http://www.wxwidgets.org/newlicen.htm
[2] http://wxcode.sourceforge.net/components/wxsqlite3/
[3] http://sourceforge.net/projects/wxcode/files/Components/wxSQLite3/
[4] http://www.dynaset.org/dogusanh/download.html

Database connectivity

Application programming interface

An **application programming interface** (**API**) specifies how some software components should interact with each other.

In addition to accessing databases or computer hardware, such as hard disk drives or video cards, an API can be used to ease the work of programming graphical user interface components. In practice, many times an API comes in the form of a library that includes specifications for routines, data structures, object classes, and variables. In some other cases, notably for SOAP and REST services, an API comes as just a specification of remote calls exposed to the API consumers.

An API specification can take many forms, including an International Standard such as POSIX, vendor documentation such as the Microsoft Windows API, the libraries of a programming language, e.g., Standard Template Library in C++ or Java API. Web APIs are also a vital component of today's web fabric. An API differs from an application binary interface (ABI) in that an API is source code based while an ABI is a binary interface. For instance POSIX is an API, while the Linux Standard Base is an ABI.

Detailed explanation

API in procedural languages

In most procedural languages, an API specifies a set of functions or routines that accomplish a specific task or are allowed to interact with a specific software component. This specification is presented in a human readable format in paper books, or in electronic formats like ebooks or as man pages. For example, the math API on Unix systems is a specification on how to use the mathematical functions included in the math library. Among these functions there is a function, named `sqrt()`, that can be used to compute the square root of a given number.

The Unix command `man 3 sqrt` presents the signature of the function `sqrt` in the form:

```
SYNOPSIS
            #include <math.h>
            double sqrt(double X);
            float  sqrtf(float X);
DESCRIPTION
       sqrt computes the positive square root of the argument. ...
RETURNS
       On success, the square root is returned. If X is real and
positive...
```

This description means that `sqrt()` function returns the square root of a positive floating point number (`single` or `double` precision), as another floating point number.

Hence the API in this case can be interpreted as the collection of the include files used by a program, written in the C language, to reference that library function, and its human readable description provided by the man pages.

Similarly, other languages have procedural libraries; for example, Perl has dedicated APIs for the same mathematical task with built-in documentation available, which is accessible using the perldoc utility:

```
$ perldoc -f sqrt
       sqrt EXPR
       sqrt    #Return the square root of EXPR.  If EXPR is omitted, returns
               #square root of $_.  Only works on non-negative operands, unless
               #you've loaded the standard Math::Complex module.
```

API in object-oriented languages

In its simplest form, an object API is a prescription of how *objects work* in a given object-oriented language – usually it is expressed as a set of classes with an associated list of class methods.

For example, in the Java language, if the class `Scanner` is to be used (a class that reads input from the user in text-based programs), it is required to import the `java.util.Scanner` library, so objects of type `Scanner` can be used by invoking some of the class' methods:

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
        System.out.println("Enter your name: ");
        Scanner inputScanner = new Scanner(System.in);
        String name = inputScanner.nextLine();
        System.out.println("Your name is " + name + ".");
        inputScanner.close();
  }
}
```

In the example above, methods `nextLine()` and `close()` are part of the API for the `Scanner` class, and hence are described in the documentation for that API.

More generally, in object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. This abstract concept is associated with the real functionality exposed, or made available, by the classes that are implemented in terms of class methods (or more generally by all its public components hence all public methods, but also possibly including any internal entity made public, like fields, constants, nested objects, enums, etc.).

The API in this case can be conceived of as the totality of all the methods publicly exposed by the classes (usually called the class *interface*). This means that the API prescribes the methods by which one interacts with/handles the objects derived from the class definitions.

More generally, one can see the API as the collection of all the *kinds* of objects one can derive from the class definitions, and their associated possible behaviors. Again: the use is mediated by the public methods, but in this interpretation, the methods are seen as a *technical detail* of how the behavior is implemented.

For instance: a class representing a `Stack` can simply expose publicly two methods `push()` (to add a new item to the stack), and `pop()` (to extract the last item, ideally placed on top of the stack).

In this case the API can be interpreted as the two methods `pop()` and `push()`, or, more generally, as the *idea* that one can use an item of type `Stack` that implements the behavior of a stack: a pile *exposing* its top to add/remove elements. The second interpretation appears more appropriate in the spirit of object orientation.

This concept can be carried to the point where a class interface in an API has no methods at all, but only behaviors associated with it. For instance, the Java and Lisp language APIs include the interface named `Serializable`, which is a marker interface that requires each class implementing it to behave in a serialized fashion. This does not require implementation of a public method, but rather requires any class which implements this interface to be based

on a representation that can be *saved* (serialized) at any time.[1]

Similarly the behavior of an object in a concurrent (multi-threaded) environment is not necessarily determined by specific methods, belonging to the interface implemented, but still belongs to the API for that Class of objects, and should be described in the documentation.

In this sense, in object-oriented languages, the API defines a set of object behaviors, possibly mediated by a set of class methods.

In such languages, the API is still distributed as a library. For example, the Java language libraries include a set of APIs that are provided in the form of the JDK used by the developers to build new Java programs. The JDK includes the documentation of the API in JavaDoc notation.

The quality of the documentation associated with an API is often a factor determining its success in terms of ease of use.

API libraries and frameworks

An API is usually related to a software library: the API describes and prescribes the *expected behavior* while the library is an *actual implementation* of this set of rules. A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface.

An API can also be related to a software framework: a framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the *access* to the behavior *built into the framework* is mediated by extending its content with new classes plugged into the framework itself. Moreover the overall program flow of control can be out of the control of the caller, and in the hands of the framework via inversion of control or a similar mechanism.

API and protocols

An API can also be an implementation of a protocol.

When an API implements a protocol it can be based on proxy methods for remote invocations that underneath rely on the communication protocol. The role of the API can be exactly to hide the detail of the transport protocol. E.g.: RMI is an API that implements the JRMP protocol or the IIOP as RMI-IIOP.

Protocols are usually shared between different technologies (system based on given computer programming languages in a given operating system) and usually allow the different technologies to exchange information, acting as an abstraction/mediation level between the two different environments. APIs are usually specific to a given technology: hence the APIs of a given language cannot be used in other languages, unless the function calls are wrapped with specific adaptation libraries.

To enable the exchange of information among systems that use different technologies, when an API implements a protocol, it can prescribe a *language-neutral* message format: e.g. SOAP uses XML as a general container for the messages to be exchanged.

Object exchange API and protocols

An object API can prescribe a specific (local) object exchange format, an object exchange protocol can define a way to transfer the same kind of information in a message sent to a remote system.

When a message is exchanged via a protocol between two different platforms using objects on both sides, the object in a programming language can be transformed (marshalled and unmarshalled) in an object in a remote and different language: so, e.g., a program written in Java invokes a service via SOAP or IIOP written in C# both programs use APIs for remote invocation (each locally to the machine where they are working) to (remotely) exchange information that they both convert from/to an object in local memory.

Instead when a similar object is exchanged via an API local to a single machine the object is effectively exchanged (or a reference to it) in memory: e.g. via memory allocated by a single process, or among multiple processes using shared memory, an application server, or other sharing technologies like tuple spaces.

Object remoting API and protocols

An object remoting API is based on a remoting protocol, such as CORBA, that allows remote object method invocation. A method call, executed locally on a proxy object, invokes the corresponding method on the remote object, using the remoting protocol, and acquires the result to be used locally as return value.

When remoting is in place, a modification on the proxy object corresponds to a modification on the remote object. When only an object transfer takes place, the modification to the local copy of the object is not reflected on the original object, unless the object is sent back to the sending system.

API sharing and reuse via virtual machine

Some languages like those running in a virtual machine (e.g. .NET CLI compliant languages in the Common Language Runtime (CLR), and JVM compliant languages in the Java Virtual Machine) can share an API. In this case, a virtual machine enables language interoperability, by abstracting a programming language using an intermediate bytecode and its language bindings. In these languages, the compiler performs just-in-time compilation or ahead-of-time compilation transforming the source code, possibly written in multiple languages, into its language-independent bytecode representation.

For instance, through the bytecode representation, a program written in Groovy or Scala language can use any standard Java class and hence any Java API. This is possible thanks to the fact both Groovy and Scala have an object model that is a superset of that of the Java language; thus, any API exposed via a Java object is accessible via Groovy or Scala by an equivalent object invocation translated in bytecode.

On the other side, Groovy and Scala introduce first-class *entities* that are not present in Java, like closures. These entities cannot be natively represented in Java language (Java 8 will include a concept of closure); thus, in order to enable interoperation a closure is encapsualted in a *standard* object. In this case the *closure invocation* is mediated by a method named `call()` which is always present in an closure object as seen by Java.

Web APIs

When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. While "web API" historically has been virtually synonymous for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based web services and service-oriented architecture (SOA) towards more direct representational state transfer (REST) style web resources and resource-oriented architecture (ROA). Part of this trend is related to the Semantic Web movement toward Resource Description Framework (RDF), a concept to promote web-based ontology engineering technologies. Web APIs allow the combination of multiple APIs into new applications known as mashups.

Web use to share content

The practice of publishing APIs has allowed web communities to create an open architecture for sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web:

- Photos can be shared from sites like Flickr and Photobucket to social network sites like Facebook and MySpace.
- Content can be embedded, e.g. embedding a presentation from SlideShare on a LinkedIn profile.
- Content can be dynamically posted. Sharing live comments made on Twitter with a Facebook account, for example, is enabled by their APIs.
- Video content can be embedded on sites served by another host.
- User information can be shared from web communities to outside applications, delivering new functionality to the web community that shares its user data via an open API. One of the best examples of this is the Facebook Application platform. Another is the Open Social platform.
- If content is a direct representation of the physical world (e.g., temperature at a geospatial location on earth) then an API can be considered an "Environmental Programming Interface" (EPI). EPIs are characterized by their ability to provide a means for universally sequencing events sufficient to utilize real-world data for decision making.

Implementations

The POSIX standard defines an API that allows writing a wide range of common computing functions in a way such that they can operate on many different systems (Mac OS X, and various Berkeley Software Distributions (BSDs) implement this interface). However, using this requires re-compiling for each platform. A compatible API, on the other hand, allows compiled object code to function with no changes to the system that implements that API. This is beneficial to both software providers (where they may distribute existing software on new systems without producing and distributing upgrades) and users (where they may install older software on their new systems without purchasing upgrades), although this generally requires that various software libraries implement the necessary APIs as well.

Microsoft has shown a strong commitment to a backward compatible API, particularly within their Windows API (Win32) library, such that older applications may run on newer versions of Windows using an executable-specific setting called "Compatibility Mode".

Among Unix-like operating systems, there are many related but incompatible operating systems running on a common hardware platform (particularly Intel 80386-compatible systems). There have been several attempts to standardize the API such that software vendors may distribute one binary application for all these systems; however, to date, none of these has met with much success. The Linux Standard Base is attempting to do this for the Linux platform, while many of the BSD Unixes, such as FreeBSD, NetBSD, and OpenBSD, implement various levels of API compatibility for both backward compatibility (allowing programs written for older versions to run on newer distributions of the system) and cross-platform compatibility (allowing execution of foreign code without recompiling).

Release policies

The main policies for releasing an API are:

- Protecting information on APIs from the general public. For example, Sony used to make its official PlayStation 2 API available only to licensed PlayStation developers. This enabled Sony to control who wrote PlayStation 2 games. This gives companies quality control privileges and can provide them with potential licensing revenue streams.
- Making APIs freely available. For example, Microsoft makes the Microsoft Windows API public, and Apple releases its APIs Carbon and Cocoa, so that software can be written for their platforms.

A mix of the two behaviors can be used as well.

Public API implications

An API can be developed for a restricted group of users, or it can be released to the public.

An important factor when an API becomes public is its *interface stability*: if the developer of an API changes a part of it, for example by adding new parameters to some function calls, it could break the compatibility with all clients that depend on or use that API.

When parts of a publicly presented API are subject to change and thus not stable, such parts of a particular API should be explicitly documented as *unstable*. For example, in the Google Guava library the parts that are considered unstable, and that might change in a near future, are marked with the Java annotation `@Beta`.

API deprecation

A public API can sometimes declare parts of itself as *deprecated*. This usually means that such part of an API should be considered candidated for being removed, or modified in a backward incompatible way.

When adopting a third-party public API, developers should consider the deprecation policy used by the producer of that API; if a developer publicly releases a solution based on an API that becomes deprecated, he/she might be unable to guarantee the provided service.

APIs and copyrights

In 2010, Oracle sued Google for having distributed a new implementation of Java embedded in the Android operating system. Google had not acquired any permission to reproduce the Java API, although a similar permission had been given to the OpenJDK project. Judge William Alsup ruled in the Oracle v. Google case that APIs cannot be copyrighted in the U.S, and that a victory for Oracle would have widely expanded copyright protection and allowed the copyrighting of simple software commands:

> To accept Oracle's claim would be to allow anyone to copyright one version of code to carry out a system of commands and thereby bar all others from writing their own different versions to carry out all or part of the same commands.

2013 saw the creation of the "API Commons" initiative. API Commons is a common place to publish and share your own API specifications and data models in any format such as Swagger, API Blueprint or RAML, as well as to explore and discover the API designs of others. The API specifications and data models declared in API Commons are available publicly under the Creative Commons license.

API examples

- ASPI for SCSI device interfacing
- Cocoa and Carbon for the Macintosh
- DirectX for Microsoft Windows
- EHLLAPI
- Java APIs
- ODBC for Microsoft Windows
- OpenAL cross-platform sound API
- OpenCL cross-platform API for general-purpose computing for CPUs & GPUs
- OpenGL cross-platform graphics API
- OpenMP API that supports multi-platform shared memory multiprocessing programming in C, C++ and Fortran on many architectures, including Unix and Microsoft Windows platforms.
- Server Application Programming Interface (SAPI)
- Simple DirectMedia Layer (SDL)

Language bindings and interface generators

APIs that are intended to be used by more than one high-level programming language often provide, or are augmented with, facilities to automatically map the API to features (syntactic or semantic) that are more natural in those languages. This is known as language binding, and is itself an API. The aim is to encapsulate most of the required functionality of the API, leaving a "thin" layer appropriate to each language.

Below are listed some interface generator tools that bind languages to APIs at compile time:

- SWIG – an open-source interfaces bindings generator supporting numerous programming languages
- F2PY – a Fortran to Python interface generator

Notes

[1] This is typically true for any class containing simple data and no link to external resources, like an open connection to a file, a remote system, or an external device.

References

External links

- What is an API? (http://www.3scale.net/wp-content/uploads/2012/06/What-is-an-API-1.0.pdf)
- How to design a good API and why it matters (http://lcsd05.cs.tamu.edu/slides/keynote.pdf)
- How to Write an API (http://www.lior.ca/publications/api_design.pdf)

Structured Query Language Interface

The **Structured Query Language Interface (SQLI)** is the internal interface between an application and the INFORMIX Online Dynamic Server. Starting from v10.0 Informix Dynamic Server also supports DRDA.

External links

- Setting SQLIDEBUG on the client side [1]

References

[1] http://www-1.ibm.com/support/docview.wss?rs=630&context=SSGU8G&dc=DB520&uid=swg21104625&loc=en_US&cs=UTF-8&lang=en&rss=ct630db2

Database transaction

A **transaction** comprises a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. Transactions in a database environment have two main purposes:

1. To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.
2. To provide isolation between programs accessing a database concurrently. If this isolation is not provided, the program's outcome are possibly erroneous.

A database transaction, by definition, must be atomic, consistent, isolated and durable.[1] Database practitioners often refer to these properties of database transactions using the acronym ACID.

Transactions provide an "all-or-nothing" proposition, stating that each work-unit performed in a database must either complete in its entirety or have no effect whatsoever. Further, the system must isolate each transaction from other transactions, results must conform to existing constraints in the database, and transactions that complete successfully must get written to durable storage.

Purpose

Databases and other data stores which treat the integrity of data as paramount often include the ability to handle transactions to maintain the integrity of data. A single transaction consists of one or more independent units of work, each reading and/or writing information to a database or other data store. When this happens it is often important to ensure that all such processing leaves the database or data store in a consistent state.

Examples from double-entry accounting systems often illustrate the concept of transactions. In double-entry accounting every debit requires the recording of an associated credit. If one writes a check for $100 to buy groceries, a transactional double-entry accounting system must record the following two entries to cover the single transaction:

1. Debit $100 to Groceries Expense Account
2. Credit $100 to Checking Account

A transactional system would make both entries pass or both entries would fail. By treating the recording of multiple entries as an atomic transactional unit of work the system maintains the integrity of the data recorded. In other words, nobody ends up with a situation in which a debit is recorded but no associated credit is recorded, or vice versa.

Transactional databases

A **transactional database** is a DBMS where write transactions on the database are able to be rolled back if they are not completed properly (e.g. due to power or connectivity loss).

Most modern[2] relational database management systems fall into the category of databases that support transactions.

In a database system a transaction might consist of one or more data-manipulation statements and queries, each reading and/or writing information in the database. Users of database systems consider consistency and integrity of data as highly important. A simple transaction is usually issued to the database system in a language like SQL wrapped in a transaction, using a pattern similar to the following:

1. Begin the transaction
2. Execute a set of data manipulations and/or queries
3. If no errors occur then commit the transaction and end it
4. If errors occur then rollback the transaction and end it

If no errors occurred during the execution of the transaction then the system commits the transaction. A transaction commit operation applies all data manipulations within the scope of the transaction and persists the results to the database. If an error occurs during the transaction, or if the user specifies a rollback operation, the data manipulations within the transaction are not persisted to the database. In no case can a partial transaction be committed to the database since that would leave the database in an inconsistent state.

Internally, multi-user databases store and process transactions, often by using a transaction ID or XID.

There are multiple varying ways for transactions to be implemented other than the simple way documented above. Nested transactions, for example, are transactions which contain statements within them that start new transactions (i.e. sub-transactions). *Multi-level transactions* are a variant of nested transactions where the sub-transactions take place at different levels of a layered system architecture (e.g., with one operation at the database-engine level, one operation at the operating-system level) [3] Another type of transaction is the compensating transaction.

In SQL

Transactions are available in most SQL database implementations, though with varying levels of robustness. (MySQL, for example, does not support transactions in the MyISAM storage engine, which was its default storage engine before version 5.5.)

A transaction is typically started using the command `BEGIN` (although the SQL standard specifies `START TRANSACTION`). When the system processes a `COMMIT` statement, the transaction ends with successful completion. A `ROLLBACK` statement can also end the transaction, undoing any work performed since `BEGIN TRANSACTION`. If autocommit was disabled using `START TRANSACTION`, autocommit will also be re-enabled at the transaction's end.

One can set the isolation level for individual transactional operations as well as globally. At the READ COMMITTED level, the result of any work done after a transaction has commenced, but before it has ended, will remain invisible to other database-users until it has ended. At the lowest level (READ UNCOMMITTED), which may occasionally be used to ensure high concurrency, such changes will be visible.

Distributed transactions

Database systems implement distributed transactions as transactions against multiple applications or hosts. A distributed transaction enforces the ACID properties over multiple systems or data stores, and might include systems such as databases, file systems, messaging systems, and other applications. In a distributed transaction a coordinating service ensures that all parts of the transaction are applied to all relevant systems. As with database and other transactions, if any part of the transaction fails, the entire transaction is rolled back across all affected systems.

Transactional filesystems

The Namesys Reiser4 filesystem for Linux[4] supports transactions, and as of Microsoft Windows Vista, the Microsoft NTFS filesystem[5] supports distributed transactions across networks.

References

[1] A transaction is a group of operations that are atomic, consistent, isolated, and durable ([[ACID (http://msdn.microsoft.com/en-us/library/aa366402(VS.85).aspx)]).]

[2] http://en.wikipedia.org/w/index.php?title=Database_transaction&action=edit

[3] Beeri, C., Bernstein, P.A., and Goodman, N. A model for concurrency in nested transactions systems. Journal of the ACM, 36(1):230-269, 1989

[4] namesys.com (http://namesys.com/v4/v4.html#committing)

[5] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/portal.asp

Further reading

- Philip A. Bernstein, Eric Newcomer (2009): *Principles of Transaction Processing*, 2nd Edition (http://www.elsevierdirect.com/product.jsp?isbn=9781558606234), Morgan Kaufmann (Elsevier), ISBN 978-1-55860-623-4
- Gerhard Weikum, Gottfried Vossen (2001), *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, ISBN 1-55860-508-8

Prepared statement

In database management systems, a **prepared statement** or **parameterized statement** is a feature used to execute the same or similar database statements repeatedly with high efficiency. Typically used with SQL statements such as queries or updates, the prepared statement takes the form of a template into which certain constant values are substituted during each execution.

The typical workflow of using a prepared statement is as follows:

1. **Prepare**: The statement template is created by the application and sent to the database management system (DBMS). Certain values are left unspecified, called *parameters*, *placeholders* or *bind variables* (labelled "?" below):
   - `INSERT INTO PRODUCT (name, price) VALUES (?, ?)`
2. The DBMS parses, compiles, and performs query optimization on the statement template, and stores the result without executing it.
3. **Execute**: At a later time, the application supplies (or *binds*) values for the parameters, and the DBMS executes the statement (possibly returning a result). The application may execute the statement as many times as it wants with different values. In this example, it might supply 'Bread' for the first parameter and '1.00' for the second parameter.

As compared to executing SQL statements directly, prepared statements offer two main advantages:

- The overhead of compiling and optimizing the statement is incurred only once, although the statement is executed multiple times. Not all optimization can be performed at the time the prepared statement is compiled, for two reasons: the best plan may depend on the specific values of the parameters, and the best plan may change as tables and indexes change over time.
- Prepared statements are resilient against SQL injection, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

On the other hand, if a query is executed only once, server-side prepared statements can be slower because of the additional round-trip to the server. Implementation limitations may also lead to performance penalties: some versions of MySQL did not cache results of prepared queries, and some DBMSs such as PostgreSQL do not perform additional query optimization during execution.

A stored procedure, which is also precompiled and stored on the server for later execution, has similar advantages. Unlike a stored procedure, a prepared statement is not normally written in a procedural language and cannot use or modify variables or use control flow structures, relying instead on the declarative database query language. Due to their simplicity and client-side emulation, prepared statements are more portable across vendors.

Software support

Prepared statements are widely supported by major DBMSs, including MySQL, Oracle, DB2, Microsoft SQL Server, and PostgreSQL. Prepared statements are normally executed through a non-SQL binary protocol, for efficiency and protection from SQL injection, but with some DBMSs such as MySQL are also available using a SQL syntax for debugging purposes.

A number of programming languages support prepared statements in their standard libraries and will emulate them on the client side even if the underlying DBMS does not support them, including Java's JDBC, Perl's DBI, PHP's PDO and Python's DB-API. Client-side emulation can be faster for queries which are executed only once, by reducing the number of round trips to the server, but is usually slower for queries executed many times. It resists SQL injection attacks equally effectively.

Many types of SQL injection attacks can be eliminated by *disabling literals*, effectively requiring the use of prepared statements; as of 2007 only H2 supports this feature.

Examples

Java JDBC

This example uses Java and the JDBC API:

```
java.sql.PreparedStatement stmt = connection.prepareStatement(
                "SELECT * FROM users WHERE USERNAME = ? AND ROOM = ?");
stmt.setString(1, username);
stmt.setInt(2, roomNumber);
stmt.executeQuery();
```

Java `PreparedStatement` provides "setters" (`setInt(int)`, `setString(String)`, `setDouble(double)`, etc.) for all major built-in data types.

PHP PDO

This example uses PHP and PHP Data Objects (PDO):

```
$stmt = $dbh->prepare("SELECT * FROM users WHERE USERNAME = ? AND
PASSWORD = ?");
$stmt->execute(array($username, $password));
```

PERL DBI

This example uses Perl and DBI:

```
my $stmt = $dbh->prepare('SELECT * FROM users WHERE USERNAME = ? AND
PASSWORD = ?');
$stmt->execute($username, $password);
```

C# ADO.NET

This example uses C# and ADO.NET:

```
using (SqlCommand command = connection.CreateCommand())
{
    command.CommandText = "SELECT * FROM users WHERE USERNAME =
@username AND ROOM = @room";

    command.Parameters.AddWithValue("@username", username);
    command.Parameters.AddWithValue("@room", room);

    using (SqlDataReader dataReader = command.ExecuteReader())
    {
        // ...
    }
}
```

ADO.NET `SqlCommand` will accept any type for the `value` parameter of `AddWithValue`, and type conversion occurs automatically. Note the use of "named parameters" (i.e. "@username") rather than "?" - this allows you to use a parameter multiple times and in any arbitrary order within the query command text.

However, the AddWithValue method should not be used with variable length data types, like varchar and nvarchar. This is because .NET assumes the length of the parameter to be the length of the given value, rather than getting the actual length of from the database via reflection. The consequence of this is that a different query plan is compiled and stored for each different length. In general, the maximum number of 'duplicate' plans is the product of the lengths of the variable length columns as specified in the database. For this reason, it is important to use the standard Add method for variable length columns:

command.Parameters.Add(ParamName, VarChar, ParamLength).Value = ParamValue, where ParamLength is the length as specified in the database.

Since the standard Add method needs to be used for variable length data types, it is a good habit to use it for all parameter types.

Python DB-API

This example uses Python DB-API with SQLite and `paramstyle='qmark'`:

```
import sqlite3
conn = sqlite3.connect(':memory:')
c = conn.cursor()

_users = [('mother', 'red'),
          ('father', 'green'),
          ('me', 'blue')]
c.executemany('INSERT INTO users VALUES (?,?)', _users)

params = ('B', 'green')
c.execute('SELECT * FROM users WHERE username=? AND room=?', params)
c.fetchone()
```

References

Open Database Connectivity

In computing, **ODBC** (**Open Database Connectivity**) is a standard programming language middleware API for accessing database management systems (DBMS). The designers of ODBC aimed to make it independent of database systems and operating systems; an application written using ODBC can be ported to other platforms, both on the client and server side, with few changes to the data access code.

ODBC accomplishes DBMS independence by using an **ODBC driver** as a translation layer between the application and the DBMS. The application uses ODBC functions through an **ODBC driver manager** with which it is linked, and the driver passes the query to the DBMS. An ODBC driver can be thought of as analogous to a printer or other driver, providing a standard set of functions for the application to use, and implementing DBMS-specific functionality. An application that can use ODBC is referred to as "ODBC-compliant". Any ODBC-compliant application can access any DBMS for which a driver is installed. Drivers exist for all major DBMSs, many other data sources like address book systems and Microsoft Excel, and even for text or CSV files.

ODBC was originally developed by Microsoft during the early 1990s, and became the basis for the Call Level Interface (CLI) standardized by SQL Access Group in the Unix and mainframe world. ODBC retained a number of features that were removed as part of the CLI effort. Full ODBC was later ported back to those platforms, and became a de facto standard considerably better known than CLI. The CLI remains similar to ODBC, and applications can be ported from one platform to the other with few changes.

History

Prior to ODBC

The introduction of the mainframe-based relational database during the 1970s led to a proliferation of data access methods. Generally these systems operated hand-in-hand with a simple command processor that allowed the user to type in English-like commands, and receive output. The best-known examples are SQL from IBM and QUEL from the Ingres project. These systems may or may not allow other applications to access the data directly, and those that did used a wide variety of methodologies. The introduction of SQL aimed to solve the problem of *language* standardization, although substantial differences in implementation remained.

Additionally, since the SQL language had only rudimentary programming features, it was often desired to use SQL within a program written in another language, say Fortran or C. This led to the concept of Embedded SQL, which allowed SQL code to be "embedded" within another language. For instance, a SQL statement like `SELECT * FROM city` could be inserted as text within C source code, and during compilation it would be converted into a custom format that directly called a function within a library that would pass the statement into the SQL system. Results returned from the statements would be interpreted back into C data formats like `char *` using similar library code.

There were a number of problems with the Embedded SQL approach. Like the different varieties of SQL, the Embedded SQL's that used them varied widely, not only from platform to platform, but even across languages on a single platform - a system that allowed calls into IBM's DB2 would look entirely different from one that called into their own SQL/DSWikipedia:Disputed statement. Another key problem to the Embedded SQL concept was that the SQL code could only be changed in the program's source code, so that even small changes to the query required considerable programmer effort to modify. The SQL market referred to this as "static SQL", as opposed to "dynamic SQL" which could be changed at any time - like the command-line interfaces that shipped with almost all SQL systems, or a programming interface that left the SQL as plain text until it was called. Dynamic SQL systems became a major focus for SQL vendors during the 1980s.

Older mainframe databases, and the newer microcomputer based systems that were based on them, generally did not have a SQL-like command processor between the user and the database engine. Instead, the data was accessed directly by the program - a programming library in the case of large mainframe systems, or a command line interface or interactive forms system in the case of dBASE and similar applications. Data from dBASE could not generally be accessed directly by other programs running on the machine. Those programs may be given a way to access this data, often through libraries, but it would not work with any other database engine, or even different databases in the same engine. In effect, all such systems were static, which presented considerable problems.

Early efforts

By the mid-1980s the rapid improvement in microcomputers, and especially the introduction of the graphical user interface and data-rich application programs like Lotus 1-2-3 led to an increasing interest in using personal computers as the client-side platform of choice in client-server computing. Under this model, large mainframes and minicomputers would be used primarily to serve up data over local area networks to microcomputers that would interpret, display and manipulate that data. For this model to work, a data access standard was a requirement - in the mainframe world it was highly likely that all of the computers in a shop were from a single vendor and clients were computer terminals talking directly to them, but in the micro world there was no such standardization and any client might access any server using any networking system.

By the late 1980s there were a number of efforts underway to provide an abstraction layer for this purpose. Some of these were mainframe related, designed to allow programs running on those machines to translate between the variety of SQL's and provide a single common interface which could then be called by other mainframe or microcomputer programs. These solutions included IBM's Distributed Relational Database Architecture (DRDA) and Apple Computer's Data Access Language. Much more common, however, were systems that ran entirely on microcomputers, including a complete protocol stack that included any required networking or file translation support.

One of the early examples of such a system was Lotus Development's DataLens, initially known as Blueprint. Blueprint, developed for 1-2-3, supported a variety of data sources, including SQL/DS, DB2, FOCUS and a variety of similar mainframe systems, as well as microcomputer systems like dBase and the early Microsoft/Ashton-Tate efforts that would eventually develop into Microsoft SQL Server.[1] Unlike the later ODBC, Blueprint was a purely code-based system, lacking anything approximating a command language like SQL. Instead, programmers used data structures to store the query information, constructing a query by linking many of these structures together. Lotus referred to these compound structures as "query trees".[2]

Around the same time, an industry team including members from Sybase, Tandem Computers and Microsoft were working on a standardized dynamic SQL concept. Much of the system was based on Sybase's DB-Library system, with the Sybase-specific sections removed and several additions to support other platforms.[3] DB-Library was aided by an industry-wide move from library systems that were tightly linked to a particular language, to library systems that were provided by the operating system and required the languages on that platform to conform to its standards. This meant that a single library could be used with (potentially) any programming language on a given platform.

The first draft of the **Microsoft Data Access API** was published in April 1989, about the same time as Lotus' announcement of Blueprint.[4] In spite of Blueprint's great lead - it was running when MSDA was still a paper project - Lotus eventually joined the MSDA efforts as it became clear that SQL would become the de facto database standard.[2] After considerable industry input, in the summer of 1989 the standard became **SQL Connectivity**, or **SQLC** for short,.[5]

SAG and CLI

In 1988 a number of vendors, mostly from the Unix and database communities, formed the SQL Access Group (SAG) in an effort to produce a single basic standard for the SQL language. At the first meeting there was considerable debate over whether or not the effort should work solely on the SQL language itself, or attempt a wider standardization which included a dynamic SQL language-embedding system as well, what they called a Call Level Interface (CLI).[6] While attending the meeting with an early draft of what was then still known as MS Data Access, Kyle Geiger of Microsoft invited Jeff Balboni and Larry Barnes of Digital Equipment Corporation (DEC) to join the SQLC meetings as well. SQLC was a potential solution to the call for the CLI, which was being led by DEC.

The new SQLC "gang of four", MS, Lotus, DEC and Sybase, brought an updated version of SQLC to the next SAG meeting in June 1990.[7] The SAG responded by opening the standard effort to any competing design, but of the many proposals, only Oracle Corp had a system that presented serious competition. In the end, SQLC won the votes and became the draft standard, but only after large portions of the API were removed - the standards document was trimmed from 120 pages to 50 during this time. It was also during this period that the name Call Level Interface was formally adopted.[7] In 1995 SQL/CLI became part of the international SQL standard, ISO/IEC 9075-3.[8] The SAG itself was taken over by the X/Open group in 1996, and, over time, became part of The Open Group's Common Application Environment.

MS continued working with the original SQLC standard, retaining many of the advanced features that were removed from the CLI version. These included features like scrollable cursors, and metadata information queries. The commands in the API were split into groups; the Core group was identical to the CLI, the Level 1 extensions were commands that would be easy to implement in drivers, while Level 2 commands contained the more advanced features like cursors. A proposed standard was released in December 1991, and industry input was gathered and worked into the system through 1992, resulting in yet another name change to **ODBC**.[9]

JET and ODBC

During this time, Microsoft was in the midst of developing their Jet database system. Jet combined three primary subsystems; an ISAM-based database engine (also known as "Jet", confusingly), a C-based interface allowing applications to access that data, and a selection of driver DLLs that allowed the same C interface to redirect input and output to other ISAM-based databases, like Paradox and xBase. Jet allowed programmers to use a single set of calls to access common microcomputer databases in a fashion similar to Blueprint (by this point known as DataLens). However, Jet did not use SQL; like DataLens, the interface was in C and consisted of data structures and function calls.

The SAG standardization efforts presented an opportunity for Microsoft to adapt their Jet system to the new CLI standard. This would not only make Windows a premier platform for CLI development, but also allow users to use SQL to access both Jet and other databases as well. What was missing was the SQL parser that could convert those calls from their text form into the C-interface used in Jet. To solve this, MS partnered with PageAhead Software to use their existing query processor, "SIMBA". SIMBA was used as a parser above Jet's C library, turning Jet into an SQL database. And because Jet could forward those C-based calls to other databases, this also allowed SIMBA to query other systems. Microsoft included drivers for Excel to turn its spreadsheet documents into SQL-accessible database tables.

Release and continued development

ODBC 1.0 was released in September 1992. At the time, there was little direct support for SQL databases (as opposed to ISAM), and early drivers were noted for poor performance. Some of this was unavoidable due to the path that the calls took through the Jet-based stack; ODBC calls to SQL databases were first converted from SIMBA's SQL dialect to Jet's internal C-based format, then passed to a driver for conversion back into SQL calls for the database. Digital Equipment and Oracle both contracted Simba to develop drivers for their databases as well.[10]

Circa 1993, OpenLink Software shipped one of the first independently developed third-party ODBC drivers, for the PROGRESS DBMS, and soon followed with their UDBC (a cross-platform API equivalent of ODBC and the SAG/CLI) SDK and associated drivers for PROGRESS, Sybase, Oracle, and other DBMS, for use on Unix-like OS (AIX, HP-UX, Solaris, Linux, etc.), VMS, Windows NT, OS/2, and other OS.

Meanwhile the CLI standard effort dragged on, and it was not until March 1995 that the definitive version was finalized. By this time Microsoft had already granted Visigenic Software a source code license to develop ODBC on non-Windows platforms. Visigenic ported ODBC to a wide variety of Unix platforms, where ODBC quickly became the de facto standard.[11] "Real" CLI is rare today. The two systems remain similar, and many applications can be ported from ODBC to CLI with few or no changes.[12]

Over time, database vendors took over the driver interfaces and provided direct links to their products. Skipping the intermediate conversions to and from Jet or similar wrappers often resulted in higher performance. However, by this time Microsoft had changed focus to their OLE DB concept, which provided direct access to a wider variety of data sources from address books to text files. Several new systems followed which further turned their attention from ODBC, including DAO, ADO and ADO.net, which interacted more or less with ODBC over their lifetimes.

As Microsoft turned its attention away from working directly on ODBC, the Unix world was increasingly embracing it. This was propelled by two changes within the market, the introduction of GUIs like GNOME that provided the need for access to these sources in non-text form, and the emergence of open software database systems like PostgreSQL and MySQL, initially under Unix. The later adoption of ODBC by Apple for using the standard Unix-side iODBC package Mac OS X 10.2 (Jaguar) (which OpenLink Software had been independently providing for Mac OS X 10.0 and even Mac OS 9 since 2001) further cemented ODBC as the standard for cross-platform data access.

Sun Microsystems used the ODBC system as the basis for their own open standard, JDBC. In most ways, JDBC can be considered a version of ODBC for the Java programming language as opposed to C. JDBC-to-ODBC "bridges" allow Java-based programs to access data sources through ODBC drivers on platforms lacking a native JDBC driver, although these are now relatively rare. Inversely, ODBC-to-JDBC "bridges" allow C-based programs to access data sources through JDBC drivers on platforms or from databases lacking suitable ODBC drivers.

ODBC today

ODBC remains largely universal today, with drivers available for most platforms and most databases. It is not uncommon to find ODBC drivers for database engines that are meant to be embedded, like SQLite, as a way to allow existing tools to act as front-ends to these engines for testing and debugging.[13]

However, the rise of thin client computing using HTML as an intermediate format has reduced the need for ODBC. Many web development platforms contain direct links to target databases - MySQL being particularly common. In these scenarios, there is no direct client-side access nor multiple client software systems to support; everything goes through the programmer-supplied HTML application. The virtualization that ODBC offers is no longer a strong requirement, and development of ODBC is no longer as active as it once was.

Version history

Version history:

- 1.0: released in September 1992
- 2.0: ca 1994
- 2.5
- 3.0: ca 1995, John Goodson of Intersolv and Frank Pellow and Paul Cotton of IBM provided significant input to ODBC 3.0[14]
- 3.5: ca 1997
- 3.8: ca 2009, with Windows 7

Drivers and Managers

Drivers

ODBC is based on the device driver model, where the driver encapsulates the logic needed to convert a standard set of commands and functions into the specific calls required by the underlying system. For instance, a printer driver presents a standard set of printing commands, the API, to applications using the printing system. Calls made to those APIs are converted by the driver into the format used by the actual hardware, say PostScript or PCL.

In the case of ODBC, the drivers encapsulate a number of functions that can be broken down into several broad categories. One set of functions is primarily concerned with finding, connecting to and disconnecting from the DBMS that driver talks to. A second set is used to send SQL commands from the ODBC system to the DBMS, converting or interpreting any commands that are not supported internally. For instance, a DBMS that does not support cursors can emulate this functionality in the driver. Finally, another set of commands, mostly used internally, is used to convert data from the DBMS's internal formats to a set of standardized ODBC formats, which are based on the C language formats.

An ODBC driver enables an ODBC-compliant application to use a *data source*, normally a DBMS. Some non-DBMS drivers exist, for such data sources as CSV files, by implementing a small DBMS inside the driver itself. ODBC drivers exist for most DBMSs, including Oracle, PostgreSQL, MySQL, Microsoft SQL Server (but not for the Compact aka CE edition), Sybase ASE, and DB2. Because different technologies have different capabilities, most ODBC drivers do not implement all functionality defined in the ODBC standard. Some drivers offer extra functionality not defined by the standard.

Driver Manager

Device drivers are normally enumerated, set up and managed by a separate Manager layer, which may provide additional functionality. For instance, printing systems often include functionality to provide spooling functionality on top of the drivers, providing print spooling for any supported printer.

In ODBC the Driver Manager (DM) provides these features. The DM can enumerate the installed drivers and present this as a list, often in a GUI-based form.

But more important to the operation of the ODBC system is the DM's concept of **Data Source Name**s, or **DSN**. DSNs collect additional information needed to connect to a *particular* data source, as opposed to the DBMS itself. For instance, the same MySQL driver can be used to connect to any MySQL server, but the connection information to connect to a local private server is different from the information needed to connect to an internet-hosted public server. The DSN stores this information in a standardized format, and the DM provides this to the driver during connection requests. The DM also includes functionality to present a list of DSNs using human readable names, and to select them at run-time to connect to different resources.

The DM also includes the ability to save partially complete DSN's, with code and logic to ask the user for any missing information at runtime. For instance, a DSN can be created without a required password. When an ODBC application attempts to connect to the DBMS using this DSN, the system will pause and ask the user to provide the password before continuing. This frees the application developer from having to create this sort of code, as well as having to know which questions to ask. All of this is included in the driver and the DSNs.

Bridging configurations

A *bridge* is a special kind of driver: a driver that uses another driver-based technology.

ODBC-to-JDBC (or simply ODBC-JDBC) bridges

An ODBC-JDBC bridge consists of an **ODBC** driver which uses the services of a JDBC driver to connect to a database. This driver translates ODBC function-calls into JDBC method-calls. Programmers usually use such a bridge when they lack an ODBC driver for a particular database but have access to a JDBC driver.

JDBC-to-ODBC (or simply JDBC-ODBC) bridges

A JDBC-ODBC bridge consists of a JDBC driver which employs an ODBC driver to connect to a target database. This driver translates JDBC method calls into ODBC function calls. Programmers usually use such a bridge when a particular database lacks a JDBC driver. Sun Microsystems included one such bridge in the JVM, but viewed it as a stop-gap measure while few JDBC drivers existed. Sun never intended its bridge for production environments, and generally recommended against its use. As of 2008[15] independent data-access vendors deliver JDBC-ODBC bridges which support current standards for both mechanisms, and which far outperform the JVM built-in.[*citation needed*]

OLE DB-to-ODBC bridges

An OLE DB-ODBC bridge consists of an OLE DB Provider which uses the services of an ODBC driver to connect to a target database. This provider translates OLE DB method calls into ODBC function calls. Programmers usually use such a bridge when a particular database lacks an OLE DB provider. Microsoft ships one, MSDASQL.DLL, as part of the MDAC system component bundle, together with other database drivers, to simplify development in COM-aware languages (e.g. Visual Basic). Third parties have also developed such, notably OpenLink Software whose 64-bit OLE DB Provider for ODBC Data Sources filled the gap when Microsoft initially deprecated this bridge for their 64-bit OS.[16] (Microsoft later relented, and 64-bit Windows starting with Windows Server 2008 and Windows Vista SP1 have shipped with a 64-bit version of MSDASQL.)

ADO.NET-to-ODBC bridges

An ADO.NET-ODBC bridge consists of an ADO.NET Provider which uses the services of an OBDC driver to connect to a target database. This provider translates ADO.NET method calls into ODBC function calls. Programmers usually use such a bridge when a particular database lacks an ADO.NET provider. Microsoft ships one as part of the MDAC system component bundle, together with other database drivers, to simplify development in C#. Third parties have also developed such.

References

Citations

[1] Evan McGlinn, Blueprint Lets 1-2-3 Access Outside Data" (http://books.google.ca/books?id=6D4EAAAAMBAJ), *InfoWorld*, 4 April 1988, p. 1, 69
[2] Geiger 1995, p. 65.
[3] Geiger 1995, p. 86-87.
[4] Geiger 1995, p. 56.
[5] Geiger 1995, p. 106.
[6] Geiger 1995, p. 165.
[7] Geiger 1995, p. 186-187.
[8] ISO/IEC 9075-3 -- Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI)
[9] Geiger 1995, p. 203.
[10] "Our History" (http://www.simba.com/simba-history.htm), Simba Technologies
[11] Roger Sippl, "SQL Access Group's Call-Level Interface" (http://www.drdobbs.com/sql-access-groups-call-level-interface/184410032), Dr. Dobbs, 1 February 1996
[12] "Similarities and differences between ODBC and CLI" (http://publib.boulder.ibm.com/infocenter/iisclzos/v9r5/index.jsp?topic=/com.ibm.swg.im.iis.fed.classic.clientsref.doc/topics/iiyfcodbcclisimdiff.html), InfoSphere Classic documentation, IBM, 26 September 2008
[13] Christian Werner, "SQLite ODBC Driver" (http://www.ch-werner.de/sqliteodbc/)
[14] Microsoft Corporation. Microsoft ODBC 3.0 Programmer's Reference and SDK Guide, Volume 1. Microsoft Press. February 1997. (ISBN13: 9781572315167)
[15] http://en.wikipedia.org/w/index.php?title=ODBC&action=edit
[16] *Microsoft*, "Data Access Technologies Road Map", Deprecated MDAC Components, *Microsoft* "ADO Programmer's Guide" Appendix A: Providers, Microsoft OLE DB Provider for ODBC (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdrefodbcprovspec.asp), retrieved July 30, 2005.

Bibliography

- Geiger, Kyle (1995). *Inside ODBC* (http://books.google.ca/books?id=G-ZQAAAAMAAJ&). Microsoft Press.

External links

- Microsoft ODBC Overview (http://support.microsoft.com/kb/110093)
- List of ODBC Drivers at databasedrivers.com (http://www.databasedrivers.com/odbc/)
- List of ODBC Drivers at SQLSummit.com (http://www.SQLSummit.com/ODBCVend.htm)
- OS400 and i5OS ODBC Administration (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/topic/rzaii/rzaiiodbcadm.htm)
- Presentation slides from www.roth.net (http://www.roth.net/perl/odbc/conf/sld002.htm)
- Early ODBC White Paper (http://www.openlinksw.com/info/docs/odbcwhp/tableof.htm)
- Microsoft ODBC & Data Access APIs History Article (http://blogs.msdn.com/data/archive/2006/12/05/data-access-api-of-the-day-part-i.aspx)

Java Database Connectivity

JDBC

Type	Data Access API
Website	Java SE 7 [6]

JDBC is a Java-based data access technology (Java Standard Edition platform) from Oracle Corporation. This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the JVM host environment.

History and implementation

Sun Microsystems released JDBC as part of JDK 1.1 on February 19, 1997. It has since formed part of the Java Standard Edition.

The JDBC classes are contained in the Java package `java.sql` [1] and `javax.sql` [2].

Starting with version 3.1, JDBC has been developed under the Java Community Process. JSR 54 specifies JDBC 3.0 (included in J2SE 1.4), JSR 114 specifies the JDBC Rowset additions, and JSR 221 is the specification of JDBC 4.0 (included in Java SE 6).[3]

The latest version, JDBC 4.1, is specified by a maintenance release of JSR 221[4] and is included in Java SE 7.[5]

Functionality

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

- `Statement` [6] – the statement is sent to the database server each and every time.
- `PreparedStatement` [7] – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.
- `CallableStatement` [8] – used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API in the `javax.sql` [2].

JDBC connections are often managed via a connection pool rather than obtained directly from the driver. Examples of connection pools include BoneCP [9], C3P0 [10] and DBCP [11]

Examples

The method `Class.forName(String)` [12] is used to load the JDBC driver class. The line below causes the JDBC driver from *some jdbc vendor* to be loaded into the application. (Some JVMs also require the class to be instantiated with `.newInstance()` [13].)

```
Class.forName( "com.somejdbcvendor.TheirJdbcDriver" );
```

In JDBC 4.0, it is no longer necessary to explicitly load JDBC drivers using `Class.forName()`. See JDBC 4.0 Enhancements in Java SE 6 [14].

When a `Driver` [15] class is loaded, it creates an instance of itself and registers it with the `DriverManager` [16]. This can be done by including the needed code in the driver class's `static` block. E.g., `DriverManager.registerDriver(Driver driver)`

Now when a connection is needed, one of the `DriverManager.getConnection()` methods is used to create a JDBC connection.

```
Connection conn = DriverManager.getConnection(
     "jdbc:somejdbcvendor:other data needed by some jdbc vendor",
     "myLogin",
     "myPassword" );
try {
     /* you use the connection here */
} finally {
    //It's important to close the connection when you are done with it
    try { conn.close(); } catch (Throwable ignore) { /* Propagate the original exception
instead of this one that you may want just logged */ }
}
```

Using Java's try-with-resources statement will make the above code cleaner:

```
try (Connection conn = DriverManager.getConnection(
     "jdbc:somejdbcvendor:other data needed by some jdbc vendor",
     "myLogin",
     "myPassword" ) ) {
     /* you use the connection here */
}  // the VM will take care of closing the connection
```

The URL used is dependent upon the particular JDBC driver. It will always begin with the "jdbc:" protocol, but the rest is up to the particular vendor. Once a connection is established, a statement can be created.

```
try (Statement stmt = conn.createStatement()) {
    stmt.executeUpdate( "INSERT INTO MyTable( name ) VALUES ( 'my name' ) " );
}
```

Note that Connections, Statements, and ResultSets often tie up operating system resources such as sockets or file descriptors. In the case of Connections to remote database servers, further resources are tied up on the server, e.g., cursors for currently open ResultSets. It is vital to `close()` any JDBC object as soon as it has played its part; garbage collection should not be relied upon. Forgetting to `close()` things properly results in spurious errors and misbehaviour. The above try-with-resources construct is a recommendedWikipedia:Manual of Style/Words to watch#Unsupported attributions code pattern to use with JDBC objects.

Data is retrieved from the database using a database query mechanism. The example below shows creating a statement and executing a query.

```
try (Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery( "SELECT * FROM MyTable" )
) {
    while ( rs.next() ) {
        int numColumns = rs.getMetaData().getColumnCount();
        for ( int i = 1 ; i <= numColumns ; i++ ) {
           // Column numbers start at 1.
           // Also there are many methods on the result set to return
           //  the column as a particular type. Refer to the Sun documentation
           //  for the list of valid conversions.
           System.out.println( "COLUMN " + i + " = " + rs.getObject(i) );
        }
    }
}
```

Typically, however, it would be rare for a seasoned Java programmer to code in such a fashion. The usual practice would be to abstract the database logic into an entirely different class and to pass preprocessed strings (perhaps derived themselves from a further abstracted class) containing SQL statements and the connection to the required methods. Abstracting the data model from the application code makes it more likely that changes to the application and data model can be made independently.

An example of a `PreparedStatement` query, using `conn` and class from first example.

```
try (PreparedStatement ps =
    conn.prepareStatement( "SELECT i.*, j.* FROM Omega i, Zappa j WHERE i.name = ? AND j.num = ?" )
){
    // In the SQL statement being prepared, each question mark is a placeholder
    // that must be replaced with a value you provide through a "set" method invocation.
    // The following two method calls replace the two placeholders; the first is
    // replaced by a string value, and the second by an integer value.
    ps.setString(1, "Poor Yorick");
    ps.setInt(2, 8008);

    // The ResultSet, rs, conveys the result of executing the SQL statement.
    // Each time you call rs.next(), an internal row pointer, or cursor,
    // is advanced to the next row of the result.  The cursor initially is
    // positioned before the first row.
    try (ResultSet rs = ps.executeQuery()) {
        while ( rs.next() ) {
            int numColumns = rs.getMetaData().getColumnCount();
            for ( int i = 1 ; i <= numColumns ; i++ ) {
                // Column numbers start at 1.
                // Also there are many methods on the result set to return
                // the column as a particular type. Refer to the Sun documentation
                // for the list of valid conversions.
                System.out.println( "COLUMN " + i + " = " + rs.getObject(i) );
            } // for
```

```
        } // while
    } // try
} // try
```

If a database operation fails, JDBC raises an `SQLException` [17]. There is typically very little one can do to recover from such an error, apart from logging it with as much detail as possible. It is recommended that the SQLException be translated into an application domain exception (an unchecked one) that eventually results in a transaction rollback and a notification to the user.

An example of a database transaction:

```
boolean autoCommitDefault = conn.getAutoCommit();
try {
    conn.setAutoCommit(false);

    /* You execute statements against conn here transactionally */

    conn.commit();
} catch (Throwable e) {
    try { conn.rollback(); } catch (Throwable ignore) {}
    throw e;
} finally {
    try { conn.setAutoCommit(autoCommitDefault); } catch (Throwable ignore) {}
}
```

Here are examples of host database types which Java can convert to with a function.

`setXXX()` Methods

Oracle Datatype	`setXXX()`
CHAR	`setString()`
VARCHAR2	`setString()`
NUMBER	`setBigDecimal()`
	`setBoolean()`
	`setByte()`
	`setShort()`
	`setInt()`
	`setLong()`
	`setFloat()`
	`setDouble()`
INTEGER	`setInt()`
FLOAT	`setDouble()`
CLOB	`setClob()`
BLOB	`setBlob()`
RAW	`setBytes()`
LONGRAW	`setBytes()`

DATE	`setDate()`
	`setTime()`
	`setTimestamp()`

For an example of a `CallableStatement` (to call stored procedures in the database), see the Java SE 7 [6].

JDBC drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand.

Types

There are commercial and free drivers available for most relational database servers. These drivers fall into one of the following types:

- Type 1 that calls native code of the locally available ODBC driver.
- Type 2 that calls database vendor native library on a client side. This code then talks to database over network.
- Type 3, the pure-java driver that talks with the server-side middleware that then talks to database.
- Type 4, the pure-java driver that uses database native protocol.

There is also a type called internal JDBC driver, driver embedded with JRE in Java-enabled SQL databases. It's used for Java stored procedures. This does not belong to the above classification, although it would likely be either a type 2 or type 4 driver (depending on whether the database itself is implemented in Java or not). An example of this is the KPRB driver supplied with Oracle RDBMS. "jdbc:default:connection" is a relatively standard way of referring making such a connection (at least Oracle and Apache Derby support it). The distinction here is that the JDBC client is actually running as part of the database being accessed, so access can be made directly rather than through network protocols.

Sources

- SQLSummit.com publishes list of drivers, including JDBC drivers and vendors
- Oracle provides a list of some JDBC drivers and vendors [18]
- Simba Technologies ships an SDK for building custom JDBC Drivers for any custom/proprietary relational data source
- RSSBus Type 4 JDBC Drivers for applications, databases, and web services [19].
- DataDirect Technologies provides a comprehensive suite of fast Type 4 JDBC drivers for all major database they advertise as Type 5
- IDS Software provides a Type 3 JDBC driver for concurrent access to all major databases. Supported features include resultset caching, SSL encryption, custom data source, dbShield
- OpenLink Software ships JDBC Drivers for a variety of databases, including Bridges to other data access mechanisms (e.g., ODBC, JDBC) which can provide more functionality than the targeted mechanism
- JDBaccess is a Java persistence library for MySQL and Oracle which defines major database access operations in an easy usable API above JDBC
- JNetDirect provides a suite of fully Sun J2EE certified high performance JDBC drivers.
- HSQLDB is a RDBMS with a JDBC driver and is available under a BSD license.
- SchemaCrawler is an open source API that leverages JDBC, and makes database metadata available as plain old Java objects (POJOs)

References

[1] http://download.oracle.com/javase/7/docs/api/java/sql/package-summary.html
[2] http://download.oracle.com/javase/7/docs/api/javax/sql/package-summary.html
[3] JDBC API Specification Version: 4.0 (http://java.sun.com/products/jdbc/download.html#corespec40).
[4] JSR-000221 JDBC API Specification 4.1 (Maintenance Release) (http://jcp.org/aboutJava/communityprocess/mrel/jsr221/index.html)
[5] http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/jdbc_41.html
[6] http://download.oracle.com/javase/7/docs/api/java/sql/Statement.html
[7] http://download.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html
[8] http://download.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html
[9] http://jolbox.com
[10] http://sourceforge.net/projects/c3p0
[11] http://commons.apache.org/dbcp
[12] http://download.oracle.com/javase/7/docs/api/java/lang/Class.html#forName(java.lang.String)
[13] http://download.oracle.com/javase/7/docs/api/java/lang/Class.html#newInstance()
[14] http://www.onjava.com/pub/a/onjava/2006/08/02/jjdbc-4-enhancements-in-java-se-6.html
[15] http://download.oracle.com/javase/7/docs/api/java/sql/Driver.html
[16] http://download.oracle.com/javase/7/docs/api/java/sql/DriverManager.html
[17] http://download.oracle.com/javase/7/docs/api/java/sql/SQLException.html
[18] http://devapp.sun.com/product/jdbc/drivers
[19] http://www.rssbus.com/jdbc/

External links

- Java SE 7 (http://download.oracle.com/javase/7/docs/) This documentation has examples where the JDBC resources are not closed appropriately (swallowing primary exceptions and being able to cause NullPointerExceptions) and has code prone to SQL injection[*citation needed*]
- `java.sql` (http://download.oracle.com/javase/7/docs/api/java/sql/package-summary.html) API Javadoc documentation
- `javax.sql` (http://download.oracle.com/javase/7/docs/api/javax/sql/package-summary.html) API Javadoc documentation
- O/R Broker (http://www.orbroker.org) Scala JDBC framework
- SqlTool (http://www.hsqldb.org/doc/2.0/util-guide/sqltool-chapt.html) Open source, command-line, generic JDBC client utility. Works with any JDBC-supporting database.
- JDBC URL Strings and related information of All Databases. (http://codeoftheday.blogspot.com/2012/12/java-database-connectivity-jdbc-url.html)

ODBC

In computing, **ODBC** (**Open Database Connectivity**) is a standard programming language middleware API for accessing database management systems (DBMS). The designers of ODBC aimed to make it independent of database systems and operating systems; an application written using ODBC can be ported to other platforms, both on the client and server side, with few changes to the data access code.

ODBC accomplishes DBMS independence by using an **ODBC driver** as a translation layer between the application and the DBMS. The application uses ODBC functions through an **ODBC driver manager** with which it is linked, and the driver passes the query to the DBMS. An ODBC driver can be thought of as analogous to a printer or other driver, providing a standard set of functions for the application to use, and implementing DBMS-specific functionality. An application that can use ODBC is referred to as "ODBC-compliant". Any ODBC-compliant application can access any DBMS for which a driver is installed. Drivers exist for all major DBMSs, many other data sources like address book systems and Microsoft Excel, and even for text or CSV files.

ODBC was originally developed by Microsoft during the early 1990s, and became the basis for the Call Level Interface (CLI) standardized by SQL Access Group in the Unix and mainframe world. ODBC retained a number of features that were removed as part of the CLI effort. Full ODBC was later ported back to those platforms, and became a de facto standard considerably better known than CLI. The CLI remains similar to ODBC, and applications can be ported from one platform to the other with few changes.

History

Prior to ODBC

The introduction of the mainframe-based relational database during the 1970s led to a proliferation of data access methods. Generally these systems operated hand-in-hand with a simple command processor that allowed the user to type in English-like commands, and receive output. The best-known examples are SQL from IBM and QUEL from the Ingres project. These systems may or may not allow other applications to access the data directly, and those that did used a wide variety of methodologies. The introduction of SQL aimed to solve the problem of *language* standardization, although substantial differences in implementation remained.

Additionally, since the SQL language had only rudimentary programming features, it was often desired to use SQL within a program written in another language, say Fortran or C. This led to the concept of Embedded SQL, which allowed SQL code to be "embedded" within another language. For instance, a SQL statement like `SELECT * FROM city` could be inserted as text within C source code, and during compilation it would be converted into a custom format that directly called a function within a library that would pass the statement into the SQL system. Results returned from the statements would be interpreted back into C data formats like `char *` using similar library code.

There were a number of problems with the Embedded SQL approach. Like the different varieties of SQL, the Embedded SQL's that used them varied widely, not only from platform to platform, but even across languages on a single platform - a system that allowed calls into IBM's DB2 would look entirely different from one that called into their own SQL/DSWikipedia:Disputed statement. Another key problem to the Embedded SQL concept was that the SQL code could only be changed in the program's source code, so that even small changes to the query required considerable programmer effort to modify. The SQL market referred to this as "static SQL", as opposed to "dynamic SQL" which could be changed at any time - like the command-line interfaces that shipped with almost all SQL systems, or a programming interface that left the SQL as plain text until it was called. Dynamic SQL systems became a major focus for SQL vendors during the 1980s.

Older mainframe databases, and the newer microcomputer based systems that were based on them, generally did not have a SQL-like command processor between the user and the database engine. Instead, the data was accessed directly by the program - a programming library in the case of large mainframe systems, or a command line interface or interactive forms system in the case of dBASE and similar applications. Data from dBASE could not generally be accessed directly by other programs running on the machine. Those programs may be given a way to access this data, often through libraries, but it would not work with any other database engine, or even different databases in the same engine. In effect, all such systems were static, which presented considerable problems.

Early efforts

By the mid-1980s the rapid improvement in microcomputers, and especially the introduction of the graphical user interface and data-rich application programs like Lotus 1-2-3 led to an increasing interest in using personal computers as the client-side platform of choice in client-server computing. Under this model, large mainframes and minicomputers would be used primarily to serve up data over local area networks to microcomputers that would interpret, display and manipulate that data. For this model to work, a data access standard was a requirement - in the mainframe world it was highly likely that all of the computers in a shop were from a single vendor and clients were computer terminals talking directly to them, but in the micro world there was no such standardization and any client might access any server using any networking system.

By the late 1980s there were a number of efforts underway to provide an abstraction layer for this purpose. Some of these were mainframe related, designed to allow programs running on those machines to translate between the variety of SQL's and provide a single common interface which could then be called by other mainframe or microcomputer programs. These solutions included IBM's Distributed Relational Database Architecture (DRDA) and Apple Computer's Data Access Language. Much more common, however, were systems that ran entirely on microcomputers, including a complete protocol stack that included any required networking or file translation support.

One of the early examples of such a system was Lotus Development's DataLens, initially known as Blueprint. Blueprint, developed for 1-2-3, supported a variety of data sources, including SQL/DS, DB2, FOCUS and a variety of similar mainframe systems, as well as microcomputer systems like dBase and the early Microsoft/Ashton-Tate efforts that would eventually develop into Microsoft SQL Server.[1] Unlike the later ODBC, Blueprint was a purely code-based system, lacking anything approximating a command language like SQL. Instead, programmers used data structures to store the query information, constructing a query by linking many of these structures together. Lotus referred to these compound structures as "query trees".[2]

Around the same time, an industry team including members from Sybase, Tandem Computers and Microsoft were working on a standardized dynamic SQL concept. Much of the system was based on Sybase's DB-Library system, with the Sybase-specific sections removed and several additions to support other platforms.[3] DB-Library was aided by an industry-wide move from library systems that were tightly linked to a particular language, to library systems that were provided by the operating system and required the languages on that platform to conform to its standards. This meant that a single library could be used with (potentially) any programming language on a given platform.

The first draft of the **Microsoft Data Access API** was published in April 1989, about the same time as Lotus' announcement of Blueprint.[4] In spite of Blueprint's great lead - it was running when MSDA was still a paper project - Lotus eventually joined the MSDA efforts as it became clear that SQL would become the de facto database standard.[2] After considerable industry input, in the summer of 1989 the standard became **SQL Connectivity**, or **SQLC** for short,.[5]

SAG and CLI

In 1988 a number of vendors, mostly from the Unix and database communities, formed the SQL Access Group (SAG) in an effort to produce a single basic standard for the SQL language. At the first meeting there was considerable debate over whether or not the effort should work solely on the SQL language itself, or attempt a wider standardization which included a dynamic SQL language-embedding system as well, what they called a Call Level Interface (CLI).[6] While attending the meeting with an early draft of what was then still known as MS Data Access, Kyle Geiger of Microsoft invited Jeff Balboni and Larry Barnes of Digital Equipment Corporation (DEC) to join the SQLC meetings as well. SQLC was a potential solution to the call for the CLI, which was being led by DEC.

The new SQLC "gang of four", MS, Lotus, DEC and Sybase, brought an updated version of SQLC to the next SAG meeting in June 1990.[7] The SAG responded by opening the standard effort to any competing design, but of the many proposals, only Oracle Corp had a system that presented serious competition. In the end, SQLC won the votes and became the draft standard, but only after large portions of the API were removed - the standards document was trimmed from 120 pages to 50 during this time. It was also during this period that the name Call Level Interface was formally adopted.[7] In 1995 SQL/CLI became part of the international SQL standard, ISO/IEC 9075-3.[8] The SAG itself was taken over by the X/Open group in 1996, and, over time, became part of The Open Group's Common Application Environment.

MS continued working with the original SQLC standard, retaining many of the advanced features that were removed from the CLI version. These included features like scrollable cursors, and metadata information queries. The commands in the API were split into groups; the Core group was identical to the CLI, the Level 1 extensions were commands that would be easy to implement in drivers, while Level 2 commands contained the more advanced features like cursors. A proposed standard was released in December 1991, and industry input was gathered and worked into the system through 1992, resulting in yet another name change to **ODBC**.[9]

JET and ODBC

During this time, Microsoft was in the midst of developing their Jet database system. Jet combined three primary subsystems; an ISAM-based database engine (also known as "Jet", confusingly), a C-based interface allowing applications to access that data, and a selection of driver DLLs that allowed the same C interface to redirect input and output to other ISAM-based databases, like Paradox and xBase. Jet allowed programmers to use a single set of calls to access common microcomputer databases in a fashion similar to Blueprint (by this point known as DataLens). However, Jet did not use SQL; like DataLens, the interface was in C and consisted of data structures and function calls.

The SAG standardization efforts presented an opportunity for Microsoft to adapt their Jet system to the new CLI standard. This would not only make Windows a premier platform for CLI development, but also allow users to use SQL to access both Jet and other databases as well. What was missing was the SQL parser that could convert those calls from their text form into the C-interface used in Jet. To solve this, MS partnered with PageAhead Software to use their existing query processor, "SIMBA". SIMBA was used as a parser above Jet's C library, turning Jet into an SQL database. And because Jet could forward those C-based calls to other databases, this also allowed SIMBA to query other systems. Microsoft included drivers for Excel to turn its spreadsheet documents into SQL-accessible database tables.

Release and continued development

ODBC 1.0 was released in September 1992. At the time, there was little direct support for SQL databases (as opposed to ISAM), and early drivers were noted for poor performance. Some of this was unavoidable due to the path that the calls took through the Jet-based stack; ODBC calls to SQL databases were first converted from SIMBA's SQL dialect to Jet's internal C-based format, then passed to a driver for conversion back into SQL calls for the database. Digital Equipment and Oracle both contracted Simba to develop drivers for their databases as well.[10]

Circa 1993, OpenLink Software shipped one of the first independently developed third-party ODBC drivers, for the PROGRESS DBMS, and soon followed with their UDBC (a cross-platform API equivalent of ODBC and the SAG/CLI) SDK and associated drivers for PROGRESS, Sybase, Oracle, and other DBMS, for use on Unix-like OS (AIX, HP-UX, Solaris, Linux, etc.), VMS, Windows NT, OS/2, and other OS.

Meanwhile the CLI standard effort dragged on, and it was not until March 1995 that the definitive version was finalized. By this time Microsoft had already granted Visigenic Software a source code license to develop ODBC on non-Windows platforms. Visigenic ported ODBC to a wide variety of Unix platforms, where ODBC quickly became the de facto standard.[11] "Real" CLI is rare today. The two systems remain similar, and many applications can be ported from ODBC to CLI with few or no changes.[12]

Over time, database vendors took over the driver interfaces and provided direct links to their products. Skipping the intermediate conversions to and from Jet or similar wrappers often resulted in higher performance. However, by this time Microsoft had changed focus to their OLE DB concept, which provided direct access to a wider variety of data sources from address books to text files. Several new systems followed which further turned their attention from ODBC, including DAO, ADO and ADO.net, which interacted more or less with ODBC over their lifetimes.

As Microsoft turned its attention away from working directly on ODBC, the Unix world was increasingly embracing it. This was propelled by two changes within the market, the introduction of GUIs like GNOME that provided the need for access to these sources in non-text form, and the emergence of open software database systems like PostgreSQL and MySQL, initially under Unix. The later adoption of ODBC by Apple for using the standard Unix-side iODBC package Mac OS X 10.2 (Jaguar) (which OpenLink Software had been independently providing for Mac OS X 10.0 and even Mac OS 9 since 2001) further cemented ODBC as the standard for cross-platform data access.

Sun Microsystems used the ODBC system as the basis for their own open standard, JDBC. In most ways, JDBC can be considered a version of ODBC for the Java programming language as opposed to C. JDBC-to-ODBC "bridges" allow Java-based programs to access data sources through ODBC drivers on platforms lacking a native JDBC driver, although these are now relatively rare. Inversely, ODBC-to-JDBC "bridges" allow C-based programs to access data sources through JDBC drivers on platforms or from databases lacking suitable ODBC drivers.

ODBC today

ODBC remains largely universal today, with drivers available for most platforms and most databases. It is not uncommon to find ODBC drivers for database engines that are meant to be embedded, like SQLite, as a way to allow existing tools to act as front-ends to these engines for testing and debugging.[13]

However, the rise of thin client computing using HTML as an intermediate format has reduced the need for ODBC. Many web development platforms contain direct links to target databases - MySQL being particularly common. In these scenarios, there is no direct client-side access nor multiple client software systems to support; everything goes through the programmer-supplied HTML application. The virtualization that ODBC offers is no longer a strong requirement, and development of ODBC is no longer as active as it once was.

Version history

Version history:

- 1.0: released in September 1992
- 2.0: ca 1994
- 2.5
- 3.0: ca 1995, John Goodson of Intersolv and Frank Pellow and Paul Cotton of IBM provided significant input to ODBC 3.0[14]
- 3.5: ca 1997
- 3.8: ca 2009, with Windows 7

Drivers and Managers

Drivers

ODBC is based on the device driver model, where the driver encapsulates the logic needed to convert a standard set of commands and functions into the specific calls required by the underlying system. For instance, a printer driver presents a standard set of printing commands, the API, to applications using the printing system. Calls made to those APIs are converted by the driver into the format used by the actual hardware, say PostScript or PCL.

In the case of ODBC, the drivers encapsulate a number of functions that can be broken down into several broad categories. One set of functions is primarily concerned with finding, connecting to and disconnecting from the DBMS that driver talks to. A second set is used to send SQL commands from the ODBC system to the DBMS, converting or interpreting any commands that are not supported internally. For instance, a DBMS that does not support cursors can emulate this functionality in the driver. Finally, another set of commands, mostly used internally, is used to convert data from the DBMS's internal formats to a set of standardized ODBC formats, which are based on the C language formats.

An ODBC driver enables an ODBC-compliant application to use a *data source*, normally a DBMS. Some non-DBMS drivers exist, for such data sources as CSV files, by implementing a small DBMS inside the driver itself. ODBC drivers exist for most DBMSs, including Oracle, PostgreSQL, MySQL, Microsoft SQL Server (but not for the Compact aka CE edition), Sybase ASE, and DB2. Because different technologies have different capabilities, most ODBC drivers do not implement all functionality defined in the ODBC standard. Some drivers offer extra functionality not defined by the standard.

Driver Manager

Device drivers are normally enumerated, set up and managed by a separate Manager layer, which may provide additional functionality. For instance, printing systems often include functionality to provide spooling functionality on top of the drivers, providing print spooling for any supported printer.

In ODBC the Driver Manager (DM) provides these features. The DM can enumerate the installed drivers and present this as a list, often in a GUI-based form.

But more important to the operation of the ODBC system is the DM's concept of **Data Source Name**s, or **DSN**. DSNs collect additional information needed to connect to a *particular* data source, as opposed to the DBMS itself. For instance, the same MySQL driver can be used to connect to any MySQL server, but the connection information to connect to a local private server is different from the information needed to connect to an internet-hosted public server. The DSN stores this information in a standardized format, and the DM provides this to the driver during connection requests. The DM also includes functionality to present a list of DSNs using human readable names, and to select them at run-time to connect to different resources.

The DM also includes the ability to save partially complete DSN's, with code and logic to ask the user for any missing information at runtime. For instance, a DSN can be created without a required password. When an ODBC application attempts to connect to the DBMS using this DSN, the system will pause and ask the user to provide the password before continuing. This frees the application developer from having to create this sort of code, as well as having to know which questions to ask. All of this is included in the driver and the DSNs.

Bridging configurations

A *bridge* is a special kind of driver: a driver that uses another driver-based technology.

ODBC-to-JDBC (or simply ODBC-JDBC) bridges

An ODBC-JDBC bridge consists of an **ODBC** driver which uses the services of a JDBC driver to connect to a database. This driver translates ODBC function-calls into JDBC method-calls. Programmers usually use such a bridge when they lack an ODBC driver for a particular database but have access to a JDBC driver.

JDBC-to-ODBC (or simply JDBC-ODBC) bridges

A JDBC-ODBC bridge consists of a JDBC driver which employs an ODBC driver to connect to a target database. This driver translates JDBC method calls into ODBC function calls. Programmers usually use such a bridge when a particular database lacks a JDBC driver. Sun Microsystems included one such bridge in the JVM, but viewed it as a stop-gap measure while few JDBC drivers existed. Sun never intended its bridge for production environments, and generally recommended against its use. As of 2008[15] independent data-access vendors deliver JDBC-ODBC bridges which support current standards for both mechanisms, and which far outperform the JVM built-in.[*citation needed*]

OLE DB-to-ODBC bridges

An OLE DB-ODBC bridge consists of an OLE DB Provider which uses the services of an ODBC driver to connect to a target database. This provider translates OLE DB method calls into ODBC function calls. Programmers usually use such a bridge when a particular database lacks an OLE DB provider. Microsoft ships one, MSDASQL.DLL, as part of the MDAC system component bundle, together with other database drivers, to simplify development in COM-aware languages (e.g. Visual Basic). Third parties have also developed such, notably OpenLink Software whose 64-bit OLE DB Provider for ODBC Data Sources filled the gap when Microsoft initially deprecated this bridge for their 64-bit OS.[15] (Microsoft later relented, and 64-bit Windows starting with Windows Server 2008 and Windows Vista SP1 have shipped with a 64-bit version of MSDASQL.)

ADO.NET-to-ODBC bridges

An ADO.NET-ODBC bridge consists of an ADO.NET Provider which uses the services of an OBDC driver to connect to a target database. This provider translates ADO.NET method calls into ODBC function calls. Programmers usually use such a bridge when a particular database lacks an ADO.NET provider. Microsoft ships one as part of the MDAC system component bundle, together with other database drivers, to simplify development in C#. Third parties have also developed such.

References

Citations

[1] Evan McGlinn, Blueprint Lets 1-2-3 Access Outside Data" (http://books.google.ca/books?id=6D4EAAAAMBAJ), *InfoWorld*, 4 April 1988, p. 1, 69
[2] Geiger 1995, p. 65.
[3] Geiger 1995, p. 86-87.
[4] Geiger 1995, p. 56.
[5] Geiger 1995, p. 106.
[6] Geiger 1995, p. 165.
[7] Geiger 1995, p. 186-187.
[8] ISO/IEC 9075-3 -- Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI)
[9] Geiger 1995, p. 203.
[10] "Our History" (http://www.simba.com/simba-history.htm), Simba Technologies
[11] Roger Sippl, "SQL Access Group's Call-Level Interface" (http://www.drdobbs.com/sql-access-groups-call-level-interface/184410032), Dr. Dobbs, 1 February 1996
[12] "Similarities and differences between ODBC and CLI" (http://publib.boulder.ibm.com/infocenter/iisclzos/v9r5/index.jsp?topic=/com.ibm.swg.im.iis.fed.classic.clientsref.doc/topics/iiyfcodbcclisimdiff.html), InfoSphere Classic documentation, IBM, 26 September 2008
[13] Christian Werner, "SQLite ODBC Driver" (http://www.ch-werner.de/sqliteodbc/)
[14] Microsoft Corporation. Microsoft ODBC 3.0 Programmer's Reference and SDK Guide, Volume 1. Microsoft Press. February 1997. (ISBN13: 9781572315167)
[15] *Microsoft*, "Data Access Technologies Road Map", Deprecated MDAC Components, *Microsoft* "ADO Programmer's Guide" Appendix A: Providers, Microsoft OLE DB Provider for ODBC (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdrefodbcprovspec.asp), retrieved July 30, 2005.

Bibliography

- Geiger, Kyle (1995). *Inside ODBC* (http://books.google.ca/books?id=G-ZQAAAAMAAJ&). Microsoft Press.

External links

- Microsoft ODBC Overview (http://support.microsoft.com/kb/110093)
- List of ODBC Drivers at databasedrivers.com (http://www.databasedrivers.com/odbc/)
- List of ODBC Drivers at SQLSummit.com (http://www.SQLSummit.com/ODBCVend.htm)
- OS400 and i5OS ODBC Administration (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/topic/rzaii/rzaiiodbcadm.htm)
- Presentation slides from www.roth.net (http://www.roth.net/perl/odbc/conf/sld002.htm)
- Early ODBC White Paper (http://www.openlinksw.com/info/docs/odbcwhp/tableof.htm)
- Microsoft ODBC & Data Access APIs History Article (http://blogs.msdn.com/data/archive/2006/12/05/data-access-api-of-the-day-part-i.aspx)

OLE DB provider

An **OLE DB provider** is a software component enabling an OLE DB consumer to interact with a data source. OLE DB providers are analogous to ODBC drivers, JDBC drivers, and ADO.NET data providers.

OLE DB providers can be created to access such simple data stores as a text file and spreadsheet, through to such complex databases as Oracle, Microsoft SQL Server, Sybase ASE, and many others. It can also provide access to hierarchical data stores such as email systems.

However, because different data store technologies can have different capabilities, every OLE DB provider cannot implement every possible interface available in the OLE DB standard. The capabilities that are available are implemented through the use of COM objects; an OLE DB provider will map the data store technologies functionality to a particular COM interface. Microsoft describes the availability of an interface as "provider-specific," as it may not be applicable depending on the data store technology involved. Note also that providers may augment the capabilities of a data store; these capabilities are known as *services* in Microsoft parlance.

OLE DB providers

- SQLSummit.com: Catalogue of OLE DB Providers [1]
- Microsoft [2] ships a few OLE DB Providers as part of its MDAC and JET kits
- Simba Technologies ships SimbaProvider, an SDK used to build custom OLE DB for OLAP providers for multi-dimensional and star schema database connectivity.
- OpenLink Software ships components supporting OLE DB access [3] to a number of data sources, including several SQL DBMS, as well as Bridges to ODBC- and JDBC-accessible data sources
- OLE DB Provider for Interbase and Firebird (supports 14 database types, free and pro versions are available) [4]
- OLE DB Provider for PostgreSQL [5]

External links

- "OLE DB Providers Overview" [6]. Microsoft. MSDN: Data Developer Center. Retrieved 23 March 2011.

References

[1] http://www.sqlsummit.com/oledbVen.htm
[2] http://msdn.microsoft.com/data/
[3] http://uda.openlinksw.com/oledb/
[4] http://www.ibprovider.com/
[5] http://www.pgoledb.com/
[6] http://msdn.microsoft.com/en-us/library/ms709836%28v=vs.85%29.aspx

OLE DB

OLE DB (*Object Linking and Embedding, Database*, sometimes written as **OLEDB** or **OLE-DB**), an API designed by Microsoft, allows accessing data from a variety of sources in a uniform manner. The API provides a set of interfaces implemented using the Component Object Model (COM); it is otherwise unrelated to OLE. Microsoft originally intended OLE DB as a higher-level replacement for, and successor to, ODBC, extending its feature set to support a wider variety of non-relational databases, such as object databases and spreadsheets that do not necessarily implement SQL.

Methodology

OLE DB separates the data store from the application that needs access to it through a set of abstractions that include the datasource, session, command, and rowsets. This was done because different applications need access to different types and sources of data, and do not necessarily want to know how to access functionality with technology-specific methods. OLE DB is conceptually divided into *consumers* and *providers*. The consumers are the applications that need access to the data, and the providers are the software components that implement the interface and thereby provides the data to the consumer. OLE DB is part of the Microsoft Data Access Components (MDAC) stack.

Support status

Microsoft's release of SQL Server 2012 (internal code: 'Denali') is the last to include an OLE DB provider for SQL Server, but support will continue for 7 years.[1] According to a related Microsoft FAQ,[2] "Providers like ADO.Net which can run on top of OLE DB will not support OLE DB once the latter is deprecated", but the same answer in the FAQ states that the original post relates only to the OLE DB provider for SQL Server, so the position of OLE DB itself remains unclear. The same FAQ states that ODBC performs better than OLE DB in most cases.

References

[1] Microsoft SQLNCli team blog: Microsoft is Aligning with ODBC for Native Relational Data Access (http://blogs.msdn.com/b/sqlnativeclient/archive/2011/08/29/microsoft-is-aligning-with-odbc-for-native-relational-data-access.aspx)

[2] SQL Server Forums - SQL Server Data Access: Microsoft is Aligning with ODBC for Native Relational Data Access (http://social.technet.microsoft.com/Forums/en/sqldataaccess/thread/e696d0ac-f8e2-4b19-8a08-7a357d3d780f)

UnixODBC

unixODBC

Stable release	2.3.2 / October 8, 2013
Operating system	Cross-platform
Type	Data Access API
License	GNU GPL/LGPL
Website	www.unixODBC.org [1]

unixODBC is an open source project that implements the ODBC API. The code is provided under the GNU GPL/LGPL license and can be built and used on many different operating systems, including most versions of Unix, Linux, Mac OS X, IBM OS/2 and Microsoft's Interix.

The goals of the project include:

- Provide developers with the tools to port Microsoft Windows ODBC applications to other platforms with the minimum of code changes.
- Maintain the project as a vendor neutral interface database SDK
- Provide people who write ODBC drivers the tools to port their drivers to non Windows platforms
- Provide the user with a set of GUI and command line tools for managing their database access
- Maintain links with both the free software community and commercial database vendors, to ensure interoperability

History

1999

The unixODBC project was first started in the early months of 1999 (by Peter Harvey) and was created as at that time the developers of iODBC (another open source ODBC implementation) were not then willing to LGPL the code, expand the API to include the current ODBC 3 API specification, and did not consider the addition of GUI based configuration tools worthwhile. iODBC now has these parts added, and applications that use the ODBC interface may use both iODBC and unixODBC, without change in most cases, as a result of both projects adhering to the single ODBC specification.

1999 July

The original driver manager was very basic. The driver manager was rewritten by Easysoft's [2] Nick Gorham soon after the project started. Nick assumed leadership of the project in July 1999 with Peter Harvey continuing work on supporting code.

The development of unixODBC progressed since its origin, with contributions from many developers, both in the open source community and also from commercial database companies, including IBM, Oracle Corporation and SAP.

It is included as part of the standard installation of many Linux distributions.

2009

The unixODBC project was split into several projects (all hosted on SourceForge);

- unixODBC [3] ("Core" and "Dev" bits)
- unixODBC-GUI-Qt [4] (Qt based GUI bits)
- unixODBC-Test [5] (multiple test frameworks)

This split was done to allow faster releases of supporting work while maintaining focus on stability and consistency for the core code.

External links

- unixODBC homepage [6]
- UnixODBC & MySQL Sample Program [7]

References

[1] http://www.unixODBC.org
[2] http://www.easysoft.com
[3] http://www.unixodbc.org
[4] http://sourceforge.net/projects/unixodbc-gui-qt
[5] http://sourceforge.net/projects/unixodbc-test
[6] http://www.unixODBC.org/
[7] http://compscinotes.wordpress.com/2010/04/18/unixodbc-mysql-sample-program/

IODBC

iODBC

Stable release	3.52.7 / September 10, 2009
Operating system	Cross-platform
Type	Data Access API
License	BSD, LGPL
Website	www.iodbc.org [1]

iODBC is an open source initiative managed by OpenLink Software. It is a platform-independent ODBC SDK and runtime offering that enables the development of ODBC-compliant applications and drivers outside the Windows platform. The prime goals of this project are as follows:

- Simplify the effort of porting ODBC applications from Windows to other platforms
- Simplify the effort of porting ODBC drivers from Windows to other platforms
- Create consistent ODBC-utilization experience across all platforms

History

iODBC emerged from a cooperative effort between OpenLink Software and Ke Jin. OpenLink Software produced a Driver Manager-less ODBC SDK that it branded as Universal DataBase Connectivity (UDBC) in 1993, because of the sporadic nature of shared library implementations across Unix platforms. Ke Jin used UDBC as inspiration for building a Driver Manager for ODBC outside the windows platform.

Over time Ke Jin and OpenLink Software decided to merge this effort into a single Open Source offering under the LGPL license.

This process occurred at a time when the Free Software Foundation sought to have iODBC as a GPL offering. The delay in determining final licensing status for iODBC led to the emergence of UnixODBC and led to a fork in the platform-independent ODBC SDK and runtime that exists today. Drivers and applications written using either SDK have remained compatible (a tribute to both projects).

External links

- iODBC homepage [1]

References

[1] http://www.iodbc.org

Pool (computer science)

A **pool** in computer science is a set of initialised resources that are kept ready to use, rather than allocated and destroyed on demand. A client of the pool will request an object from the pool and perform operations on the returned object. When the client has finished with an object (or resource), it returns it to the pool, rather than destroying it.

Pooling of resources can offer a significant performance boost in situations where the cost of initializing a class instance is high, the rate of instantiation of a class is high, and the number of instances in use at any one time is low. The pooled resource is obtained in predictable time when creation of the new objects (especially over network) may take variable time.

However these benefits are mostly true for objects which are expensive with respect to time, such as database connections, socket connections, threads and large graphic objects like fonts or bitmaps. In certain situations, simple object pooling (which hold no external resources, but only occupy memory) may not be efficient and could decrease performance.

Special cases are:

- Connection pool
- Thread pool
- Memory pool

Pool can also refer to a design pattern for implementing them in object-oriented languages, such as the object pool pattern.

References

Connection pool

In software engineering, a **connection pool** is a cache of database connections maintained so that the connections can be reused when future requests to the database are required. Connection pools are used to enhance the performance of executing commands on a database. Opening and maintaining a database connection for each user, especially requests made to a dynamic database-driven website application, is costly and wastes resources. In connection pooling, after a connection is created, it is placed in the pool and it is used over again so that a new connection does not have to be established. If all the connections are being used, a new connection is made and is added to the pool. Connection pooling also cuts down on the amount of time a user must wait to establish a connection to the database.

Applications

Connection pooling is used in web-based and enterprise applications and is handled by the application server. Dynamic web pages without connection pooling open connections to database services when they are needed and close them when the page is done servicing a particular request. Pages that use connection pooling instead maintain open connections in a pool. When the page requires access to the database, it simply uses an existing connection from the pool, and establishes a new connection only if no pooled connections are available. This reduces the overhead associated with connecting to the database to service individual requests.

Local applications that need frequent access to databases can also benefit from connection pooling. Open connections can be maintained in local applications that don't need to service separate remote requests like application servers, but implementations of connection pooling can be complicated. There are a number of libraries available that implement connection pooling and related SQL query pooling, simplifying implementation of connection pools in database-intensive applications.

Connection pools can be configured with restrictions on the numbers of minimum connections, maximum connections and idle connections to optimize the performance of pooling in specific problem contexts and environments.

Database support

Connection pooling is supported by IBM DB2,[1] Microsoft SQL Server,[2] Oracle,[3] MySQL,[4] and PostgreSQL.[5]

References

[1] IBM Connection Pooling Support (http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z10.doc.java/src/tpc/imjcc_cjvjpool.htm)

[2] SQL Server Connection Pooling (ADO.NET) (http://msdn.microsoft.com/en-us/library/8xx3tyca.aspx)

[3] OCI Driver Connection Pooling (http://download.oracle.com/docs/cd/B10501_01/java.920/a96654/oci_func.htm#1014118&displaylang=en)

[4] MySQL Connection Pooling (http://dev.mysql.com/tech-resources/articles/connection_pooling_with_connectorj.html)

[5] PostgreSQL Replication, Clustering and Connection Pooling (http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling)

Remote Database Access

Remote database access (RDA) is a protocol standard for database access.

Purpose

RDA describes the connection of a database client to a database server. It includes features for

- communicating database operations and parameters from the client to the server,
- in return, transporting result data from the server to the client,
- database transaction management.
- exchange of information.

RDA is an application-level protocol, inasmuch that it builds on an existing network connection between client and server. In the case of TCP/IP connections, RFC 1066 is used for implementing RDA.

History

RDA was published in 1993, as a combined standard of ANSI, ISO and IEC. The standards definition comprises two parts:

- ANSI/ISO/IEC 9579-1:1993
- ANSI/ISO/IEC 9579-2:1993

Sources

- "Remote Database Access" [1]. *NIST SQL Project*. National Institute of Standards and Technology. Retrieved 2008-04-12.

References

[1] http://www.itl.nist.gov/div897/ctg/dm/rda_info.html

SQLJ

SQLJ is an outdated working title for efforts to combine Java and SQL. It was a common effort started around 1997 by engineers from IBM, Oracle, Compaq, Informix, Sybase, Cloudscape and Sun Microsystems.

It consists of the three parts: 0, 1 and 2. Part 0 describes the embedding of SQL statements into Java programs. SQLJ part 0 is the basis for part 10 of the SQL:1999 standard, aka SQL Object Language Bindings (SQL/OLB). SQLJ parts 1 and 2 describes the converse possibility to use Java classes (routines and types) from SQL statements. Parts 1 and 2 are the basis for part 13 of the SQL standard, SQL Routines and Types Using the Java Programming Language (SQL/JRT).

"SQLJ" is commonly used to refer to just SQLJ part 0, usually when it is contrasted with other means of embedding SQL in Java, like JDBC.

ANSI and ISO standards

- SQLJ part 0: ANSI X3.135.10-1998, "Database Language SQL—Part 10: Object Language Bindings (SQL/OLB)"
- SQLJ part 1: ANSI NCITS 331.1-1999, "SQLJ—Part 1: SQL Routines Using the Java Programming Language"
- SQLJ part 2: ANSI NCITS 331.2-2000, "SQLJ—Part 2: SQL Types Using the Java Programming Language"

Part 0 was updated for JDBC 2.0 compatibility and ratified by ISO in 2000. The last two parts were combined when submitted to ISO. Part 2 was substantially rewritten for the ISO submission because the ANSI version was not formal enough for a specification, being closer to the style of a user manual. The combined version was ratified in 2002.

- ISO/IEC 9075-10:2000, *Information technology—Database languages—SQL—Part 10: Object Language Bindings (SQL/OLB)*
- ISO/IEC 9075-13:2002, *Information technology—Database languages—SQL—Part 13: SQL Routines and Types Using the Java Programming Language (SQL/JRT)*.

SQLJ part 0

The SQLJ part 0 specification largely originated from Oracle, who also provided the first reference implementation.

In the following SQLJ is a synonym for SQLJ part 0.

Whereas JDBC provides an API, SQLJ consists of a language extension. Thus programs containing SQLJ must be run through a preprocessor (the SQLJ translator) before they can be compiled.

Advantages and disadvantages

Some advantages of SQLJ over JDBC include:

- SQLJ commands tend to be shorter than equivalent JDBC programs.
- SQL syntax can be checked at compile time. The returned query results can also be checked strictly.
- Preprocessor might generate static SQL which performs better than dynamic SQL because query plan is created on program compile time, stored in database and reused at runtime. Static SQL can guarantee access plan stability. IBM DB2 supports static SQL use in SQLJ programs.

Disadvantages include:

- SQLJ requires a preprocessing step.
- Many IDEs do not have SQLJ support.
- SQLJ lacks support for most of the common persistence frameworks, such as Hibernate.

Examples

The following examples compare SQLJ syntax with JDBC usage.

Multi-row query

JDBC

```
 size="7.22">
iredStatement stmt = conn.prepareStatement(
ELECT LASTNAME"
, FIRSTNME"
, SALARY"
FROM DSN8710.EMP"
WHERE SALARY BETWEEN ? AND ?");
setBigDecimal(1, min);
setBigDecimal(2, max);
tSet rs = stmt.executeQuery();
 (rs.next()) {
tname = rs.getString(1);
rstname = rs.getString(2);
ary = rs.getBigDecimal(3);
Print row...

ose();
close();
t>
```

SQLJ

```
<font size="7.22">
#sql private static iterator EmployeeIterator(String, Str
BigDecimal);
...
EmployeeIterator iter;
#sql [ctx] iter = {
  SELECT LASTNAME
       , FIRSTNME
       , SALARY
    FROM DSN8710.EMP
   WHERE SALARY BETWEEN :min AND :max
};
do {
  #sql {
    FETCH :iter
     INTO :lastname, :firstname, :salary
  };
  // Print row...
} while (!iter.endFetch());
iter.close();
</font>
```

Single-row query

JDBC

```
PreparedStatement stmt = conn.prepareStatement(
    "SELECT MAX(SALARY), AVG(SALARY)"
  + " FROM DSN8710.EMP");
rs = stmt.executeQuery();
if (!rs.next()) {
  // Error—no rows found
}
maxSalary = rs.getBigDecimal(1);
avgSalary = rs.getBigDecimal(2);
if (rs.next()) {
  // Error—more than one row found
}
rs.close();
stmt.close();
```

SQLJ

```
#sql [ctx] {
  SELECT MAX(SALARY), AVG(SALARY)
    INTO :maxSalary, :avgSalary
    FROM DSN8710.EMP
};
```

INSERT

JDBC

```
16">
epareStatement(
O DSN8710.EMP " +
RSTNME, MIDINIT, LASTNAME, HIREDATE, SALARY) "
 ?, ?, ?, CURRENT DATE, ?)");
(1, empno);
(2, firstname);
(3, midinit);
(4, lastname);
imal(5, salary);
date();
```

SQLJ

```
<font size="6.16">
#sql [ctx] {
  INSERT INTO DSN8710.EMP
    (EMPNO,  FIRSTNME,  MIDINIT,  LASTNAME,  HIREDATI
  VALUES
    (:empno, :firstname, :midinit, :lastname, CURRENT
};
</font>
```

References

Further reading

- Connie Tsui, Considering SQLJ for Your DB2 V8 Java Applications (http://www.ibm.com/developerworks/data/library/techarticle/0302tsui/0302tsui.html), IBM developerworks, 13 Feb 2003
- Owen Cline, Develop your applications using SQLJ (http://www.ibm.com/developerworks/data/library/techarticle/dm-0412cline/), IBM developerworks, 16 Dec 2004
- Jason Price (2001). *Java Programming With Oracle SQLJ*. O'Reilly Media. ISBN 978-0-596-00087-5.

External links

- http://sqlj.org/
- IBM Redbook: DB2 for z/OS and OS/390: Ready for Java (http://www.redbooks.ibm.com/abstracts/sg246435.html)

Native Queries

Native Queries are a concise and type-safe way to express queries directly as Java and C# methods.

Native Queries are based on Safe Queries by Cook and Rai and were first[*citation needed*] implemented in db4o's open source object database as well as POJQ on Java.net.

Products

- db4o [1] - database for objects
- JODB [2] - Java Objects Database
- pojq [3] - Plain Old Java Queries
- NeoDatis [4] - NeoDatis ODB

External links

- Native Queries for Persistent Objects [5] - Whitepaper by Cook, Rosenberger

References

[1] http://www.db4o.com/
[2] http://www.java-objects-database.com/
[3] https://pojq.dev.java.net/
[4] http://www.neodatis.org/
[5] http://www.ddj.com/windows/184406432

Meta-SQL

'**Meta-SQL**' Use (with reference to PeopleSoft)

Meta-SQL expands to platform-specific SQL substrings, causes another function to be called, or substitutes a value. Meta-SQL constructs are used in functions that pass SQL strings, such as the following:

- SQLExec.
- Scroll buffer functions (ScrollSelect and its relatives)
- PeopleSoft Application Designer dynamic and SQL views
- Some Rowset class methods (Select, SelectNew, Fill, and so on.)
- The SQL class
- PeopleSoft Application Engine programs
- Some Record class methods (Insert, Update, and so on.)
    - COBOL functions

Meta-SQL Element

Types There are three types of meta-SQL elements:

- Constructs are a direct substitution of a value, and help to build or modify a SQL statement. Examples include %Bind, %InsertSelect, and %List.
- Functions perform actions or cause another function to be called. Examples include %ClearCursor, %Execute, and %ExecuteEdits.
- Meta-variables enable substitution of text within SQL statements. Examples include %AsOfDate, %Comma, and %JobInstance.

Meta-SQL Placement Considerations: Not all meta-SQL can be used by all programs. Some meta-SQL can be used only in Application Engine programs. Other meta-SQL can only be used as part of a SQL statement in a SQL or dynamic view. The following table lists available meta-SQL elements and where each element can be used.

If a meta-SQL construct, function, or meta-variable is supported in PeopleCode, it is supported in all types of PeopleCode programs; that is, in Application Engine PeopleCode programs (actions), component interface PeopleCode programs, and so on.

Note: Even if a meta-SQL element is used in PeopleCode, you cannot use meta-SQL like a built-in function. You can use meta-SQL in the SQLExec function, the Select method, the Fill method, and so on. Note: Meta-SQL is not available in SQR

ADO.NET

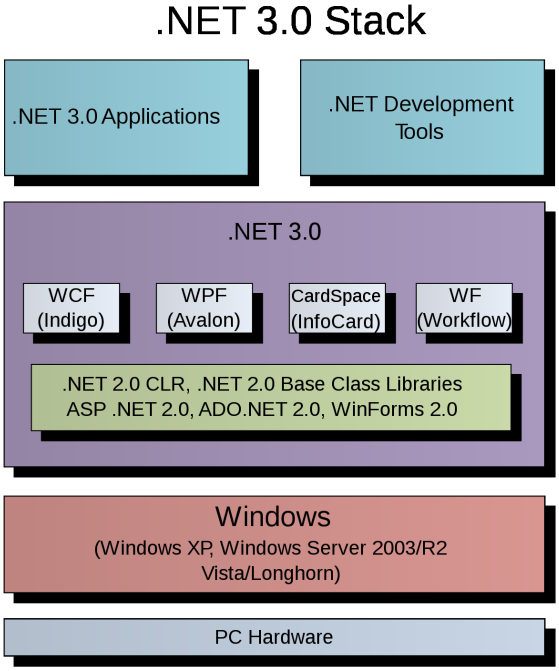
ADO.NET

Operating system	Microsoft Windows
Type	Software framework
License	MS-EULA, BCL under Microsoft Reference License
Website	ADO.NET Overview on MSDN [1]

ADO.NET is a set of computer software components that programmers can use to access data and data services based on disconnected DataSets and XML. It is a part of the base class library that is included with the Microsoft .NET Framework. It is commonly used by programmers to access and modify data stored in relational database systems, though it can also access data in non-relational sources. ADO.NET is sometimes considered an evolution of ActiveX Data Objects (ADO) technology, but was changed so extensively that it can be considered an entirely new product.

Architecture

ADO.NET is conceptually divided into *consumers* and *data providers*. The consumers are the applications that need access to the data, and the providers are the software components that implement the interface and thereby provide the data to the consumer.



This technology forms a Batata part of .NET Framework 3.0 (having been part of the framework since version 1.0)

ADO.NET and Visual Studio

Functionality exists in Visual Studio IDE to create specialized subclasses of the DataSet classes for a particular database schema, allowing convenient access to each field through strongly typed properties. This helps catch more programming errors at compile-time and enhances the IDE's Intellisense feature.

ADO.NET and O/R Mapping

Entity Framework

The **ADO.NET Entity Framework** is a set of data-access APIs for the Microsoft .NET Framework, similar to the Java Persistence API, targeting the version of ADO.NET that ships with .NET Framework 4.0. ADO.NET Entity Framework is included with .NET Framework 4.0 and Visual Studio 2010, released in April 2010. An Entity Framework *Entity* is an object which has a key representing the primary key of a logical datastore entity. A conceptual *Entity Data Model* (Entity-relationship model) is mapped to a datastore schema model. Using the Entity Data Model, the Entity Framework allows data to be treated as entities independently of their underlying datastore representations.

Entity SQL, a SQL-like language, serves for querying the Entity Data Model (instead of the underlying datastore). Similarly, LINQ extension *LINQ to Entities* provides typed querying on the Entity Data Model. Entity SQL and LINQ to Entities queries are converted internally into a *Canonical Query Tree* which is then converted into a query understandable to the underlying database.

External links

ADO.NET

- ADO.NET Overview on MSDN [1]
- ADO.NET for the ADO Programmer [2]
- ADO.NET Connection Strings [3]
- ADO.NET Team Blog [4]
- List of ADO.NET Providers at databasedrivers.com [5]

Incubation Projects

- Data Access Incubation Projects [6]
- Jasper [7], download [8]

References

[1] http://msdn2.microsoft.com/en-us/library/aa286484.aspx
[2] http://msdn2.microsoft.com/en-us/library/ms973217.aspx
[3] http://www.devlist.com/ConnectionStringsPage.aspx
[4] http://blogs.msdn.com/adonet/
[5] http://www.databasedrivers.com/ado/
[6] http://msdn2.microsoft.com/en-us/data/bb419139.aspx
[7] http://blogs.msdn.com/adonet/archive/2007/04/30/project-codename-jasper-announced-at-mix-07.aspx
[8] http://www.microsoft.com/downloads/details.aspx?FamilyId=471BB3AC-B31A-49CD-A567-F2E286715C8F&displaylang=en

Software and Tools

List of relational database management systems

This is a **list of relational database management systems**.

List of Software

- 4th Dimension
- Adabas D
- Alpha Five
- Apache Cassandra
- Apache Derby
- Aster Data
- Altibase
- BlackRay
- CA-Datacom
- Clarion
- Clustrix
- CSQL
- CUBRID
- Daffodil database
- DataEase
- Database Management Library
- Dataphor
- dBase
- Derby aka Java DB
- Empress Embedded Database
- EXASolution
- EnterpriseDB
- eXtremeDB
- FileMaker Pro
- Firebird
- Greenplum
- GroveSite
- H2
- Helix database
- HSQLDB
- IBM DB2
- IBM Lotus Approach
- IBM DB2 Express-C
- Infobright
- Informix
- Ingres
- InterBase

- InterSystems Caché
- GT.M
- Linter
- MariaDB
- MaxDB
- MemSQL
- Microsoft Access
- Microsoft Jet Database Engine (part of Microsoft Access)
- Microsoft SQL Server
- Microsoft SQL Server Express
- Microsoft Visual FoxPro
- Mimer SQL
- MonetDB
- MongoDB
- mSQL
- MySQL
- Netezza
- NexusDB
- NonStop SQL
- NoSQL
- Openbase
- OpenLink Virtuoso (Open Source Edition)
- OpenLink Virtuoso Universal Server
- OpenOffice.org Base
- Oracle
- Oracle Rdb for OpenVMS
- Panorama
- Pervasive PSQL
- Polyhedra
- PostgreSQL
- Postgres Plus Advanced Server
- Progress Software
- RDM Embedded
- RDM Server
- The SAS system
- SAND CDBMS
- SAP HANA
- SAP Sybase Adaptive Server Enterprise
- SAP Sybase IQ
- SQL Anywhere (formerly known as Sybase Adaptive Server Anywhere and Watcom SQL)
- ScimoreDB
- SmallSQL
- solidDB
- SQLBase
- SQLite
- Sybase Advantage Database Server
- Teradata

- TimesTen
- txtSQL
- mizanSQL
- Unisys RDMS 2200
- UniData
- UniVerse
- Vectorwise
- Vertica
- VMDS

Historical

- Britton Lee IDMs
- Cornerstone
- IBM System R
- MICRO Information Management System
- Oracle Rdb
- Paradox
- Pick
- PRTV
- QBE
- IBM SQL/DS
- Sybase SQL Server

Relational by the Date-Darwen-Pascal Model

Current

- Alphora Dataphor (a proprietary virtual, federated DBMS and RAD MS .Net IDE).
- Rel (free Java implementation).

Obsolete

- IBM Business System 12
- IBM IS1
- IBM PRTV (ISBL)
- Multics Relational Data Store

Comparison of relational database management systems

The following tables compare general and technical information for a number of relational database management systems. Please see the individual products' articles for further information. Unless otherwise specified in footnotes, comparisons are based on the stable versions without any add-ons, extensions or external programs.

General information

	Maintainer	First public release date	Latest stable version	Latest release date	Software license
4D (4th Dimension)	4D S.A.S.	1984	v13.2	2012-11-12	Proprietary
ADABAS	Software AG	1970	8.1	2013-06	Proprietary
Adaptive Server Enterprise	Sybase	1987	15.7		Proprietary
Advantage Database Server (ADS)	Sybase	1992	11.1	2012	Proprietary
Altibase	Altibase Corp.	2000	6.1.1	2012-04-01	Proprietary
Apache Derby	Apache	2004	10.10.1.1	2013-04-15	Apache License
Clustrix	Clustrix	2010	v5.0	2013-05-01	Proprietary
CUBRID	NHN Corporation	2008-11	8.4.1	2012-02-24	GPL v2 or later
Datacom	CA, Inc.	*?*	11.2		Proprietary
DB2	IBM	1983	10.5	2013-04-23	Proprietary
Drizzle	Brian Aker	2008	7.1.36	2012-05-23	GPL v2 and v3, with some BSD components
Empress Embedded Database	Empress Software Inc	1979	10.20	2010-03	Proprietary
EXASolution	EXASOL AG	2004	4.2.2	2013-10-17	Proprietary
Firebird	Firebird project	2000-07-25	2.5.2	2013-03-24	IPL and IDPL
HSQLDB	HSQL Development Group	2001	2.3.1	2013-10-08	BSD
H2	H2 Software	2005	1.3.171	2013-03-17	EPL and modified MPL
Informix Dynamic Server	IBM	1980	12.10.xC2	2013-10-01	Proprietary
Ingres	Ingres Corp.	1974	Ingres Database 10	2010-10-12	GPL and Proprietary
InterBase	Embarcadero	1984	InterBase XE	2010-09-21	Proprietary
Linter SQL RDBMS	RELEX Group	1990	6.x	2013-08-26	Proprietary
LucidDB	The Eigenbase Project	2007-01	0.9.3		GPL v2
MariaDB	MariaDB Community	2010-02-01	5.5.35[1]	2014-01-29	GPL v2 and LGPL for client-libraries
MaxDB	SAP AG	2003-05	7.6	2008-01	Proprietary
Microsoft Access (JET)	Microsoft	1992	15 (2013)	2012-10-02	Proprietary
Microsoft Visual Foxpro	Microsoft	1984	9 (2005)	2007-10-11	Proprietary
Microsoft SQL Server	Microsoft	1989	2012 (v11)		Proprietary

Microsoft SQL Server Compact (Embedded Database)	Microsoft	2000	2011 (v4.0)		Proprietary
MonetDB/SQL	The MonetDB Developer Team	2004	11.9.1	2012-04	MonetDB License v1.1 (based on the MPL 1.1)
mSQL	Hughes Technologies	1994	3.9	2011-02	Proprietary
MySQL	Sun Microsystems (now Oracle Corporation)	1995-11	5.6.31	2013-07-30	GPL v2 or Proprietary
MemSQL	MemSQL	2012-06	1.8 (2012)	2012-12	Proprietary
Nexusdb	Nexus Database Systems Pty Ltd	2003-09	3.04	2010-05-08	Proprietary
HP NonStop SQL	Hewlett-Packard	1987	SQL/MX 2.3		Proprietary
Omnis Studio	TigerLogic Inc	1982-07	4.3.1 Release 1no	2008-05	Proprietary
OpenBase SQL	OpenBase International	1991	11.0.0		Proprietary
OpenEdge	Progress Software Corporation	1984	11.0		Proprietary
OpenLink Virtuoso	OpenLink Software	1998	7.x	2013-08-05	GPL v2 or Proprietary
Oracle	Oracle Corporation	1979-11	12c Release 1	2013-06-25	Proprietary
Oracle Rdb	Oracle Corporation	1984	7.2.5.3.0	2013-07-16	Proprietary
Paradox	Corel Corporation	1985	11	2003	Proprietary
Pervasive PSQL	Pervasive Software	1982	v11 SP3	2013	Proprietary
Polyhedra DBMS	ENEA AB	1993	8.7	2013-03	Proprietary
PostgreSQL	PostgreSQL Global Development Group	1989-06	9.3.3	2014-02-20	PostgreSQL Licence (a liberal Open Source license)
R:Base	R:BASE Technologies	1982	9.5		Proprietary
RDM	Raima Inc.	1984	11.0	2012-06-29	Proprietary
RDM Server	Raima Inc.	1993	8.4	2012-10-31	Proprietary
SAP HANA	SAP AG	2010	1.0		Proprietary
ScimoreDB	Scimore	2005	3.0	2008-03-03	Proprietary
SmallSQL	SmallSQL	2005-04-16	0.20	2008-12	LGPL
SQL Anywhere	Sybase	1992	12.0	2010-07-09	Proprietary
SQLBase	Unify Corp.	1982	11.5	2008-11	Proprietary
SQLite	D. Richard Hipp	2000-08-17	3.8.0.2	2013-09-03	Public domain
Superbase	Superbase	1984	Scientific (2004)		Proprietary
Teradata	Teradata	1984	14.10		Proprietary
UniData	Rocket Software	1988	7.2.12	2011-10	Proprietary
Xeround Cloud Database	Xeround Systems	2010	3.1	2011-10-11	SaaS

Operating system support

The operating systems that the RDBMSes can run on.

	Windows	OS X	Linux	BSD	UNIX	AmigaOS	Symbian	z/OS	iOS	Android
4th Dimension	Yes	Yes	No	No	No	No	No	No	No	No
ADABAS	Yes	No	Yes	No	Yes	No	No	Yes	No	No
Adaptive Server Enterprise	Yes	No	Yes	Yes	Yes	No	No	No	Yes	Yes
Advantage Database Server	Yes	No	Yes	No	No	No	No	No	No	No
Altibase	Yes	No	Yes	No	Yes	No	No	No	No	No
Apache Derby	Yes	Yes	Yes	Yes	Yes	No	No	Yes	*?*	No
Clustrix	No	No	Yes	No	Yes	No	No	No	No	No
CUBRID	Yes	Partial	Yes	No	No	No	No	No	No	No
Drizzle	No	Yes	Yes	Yes	Yes	No	No	No	No	No
DB2	Yes	Yes (Express C)	Yes	No	Yes	No	No	Yes	Yes	No
Empress Embedded Database	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes
EXASolution	No	No	Yes	No	No	No	No	No	No	No
Firebird	Yes	Yes	Yes	Yes	Yes	No	No	Maybe	No	No
HSQLDB	Yes	Yes	Yes	Yes	Yes	No	No	Yes	*?*	*?*
H2	Yes	Yes	Yes	Yes	Yes	No	No	Yes	*?*	Yes
FileMaker	Yes	Yes	No	No	No	No	No	No	Yes	No
Informix Dynamic Server	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
Ingres	Yes	Yes	Yes	Yes	Yes	No	No	Partial	No	No
InterBase	Yes	Yes	Yes	No	Yes (Solaris)	No	No	No	No	No
Linter SQL RDBMS	Yes	Yes	Yes	Yes	Yes	No	No	Under Linux on System z	No	Yes
LucidDB	Yes	Yes	Yes	No	No	No	No	No	No	No
MariaDB	Yes	Yes	Yes	Yes	Yes	No	No	No	*?*	*?*
MaxDB	Yes	No	Yes	No	Yes	No	No	Maybe	No	No
Microsoft Access (JET)	Yes	No	No	No	No	No	No	No	No	No
Microsoft Visual Foxpro	Yes	No	No	No	No	No	No	No	No	No
Microsoft SQL Server	Yes	No	No	No	No	No	No	No	No	No
Microsoft SQL Server Compact (Embedded Database)	Yes	No	No	No	No	No	No	No	No	No
MonetDB/SQL	Yes	Yes	Yes	No	Yes	No	No	No	*?*	*?*
MySQL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*	Yes[2]
Omnis Studio	Yes	Yes	Yes	No	No	No	No	No	No	No
OpenBase SQL	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
OpenEdge	Yes	No	Yes	No	Yes	No	No	No	No	No
OpenLink Virtuoso	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No

Oracle	Yes	Yes	Yes	No	Yes	No	No	Yes	No	No
Oracle Rdb	No	No	No	No	No	No	No	No	No	No
Pervasive PSQL	Yes	Yes (OEM only)	Yes	No	No	No	No	No	No	No
Polyhedra	Yes	No	Yes	No	Yes	No	No	No	No	No
PostgreSQL	Yes	Yes	Yes	Yes	Yes	No	No	Under Linux on System z[3]	No	Yes
R:Base	Yes	No	No	No	No	No	No	No	No	No
RDM	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	No
RDM Server	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
ScimoreDB	Yes	No	No	No	No	No	No	No	No	No
SmallSQL	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No
SQL Anywhere	Yes	Yes	Yes	No	Yes	No	No	No	No	Yes
SQLBase	Yes	No	Yes	No	No	No	No	No	No	No
SQLite	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Maybe	Yes	Yes
Superbase	Yes	No	No	No	No	Yes	No	No	No	No
Teradata	Yes	No	Yes	No	Yes	No	No	No	No	No
UniData	Yes	No	Yes	No	Yes	No	No	No	No	No
UniVerse	Yes	No	Yes	No	Yes	No	No	No	No	No
Xeround Cloud Database	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Fundamental features

Information about what fundamental RDBMS features are implemented natively.

	ACID	Referential integrity	Transactions	Unicode	Interface
4th Dimension	Yes	Yes	Yes	Yes	GUI & SQL
ADABAS	Yes	No	Yes	Yes	proprietary direct call & SQL (via 3rd party)
Adaptive Server Enterprise	Yes	Yes	Yes	Yes	SQL
Advantage Database Server	Yes	Yes	Yes	Yes[4]	API & SQL
Altibase	Yes	Yes	Yes	Yes	API & GUI & SQL
Apache Derby	Yes	Yes	Yes	Yes	SQL
Clustrix	Yes	Yes	Yes	Yes	SQL
CUBRID	Yes	Yes	Yes	Yes	GUI & SQL
Drizzle	Yes	Yes	Yes	Yes	SQL
DB2	Yes	Yes	Yes	Yes	GUI & SQL
Empress Embedded Database	Yes	Yes	Yes	Yes	API & SQL
EXASolution	Yes	Yes	Yes	Yes	API & GUI & SQL
Firebird	Yes	Yes	Yes	Yes	SQL

HSQLDB	Yes	Yes	Yes	Yes	SQL
H2	Yes	Yes	Yes	Yes	SQL
Informix Dynamic Server	Yes	Yes	Yes	Yes	SQL and JSON
Ingres	Yes	Yes	Yes	Yes	SQL & QUEL
InterBase	Yes	Yes	Yes	Yes	SQL
Linter SQL RDBMS	Yes	Yes	Yes	Yes	GUI & SQL
LucidDB	Yes	No	No	Yes	SQL
MariaDB	Yes[2]	Partial[3]	Yes[2] except for DDL[4]	Yes	SQL
MaxDB	Yes	Yes	Yes	Yes	SQL
Microsoft Access (JET)	Yes	Yes	Yes	Yes	GUI & SQL
Microsoft Visual FoxPro	No	Yes	Yes	No	GUI & SQL
Microsoft SQL Server	Yes	Yes	Yes	Yes	GUI & SQL
Microsoft SQL Server Compact (Embedded Database)	Yes	Yes	Yes	Yes	GUI & SQL
MonetDB/SQL	Yes	Yes	Yes	Yes	SQL
MySQL	Yes[2]	Partial[3]	Yes[2] except for DDL	Yes	GUI [5] & SQL
OpenBase SQL	Yes	Yes	Yes	Yes	GUI & SQL
Oracle	Yes	Yes	Yes except for DDL	Yes	API & GUI & SQL
Oracle Rdb	Yes	Yes	Yes	Yes	SQL
OpenLink Virtuoso	Yes	Yes	Yes	Yes	API & GUI & SQL
Pervasive PSQL	Yes	Yes	Yes	Yes[6]	API & GUI & SQL
Polyhedra DBMS	Yes	Yes	Yes	Yes	API & SQL
PostgreSQL	Yes	Yes	Yes	Yes	API & GUI & SQL
RDM	Yes	Yes	Yes	Yes	SQL & API
RDM Server	Yes	Yes	Yes	Yes	SQL & API
ScimoreDB	Yes	Yes	Yes	Partial	SQL
SQL Anywhere	Yes	Yes	Yes	Yes	SQL
SQLBase	Yes	Yes	Yes	Yes	API & GUI & SQL
SQLite	Yes	Yes	Yes	Optional[5]	API & SQL
Teradata	Yes	Yes	Yes	Yes	SQL
UniData	Yes	No	Yes	Yes	Multiple
UniVerse	Yes	No	Yes	Yes	Multiple
Xeround Cloud Database	Yes	No	Yes	Yes	SQL
	ACID	**Referential integrity**	**Transactions**	**Unicode**	**Interface**

Note (1): Currently only supports read uncommited transaction isolation. Version 1.9 adds serializable isolation and version 2.0 will be fully ACID compliant.

Note (2): MySQL provides ACID compliance through the default InnoDB storage engine.

Note (3): "For other [than InnoDB] storage engines, MySQL Server parses and ignores the FOREIGN KEY and REFERENCES syntax in CREATE TABLE statements. The CHECK clause is parsed but ignored by all storage engines."

Note (4): Support for Unicode is new in version 10.0.

Note (5): MySQL provides GUI interface through MySQL Workbench.

Note (6): Pervasive PSQL provides UTF-8 storage.

Limits

Information about data size limits.

	Max DB size	Max table size	Max row size	Max columns per row	Max Blob/Clob size	Max CHAR size	Max NUMBER size	Min DATE value	Max DATE value	Max column name size
4th Dimension	Limited	?	?	65,135	200 GB (2 GiB Unicode)	200 GB (2 GiB Unicode)	64 bits	?	?	?
Advantage Database Server	Unlimited	16 EiB	65,530 B	65,135 / (10+ AvgFieldNameLength)	4 GiB	?	64 bits	?	?	128
Apache Derby	Unlimited	Unlimited	Unlimited	1,012 (5,000 in views)	2,147,483,647 chars	254 (VARCHAR: 32,672)	64 bits	0001-01-01	9999-12-31	128
Clustrix	Unlimited	Unlimited	64 MB on Appliance, 4 MB on AWS	?	64 MB	64 MB	64 MB	0001-01-01	9999-12-31	254
CUBRID	2 EB	2 EB	Unlimited	6400	Unlimited	1 GB	64 bits	0001-01-01	9999-12-31	254
Drizzle	Unlimited	64 TB	8 KB	1,000	4 GB (longtext, longblob)	64 KB (text)	64 bits	0001	9999	64
DB2	Unlimited	2 ZB	32,677 B	1,012	2 GB	32 KiB)	64 bits	0001-01-01	9999-12-31	128
Empress Embedded Database	Unlimited	2^{63}-1 bytes	2 GB	32,767	2 GB	2 GB	64 bits	0000-01-01	9999-12-31	32
EXASolution	Unlimited	Unlimited	Unlimited	10,000	N/A	2 MB	128 bits	0001-01-01	9999-12-31	256
FileMaker	8 TB	8 TB	8 TB	256,000,000	4 GB	10^9 characters	10^9 numbers w/ range 10^-400 to 10^400	0001-01-01	4000-12-31	100
Firebird	Unlimited[1]	~32 TB	65,536 B	Depends on data types used	2 GB	32,767 B	64 bits	100	32768	31
HSQLDB	64 TB	Unlimited[8]	Unlimited[8]	Unlimited[8]	64 TB[7]	Unlimited[8]	Unlimited[8]	0001-01-01	9999-12-31	128
H2	64 TB	Unlimited[8]	Unlimited[8]	Unlimited[8]	64 TB[7]	Unlimited[8]	64 bits	-99999999	99999999	Unlimited[8]
	Max DB size	**Max table size**	**Max row size**	**Max columns per row**	**Max Blob/Clob size**	**Max CHAR size**	**Max NUMBER size**	**Min DATE value**	**Max DATE value**	**Max column name size**

Informix Dynamic Server	~128 PB	~128 PB	32,765 bytes (exclusive of large objects)	32,765	4 TB	32,765	10^{32}	01/01/0001[10]	12/31/9999	128 bytes
Ingres	Unlimited	Unlimited	256 KB	1,024	2 GB	32 000 B	64 bits	0001	9999	256
InterBase	Unlimited[1]	~32 TB	65,536 B	Depends on data types used	2 GB	32,767 B	64 bits	100	32768	31
Linter SQL RDBMS	Unlimited	2^{30} rows	64 KB (w/o BLOBs), 4 GB (BLOB)	250	4 GB	4 KB	64 bits	0001-01-01	9999-12-31	66
Microsoft Access (JET)	2 GB	2 GB	16 MB	255	64 KB (memo field), 1 GB ("OLE Object" field)	255 B (text field)	32 bits	0100	9999	64
Microsoft Visual Foxpro	Unlimited	2 GB	65,500 B	255	2 GB	16 MB	32 bits	0001	9999	10
Microsoft SQL Server	524,272 TB (32 767 files * 16 TB max file size)	524,272 TB	8,060 bytes (Unlimited)[6]	30,000	2 GB	2 GB[6]	126 bits[2]	0001	9999	128
Microsoft SQL Server Compact (Embedded Database)	4 GB	4 GB	8,060 bytes	1024	2 GB	4000	154 bits	0001	9999	128
MySQL 5	Unlimited	MyISAM storage limits: 256 TB; Innodb storage limits: 64 TB	64 KB[3]	4,096[4]	4 GB (longtext, longblob)	64 KB (text)	64 bits	1000	9999	64
OpenLink Virtuoso	32 TB per instance (Unlimited via elastic cluster)	DB size (or 32 TB)	4 KB	200	2 GB	2 GB	2^{31}	0	9999	100
Oracle	Unlimited (4 GB * block size per tablespace)	4 GB * block size (with BIGFILE tablespace)	8 KB	1,000	128 TB	32,767 B[11]	126 bits	–4712	9999	30
	Max DB size	**Max table size**	**Max row size**	**Max columns per row**	**Max Blob/Clob size**	**Max CHAR size**	**Max NUMBER size**	**Min DATE value**	**Max DATE value**	**Max column name size**
Pervasive PSQL	4 billion objects	256 GB	2 GB	1,536	2 GB	8,000 bytes	64 bits	01-01-0001	12-31-9999	128 bytes

Polyhedra	Limited by available RAM, address space	2^{32} rows	Unlimited	65,536	4 GB (subject to RAM)	4 GB (subject to RAM)	32 bits	0001-01-01	8000-12-31	255
PostgreSQL	Unlimited	32 TB	1.6 TB	250–1600 depending on type	1 GB (text, bytea)[6] - stored inline or 4 TB (stored in pg_largeobject)[7]	1 GB	Unlimited	–4,713	5,874,897	63
RDM Embedded	Unlimited	2^{48}-1 rows	32 KB	1,000	4 GB	char: 256, varchar: 4 KB	64 bits	0001-01-01	11758978-12-31	31
RDM Server	Unlimited	2^{64}-1 rows	32 KB	32,768	Unlimited	32 KB	64 bits	0001-01-01	11758978-12-31	32
ScimoreDB	Unlimited	16 EB	8,050 B	255	16 TB	8,000 B	64 bits	?	?	?
SQL Anywhere	104 TB (13 files, each file up to 8 TB (32 KB pages))	Limited by file size	Limited by file size	45,000	2 GB	2 GB	64 bits	0001-01-01	9999-12-31	?
SQLite	128 TB (2^{31} pages * 64 KB max page size)	Limited by file size	Limited by file size	32,767	2 GB	2 GB	64 bits	No DATE type[9]	No DATE type[9]	Unlimited
Teradata	Unlimited	Unlimited	64 KB wo/lobs (64 GB w/lobs)	2,048	2 GB	10,000	64 bits	?	9999-12-31 Select 80991231 (date);	30
UniVerse	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Xeround Cloud Database	Unlimited	Unlimited	32 GB, depending on available memory	1,000	4 GB	64 KB	64 bits	1000	9999	64
	Max DB size	**Max table size**	**Max row size**	**Max columns per row**	**Max Blob/Clob size**	**Max CHAR size**	**Max NUMBER size**	**Min DATE value**	**Max DATE value**	**Max column name size**

Note (1): Firebird 2.x maximum database size is effectively unlimited with the largest known database size >980 GB. Firebird 1.5.x maximum database size: 32 TB.

Note (2): Limit is 10^{38} using `DECIMAL` datatype.

Note (3): InnoDB is limited to 8,000 bytes (excluding `VARBINARY`, `VARCHAR`, `BLOB`, or `TEXT` columns).

Note (4): InnoDB is limited to 1,000 columns.

Note (6): Using `VARCHAR (MAX)` in SQL 2005 and later.

Note (7): When using a page size of 32 KB, and when BLOB/CLOB data is stored in the database file.

Note (8): Java array size limit of 2,147,483,648 (2^{31}) objects per array applies. This limit applies to number of characters in names, rows per table, columns per table, and characters per `CHAR`/`VARCHAR`.

Note (9): Despite the lack of a date datatype, SQLite does include date and time functions, which work for timestamps between 0000-01-01 00:00:00 and 5352-11-01 10:52:47.

Note (10): Informix DATETIME type has adjustable range from YEAR only through 1/10000th second. DATETIME date range is 0001-01-01 00:00:00.00000 through 9999-12-31 23:59:59.99999.

Note (11): Since version 12c. Earlier versions support up to 4000 B.

Tables and views

Information about what tables and views (other than basic ones) are supported natively.

	Temporary table	Materialized view
4th Dimension	Yes	Planned for inclusion in next major release
ADABAS	*?*	*?*
Adaptive Server Enterprise	Yes[1]	Yes - see precomputed result sets
Advantage Database Server	Yes	No (only common views)
Altibase	Yes	No (only common views)
Apache Derby	Yes	No
Clustrix	Yes	No
CUBRID	No	No
Drizzle	Yes	No[4]
DB2	Yes	Yes
Empress Embedded Database	Yes	Yes
EXASolution	Yes	No
Firebird	Yes	No (only common views)
HSQLDB	Yes	No
H2	Yes	No
Informix Dynamic Server	Yes	No[2]
Ingres	Yes	Planned for inclusion in next major release
InterBase	Yes	No
Linter SQL RDBMS	Yes	Yes
LucidDB	No	No
MaxDB	Yes	No
Microsoft Access (JET)	No	No
Microsoft Visual Foxpro	Yes	Yes
Microsoft SQL Server	Yes	Yes[3]
Microsoft SQL Server Compact (Embedded Database)	Yes	No
MonetDB/SQL	Yes	No
MySQL	Yes	No[4]
OpenBase SQL	Yes	Yes
Oracle	Yes	Yes

Oracle Rdb	Yes	Yes
OpenLink Virtuoso	Yes	Yes
Pervasive PSQL	Yes	No
Polyhedra DBMS	No	No (only common views)
PostgreSQL	Yes	Yes[5]
RDM Embedded	Yes	No
RDM Server	Yes	No
SQL Anywhere	Yes	Yes
ScimoreDB	No	No
SQLite	Yes	No
Teradata	Yes	Yes
UniData	Yes	No
UniVerse	Yes	No
Xeround Cloud Database	Yes	No

Note (1): Server provides tempdb, which can be used for public and private (for the session) temp tables.

Note (2): Materialized views are not supported in Informix; the term is used in IBM's documentation to refer to a temporary table created to run the view's query when it is too complex, but one cannot for example define the way it is refreshed or build an index on it. The term is defined in the Informix Performance Guide.

Note (3): Query optimizer support only in Developer and Enterprise Editions. In other versions, a direct reference to materialized view and a query hint are required.

Note (4): Materialized views can be emulated using stored procedures and triggers.

Note (5): Materialized views are now standard but can be emulated in versions prior to 9.3 with stored procedures and triggers using PL/pgSQL, PL/Perl, PL/Python, or other procedural languages.

Indices

Information about what indices (other than basic B-/B+ tree indices) are supported natively.

	R-/R+ tree	Hash	Expression	Partial	Reverse	Bitmap	GiST	GIN	Full-text	Spatial	FOT
4th Dimension	*?*	Cluster	*?*	*?*	*?*	*?*	*?*	*?*	Yes	*?*	*?*
ADABAS	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*
Adaptive Server Enterprise	No	No	Yes	No	Yes	No	No	No	Yes	*?*	*?*
Advantage Database Server	No	No	Yes	No	Yes	Yes	No	No	Yes	*?*	*?*
Apache Derby	No	No	No	No	No	No	No	No	No	*?*	*?*
Clustrix	No	Yes	No	No	No	No	No	No	No	No	*?*
CUBRID	No	No	Yes	Yes	Yes	No	No	No	No	No	No
Drizzle	No	No	No	No	No	No	No	No	No	*?*	*?*
DB2	No	*?*	Yes	No	Yes	Yes	No	No	Yes	*?*	*?*

Empress Embedded Database	Yes	No	No	Yes	No	Yes	No	No	No	*?*	*?*
EXASolution	No	Yes	No	No	No	No	No	No	No	*?*	*?*
Firebird	No	No	Yes	No	Yes [1]	No	No	No	No	*?*	*?*
HSQLDB	No	No	No	No	No	No	No	No	No	*?*	*?*
H2	No	Yes	No	No	No	No	No	No	Yes	Yes	*?*
Informix Dynamic Server	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ingres	Yes	Yes	Ingres v10	No	No	Ingres v10	No	No	No	*?*	*?*
InterBase	No	No	No	No	No	No	No	No	No	*?*	*?*
Linter SQL RDBMS[10]	No	No	No	No	No	No	No	No	Yes	No	No
LucidDB	No	No	No	No	No	Yes	No	No	No	*?*	*?*
MaxDB	No	No	No	No	No	No	No	No	No	*?*	*?*
Microsoft Access (JET)	No	No	No	No	No	No	No	No	No	*?*	*?*
Microsoft Visual Foxpro	No	No	Yes	Yes	Yes[2]	Yes	No	No	No	*?*	*?*
Microsoft SQL Server	*?*	Non/Cluster & fill factor	Yes[3]	Yes[4]	No[3]	No	No	No	Yes	Yes	*?*
Microsoft SQL Server Compact (Embedded Database)	No	No	No	No	No	No	No	No	No	*?*	*?*
MonetDB/SQL	No	Yes	No	No	No	No	No	No	*?*	*?*	*?*
MySQL	MyISAM tables only	MEMORY, Cluster (NDB), InnoDB,[5] tables only	No	No	No	No	No	No	MyISAM tables and, since v5.6.4, InnoDB tables	MyISAM tables only	*?*
Oracle	Yes [11]	Cluster Tables	Yes	Yes [6]	Yes	Yes	No	No	Yes	Yes	*?*
Oracle Rdb	No	Yes	*?*	No	No	*?*	No	No	*?*	*?*	*?*
OpenLink Virtuoso	Yes	Cluster	Yes	Yes	No	Yes	No	No	Yes	Yes (Commercial only)	No
Pervasive PSQL	No	No	No	No	No	No	No	No	No	No	No
Polyhedra DBMS	No	Yes	No	No	No	No	No	No	No	No	*?*
PostgreSQL	Yes	Yes	Yes	Yes	Yes[7]	Yes[8]	Yes	Yes	Yes	PostGIS	*?*
RDM Embedded	No	Yes	No	Yes	Yes	No	No	No	No	No	No
RDM Server	No	No	No	Yes	Yes	No	No	No	No	No	No
ScimoreDB	No	No	No	No	No	No	No	No	Yes	*?*	*?*
SQL Anywhere	No	No	No	No	No	No	No	No	Yes	*?*	*?*

SQLite	Yes[8]	No	No	Yes[9]	Yes	No	No	No	Yes[10]	SpatiaLite	*?*
Teradata	No	Yes	Yes	Yes	No	Yes	No	No	*?*	*?*	*?*
UniVerse	Yes	Yes	Yes[3]	Yes[3]	Yes[3]	No	No	No	*?*	Yes[11]	*?*
Xeround Cloud Database	No	Yes	No	No	No	No	No	No	No	No	*?*
	R-/R+ tree	**Hash**	**Expression**	**Partial**	**Reverse**	**Bitmap**	**GiST**	**GIN**	**Full-text**	**Spatial**	**FOT**

Note (1): The users need to use a function from freeAdhocUDF library or similar.

Note (2): Can be implemented for most data types using expression-based indexes.

Note (3): Can be emulated by indexing a computed column (doesn't easily update) or by using an "Indexed View" (proper name not just any view works).

Note (4): Can be implemented by using an indexed view.

Note (5): InnoDB automatically generates adaptive hash index entries as needed.

Note (6): Can be implemented using Function-based Indexes in Oracle 8i and higher, but the function needs to be used in the sql for the index to be used.

Note (7): A PostgreSQL functional index can be used to reverse the order of a field.

Note (8): PostgreSQL will likely support on-disk bitmap indexes in a future version. Version 8.2 supports a related technique known as "in-memory bitmap scans".

Note (10): B+ tree and full-text only for now.

Note (11): R-Tree indexing available in base edition with Locator but some functionality requires Personal Edition or Enterprise Edition with Spatial option.

Database capabilities

	Union	**Intersect**	**Except**	**Inner joins**	**Outer joins**	**Inner selects**	**Merge joins**	**Blobs and Clobs**	**Common Table Expressions**	**Windowing Functions**	**Parallel Query**
4th Dimension	Yes	Yes	Yes	Yes	Yes	No	No	Yes	*?*	*?*	*?*
ADABAS	Yes	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*
Adaptive Server Enterprise	Yes	*?*	*?*	Yes	Yes	Yes	Yes	Yes	*?*	*?*	Yes
Advantage Database Server	Yes	No	No	Yes	Yes	Yes	Yes	Yes	*?*	No	*?*
Altibase	Yes	Yes	Yes, via MINUS	Yes	Yes	Yes	Yes	Yes	No	No	No
Apache Derby	Yes	Yes	Yes	Yes	Yes	*?*	*?*	Yes	No	No	*?*
Clustrix	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
CUBRID	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	*?*
Drizzle	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No	No[12]
DB2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Empress Embedded Database	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*	*?*	*?*
EXASolution	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Firebird	Yes	*?*	*?*	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*
HSQLDB	Yes	Yes	Yes	Yes	Yes	Yes	Yes[13]	Yes	Yes	No	Yes
H2	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	experimental[14]	No[15]	*?*
Informix Dynamic Server	Yes	Yes	Yes, via MINUS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes[16]
Ingres	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	*?*
InterBase	Yes	*?*	*?*	Yes	Yes	*?*	*?*	Yes	*?*	*?*	*?*
Linter SQL RDBMS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
LucidDB	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	*?*	*?*	*?*
MaxDB	Yes	*?*	*?*	Yes	Yes	Yes	No	Yes	*?*	*?*	*?*
Microsoft Access (JET)	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No	*?*
Microsoft Visual Foxpro	Yes	*?*	*?*	Yes	Yes	Yes	*?*	Yes	*?*	*?*	*?*
Microsoft SQL Server	Yes	Yes (2005 and beyond)	Yes (2005 and beyond)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Microsoft SQL Server Compact (Embedded Database)	Yes	No	No	Yes	Yes	*?*	No	Yes	No	No	*?*
MonetDB/SQL	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*
MySQL	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No	No
OpenBase SQL	No	No	No	Yes	Yes	Yes	Yes	Yes	*?*	*?*	*?*
OpenLink Virtuoso	Yes	*?*	*?*	Yes	Yes	Yes	*?*	Yes	*?*	*?*	*?*
Oracle	Yes	Yes	Yes, via MINUS	Yes	Yes	Yes	Yes	Yes	Yes [1]	Yes	Yes
Oracle Rdb	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*	*?*	*?*
Pervasive PSQL	Yes	No	No	Yes	Yes	*?*	*?*	Yes	No	No	No
Polyhedra DBMS	Yes	Yes	Yes	Yes	No	*?*	*?*	Yes	*?*	*?*	*?*
PostgreSQL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
RDM Embedded	No	No	No	Yes	Yes	No	No	Yes	No	No	No
RDM Server	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No	No
ScimoreDB	Yes	*?*	*?*	Yes	LEFT only	Yes	Yes	Yes	*?*	*?*	*?*
SmallSQL	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*
SQL Anywhere	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

SQLite	Yes	Yes	Yes	Yes	LEFT only	Yes	No	Yes	No	No	No
Teradata	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
UniVerse	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	*?*
Xeround Cloud Database	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No	No
	Union	**Intersect**	**Except**	**Inner joins**	**Outer joins**	**Inner selects**	**Merge joins**	**Blobs and Clobs**	**Common Table Expressions**	**Windowing Functions**	**Parallel Query**

Note (1): Recursive CTEs introduced in 11gR2 supersedes similar construct called CONNECT BY.

Data types

	Type system	**Integer**	**Floating point**	**Decimal**	**String**	**Binary**	**Date/Time**	**Boolean**	**Other**
4th Dimension	Static	UUID (16-bit), SMALLINT (16-bit), INT (32-bit), BIGINT (64-bit), NUMERIC (64-bit)	REAL, FLOAT	REAL, FLOAT	CLOB, TEXT, VARCHAR	BIT, BIT VARYING, BLOB	DURATION, INTERVAL, TIMESTAMP	BOOLEAN	PICTURE
Altibase	Static	SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit)	REAL(32-bit), DOUBLE(64-bit)	DECIMAL, NUMERIC, NUMBER, FLOAT	CHAR, VARCHAR, NCHAR, NVARCHAR, CLOB	BLOB, BYTE, NIBBLE, BIT, VARBIT	DATE		GEOMETRY
Clustrix	Static	TINYINT (8-bit), SMALLINT (16-bit), MEDIUMINT (24-bit), INT (32-bit), BIGINT (64-bit)	FLOAT (32-bit), DOUBLE	DECIMAL	CHAR, BINARY, VARCHAR, VARBINARY, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	DATETIME, DATE, TIMESTAMP, YEAR	BIT(1), BOOLEAN	ENUM, SET,
CUBRID	Static	SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit)	FLOAT, REAL(32-bit), DOUBLE(64-bit)	DECIMAL, NUMERIC	CHAR, VARCHAR, NCHAR, NVARCHAR, CLOB	BLOB	DATE, DATETIME, TIME, TIMESTAMP	BIT	MONETARY, BIT VARYING, SET, MULTISET, SEQUENCE, ENUM
Drizzle	Static	INT (32-bit), BIGINT (64-bit)	DOUBLE (aka REAL) (64-bit)	DECIMAL	BINARY, VARCHAR, VARBINARY, TEXT,	BLOB	DATETIME, DATE, TIMESTAMP		ENUM, SERIAL

Empress Embedded Database	Static	TINYINT, SQL_TINYINT, or INTEGER8; SMALLINT, SQL_SMALLINT, or INTEGER16; INTEGER, INT, SQL_INTEGER, or INTEGER32; BIGINT, SQL_BIGINT, or INTEGER64	REAL, SQL_REAL, or FLOAT32; DOUBLE PRECISION, SQL_DOUBLE, or FLOAT64; FLOAT, or SQL_FLOAT; EFLOAT	DECIMAL, DEC, NUMERIC, SQL_DECIMAL, or SQL_NUMERIC; DOLLAR	CHARACTER, ECHARACTER, CHARACTER VARYING, NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, NLSCHARACTER, CHARACTER LARGE OBJECT, TEXT, NATIONAL CHARACTER LARGE OBJECT, NLSTEXT	BINARY LARGE OBJECT or BLOB; BULK	DATE, EDATE, TIME, ETIME, EPOCH_TIME, TIMESTAMP, MICROTIMESTAMP	BOOLEAN	SEQUENCE 32, SEQUENCE
EXASolution	Static	TINYINT, SMALLINT, INTEGER, BIGINT,	REAL, FLOAT, DOUBLE	DECIMAL, DEC, NUMERIC, NUMBER	CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR, NVARCHAR2, CLOB, NCLOB	N/A	DATE, TIMESTAMP, INTERVAL	BOOLEAN, BOOL	GEOMETRY
HSQLDB	Static	TINYINT (8-bit), SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit)	DOUBLE (64-bit)	DECIMAL, NUMERIC	CHAR, VARCHAR, LONGVARCHAR, CLOB	BINARY, VARBINARY, LONGVARBINARY, BLOB	DATE, TIME, TIMESTAMP, INTERVAL	BOOLEAN	OTHER (object), BIT, BIT VARYING, ARRAY
Informix Dynamic Server	Static	SMALLINT (16-bit), INT (32-bit), INT8 (64-bit proprietary), BIGINT (64-bit)	SMALLFLOAT (32-bit), FLOAT (64-bit)	DECIMAL (32 digits float/fixed), MONEY	CHAR, VARCHAR, NCHAR, NVARCHAR, LVARCHAR, CLOB, TEXT	TEXT, BYTE, BLOB, CLOB	DATE, DATETIME, INTERVAL	BOOLEAN	SET, LIST, MULTISET, ROW, TIMESERIES, SPATIAL, JSON, BSON, USER DEFINED TYPES
Ingres	Static	TINYINT (8-bit), SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit)	FLOAT4 (32-bit), FLOAT (64-bit)	DECIMAL	C, CHAR, VARCHAR, LONG VARCHAR, NCHAR, NVARCHAR, LONG NVARCHAR, TEXT	BYTE, VARBYTE, LONG VARBYTE (BLOB)	DATE, ANSIDATE, INGRESDATE, TIME, TIMESTAMP, INTERVAL	N/A	MONEY, OBJECT_KEY, TABLE_KEY, USER-DEFINED DATA TYPES (via OME)
Linter SQL RDBMS	Static	SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit)	REAL(32-bit), DOUBLE(64-bit)	DECIMAL, NUMERIC	CHAR, VARCHAR, NCHAR, NVARCHAR, BLOB	BYTE, VARBYTE, BLOB	DATE	BOOLEAN	GEOMETRY, EXTFILE
Microsoft SQL Server	Static	TINYINT, SMALLINT, INT, BIGINT	FLOAT, REAL	NUMERIC, DECIMAL, SMALLMONEY, MONEY	CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT	BINARY, VARBINARY, IMAGE, FILESTREAM	DATE, DATETIMEOFFSET, DATETIME2, SMALLDATETIME, DATETIME, TIME	BIT	CURSOR, TIMESTAMP, HIERARCHYID, UNIQUEIDENTIFIER, SQL_VARIANT, XML, TABLE
Microsoft SQL Server Compact (Embedded Database)	Static	TINYINT, SMALLINT, INT, BIGINT	FLOAT, REAL	NUMERIC, DECIMAL, MONEY	NCHAR, NVARCHAR, NTEXT	BINARY, VARBINARY, IMAGE	DATETIME	BIT	TIMESTAMP, ROWVERSION, UNIQUEIDENTIFIER, IDENTITY, ROWGUIDCOL

MySQL	Static	TINYINT (8-bit), SMALLINT (16-bit), MEDIUMINT (24-bit), INT (32-bit), BIGINT (64-bit)	FLOAT (32-bit), DOUBLE (aka REAL) (64-bit)	DECIMAL	CHAR, BINARY, VARCHAR, VARBINARY, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	DATETIME, DATE, TIMESTAMP, YEAR	BIT(1), BOOLEAN (aka BOOL) = synonym for TINYINT	ENUM, SET, GIS data types (Geometry, Point, Curve, LineString, Surface, Polygon, GeometryCollection, MultiPoint, MultiCurve, MultiLineString, MultiSurface, MultiPolygon)
OpenLink Virtuoso	Static + Dynamic	INT, INTEGER, SMALLINT	REAL, DOUBLE PRECISION, FLOAT, FLOAT('INTNUM')'	DECIMAL, DECIMAL('INTNUM')', DECIMAL('INTNUM','INTNUM')', NUMERIC, NUMERIC('INTNUM')', NUMERIC('INTNUM','INTNUM')'	CHARACTER, CHAR('INTNUM')', VARCHAR, VARCHAR('INTNUM')', NVARCHAR, NVARCHAR('INTNUM')'	BLOB	TIMESTAMP, DATETIME, TIME, DATE	n/a	GEOMETRY, REFERENCE (URI), UDT (User Defined Type)
Oracle	Static + Dynamic (through ANYDATA)	NUMBER	BINARY_FLOAT, BINARY_DOUBLE	NUMBER	CHAR, VARCHAR2, CLOB, NCLOB, NVARCHAR2, NCHAR, LONG (deprecated)	BLOB, RAW, LONG RAW (deprecated), BFILE	DATE, TIMESTAMP (with/without TIMEZONE), INTERVAL	N/A	SPATIAL, IMAGE, AUDIO, VIDEO, DICOM, XMLType
Pervasive PSQL	Static	BIGINT, INTEGER, SMALLINT, TINYINT, UBIGINT, UINTEGER, USMALLINT, UTINYINT	BFLOAT4, BFLOAT8, DOUBLE, FLOAT	DECIMAL, NUMERIC, NUMERICSA, NUMERICSLB, NUMERICSLS, NUMERICSTB, NUMERICSTS	CHAR, LONGVARCHAR, VARCHAR	BINARY, LONGVARBINARY, VARBINARY	DATE, DATETIME, TIME	BIT	CURRENCY, IDENTITY, SMALLIDENTITY, TIMESTAMP, UNIQUEIDENTIFIER
Polyhedra	Static	INTEGER8 (8-bit), INTEGER(16-bit), INTEGER (32-bit), INTEGER64 (64-bit)	FLOAT32 (32-bit), FLOAT (aka REAL; 64-bit)	N/A	VARCHAR, LARGE VARCHAR (aka CHARACTER LARGE OBJECT)	LARGE BINARY (aka BINARY LARGE OBJECT)	DATETIME	BOOLEAN	N/A
PostgreSQL	Static	SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit)	REAL (32-bit), DOUBLE PRECISION (64-bit)	DECIMAL, NUMERIC	CHAR, VARCHAR, TEXT	BYTEA	DATE, TIME (with/without TIMEZONE), TIMESTAMP (with/without TIMEZONE), INTERVAL	BOOLEAN	ENUM, POINT, LINE, LSEG, BOX, PATH, POLYGON, CIRCLE, CIDR, INET, MACADDR, BIT, UUID, XML, JSON, arrays, composites, ranges, custom
RDM Embedded	Static	tinyint, smallint, integer, bigint	real, float, double	N/A	char, varchar, wchar, varwchar, long varchar, long varwchar	binary, varbinary, long varbinary	date, time, timestamp	bit	N/A
RDM Server	Static	tinyint, smallint, integer, bigint	real, float, double	decimal, numeric	char, varchar, wchar, varwchar, long varchar, long varwchar	binary, varbinary, long varbinary	date, time, timestamp	bit	rowid
SQLite	Dynamic	INTEGER (64-bit)	REAL (aka FLOAT, DOUBLE) (64-bit)	N/A	TEXT (aka CHAR, CLOB)	BLOB	N/A	N/A	N/A
UniData	Dynamic	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
UniVerse	Dynamic	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Xeround Cloud Database	Static	TINYINT (8-bit), SMALLINT (16-bit), MEDIUMINT (24-bit), INT (32-bit), BIGINT (64-bit)	FLOAT (32-bit), DOUBLE (aka REAL) (64-bit)	DECIMAL	CHAR, BINARY, VARCHAR, VARBINARY, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	DATETIME, DATE, TIMESTAMP, YEAR	BOOLEAN (aka BOOL) = synonym for TINYINT	ENUM, SET
	Type system	**Integer**	**Floating point**	**Decimal**	**String**	**Binary**	**Date/Time**	**Boolean**	**Other**

Other objects

Information about what other objects are supported natively.

	Data Domain	Cursor	Trigger	Function [1]	Procedure [1]	External routine [1]
4th Dimension	Yes	No	Yes	Yes	Yes	Yes
ADABAS	*?*	Yes	*?*	Yes?	Yes?	Yes
Adaptive Server Enterprise	Yes	Yes	Yes	Yes	Yes	Yes
Advantage Database Server	Yes	Yes	Yes	Yes	Yes	Yes
Altibase	Yes	Yes	Yes	Yes	Yes	Yes
Apache Derby	No	Yes	Yes	Yes [2]	Yes [2]	Yes [2]
Clustrix	No	Yes	No	Yes	Yes	Yes
CUBRID	Yes	Yes	Yes	Yes	Yes [2]	Yes
Drizzle	Yes	Yes	Yes [4]	Yes [4]	Yes [4]	Yes [4]
Empress Embedded Database	Yes via RANGE CHECK	Yes	Yes	Yes	Yes	Yes
EXASolution	Yes	No	No	Yes	Yes	Yes
DB2	Yes via CHECK CONSTRAINT	Yes	Yes	Yes	Yes	Yes
Firebird	Yes	Yes	Yes	Yes	Yes	Yes
HSQLDB	Yes	No	Yes	Yes	Yes	Yes
H2	Yes	No	Yes [2]	Yes [2]	Yes [2]	Yes
Informix Dynamic Server	Yes via CHECK	Yes	Yes	Yes	Yes	Yes [5]
Ingres	Yes	Yes	Yes	Yes	Yes	Yes
InterBase	Yes	Yes	Yes	Yes	Yes	Yes
Linter SQL RDBMS	No	Yes	Yes	Yes	Yes	No
LucidDB	No	Yes	No	Yes [2]	Yes [2]	Yes [2]
MaxDB	Yes	Yes	Yes	Yes	Yes	*?*
Microsoft Access (JET)	Yes	No	No	No	Yes, But single DML/DDL Operation	Yes
Microsoft Visual Foxpro	No	Yes	Yes	Yes	Yes	Yes
Microsoft SQL Server	Yes (2000 and beyond)	Yes	Yes	Yes	Yes	Yes

Microsoft SQL Server Compact (Embedded Database)	No	Yes	No	No	No	No
MonetDB	No	No	Yes	Yes	Yes	Yes
MySQL	No [3]	Yes	Yes	Yes	Yes	Yes
OpenBase SQL	Yes	Yes	Yes	Yes	Yes	Yes
Oracle	Yes	Yes	Yes	Yes	Yes	Yes
Oracle Rdb	Yes	Yes	Yes	Yes	Yes	Yes
OpenLink Virtuoso	Yes	Yes	Yes	Yes	Yes	Yes
Pervasive PSQL	Yes	Yes	Yes	Yes	Yes	No
Polyhedra DBMS	No	No	Yes	Yes	Yes	Yes
PostgreSQL	Yes	Yes	Yes	Yes	Yes	Yes
RDM Embedded	No	Yes	No	No	Yes	Yes
RDM Server	No	Yes	Yes	No	Yes	Yes
ScimoreDB	No	No	No	No	Yes	Yes
SQL Anywhere	Yes	Yes	Yes	Yes	Yes	Yes
SQLite	No	No	Yes	No	No	Yes
Teradata	No	Yes	Yes	Yes	Yes	Yes
UniData	No	No	Yes	Yes	Yes	Yes
UniVerse	No	No	Yes	Yes	Yes	Yes
Xeround Cloud Database	No [3]	Yes	Yes	Yes	Yes	No
	Data Domain	**Cursor**	**Trigger**	**Function** [1]	**Procedure** [1]	**External routine** [1]

Note (1): Both **function** and **procedure** refer to internal routines written in SQL and/or procedural language like PL/SQL. **External routine** refers to the one written in the host languages, such as C, Java, Cobol, etc. "Stored procedure" is a commonly used term for these routine types. However, its definition varies between different database vendors.

Note (2): In Derby, H2, LucidDB, and CUBRID, users code **functions** and **procedures** in Java.

Note (3): ENUM datatype exist. CHECK clause is parsed, but not enforced in runtime.

Note (4): In Drizzle the user codes **functions** and **procedures** in C++.

Note (5): Informix supports external functions written in Java, C, & C++.

Partitioning

Information about what partitioning methods are supported natively.

	Range	Hash	Composite (Range+Hash)	List	Expression
4th Dimension	*?*	*?*	*?*	*?*	*?*
ADABAS	*?*	*?*	*?*	*?*	*?*
Adaptive Server Enterprise	Yes	Yes	No	Yes	*?*
Advantage Database Server	No	No	No	No	*?*
Altibase	Yes	Yes	No	Yes	*?*
Apache Derby	No	No	No	No	*?*
Clustrix	Yes	No	No	No	No
CUBRID	Yes	Yes	No	Yes	*?*
IBM DB2	Yes	Yes	Yes	Yes	*?*
Empress Embedded Database	No	No	No	No	*?*
EXASolution	No	Yes	No	No	No
Firebird	No	No	No	No	*?*
HSQLDB	No	No	No	No	*?*
H2	No	No	No	No	*?*
Informix Dynamic Server	Yes	Yes	Yes	Yes	Yes
Ingres	Yes	Yes	Yes	Yes	*?*
InterBase	No	No	No	No	*?*
Linter SQL RDBMS	No	No	No	No	*?*
MaxDB	No	No	No	No	*?*
Microsoft Access (JET)	No	No	No	No	*?*
Microsoft Visual Foxpro	No	No	No	No	*?*
Microsoft SQL Server	Yes	No	No	No	*?*
Microsoft SQL Server Compact (Embedded Database)	No	No	No	No	*?*
MonetDB	Yes (M5)	Yes (M5)	Yes (M5)	No	*?*
MySQL	Yes	Yes	Yes	Yes	*?*
OpenBase SQL	*?*	*?*	*?*	*?*	*?*
Oracle	Yes	Yes	Yes	Yes	*?*
Oracle Rdb	Yes	Yes	*?*	*?*	*?*
OpenLink Virtuoso	Yes	Yes	Yes	Yes	Yes
Pervasive PSQL	No	No	No	No	No
Polyhedra DBMS	No	No	No	No	No
PostgreSQL	Yes[1]	Yes[1]	Yes[1]	Yes[1]	*?*
RDM Embedded	Yes[2]	Yes[2]	Yes[2]	No	*?*

RDM Server	No	No	No	No	*?*
ScimoreDB	No	Yes	No	No	*?*
SQL Anywhere	No	No	No	No	*?*
SQLite	No	No	No	No	*?*
Teradata	Yes	Yes	Yes	Yes	*?*
UniVerse	Yes	Yes	Yes	Yes	*?*
Xeround Cloud Database	N/A - partitioning provided transparently	N/A - partitioning provided transparently	N/A - partitioning provided transparently	N/A - partitioning provided transparently	N/A - partitioning provided transparently
	Range	**Hash**	**Composite (Range+Hash)**	**List**	**Expression**

Note (1): PostgreSQL 8.1 provides partitioning support through check constraints. Range, List and Hash methods can be emulated with PL/pgSQL or other procedural languages.

Note (2): RDM Embedded 10.1 requires the application programs to select the correct partition (using range, hash or composite techniques) when adding data, but the *database union* functionality allows all partitions to be read as a single database.

Access control

Information about access control functionalities (*work in progress*).

	Native network encryption[1]	**Brute-force protection**	**Enterprise directory compatibility**	**Password complexity rules**[2]	**Patch access**[3]	**Run unprivileged**[4]	**Audit**	**Resource limit**	**Separation of duties (RBAC)**[5]	**Security Certification**	**Label Based Access Control (LBAC)**
4D	Yes (with SSL)	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*
Adaptive Server Enterprise	Yes (optional; to pay)	Yes	Yes (optional ?)	Yes	Partial (need to register; depend on which product)	Yes	Yes	Yes	Yes	Yes (EAL4+ [1])	*?*
Advantage Database Server	Yes	No	No	No	Yes	Yes	No	No	Yes	*?*	*?*
DB2	Yes	*?*	Yes (LDAP, Kerberos…)	Yes	*?*	Yes	Yes	Yes	Yes	Yes (EAL4+[6])	*?*
Empress Embedded Database	*?*	*?*	No	No	Yes	Yes	Yes	No	Yes	No	*?*
EXASolution	No	No	Yes (LDAP)	No	Yes	Yes	Yes	Yes	Yes	No	*?*
Firebird	No	Yes	Yes (Windows trusted authenification)	No	Partial (no security page)	Yes	No	No	No[7]	*?*	*?*
HSQLDB	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	No	*?*

H2	Yes	Yes	*?*	No	*?*	Yes	*?*	Yes	Yes	No	*?*
Informix Dynamic Server	Yes	*?*	Yes[10]	*?*[10]	Yes	Yes	Yes	Yes	Yes	*?*	Yes
Linter SQL RDBMS	Yes (with SSL)	Yes	Yes	Yes (length only)	No	Yes	Yes	Yes	Yes	Yes	Yes
MariaDB	Yes (SSL)	No	Yes (with 5.2, but not on Windows servers)	No	Partial (no security page)	Yes	*?*	*?*	*?*[8]	No	*?*
Microsoft SQL Server	Yes	*?*	Yes (Microsoft Active Directory)	Yes	Yes	Yes	Yes (From 2008)	Yes	Yes	Yes (EAL4+[11])	*?*
Microsoft SQL Server Compact (Embedded Database)	No (not relevant, only file permissions)	No (not relevant)	No (not relevant)	No (not relevant)	Yes	Yes (file access)	Yes	Yes	No	*?*	*?*
MySQL	Yes (SSL with 4.0)	No	Yes (with 5.5, but only in commercial edition)	No	Partial (no security page)	Yes	*?*	*?*	*?*[8]	No	*?*
OpenBase SQL	Yes	*?*	Yes (Open Directory, LDAP)	No	*?*	*?*	*?*	*?*	*?*	*?*	*?*
OpenLink Virtuoso	Yes	Yes	Yes	Yes (optional)	Yes (optional)	Yes	Yes (optional)	Yes (optional)	Yes	No	*?*
Oracle	Yes	Yes	Yes	Yes	*?*	Yes	Yes	Yes	Yes	Yes (EAL4+[1])	*?*
Pervasive PSQL	Yes	*?*	No	No	Yes	Yes	Yes [12]	No	No	No	*?*
Polyhedra DBMS	Yes (with SSL. Optional)	No	No	No	No	Yes	Yes [13]	Yes	Yes [13]	No	*?*
PostgreSQL	Yes	Yes (for 9.1)	Yes (LDAP, Kerberos…[9])	Yes (as of 9.0 with passwordcheck module)	Yes	Yes	No	Yes	Yes	Yes (EAL1[1])	*?*
RDM Embedded	No	No	No	No	No	Yes	No	No	No	No	*?*
RDM Server	Yes	No	No	No	No	Yes	Yes	No	Yes	No	*?*
SQL Anywhere	Yes	*?*	Yes (Kerberos)	Yes	*?*	Yes	Yes	No	Yes	Yes (EAL3+[1] as Adaptive Server Anywhere)	*?*
SQLite	No (not relevant, only file permissions)	No (not relevant)	No (not relevant)	No (not relevant)	Partial (no security page)	Yes (file access)	Yes	Yes	No	No	*?*

Xeround Cloud Database	Yes (SSL with 4.0)	No	No	No	N/A - database as a service	Yes	No	No	No	No	*?*
	Native network encryption[1]	**Brute-force protection**	**Enterprise directory compatibility**	**Password complexity rules**[2]	**Patch access**[3]	**Run unprivileged**[4]	**Audit**	**Resource limit**	**Separation of duties (RBAC)**[5]	**Security Certification**	**Label Based Access Control (LBAC)**

Note (1): Network traffic could be transmitted in a secure way (not clear-text, in general SSL encryption). Precise if option is default, included option or an extra modules to buy.

Note (2): Options are present to set a minimum size for password, respect complexity like presence of numbers or special characters.

Note (3): How do you get security updates? Is it free access, do you need a login or to pay? Is there easy access through a Web/FTP portal or RSS feed or only through offline access (mail CD-ROM, phone).

Note (4): Does database process run as root/administrator or unprivileged user? What is default configuration?

Note (5): Is there a separate user to manage special operation like backup (only dump/restore permissions), security officer (audit), administrator (add user/create database), etc.? Is it default or optional?

Note (6): Common Criteria certified product list.

Note (7): FirebirdSQL seems to only have SYSDBA user and DB owner. There are no separate roles for backup operator and security administrator.

Note (8): User can define a dedicated backup user but nothing particular in default install.

Note (9): Authentication methods.

Note (10): Informix Dynamic Server supports PAM and other configurable authentication. By default uses OS authentication.

Note (11): Authentication methods.

Note (12): With the use of Pervasive AuditMaster.

Note (13): User-based security is optional in Polyhedra, but when enabled can be enhanced to a role-based model with auditing.

Databases vs schemas (terminology)

The SQL specification makes clear what an "SQL schema" is; however, different databases implement it incorrectly. To compound this confusion the functionality can, when incorrectly implemented, overlap with that of the parent-database. An SQL schema is simply a namespace within a database, things within this namespace are addressed using the member operator dot ".". This seems to be a universal amongst all of the implementations.

A true fully (database, schema, and table) qualified query is exemplified as such: `SELECT * FROM database.schema.table`

Now, the issue, both a schema and a database can be used to isolate one table, "foo" from another like named table "foo". The following is pseudo code:

- `SELECT * FROM db1.foo` vs. `SELECT * FROM db2.foo` (no explicit schema between db and table)
- `SELECT * FROM [db1.]default.foo` vs. `SELECT * FROM [db1.]alternate.foo` (no explicit db prefix)

The problem that arises is that former MySQL users will create multiple databases for one project. In this context, MySQL databases are analogous in function to Postgres-schemas, insomuch as Postgres lacks off-the-shelf

cross-database functionality that MySQL has. Conversely, PostgreSQL has applied more of the specification implementing cross-table, cross-schema, and then left room for future cross-database functionality.

MySQL aliases *schema* with *database* behind the scenes, such that `CREATE SCHEMA` and `CREATE DATABASE` are analogs. It can therefore be said that MySQL has implemented cross-database functionality, skipped schema functionality entirely, and provided similar functionality into their implementation of a database. In summary, Postgres fully supports schemas but lacks some functionality MySQL has with databases, while MySQL does not even attempt to support true schemas.

Oracle has its own spin where creating a user is synonymous with creating a schema. Thus a database administrator can create a user called PROJECT and then create a table PROJECT.TABLE. Users can exist without schema objects, but an object is always associated with an owner (though that owner may not have privileges to connect to the database). With the Oracle 'shared-everything' RAC architecture, the same database can be opened by multiple servers concurrently. This is independent of replication, which can also be used, whereby the data is copied for use by different server. In the Oracle view, the 'database' is a set of files which contains the data while the 'instance' is a set of processes (and memory) through which a database is accessed.

Informix supports multiple databases in a server instance, like MySQL. It supports the CREATE SCHEMA syntax as a way to group DDL statements into a single unit creating all objects created as a part of the schema as a single owner. Informix supports a database mode called ANSI mode which supports creating objects with the same name but owned by different users.

The end result is confusion between the database factions. The Postgres and Oracle communities maintain that one database is all that is needed for one project, per the definition of database. MySQL and Informix proponents maintain that schemas have no legitimate purpose when the functionality can be achieved with databases. Postgres adheres to the SQL specification, in a more intuitive fashion (bottom-up), while MySQL's pragmatic counterargument allows their users to get the job done while creating conceptual confusion.

References

[1] MariaDB 5.5.35 now available (https://blog.mariadb.org/mariadb-5-5-35-now-available/), MariaDB Blog

[2] http://techotv.com/run-apache-mysql-php-http-web-server-android-os-phone-tablet/Run Apache, Mysql, Php – Web server on Android mobile or Tablet

[3] http://www.oss4zos.org/mediawiki/index.php?title=PostgreSQL#z.2FOS

[4] Transactional DDL in PostgreSQL: A Competitive Analysis (http://wiki.postgresql.org/wiki/Transactional_DDL_in_PostgreSQL:_A_Competitive_Analysis)

[5] SQLite Full Unicode support is optional and not installed by default in most systems (http://www.sqlite.org/faq.html#q18) (like Android, Debian…)

[6] http://grokbase.com/t/postgresql/pgsql-general/12bsww982c/large-insert-leads-to-invalid-memory-alloc

[7] http://www.postgresql.org/docs/9.3/static/lo-intro.html

[8] The SQLite R*Tree Module (http://www.sqlite.org/rtree.html)

[9] SQLite Partial Indexes (http://sqlite.org/partialindex.html)

[10] SQLite FTS3 Extension (http://www.sqlite.org/fts3.html)

[11] geospatial

[12] How does Drizzle handle parallel "things"? (https://answers.launchpad.net/drizzle/+question/135548)

[13] New Features in HyperSQL 2.2 (http://hsqldb.org/web/features200.html)

[14] H2 > Advanced > Recursive Queries (http://h2database.com/html/advanced.html#recursive_queries)

[15] H2 Roadmap (http://www.h2database.com/html/roadmap.html)

[16] Informix parallel data query (PDQ) (http://portal.acm.org/citation.cfm?id=382443)

External links

- Comparison of different SQL implementations against SQL standards (http://troels.arvin.dk/db/rdbms/). Includes Oracle, DB2, Microsoft SQL Server, MySQL and PostgreSQL. (08/Jun/2007)
- Features, strengths and weaknesses comparison between Oracle and MSSQL (independent). (http://www.wisdomforce.com/resources/docs/MSSQL2005_ORACLE10g_compare.pdf)
- The SQL92 standard (http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt)
- Metamarkets Druid IMDB (http://metamarkets.com/druid/)
- VM-Ware Redis IMDB (http://redis.io/)
- CSQL DB (http://www.csqldb.com)

Comparison of database tools

The following tables compare general and technical information for a number of available database administrator tools. Please see individual product articles for further information. This article is neither all-inclusive nor necessarily up to date.

General

Product	Creator	Latest stable release date	Latest stable release	Latest testing release	License	Runs on Windows	Runs on Mac OS X	Runs on Linux	Oracle	MySQL	PostgreSQL	SQL Server	ODBC	JDBC	SQLite	Other	Programming language
Adminer	Jakub Vrána	2013-06-23	3.7.1	none distributed	Apache license or GPL	Yes	Yes	Yes	Yes	Yes	Yes	Yes			Yes		PHP
Advanced Query Tool (AQT)	Cardett Associates Ltd[1]	2009-08-31	8.2.8	8.2.8 (2009-08-31)	Proprietary	Yes	No	No	Yes	Yes	Yes	Yes	Yes				C++
DaDaBIK	Eugenio Tacchini	2012-04-03	4.4	*?*	Proprietary	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes		PHP
Database Deployment Manager	The Unauthorized Frog project	2012-05-29	v0.1i	*?*	LGPL	Yes	No	Yes		Yes							Qt/C++
DatabaseSpy	Altova	2013-06-12	v2013r2sp1	*?*	Proprietary	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes		IBM DB2, Sybase, MS Access	C++
DBEdit	Jef Van Den Ouweland	2011-03-18	2.4	*?*	GPL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	IBM DB2, HSQLDB, Apache Derby, H2	Java
Devgems Data Modeler	Devgems	2009-04-30	1.0.0	*?*	Proprietary	Yes	No	No				Yes				InterBase, Firebird	Embarcadero Delphi
Embarcadero DBArtisan	Embarcadero Technologies	*?*	8.6.3	9.5	Proprietary	Yes	No	No	Yes	Yes		Yes	Yes	Yes		DB2, Sybase ASE	C++ Java
Epictetus	Antilogic Software	*?*	*?*	1.0 (2009-06-17)	Proprietary	Yes	Yes	Yes	Yes		Yes	Yes				Sybase, InterBase/Firebird, H2, HSQLDB	Java

ERwin Data Modeler	ca.com	*?*	7.3	*?*	Proprietary	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes			MS Access, DB2, Foxpro, Informix, Ingres, Sybase, Teradata	*?*
HeidiSQL	HeidiSQL	2012-02-12	7.0	Betas	GPL	Yes	requires Wine — Java edition abandoned	requires Wine — Java edition abandoned		Yes		Yes					Embarcadero Delphi
Maatkit	Baron Schwartz	2010-06-01	5247	*?*	GPL	Yes	Yes	Yes		Yes							Perl
Microsoft SQL Server Management Studio	Microsoft	*?*	*?*	*?*	Proprietary	Yes	No	No				Yes					.Net
ModelRight	ModelRight	*?*	3.6	3.7	Proprietary	Yes	No	No	Yes	Yes		Yes	Yes			SQL Server, Oracle, MySQL, PostgreSQL, DB2, DB2/zOS, MS Access	C++
MySQL Workbench	Oracle Corporation	2013-02-15	5.2.47	*?*	GPL	Yes	Yes	Yes		Yes							C++/C# Objective-C
Navicat	PremiumSoft CyberTech Ltd.	2013-07-03/ 2013-06-10	11.0.8/ 11.0.4	11.0.8 (2013-07-03)/ 11.0.4 (2013-06-10)	Proprietary	Yes	Yes	requires Wine	Yes	Yes	Yes	Yes	Yes		Yes		Borland Delphi Objective-C
Navicat Data Modeler	PremiumSoft CyberTech Ltd.	2013-02-04/ 2013-06-14	1.0.11/ 1.0.4	1.0.11 (2013-02-04)/ 1.0.4 (2013-06-14)	Proprietary	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes		Borland Delphi Objective-C
OpenAdmin Tool	IBM	2013-09	3.12	*?*	Proprietary	Yes	Yes	Yes	No	No	No	No			No	IBM Informix	PHP
Oracle Enterprise Manager	Oracle Corp.	2009-03-03	10gR5	*?*	Proprietary	Yes	No	Yes	Yes	Yes		Yes				DB2, Sybase, TimesTen	Java
Oracle SQL Developer	Oracle Corp.	2010-03-01	2.1.1	3.0 early adopter	Proprietary	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes		Microsoft Access, Sybase, DB2, Teradata	Java
Orbada	Andrzej Ka?u?a	2011-02-08	1.0.4.109	1.0.4.109 (2011-02-08)	GPL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Firebird, HSQL, Interbase, Derby all with JDBC driver	Java
pgAdmin III	pgAdmin Development Team	2012-09-11	1.18.0	*?*	PostgreSQL License	Yes	Yes	Yes			Yes						C++
phpLiteAdmin	Dane Iracleous	2012-5-31	1.9.2	*?*	GPL	Yes	Yes	Yes	No	No	No	No	No	No	Yes		PHP
phpMyAdmin	phpMyAdmin Development Team	2013-05-14	4.0.1	none	GPL	Yes	Yes	Yes		Yes						Drizzle, MariaDB	php

phpPgAdmin	The phpPgAdmin Project	2012-03-22	5.0.4	***?***	GPL	Yes	Yes	Yes			Yes						php
SQLPro SQL Client	Vive	2008-02-04	1.4	***?***	Proprietary	Yes	No	No	Yes	Yes	Yes	Yes			Yes		C++
SQLyog	Webyog Softworks Pvt. Ltd.	Template:2013-11-14	11.28	***?***	GPLv2	Yes	requires Wine	requires Wine		Yes							C++
SQuirreL SQL	Colin Bell, Gerd Wagner, Rob Manning and others	2013-05-06	3.5.0	***?***	GPLv2 & LGPLv2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Access,Axion Java RDBMS, Apache Derby, Daffodil DB, Fujitsu Siemens SESAM/SQL, Firebird, FrontBase, HSQLDB, Hypersonic SQL, H2 (DBMS), IBM DB2, Informix, Ingres, OpenIngres, InstantDB, InterBase, Mckoi SQL Database, Microsoft SQL Server, Mimer SQL, Netezza, Pointbase, SAPDB, Sybase, Sunopsis XML Driver, Teradata Warehouse, ThinkSQL RDBMS, Vertica Analytic Database, and others with JDBC drivers.	Java
Toad	Quest Software	Various	Various	Betas	Proprietary	Yes	No	No	Yes	Yes		Yes	Yes			DB2, Sybase	Embarcadero Delphi, C#.NET
Toad Data Modeler	Quest Software	2009-03-05	3.3.8	Betas [2]	Proprietary	Yes	No	No	Yes	Yes	Yes	Yes				DB2, MS Access, Sybase	Embarcadero Delphi
TOra	Community	2010-09-19	2.1.3	?	GPL	Yes	Yes	Yes	Yes	Yes	Yes					Teradata	C++/Qt

Features

Legend

- Create/alter table:
  - Yes - can create table, alter its definition and data, and add new rows
  - Some - can only create/alter table definition, not data
- Browse table:
  - Yes - can browse table definition and data
  - Some - can only browse table definition
- Multi-server support:
  - Yes - can manage from the same window/session multiple servers
  - Some - can manage from a different window/session multiple servers
- Monitoring server:
  - Yes - includes a headless server, that runs checks and reports failures

Tools	User Interface	Create & Alter wizard				Browse				Auto Completion	Syntax colored	Multi server support	Monitoring server
		Database	Table	Procedure	Trigger	Database	Table	Procedure	Trigger				
Adminer	Browser-based	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	*?*	*?*
Altova DatabaseSpy	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*
DaDaBIK	Browser-based	No	Some[3]	No	No	No	Some[4]	No	No	No	No	No	*?*
Database Deployment Manager	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	*?*
DBEdit	desktop	No	No	No	No	Yes	Yes	Yes	No	No	Yes	No	*?*
Devgems Data Modeler	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	*?*	*?*
DeZign for Databases	desktop	Yes	Some	Yes	Yes	Yes	Some	Yes	Yes	No	Yes	*?*	*?*
Embarcadero DBArtisan	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*	*?*
Epictetus	desktop	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	*?*	*?*
Microsoft SQL Server Management Studio	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No
ModelRight	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	*?*	*?*
Navicat	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*
Navicat Data Modeler	desktop	No	Yes	No	Yes	No	No	No	No	No	No	Yes	*?*
OpenAdmin Tool	Browser-based	Yes	Yes	No	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Oracle Enterprise Manager	Browser-based	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*	Yes

Oracle SQL Developer	desktop	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	*?*
Orbada	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
pgAdmin III	TDI	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		*?*
phpliteadmin	Browser-based	Yes	Yes	No	No	Yes	Yes	No	No	No	No	*?*	*?*
phpMyAdmin	Browser-based	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	*?*
phpPgAdmin	Browser-based	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*	*?*
SQLyog	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*	*?*
SQuirreL SQL	desktop	*?*	*?*	*?*	*?*	Yes	Yes	*?*	*?*	Yes	Yes	Some	*?*
SQLPro SQL Client	desktop	No	No	No	No	Yes	Yes	Yes	Yes	No	Yes	*?*	*?*
Toad	desktop	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*	*?*
Toad Data Modeler	desktop	Yes	Yes	Yes	Yes	Some	Some	Some	Some	No	Yes	*?*	*?*
TOra	desktop	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*?*

Features (continued)

Legend:

- User manager:
  - Yes - user manager with support for database and schema permissions as well as for individual object (table, view, functions) permissions
  - Some - simple user manager with support for database and schema permissions
  - No - no user manager, or read-only user manager

	user manager	Plugin	Compare	Import	Export	Debugger	Source control
Adminer	Yes	Yes	Yes	SQL script, CSV, TSV or the above in zip (as a plugin); imports of server-site file in SQL or SQL in zip, gzip or bzip2	SQL script, CSV, TSV or the above in zip, gzip, bzip2; XML (as a plugin)	No	Git
Altova DatabaseSpy	No	No	Yes	CSV, XML	XML, XML Structure, CSV, HTML, MS Excel	No	*?*
DaDaBIK	Some[5]	No	No	No	CSV	Yes	No
DBEdit	No	No	No	No	MS Excel, PDF, Text, SQL script	No	*?*
Devgems Data Modeler	No	No	No	No	SQL; HTML report; diagram as EMF, BMP	No	*?*
DeZign for Databases	No	No	Yes	No	SQL; diagram as EMF, BMP, GIF, JPEG, PNG	No	*?*
Epictetus	No	Yes	No	No	Excel	No	*?*
ModelRight	Some	Yes	Yes	Yes - from supported databases using native interfaces, or from any ODBC source	SQL; XML; DTD; Diagram as BMP, JPEG	No	*?*

Navicat	Yes	No	Yes	Yes - TXT, CSV, DBF, HTML, MS Excel, MS Access, Paradox file, WK1, WQ1, XML, or from any ODBC source (See link for limitations[6])	Yes - TXT, CSV, HTML, XML, DBF, SQL script, RTF, MS Word, MS Excel, MS Access, MS Windows Clipboard, Paradox file, WK1, WQ1, SLK, DIF, LDIF (See link for limitations)	Yes	No
Navicat Data Modeler	No	No	Yes	Yes - Import Database from server/ODBC	Yes - Export SQL	No	No
OpenAdmin Tool	Yes	Yes	Yes	Yes - CSV, Delimited, Fixed Width	Yes - CSV, Delimited, Fixed Width	No	*?*
Oracle SQL Developer	Yes	*?*	*?*	Yes	Yes	Yes	*?*
Orbada	No	Yes	Yes	SQL script	SQL script, CSV, XML, HTML, PDF, Excel, DBF, DataText	No	No
pgAdmin III	Yes	Yes	No	CSV, Text, or binary	CSV, text, HTML, XML	Yes	No
phpMyAdmin	Yes	Some	Yes	Yes - CSV, SQL, XML, Excel, ODS	Yes - CSV, LaTeX, Excel, Word, ODS, ODT, XML, SQL, YAML, Texy!, JSON, NHibernate, PHP, PDF, MediaWiki	No	Git
phpPgAdmin	*?*	*?*	*?*	*?*	*?*	No	*?*
SQLPro SQL Client	*?*	*?*	*?*	*?*	*?*	No	*?*
SQLyog	Yes	*?*	Yes	Yes	Yes	*?*	*?*
SQL Server Management Studio	Yes	*?*	*?*	*?*	*?*	Yes	*?*
SQuirreL SQL	*?*	Yes	*?*	Yes	*?*	No	*?*
Toad	Some	No	Yes	Yes	Yes	Yes	SVN, CVS, TFS, VSS
Toad Data Modeler	No	*?*	Yes	Toad for Oracle ERD, ERWin 7.1(XML) via plugin	SQL; meta data in XML; report in HTML/RTF/CSV; diagram as BMP, JPEG, PNG	No	*?*
TOra	Some	No	Yes	Yes	Yes	Yes	No

Features - visual design and reverse engineering

Legend:

- Visual schema/E-R design: the ability to draw entity-relationship diagrams for the database. If missing, the following two features will also be missing
- Reverse engineering - the ability to produce an ER diagram from a database, complete with foreign key relationships
  - Yes - supports incremental reverse engineering, preserving user modifications to the diagram and importing only changes from the database
  - Some - can only reverse engineer the entire database at once and drops any user modifications to the diagram (can't "refresh" the diagram to match the database)

- Forward engineering - the ability to update the database schema with changes made to its entities and relationships via the ER diagram visual designer
  - Yes - can update user-selected entities
  - Some - can only update the entire database at once

	Visual query builder	Visual schema/model/E-R diagram design	Reverse engineering	Forward engineering	ER diagram groupboxes
Adminer	Yes	Yes	Yes	No	No
Altova DatabaseSpy	Yes	Yes	Yes	Yes	*?*
DaDaBIK	Some[7]	No	No	No	No
Database Deployment Manager	Yes	Yes	Yes	No	No
DBEdit	No	No	No	No	No
Devgems Data Modeler	No	Yes	Yes	Yes	Yes
DeZign for Databases	No	Yes	Yes	Yes[8]	Yes
ModelRight	No	Yes	Yes	Yes	Yes
Navicat	Yes	Yes	Yes	Yes	Yes
Navicat Data Modeler	No	Yes	Yes	Yes	Yes
OpenAdmin Tool	No	No	No	No	No
Oracle SQL Developer	Yes	Yes	Yes	Yes	*?*
Orbada	No	No	No	No	No
pgAdmin III	Yes	No	No	No	No
phpMyAdmin	Yes	Yes	Yes	No	No
phpPgAdmin	No	No	No	No	No
SQL Server Management Studio	*?*	Yes	Yes	*?*	*?*
SQLPro SQL Client	*?*	No	No	No	No
SQLyog	Yes[9]	Yes[10]	Yes	Yes	*?*
SQuirreL SQL	Yes	Yes	Yes[11]	*?*	No
Toad	Yes	Yes	Yes	Yes	*?*
Toad Data Modeler	No	Yes	Yes	Yes[12]	*?*

Notes

[1] Cardett Associates Ltd (http://www.cardett.co.nz/)
[2] Toad Data Modeler Betas (http://modeling.inside.quest.com)
[3] create CRUD interfaces, so create table data, not table theriselves
[4] browse table data, not table definitions
[5] can manage its own users, which override the DBMS users
[6] title= Navicat feature matrix (http://navicat.com/en/products/navicat_mysql/mysql_feature.html)
[7] the CRUD interface created includes a search form
[8] Via SQL scripts for dropping and creating, which must be executed outside DeZign for Databases
[9] SQLYog query builder (http://www.webyog.com/en/screenshots_sqlyog.php)
[10] SQLYog schema designer (http://www.webyog.com/en/screenshots_sqlyog.php)
[11] Only incremental, by manually going through each table and clicking "Add to graph"
[12] Generated SQL must be executed outside Toad Data Modeler

External links

- Database administration and schema design tools (http://wiki.dandascalescu.com/reviews/software/database)

Comparison of object-relational database management systems

This is a **comparison of object-relational database management systems** (ORDBMSs). Each system has at least some features of an object-relational database; they vary widely in their completeness and the approaches taken.

The following tables compare general and technical information; please see the individual products' articles for further information. Unless otherwise specified in footnotes, comparisons are based on the stable versions without any add-ons, extensions or external programs.

Basic data

Name	Vendor	License	OS	Notes
Adaptive Server Enterprise	SAP	Proprietary	Cross-platform	
CUBRID	NHN Corporation	GPL/BSD	Linux, Windows	
DB2	IBM	Proprietary	Cross-platform	
GigaBASE	knizhnik	MIT	Various	SourceForge download page [1]
Greenplum Database	Greenplum division of EMC Corporation	Proprietary	?	Uses PostgreSQL codebase
Informix	IBM	Proprietary	Cross-platform	
Caché	InterSystems	Proprietary		
LogicSQL	Shanghai Shifang Software, Inc.	unknown license	Download page [2]	
Microsoft SQL Server	Microsoft Corporation	Proprietary	Windows	Supports data objects in .NET languages
Oracle Database	Oracle Corporation	Proprietary	Linux, Windows, Unix	
PostgreSQL	PostgreSQL Global Development Group	Postgres	Cross-platform	

OpenEdge Advanced Business Language (formerly Progress 4GL)	Progress Software Corporation	Proprietary	Cross-platform	
Valentina	Paradigma Software	Proprietary	Windows, Linux, Mac OS X	Web site [3]
Virtuoso Universal Server	OpenLink Software	GPLv2 or proprietary	Cross-platform	
VMDS (Version Managed Data Store)	GE Energy, a division of General Electric	Proprietary	?	GIS for public utilities; can be stored inside Oracle Database
WakandaDB	4th Dimension	AGPLv3 or proprietary	Windows, Linux, Mac OS X	Based on REST and Server-Side JavaScript
Zope Object Database	Zope Corporation	Zope Public License	Cross-platform	For Python, also included in Zope web application server

Object features

Information about what fundamental ORDBMSes features are implemented natively.

	Type	Method	Type inheritance	Table inheritance
CUBRID	Yes	Yes	Yes	Yes
LogicSQL[4]	*?*	*?*	*?*	*?*
Oracle	Yes	Yes[5]	Yes	Yes
OpenLink Virtuoso	Yes	Yes	Yes	Yes
PostgreSQL	Yes	Yes	Yes	Yes
Informix	Yes	Yes	Yes	Yes
WakandaDB	Yes	Yes	Yes	Yes

Data types

Information about what data types are implemented natively.

	Array	List	Set	Multiset	Object reference
CUBRID	Yes	Yes	Yes	Yes	Yes
LogicSQL	*?*	*?*	*?*	*?*	*?*
Oracle	Yes	Yes	Yes	Yes	Yes
OpenLink Virtuoso	Yes	Yes	Yes	Yes	Yes
PostgreSQL	Yes	Yes	Yes	Yes	Yes
Informix	No	Yes	Yes	Yes	Yes

References

[1] http://sourceforge.net/projects/gigabase/
[2] http://webdocs.cs.ualberta.ca/~yuan/databases/index.html
[3] http://www.valentina-db.com/
[4] http://webdocs.cs.ualberta.ca/~yuan/databases/index.html
[5] No private methods, no way to call super method from a child.

External links

- Arvin.dk (http://troels.arvin.dk/db/rdbms/), Comparison of different SQL implementations

Transactions per second

In a very generic sense, the term **Transactions Per Second** refers to the number of atomic actions performed by certain entity per second. In a more restricted view, the term is usually used by DBMS vendor and user community to refer to the number of database transactions performed per second.

SQL Server

Microsoft SQL Server

Microsoft SQL Server

Developer(s)	Microsoft
Stable release	SQL Server 2012 / April 2012
Development status	Active
Written in	C, C++
Operating system	Microsoft Windows Windows Server
Platform	IA-32, x64 or IA-64 .NET Framework 3.5
Available in	English, Chinese, French, German, Italian, Japanese, Korean, Portuguese (Brazil), Russian and Spanish
Type	Relational database management system
License	Proprietary software; both commercial and freeware editions are available
Website	www.microsoft.com/sqlserver [1]

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Its primary query languages are T-SQL and ANSI SQL.

History

Genesis

SQL Server Release History

Version	Year	Release Name	Codename
1.0 (OS/2)	1989	SQL Server 1.0 (16 bit)	-
1.1 (OS/2)	1991	SQL Server 1.1 (16 bit)	-
4.21 (WinNT)	1993	SQL Server 4.21	SQLNT
6.0	1995	SQL Server 6.0	SQL95
6.5	1996	SQL Server 6.5	Hydra
7.0	1998	SQL Server 7.0	Sphinx

-	1999	SQL Server 7.0 OLAP Tools	Palato mania
8.0	2000	SQL Server 2000	Shiloh
8.0	2003	SQL Server 2000 64-bit Edition	Liberty
9.0	2005	SQL Server 2005	Yukon
10.0	2008	SQL Server 2008	Katmai
10.25	2010	SQL Azure DB	CloudDatabase
10.5	2010	SQL Server 2008 R2	Kilimanjaro (aka KJ)
11.0	2012	SQL Server 2012	Denali
12.0	2014	SQL Server 2014	Hekaton

Prior to version 7.0 the code base for MS SQL Server was sold by Sybase SQL Server to Microsoft, and was Microsoft's entry to the enterprise-level database market, competing against Oracle, IBM, and, later, Sybase. Microsoft, Sybase and Ashton-Tate originally worked together to create and market the first version named SQL Server 1.0 for OS/2 (about 1989) which was essentially the same as Sybase SQL Server 3.0 on Unix, VMS, etc. Microsoft SQL Server 4.2 was shipped around 1992 (available bundled with IBM OS/2 version 1.3). Later Microsoft SQL Server 4.21 for Windows NT was released at the same time as Windows NT 3.1. Microsoft SQL Server v6.0 was the first version designed for NT, and did not include any direction from Sybase.

About the time Windows NT was released in July 1993, Sybase and Microsoft parted ways and each pursued its own design and marketing schemes. Microsoft negotiated exclusive rights to all versions of SQL Server written for Microsoft operating systems. (In 1996 Sybase changed the name of its product to Adaptive Server Enterprise to avoid confusion with Microsoft SQL Server.) Until 1994, Microsoft's SQL Server carried three Sybase copyright notices as an indication of its origin.

SQL Server 7.0 and SQL Server 2000 included modifications and extensions to the Sybase code base, adding support for the IA-64 architecture. By SQL Server 2005 the legacy Sybase code had been completely rewritten.[2]

Since the release of SQL Server 2000, advances have been made in performance, the client IDE tools, and several complementary systems that are packaged with SQL Server 2005. These include:

- an extract-transform-load (ETL) tool (SQL Server Integration Services or SSIS)
- a Reporting Server
- an OLAP and data mining server (Analysis Services)
- several messaging technologies, specifically Service Broker and Notification Services

SQL Server 2005

SQL Server 2005 (formerly codenamed "Yukon") released in October 2005. It included native support for managing XML data, in addition to relational data. For this purpose, it defined an `xml` data type that could be used either as a data type in database columns or as literals in queries. XML columns can be associated with XSD schemas; XML data being stored is verified against the schema. XML is converted to an internal binary data type before being stored in the database. Specialized indexing methods were made available for XML data. XML data is queried using XQuery; SQL Server 2005 added some extensions to the T-SQL language to allow embedding XQuery queries in T-SQL. In addition, it also defines a new extension to XQuery, called XML DML, that allows query-based modifications to XML data. SQL Server 2005 also allows a database server to be exposed over web services using Tabular Data Stream (TDS) packets encapsulated within SOAP (protocol) requests. When the data is accessed over web services, results are returned as XML.

Common Language Runtime (CLR) integration was introduced with this version, enabling one to write SQL code as Managed Code by the CLR. For relational data, T-SQL has been augmented with error handling features (try/catch) and support for recursive queries with CTEs (Common Table Expressions). SQL Server 2005 has also been enhanced with new indexing algorithms, syntax and better error recovery systems. Data pages are checksummed for better error resiliency, and optimistic concurrency support has been added for better performance. Permissions and access control have been made more granular and the query processor handles concurrent execution of queries in a more efficient way. Partitions on tables and indexes are supported natively, so scaling out a database onto a cluster is easier. SQL CLR was introduced with SQL Server 2005 to let it integrate with the .NET Framework.

SQL Server 2005 introduced "MARS" (Multiple Active Results Sets), a method of allowing usage of database connections for multiple purposes.[3]

SQL Server 2005 introduced DMVs (Dynamic Management Views), which are specialized views and functions that return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance.[4]

Service Pack 1 (SP1) of SQL Server 2005 introduced Database Mirroring,[5] a high availability option that provides redundancy and failover capabilities at the database level. Failover can be performed manually or can be configured for automatic failover. Automatic failover requires a witness partner and an operating mode of synchronous (also known as high-safety or full safety).

SQL Server 2008

SQL Server 2008 (formerly codenamed "Katmai") was released on August 6, 2008[6] and aims to make data management self-tuning, self organizing, and self maintaining with the development of *SQL Server Always On* technologies, to provide near-zero downtime. SQL Server 2008 also includes support for structured and semi-structured data, including digital media formats for pictures, audio, video and other multimedia data. In current versions, such multimedia data can be stored as BLOBs (binary large objects), but they are generic bitstreams. Intrinsic awareness of multimedia data will allow specialized functions to be performed on them. According to Paul Flessner, senior Vice President, Server Applications, Microsoft Corp., SQL Server 2008 can be a data storage backend for *different varieties of data: XML, email, time/calendar, file, document, spatial, etc* as well as perform *search, query, analysis, sharing, and synchronization* across all data types.

Other new data types include specialized date and time types and a *Spatial* data type for location-dependent data. Better support for unstructured and semi-structured data is provided using the new *FILESTREAM* data type, which can be used to reference any file stored on the file system. Structured data and metadata about the file is stored in SQL Server database, whereas the unstructured component is stored in the file system. Such files can be accessed both via Win32 file handling APIs as well as via SQL Server using T-SQL; doing the latter accesses the file data as a BLOB. Backing up and restoring the database backs up or restores the referenced files as well. SQL Server 2008 also natively supports hierarchical data, and includes T-SQL constructs to directly deal with them, without using recursive queries.

The Full-text search functionality has been integrated with the database engine. According to a Microsoft technical article, this simplifies management and improves performance.

Spatial data will be stored in two types. A "Flat Earth" (GEOMETRY or planar) data type represents geospatial data which has been projected from its native, spherical, coordinate system into a plane. A "Round Earth" data type (GEOGRAPHY) uses an ellipsoidal model in which the Earth is defined as a single continuous entity which does not suffer from the singularities such as the international dateline, poles, or map projection zone "edges". Approximately 70 methods are available to represent spatial operations for the Open Geospatial Consortium Simple Features for SQL, Version 1.1.

SQL Server includes better compression features, which also helps in improving scalability. It enhanced the indexing algorithms and introduced the notion of filtered indexes. It also includes *Resource Governor* that allows reserving

resources for certain users or workflows. It also includes capabilities for transparent encryption of data (TDE) as well as compression of backups. SQL Server 2008 supports the ADO.NET Entity Framework and the reporting tools, replication, and data definition will be built around the Entity Data Model. SQL Server Reporting Services will gain charting capabilities from the integration of the data visualization products from Dundas Data Visualization, Inc., which was acquired by Microsoft. On the management side, SQL Server 2008 includes the *Declarative Management Framework* which allows configuring policies and constraints, on the entire database or certain tables, declaratively. The version of SQL Server Management Studio included with SQL Server 2008 supports IntelliSense for SQL queries against a SQL Server 2008 Database Engine. SQL Server 2008 also makes the databases available via Windows PowerShell providers and management functionality available as Cmdlets, so that the server and all the running instances can be managed from Windows PowerShell.

SQL Server 2008 R2

SQL Server 2008 R2 (10.50.1600.1, formerly codenamed "Kilimanjaro") was announced at TechEd 2009, and was released to manufacturing on April 21, 2010. SQL Server 2008 R2 adds certain features to SQL Server 2008 including a master data management system branded as Master Data Services, a central management of master data entities and hierarchies. Also Multi Server Management, a centralized console to manage multiple SQL Server 2008 instances and services including relational databases, Reporting Services, Analysis Services & Integration Services.

SQL Server 2008 R2 includes a number of new services, including PowerPivot for Excel and SharePoint, Master Data Services, StreamInsight, Report Builder 3.0, Reporting Services Add-in for SharePoint, a Data-tier function in Visual Studio that enables packaging of tiered databases as part of an application, and a SQL Server Utility named UC (Utility Control Point), part of AMSM (Application and Multi-Server Management) that is used to manage multiple SQL Servers.

The first SQL Server 2008 R2 service pack (10.50.2500, Service Pack 1) was released on July 11, 2011.

The second SQL Server 2008 R2 service pack (10.50.4000, Service Pack 2) was released on July 26, 2012.

SQL Server 2012

At the 2011 Professional Association for SQL Server (PASS) summit on October 11, Microsoft announced that the next major version of SQL Server (codenamed "Denali"), would be SQL Server 2012. It was released to manufacturing on March 6, 2012. SQL Server 2012 Service Pack 1 was released to manufacturing on November 9, 2012.

It was announced to be the last version to natively support OLE DB and instead to prefer ODBC for native connectivity.

SQL Server 2012's new features and enhancements include AlwaysOn SQL Server Failover Cluster Instances and Availability Groups which provides a set of options to improve database availability, Contained Databases which simplify the moving of databases between instances, new and modified Dynamic Management Views and Functions, programmability enhancements including new spatial features, metadata discovery, sequence objects and the THROW statement, performance enhancements such as ColumnStore Indexes as well as improvements to OnLine and partition level operations and security enhancements including provisioning during setup, new permissions, improved role management, and default schema assignment for groups.

SQL Server 2014

SQL Server 2014 is still in Community Technology Preview stage. As of November, 2013 there have been two such revisions, CTP1 and CTP2.[7] SQL Server 2014 will provide a new in-memory capability for tables that can fit entirely in memory (also known as Hekaton). Whilst small tables may be entirely resident in memory in all versions of SQL Server, they also may reside on disk, so work is involved in reserving RAM, writing evicted pages to disk, loading new pages from disk, locking the pages in ram while they are being operated on, and many other tasks. By treating a table as guaranteed to be entirely resident in memory much of the 'plumbing' of disk-based databases can be avoided.[8]

For disk-based SQL Server applications, it also provides SSD bufferpool extension, which can improve application performance transparently by leveraging SSD as the intermediate memory hierarchy between DRAM and spinning media.

SQL Server 2014 also enhances AlwaysOn (HADR) solution by increasing the readable secondaries count and sustaining read operations upon secondary-primary disconnections, and it provides new hybrid disaster recovery and backup solutions with Windows Azure, enabling customers to use their existing skills with the on-premises product offerings to take advantage of Microsoft's global datacenters. In addition, it takes advantage of new Windows Server 2012 and Windows Server 2012 R2 capabilities for database application scalability in a physical or virtual environment.

Editions

Microsoft makes SQL Server available in multiple editions, with different feature sets and targeting different users. These editions are:

Mainstream editions

Datacenter

> SQL Server 2008 R2 Datacenter is the full-featured edition of SQL Server and is designed for datacenters that need the high levels of application support and scalability. It supports 256 logical processors and virtually unlimited memory. Comes with StreamInsight Premium edition. The Datacenter edition has been retired in SQL Server 2012, all its features are available in SQL Server 2012 Enterprise Edition.[9]

Enterprise

> SQL Server Enterprise Edition includes both the core database engine and add-on services, with a range of tools for creating and managing a SQL Server cluster. It can manage databases as large as 524 petabytes and address 2 terabytes of memory and supports 8 physical processors. SQL Server 2012 Enterprise Edition supports 160 physical processors

Standard

> SQL Server Standard edition includes the core database engine, along with the stand-alone services. It differs from Enterprise edition in that it supports fewer active instances (number of nodes in a cluster) and does not include some high-availability functions such as hot-add memory (allowing memory to be added while the server is still running), and parallel indexes.

Web

> SQL Server Web Edition is a low-TCO option for Web hosting.

Business Intelligence

> Introduced in SQL Server 2012 and focusing on Self Service and Corporate Business Intelligence. It includes the Standard Edition capabilities and Business Intelligence tools: PowerPivot, Power View, the BI Semantic Model, Master Data Services, Data Quality Services and xVelocity in-memory analytics.

Workgroup

SQL Server Workgroup Edition includes the core database functionality but does not include the additional services. Note that this edition has been retired in SQL Server 2012.

Express

SQL Server Express Edition is a scaled down, free edition of SQL Server, which includes the core database engine. While there are no limitations on the number of databases or users supported, it is limited to using one processor, 1 GB memory and 10 GB database files (4 GB database files prior to SQL Server Express 2008 R2). It is intended as a replacement for MSDE. Two additional editions provide a superset of features not in the original Express Edition. The first is **SQL Server Express with Tools**, which includes SQL Server Management Studio Basic. **SQL Server Express with Advanced Services** adds full-text search capability and reporting services.

Specialized editions

Azure

Microsoft SQL Azure Database is the cloud-based version of Microsoft SQL Server, presented as software as a service on Azure Services Platform.

Compact (SQL CE)

The compact edition is an embedded database engine. Unlike the other editions of SQL Server, the SQL CE engine is based on SQL Mobile (initially designed for use with hand-held devices) and does not share the same binaries. Due to its small size (1 MB DLL footprint), it has a markedly reduced feature set compared to the other editions. For example, it supports a subset of the standard data types, does not support stored procedures or Views or multiple-statement batches (among other limitations). It is limited to 4 GB maximum database size and cannot be run as a Windows service, Compact Edition must be hosted by the application using it. The 3.5 version includes support for ADO.NET Synchronization Services. SQL CE does not support ODBC connectivity, unlike SQL Server proper.

Developer

SQL Server Developer Edition includes the same features as SQL Server 2012 Enterprise Edition, but is limited by the license to be only used as a development and test system, and not as production server. This edition is available to download by students free of charge as a part of Microsoft's DreamSpark program.

Embedded (SSEE)

SQL Server 2005 Embedded Edition is a specially configured named instance of the SQL Server Express database engine which can be accessed only by certain Windows Services.

Evaluation

SQL Server Evaluation Edition, also known as the *Trial Edition*, has all the features of the Enterprise Edition, but is limited to 180 days, after which the tools will continue to run, but the server services will stop.

Fast Track

SQL Server Fast Track is specifically for enterprise-scale data warehousing storage and business intelligence processing, and runs on reference-architecture hardware that is optimized for Fast Track.

LocalDB

Introduced in SQL Server Express 2012, LocalDB is a minimal, on-demand, version of SQL Server that is designed for application developers. It can also be used as an embedded database.

Parallel Data Warehouse (PDW)

A massively parallel processing (MPP) SQL Server appliance optimized for large-scale data warehousing such as hundreds of terabytes.

Datawarehouse Appliance Edition

Pre-installed and configured as part of an appliance in partnership with Dell & HP base on the Fast Track architecture. This edition does not include SQL Server Integration Services, Analysis Services, or Reporting Services.

Architecture

The protocol layer implements the external interface to SQL Server. All operations that can be invoked on SQL Server are communicated to it via a Microsoft-defined format, called Tabular Data Stream (TDS). TDS is an application layer protocol, used to transfer data between a database server and a client. Initially designed and developed by Sybase Inc. for their Sybase SQL Server relational database engine in 1984, and later by Microsoft in Microsoft SQL Server, TDS packets can be encased in other physical transport dependent protocols, including TCP/IP, Named pipes, and Shared memory. Consequently, access to SQL Server is available over these protocols. In addition, the SQL Server API is also exposed over web services.

Data storage

Data storage is a database, which is a collection of tables with typed columns. SQL Server supports different data types, including primary types such as *Integer*, *Float*, *Decimal*, *Char* (including character strings), *Varchar* (variable length character strings), binary (for unstructured blobs of data), *Text* (for textual data) among others. The rounding of floats to integers uses either Symmetric Arithmetic Rounding or Symmetric Round Down (*Fix*) depending on arguments: `SELECT Round(2.5, 0)` gives 3.

Microsoft SQL Server also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log. A SQL Server database can contain a maximum of 2^{31} objects, and can span multiple OS-level files with a maximum file size of 2^{60} bytes. The data in the database are stored in primary data files with an extension `.mdf`. Secondary data files, identified with a `.ndf` extension, are used to store optional metadata. Log files are identified with the `.ldf` extension.

Storage space allocated to a database is divided into sequentially numbered *pages*, each 8 KB in size. A *page* is the basic unit of I/O for SQL Server operations. A page is marked with a 96-byte header which stores metadata about the page including the page number, page type, free space on the page and the ID of the object that owns it. Page type defines the data contained in the page - data stored in the database, index, allocation map which holds information about how pages are allocated to tables and indexes, change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While page is the basic unit of an I/O operation, space is actually managed in terms of an *extent* which consists of 8 pages. A database object can either span all 8 pages in an extent ("uniform extent") or share an extent with up to 7 more objects ("mixed extent"). A row in a database table cannot span more than one page, so is limited to 8 KB in size. However, if the data exceeds 8 KB and the row contains *Varchar* or *Varbinary* data, the data in those columns are moved to a new page (or possibly a sequence of pages, called an *Allocation unit*) and replaced with a pointer to the data.

For physical storage of a table, its rows are divided into a series of partitions (numbered 1 to n). The partition size is user defined; by default all rows are in a single partition. A table is split into multiple partitions in order to spread a database over a cluster. Rows in each partition are stored in either B-tree or heap structure. If the table has an associated index to allow fast retrieval of rows, the rows are stored in-order according to their index values, with a B-tree providing the index. The data is in the leaf node of the leaves, and other nodes storing the index values for the

leaf data reachable from the respective nodes. If the index is non-clustered, the rows are not sorted according to the index keys. An indexed view has the same storage structure as an indexed table. A table without an index is stored in an unordered heap structure. Both heaps and B-trees can span multiple allocation units.

Buffer management

SQL Server buffers pages in RAM to minimize disc I/O. Any 8 KB page can be buffered in-memory, and the set of all pages currently buffered is called the buffer cache. The amount of memory available to SQL Server decides how many pages will be cached in memory. The buffer cache is managed by the *Buffer Manager*. Either reading from or writing to any page copies it to the buffer cache. Subsequent reads or writes are redirected to the in-memory copy, rather than the on-disc version. The page is updated on the disc by the Buffer Manager only if the in-memory cache has not been referenced for some time. While writing pages back to disc, asynchronous I/O is used whereby the I/O operation is done in a background thread so that other operations do not have to wait for the I/O operation to complete. Each page is written along with its checksum when it is written. When reading the page back, its checksum is computed again and matched with the stored version to ensure the page has not been damaged or tampered with in the meantime.

Concurrency and locking

SQL Server allows multiple clients to use the same database concurrently. As such, it needs to control concurrent access to shared data, to ensure data integrity—when multiple clients update the same data, or clients attempt to read data that is in the process of being changed by another client. SQL Server provides two modes of concurrency control: pessimistic concurrency and optimistic concurrency. When pessimistic concurrency control is being used, SQL Server controls concurrent access by using locks. Locks can be either shared or exclusive. Exclusive lock grants the user exclusive access to the data—no other user can access the data as long as the lock is held. Shared locks are used when some data is being read—multiple users can read from data locked with a shared lock, but not acquire an exclusive lock. The latter would have to wait for all shared locks to be released. Locks can be applied on different levels of granularity—on entire tables, pages, or even on a per-row basis on tables. For indexes, it can either be on the entire index or on index leaves. The level of granularity to be used is defined on a per-database basis by the database administrator. While a fine grained locking system allows more users to use the table or index simultaneously, it requires more resources. So it does not automatically turn into higher performing solution. SQL Server also includes two more lightweight mutual exclusion solutions—latches and spinlocks—which are less robust than locks but are less resource intensive. SQL Server uses them for DMVs and other resources that are usually not busy. SQL Server also monitors all worker threads that acquire locks to ensure that they do not end up in deadlocks—in case they do, SQL Server takes remedial measures, which in many cases is to kill one of the threads entangled in a deadlock and rollback the transaction it started. To implement locking, SQL Server contains the *Lock Manager*. The Lock Manager maintains an in-memory table that manages the database objects and locks, if any, on them along with other metadata about the lock. Access to any shared object is mediated by the lock manager, which either grants access to the resource or blocks it.

SQL Server also provides the optimistic concurrency control mechanism, which is similar to the multiversion concurrency control used in other databases. The mechanism allows a new version of a row to be created whenever the row is updated, as opposed to overwriting the row, i.e., a row is additionally identified by the ID of the transaction that created the version of the row. Both the old as well as the new versions of the row are stored and maintained, though the old versions are moved out of the database into a system database identified as `Tempdb`. When a row is in the process of being updated, any other requests are not blocked (unlike locking) but are executed on the older version of the row. If the other request is an update statement, it will result in two different versions of the rows—both of them will be stored by the database, identified by their respective transaction IDs.

Data retrieval

The main mode of retrieving data from an SQL Server database is querying for it. The query is expressed using a variant of SQL called T-SQL, a dialect Microsoft SQL Server shares with Sybase SQL Server due to its legacy. The query declaratively specifies what is to be retrieved. It is processed by the query processor, which figures out the sequence of steps that will be necessary to retrieve the requested data. The sequence of actions necessary to execute a query is called a query plan. There might be multiple ways to process the same query. For example, for a query that contains a join statement and a select statement, executing join on both the tables and then executing select on the results would give the same result as selecting from each table and then executing the join, but result in different execution plans. In such case, SQL Server chooses the plan that is expected to yield the results in the shortest possible time. This is called query optimization and is performed by the query processor itself.

SQL Server includes a cost-based query optimizer which tries to optimize on the cost, in terms of the resources it will take to execute the query. Given a query, then the query optimizer looks at the database schema, the database statistics and the system load at that time. It then decides which sequence to access the tables referred in the query, which sequence to execute the operations and what access method to be used to access the tables. For example, if the table has an associated index, whether the index should be used or not - if the index is on a column which is not unique for most of the columns (low "selectivity"), it might not be worthwhile to use the index to access the data. Finally, it decides whether to execute the query concurrently or not. While a concurrent execution is more costly in terms of total processor time, because the execution is actually split to different processors might mean it will execute faster. Once a query plan is generated for a query, it is temporarily cached. For further invocations of the same query, the cached plan is used. Unused plans are discarded after some time.

SQL Server also allows stored procedures to be defined. Stored procedures are parameterized T-SQL queries, that are stored in the server itself (and not issued by the client application as is the case with general queries). Stored procedures can accept values sent by the client as input parameters, and send back results as output parameters. They can call defined functions, and other stored procedures, including the same stored procedure (up to a set number of times). They can be selectively provided access to. Unlike other queries, stored procedures have an associated name, which is used at runtime to resolve into the actual queries. Also because the code need not be sent from the client every time (as it can be accessed by name), it reduces network traffic and somewhat improves performance. Execution plans for stored procedures are also cached as necessary.

SQL CLR

Microsoft SQL Server 2005 includes a component named **SQL CLR** ("Common Language Runtime") via which it integrates with .NET Framework. Unlike most other applications that use .NET Framework, SQL Server itself hosts the .NET Framework runtime, i.e., memory, threading and resource management requirements of .NET Framework are satisfied by SQLOS itself, rather than the underlying Windows operating system. SQLOS provides deadlock detection and resolution services for .NET code as well. With SQL CLR, stored procedures and triggers can be written in any managed .NET language, including C# and VB.NET. Managed code can also be used to define UDT's (user defined types), which can persist in the database. Managed code is compiled to CLI assemblies and after being verified for type safety, registered at the database. After that, they can be invoked like any other procedure. However, only a subset of the Base Class Library is available, when running code under SQL CLR. Most APIs relating to user interface functionality are not available.

When writing code for SQL CLR, data stored in SQL Server databases can be accessed using the ADO.NET APIs like any other managed application that accesses SQL Server data. However, doing that creates a new database session, different from the one in which the code is executing. To avoid this, SQL Server provides some enhancements to the ADO.NET provider that allows the connection to be redirected to the same session which already hosts the running code. Such connections are called context connections and are set by setting `context connection` parameter to `true` in the connection string. SQL Server also provides several other enhancements

to the ADO.NET API, including classes to work with tabular data or a single row of data as well as classes to work with internal metadata about the data stored in the database. It also provides access to the XML features in SQL Server, including XQuery support. These enhancements are also available in T-SQL Procedures in consequence of the introduction of the new XML Datatype (query,value,nodes functions).

Services

SQL Server also includes an assortment of add-on services. While these are not essential for the operation of the database system, they provide value added services on top of the core database management system. These services either run as a part of some SQL Server component or out-of-process as Windows Service and presents their own API to control and interact with them.

Service Broker

Used inside an instance, programming environment. For cross instance applications, Service Broker communicates over TCP/IP and allows the different components to be synchronized together, via exchange of messages. The Service Broker, which runs as a part of the database engine, provides a reliable messaging and message queuing platform for SQL Server applications.

Replication Services

SQL Server Replication Services are used by SQL Server to replicate and synchronize database objects, either in entirety or a subset of the objects present, across replication agents, which might be other database servers across the network, or database caches on the client side. Replication follows a publisher/subscriber model, i.e., the changes are sent out by one database server ("publisher") and are received by others ("subscribers"). SQL Server supports three different types of replication:

Transaction replication

> Each transaction made to the publisher database (master database) is synced out to subscribers, who update their databases with the transaction. Transactional replication synchronizes databases in near real time.

Merge replication

> Changes made at both the publisher and subscriber databases are tracked, and periodically the changes are synchronized bi-directionally between the publisher and the subscribers. If the same data has been modified differently in both the publisher and the subscriber databases, synchronization will result in a conflict which has to be resolved - either manually or by using pre-defined policies. rowguid needs to be configured on a column if merge replication is configured.

Snapshot replication

> Snapshot replication publishes a copy of the entire database (the then-snapshot of the data) and replicates out to the subscribers. Further changes to the snapshot are not tracked.

Analysis Services

SQL Server Analysis Services adds OLAP and data mining capabilities for SQL Server databases. The OLAP engine supports MOLAP, ROLAP and HOLAP storage modes for data. Analysis Services supports the XML for Analysis standard as the underlying communication protocol. The cube data can be accessed using MDX and LINQ queries. Data mining specific functionality is exposed via the DMX query language. Analysis Services includes various algorithms - Decision trees, clustering algorithm, Naive Bayes algorithm, time series analysis, sequence clustering algorithm, linear and logistic regression analysis, and neural networks - for use in data mining.

Reporting Services

SQL Server Reporting Services is a report generation environment for data gathered from SQL Server databases. It is administered via a web interface. Reporting services features a web services interface to support the development of custom reporting applications. Reports are created as RDL files.

Reports can be designed using recent versions of Microsoft Visual Studio (Visual Studio.NET 2003, 2005, and 2008) with Business Intelligence Development Studio, installed or with the included Report Builder. Once created, RDL files can be rendered in a variety of formats[10] including Excel, PDF, CSV, XML, TIFF (and other image formats),[11] and HTML Web Archive.

Notification Services

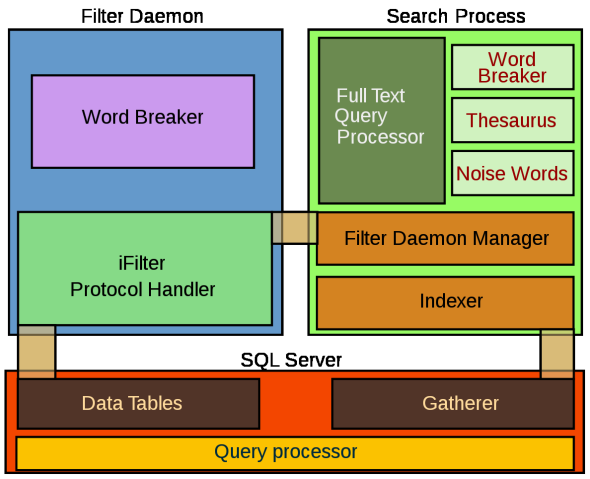
Originally introduced as a post-release add-on for SQL Server 2000, Notification Services was bundled as part of the Microsoft SQL Server platform for the first and only time with SQL Server 2005. SQL Server Notification Services is a mechanism for generating data-driven notifications, which are sent to Notification Services subscribers. A subscriber registers for a specific event or transaction (which is registered on the database server as a trigger); when the event occurs, Notification Services can use one of three methods to send a message to the subscriber informing about the occurrence of the event. These methods include SMTP, SOAP, or by writing to a file in the filesystem. Notification Services was discontinued by Microsoft with the release of SQL Server 2008 in August 2008, and is no longer an officially supported component of the SQL Server database platform.

Integration Services

SQL Server Integration Services (SSIS) provides ETL capabilities for SQL Server for data import, data integration and data warehousing needs. Integration Services includes GUI tools to build workflows such as extracting data from various sources, querying data, transforming data—including aggregation, de-duplication, de-/normalization and merging of data—and then exporting the transformed data into destination databases or files.

Full Text Search Service

SQL Server Full Text Search service is a specialized indexing and querying service for unstructured text stored in SQL Server databases. The full text search index can be created on any column with character based text data. It allows for words to be searched for in the text columns. While it can be performed with the SQL `LIKE` operator, using SQL Server Full Text Search service can be more efficient. Full allows for inexact matching of the source string, indicated by a *Rank* value which can range from 0 to 1000 - a higher rank means a more accurate match. It also allows linguistic matching ("inflectional search"), i.e., linguistic variants of a word (such as a verb in a different tense) will also be a match for a given word (but with a lower rank than an exact match). Proximity searches are also supported, i.e., if the words searched for do not occur in the sequence they are specified in the query but are near each other, they are also considered a match. T-SQL exposes special operators that can be used to access the FTS capabilities.



The SQL Server Full Text Search service architecture

The Full Text Search engine is divided into two processes - the *Filter Daemon* process (`msftefd.exe`) and the *Search* process (`msftesql.exe`). These processes interact with the SQL Server. The Search process includes the indexer (that creates the full text indexes) and the full text query processor. The indexer scans through text columns in the database. It can also index through binary columns, and use iFilters to extract meaningful text from the binary blob (for example, when a Microsoft Word document is stored as an unstructured binary file in a database). The iFilters are hosted by the Filter Daemon process. Once the text is extracted, the Filter Daemon process breaks it up into a sequence of words and hands it over to the indexer. The indexer filters out *noise words*, i.e., words like *A*, *And* etc., which occur frequently and are not useful for search. With the remaining words, an inverted index is created, associating each word with the columns they were found in. SQL Server itself includes a *Gatherer* component that monitors changes to tables and invokes the indexer in case of updates.

When a full text query is received by the SQL Server query processor, it is handed over to the FTS query processor in the Search process. The FTS query processor breaks up the query into the constituent words, filters out the noise words, and uses an inbuilt thesaurus to find out the linguistic variants for each word. The words are then queried against the inverted index and a rank of their accurateness is computed. The results are returned to the client via the SQL Server process.

SQLCMD

SQLCMD is a command line application that comes with Microsoft SQL Server, and exposes the management features of SQL Server. It allows SQL queries to be written and executed from the command prompt. It can also act as a scripting language to create and run a set of SQL statements as a script. Such scripts are stored as a `.sql` file, and are used either for management of databases or to create the database schema during the deployment of a database.

SQLCMD was introduced with SQL Server 2005 and this continues with SQL Server 2012 and 2014. Its predecessor for earlier versions was OSQL and ISQL, which is functionally equivalent as it pertains to TSQL execution, and many of the command line parameters are identical, although SQLCMD adds extra versatility.

Visual Studio

Microsoft Visual Studio includes native support for data programming with Microsoft SQL Server. It can be used to write and debug code to be executed by SQL CLR. It also includes a *data designer* that can be used to graphically create, view or edit database schemas. Queries can be created either visually or using code. SSMS 2008 onwards, provides intellisense for SQL queries as well.

SQL Server Management Studio

SQL Server Management Studio is a GUI tool included with SQL Server 2005 and later for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. SQL Server Management Studio replaces Enterprise Manager as the primary management interface for Microsoft SQL Server since SQL Server 2005. A version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as *SQL Server Management Studio Express* (SSMSE).

A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It can be used to visually observe and analyze query plans and optimize the database performance, among others. SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

Business Intelligence Development Studio

Business Intelligence Development Studio (**BIDS**) is the IDE from Microsoft used for developing data analysis and Business Intelligence solutions utilizing the Microsoft SQL Server Analysis Services, Reporting Services and Integration Services. It is based on the Microsoft Visual Studio development environment but is customized with the SQL Server services-specific extensions and project types, including tools, controls and projects for reports (using Reporting Services), Cubes and data mining structures (using Analysis Services).

Programmability

T-SQL

T-SQL (Transact-SQL) is the Secondary means of programming and managing SQL Server. It exposes keywords for the operations that can be performed on SQL Server, including creating and altering database schemas, entering and editing data in the database as well as monitoring and managing the server itself. Client applications that consume data or manage the server will leverage SQL Server functionality by sending T-SQL queries and statements which are then processed by the server and results (or errors) returned to the client application. SQL Server allows it to be managed using T-SQL. For this it exposes read-only tables from which server statistics can be read. Management functionality is exposed via system-defined stored procedures which can be invoked from T-SQL queries to perform the management operation. It is also possible to create linked Server using T-SQL. Linked server allows operation to multiple server as one query.

SQL Native Client (aka SNAC)

SQL Native Client is the native client side data access library for Microsoft SQL Server, version 2005 onwards. It natively implements support for the SQL Server features including the Tabular Data Stream implementation, support for mirrored SQL Server databases, full support for all data types supported by SQL Server, asynchronous operations, query notifications, encryption support, as well as receiving multiple result sets in a single database session. SQL Native Client is used under the hood by SQL Server plug-ins for other data access technologies, including ADO or OLE DB. The SQL Native Client can also be directly used, bypassing the generic data access layers.

On 28 Nov 2011 a preview release of the SQL Server ODBC driver for Linux was released.

Notes

[1] http://www.microsoft.com/sqlserver

[2] http://www.scriptcase.net/blog/all-about-the-history-of-sql-server/

[3] Multiple Active Result Sets (MARS) in SQL Server 2005 (http://msdn.microsoft.com/en-us/library/ms345109(SQL.90).aspx) (retrieved June 20, 2009)

[4] Dynamic Management Views and Functions (http://msdn.microsoft.com/en-us/library/ms188754.aspx) (retrieved June 06, 2010)

[5] Database Mirroring was included in the first release of SQL Server 2005 for evaluation purposes only. Prior to SP1, it was not enabled by default, and was not supported by Microsoft.

[6] Announced to the SQL Server Special Interest Group at the ESRI 2008 User's Conference on August 6, 2008 by Ed Katibah (Spatial Program Manager at Microsoft)

[7] http://blogs.msdn.com/b/data__knowledge__intelligence/archive/2013/10/17/sql-server-2014-ctp2-is-now-available.aspx

[8] http://www.microsoft.com/en-us/sqlserver/sql-server-2014.aspx

[9] http://download.microsoft.com/download/4/F/7/4F74E127-827E-420D-971F-53CECB6778BD/SQL_Server_2012_Licensing_Datasheet_and_FAQ_Mar2012.docx

[10] MSDN Library: Reporting Services Render Method (http://msdn.microsoft.com/en-us/library/microsoft.wssux.reportingserviceswebservice.rsexecutionservice2005.reportexecutionservice.render.aspx) - See Device Information Settings (http://msdn.microsoft.com/en-us/library/ms155397.aspx)

[11] Image Device Information Settings (http://msdn.microsoft.com/en-us/library/ms155373.aspx) - SSRS can render BMP, EMF, GIF, JPEG, PNG, and TIFF.

References

Further reading

- Lance Delano, Rajesh George et al. (2005). *Wrox's SQL Server 2005 Express Edition Starter Kit (Programmer to Programmer)*. *Microsoft Press*. ISBN 0-7645-8923-7
- Delaney, Kalen, et al. (2007). *Inside SQL Server 2005: Query Tuning and Optimization*. *Microsoft Press*. ISBN 0-7356-2196-9.
- Ben-Gan, Itzik, et al. (2006). *Inside Microsoft SQL Server 2005: T-SQL Programming*. *Microsoft Press*. ISBN 0-7356-2197-7.

External links

- Official website (http://technet.microsoft.com/en-us/sqlserver/default)
- Longer version of the history from a dev team member (http://blogs.msdn.com/euanga/archive/2006/01/19/514479.aspx)

Microsoft SQL Server Master Data Services

Microsoft SQL Server Master Data Services is a Master Data Management (MDM) product from Microsoft, which will ship as a part of the Microsoft SQL Server database. Originally code-named **Bulldog**, Master Data Services is the rebranding of the Stratature MDM product titled **+EDM**, which Microsoft acquired in June 2007. Master Data Services is architecturally similar to +EDM, with increased integration with other Microsoft applications as well as some new features. Master Data Services first shipped with Microsoft SQL Server 2008 R2. Microsoft SQL Server 2012 improved the analytical capabilities, transactional capabilities, and integration with other software.

Overview

Like other MDM products, Master Data Services aims to create a centralized data source and keep it synchronized, and thus reduce redundancies, across the applications which process the data.

Sharing the architectural core with Stratature +EDM, Master Data Services uses a Microsoft SQL Server database as the physical data store. It is a part of the *Master Data Hub*, which uses the database to store and manage data entities. It is a database with the software to validate and manage the data, and keep it synchronized with the systems that use the data. The master data hub has to extract the data from the source system, validate, sanitize and shape the data, remove duplicates, and update the hub repositories, as well as synchronize the external sources. The entity schemas, attributes, data hierarchies, validation rules and access control information are specified as metadata to the Master Data Services runtime. Master Data Services does not impose any limitation on the data model. Master Data Services also allows custom *Business rules*, used for validating and sanitizing the data entering the data hub, to be defined, which is then run against the data matching the specified criteria. All changes made to the data are validated against the rules, and a log of the transaction is stored persistently. Violations are logged separately, and optionally the owner is notified, automatically. All the data entities can be versioned.

Master Data Services allows the master data to be categorized by hierarchical relationships, such as employee data are a subtype of organization data. Hierarchies are generated by relating data attributes. Data can be automatically categorized using rules, and the categories are introspected programmatically. Master Data Services can also expose the data as Microsoft SQL Server views, which can be pulled by any SQL-compatible client. It uses a role-based access control system to restrict access to the data. The views are generated dynamically, so they contain the latest

data entities in the master hub. It can also push out the data by writing to some external journals. Master Data Services also includes a web-based UI for viewing and managing the data. It uses AJAX in the front-end and ASP.NET in the back-end.

Master Data Services also includes certain features not available in the Stratature +EDM product. It gains an Web service interface to expose the data, as well as an API, which internally uses the exposed web services, exposing the feature set, programmatically, to access and manipulate the data. It also integrates with Active Directory for authentication purposes. Unlike +EDM, Master Data Services supports Unicode characters, as well as support multilingual user interfaces.

Terminology

- *Model* is the highest level of an MDS instance. It is the primary container for specific groupings of master data. In many ways it is very similar to the idea of a database.
- *Entities* are containers created within a model. Entities provide a home for members, and are in many ways analogous to database tables. (e.g. Customer)
- *Members* are analogous to the records in a database table (Entity) e.g. Will Smith. Members are contained within entities. Each member is made up of two or more attributes.
- *Attributes* are analogous to the columns within a database table (Entity) e.g. Surname. Attributes exist within entities and help describe members (the records within the table). Name and Code attributes are created by default for each entity and serve to describe and uniquely identify leaf members. Attributes can be related to other attributes from other entities which are called 'domain-based' attributes. Other attributes will be of type 'free-form' (most common) or 'file'.

Master data service is the part of the sql server now the question is what are master data service

- *Attribute Groups* are explicitly defined collections of particular attributes. Say you have an entity "customer" that has 50 attributes — too much information for many of your users. Attribute groups enable the creation of custom sets of hand-picked attributes that are relevant for specific audiences. (e.g. "customer - delivery details" that would include just their name and last known delivery address)
- *Hierarchies* organise members into either Derived or Explicit hierarchical structures. Derived hierarchies, as the name suggests, are derived by the MDS engine based on the relationships that exist between attributes. Explicit hierarchies are created by hand using both leaf and consolidated members.
- *Business Rules* can be created and applied against model data to ensure that custom business logic is adhered to. In order to be committed into the system data must pass all business rule validations applied to them. e.g. Within the Customer Entity you may want to create a business rule that ensures all members of the 'Country' Attribute contain either the text "USA" or "Canada". The Business Rule once created and ran will then verify all the data is correct before it accepts it into the approved model.
- *Versions* provide system owners / administrators with the ability to Open, Lock or Commit a particular version of a model and the data contained within it at a particular point in time. As the content within a model varies, grows or shrinks over time versions provide a way of managing metadata so that subscribing systems can access to the correct content.

References

External links

- Microsoft SQL Server 2008 R2 Master Data Services (http://msdn.microsoft.com/en-us/sqlserver/ff943581.aspx)

Transact-SQL

Transact-SQL (**T-SQL**) is Microsoft's and Sybase's proprietary extension to SQL. SQL, the acronym for Structured Query Language, is a standardized computer language that was originally developed by IBM for querying, altering and defining relational databases, using declarative statements. T-SQL expands on the SQL standard to include procedural programming, local variables, various support functions for string processing, date processing, mathematics, etc. and changes to the DELETE and UPDATE statements. These additional features make Transact-SQL Turing complete.

Transact-SQL is central to using Microsoft SQL Server. All applications that communicate with an instance of SQL Server do so by sending Transact-SQL statements to the server, regardless of the user interface of the application.

Variables

Keywords for flow control in Transact-SQL include `BEGIN` and `END`, `BREAK`, `CONTINUE`, `GOTO`, `IF` and `ELSE`, `RETURN`, `WAITFOR`, and `WHILE`.

`IF` and `ELSE` allow conditional execution. This batch statement will print "It is the weekend" if the current date is a weekend day, or "It is a weekday" if the current date is a weekday. (Note: This code assumes that Sunday is configured as the first day of the week in the `@@DATEFIRST` setting.)

```
IF DATEPART(dw, GETDATE()) = 7 OR DATEPART(dw, GETDATE()) = 1
   PRINT 'It is the weekend.'
ELSE
   PRINT 'It is a weekday.'
```

`BEGIN` and `END` mark a block of statements. If more than one statement is to be controlled by the conditional in the example above, we can use `BEGIN` and `END` like this:

```
IF DATEPART(dw, GETDATE()) = 7 OR DATEPART(dw, GETDATE()) = 1
BEGIN
   PRINT 'It is the weekend.'
   PRINT 'Get some rest on the weekend!'
END
ELSE
BEGIN
   PRINT 'It is a weekday.'
   PRINT 'Get to work on a weekday!'
END
```

`WAITFOR` will wait for a given amount of time, or until a particular time of day. The statement can be used for delays or to block execution until the set time.

`RETURN` is used to immediately return from a stored procedure or function.

`BREAK` ends the enclosing `WHILE` loop, while `CONTINUE` causes the next iteration of the loop to execute. An example of a `WHILE` loop is given below.

```
DECLARE @i INT
SET @i = 0

WHILE @i < 5
BEGIN
   PRINT 'Hello world.'
   SET @i = @i + 1
END
```

Changes to DELETE and UPDATE statements

In Transact-SQL, both the `DELETE` and `UPDATE` statements allow a `FROM` clause to be added, which allows joins to be included.

This example deletes all `users` who have been flagged with the 'Idle' flag.

```
DELETE u
  FROM users AS u
  INNER JOIN user_flags AS f
    ON u.id = f.id
WHERE f.name = 'idle'
```

BULK INSERT

`BULK INSERT` is a Transact-SQL statement that implements a bulk data-loading process, inserting multiple rows into a table, reading data from an external sequential file. Use of `BULK INSERT` results in better performance than processes that issue individual `INSERT` statements for each row to be added. Additional details are available in MSDN [1].

TRY CATCH

Beginning with SQL Server 2005, Microsoft introduced additional `TRY CATCH` logic to support exception type behaviour. This behaviour enables developers to simplify their code and leave out `@@ERROR` checking after each SQL execution statement.

```
-- begin transaction
BEGIN TRAN

BEGIN TRY
   -- execute each statement
   INSERT INTO MYTABLE(NAME) VALUES ('ABC')
   INSERT INTO MYTABLE(NAME) VALUES ('123')

   -- commit the transaction
   COMMIT TRAN
END TRY
BEGIN CATCH
   -- rollback the transaction because of error
```

```
   ROLLBACK TRAN
END CATCH
```

References

[1] http://msdn2.microsoft.com/en-us/library/ms188365.aspx

External links

- Sybase Transact-SQL User's Guide (http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.sqlug/html/sqlug/title.htm)
- Transact-SQL Reference for SQL Server 2000 (MSDN) (http://msdn2.microsoft.com/en-us/library/aa260642(SQL.80).aspx)
- Transact-SQL Reference for SQL Server 2005 (MSDN) (http://msdn2.microsoft.com/en-us/library/ms189826.aspx)
- Transact-SQL Reference for SQL Server 2008 (MSDN) (http://msdn.microsoft.com/en-us/library/bb510741(SQL.100).aspx)
- Transact-SQL Reference for SQL Server 2012 (MSDN) (http://msdn.microsoft.com/en-us/library/bb510741.aspx)
- Transact-SQL Tutorial (http://www.tsql.info)

WCF Data Services

WCF Data Services (formerly **ADO.NET Data Services**, codename **"Astoria"**) is a platform for what Microsoft calls *Data Services*. It is actually a combination of the runtime and a web service through which the services are exposed. In addition, it also includes the **Data Services Toolkit** which lets Astoria Data Services be created from within ASP.NET itself. The Astoria project was announced at MIX 2007, and the first developer preview was made available on April 30, 2007. The first CTP was made available as a part of the ASP.NET 3.5 Extensions Preview. The final version was released as part of Service Pack 1 of the .NET Framework 3.5 on August 11, 2008. The name change from ADO.NET Data Services to WCF data Services was announced at the 2009 PDC.

Overview

WCF Data Services exposes data, represented as Entity Data Model (EDM) objects, via web services accessed over HTTP. The data can be addressed using a REST-like URI. The data service, when accessed via the HTTP GET method with such a URI, will return the data. The web service can be configured to return the data in either plain XML, JSON or RDF+XML. In the initial release, formats like RSS and ATOM are not supported, though they may be in the future. In addition, using other HTTP methods like PUT, POST or DELETE, the data can be updated as well. POST can be used to create new entities, PUT for updating an entity, and DELETE for deleting an entity.

The URIs representing the data will contain the physical location of the service, as well as the service name. In addition, it will also need to specify an EDM Entity-Set or a specific entity instance, as in respectively

```
http://dataserver/service.svc/MusicCollection
```

or

```
http://dataserver/service.svc/MusicCollection[SomeArtist]
```

The former will list all entities in the *Collection* set whereas the latter will list only for the entity which is indexed by *SomeArtist*.

In addition, the URIs can also specify a traversal of a relationship in the Entity Data Model. For example,

```
http://dataserver/service.svc/MusicCollection[SomeSong]/Genre
```

traverses the relationship *Genre* (in SQL parlance, joins with the *Genre* table) and retrieves all instances of *Genre* that are associated with the entity *SomeSong*. Simple predicates can also be specified in the URI, like

```
http://dataserver/service.svc/MusicCollection[SomeArtist]/ReleaseDate[Year eq 2006]
```

will fetch the items that are indexed by *SomeArtist* and had their *release* in *2006*. Filtering and partition information can also be encoded in the URL as

```
http://dataserver/service.svc/MusicCollection?$orderby=ReleaseDate&$skip=100&$top=50
```

It is important to note that although the presence of skip and top keywords indicate paging support, in Data Services version 1 there is no method of determining the number of records available and thus impossible to determine how many pages there may be. The OData 2.0 spec adds support for the **$count** path segment (to return just a count of entities) and **$inlineCount** (to retrieve a page worth of entities and a total count without a separate round-trip....).[1]

References

[1] http://msdn.microsoft.com/en-us/library/ee373845.aspx

- "Codename "Astoria": Data Services for the Web" (http://blogs.msdn.com/pablo/archive/2007/04/30/codename-astoria-data-services-for-the-web.aspx). Retrieved 2007-04-30.
- ADO.NET Data Services Framework (formerly "Project Astoria") (http://astoria.mslivelabs.com/)

External links

- Using Microsoft ADO.NET Data Services (http://msdn.microsoft.com/en-us/library/cc907912.aspx)
- ASP.NET 3.5 Extensions Preview (http://www.asp.net/downloads/3.5-extensions/)
- ADO.NET Data Services (Project Astoria) Team Blog (http://blogs.msdn.com/astoriateam/)
- Access Cloud Data with Astoria: ENT News Online (http://entmag.com/news/article.asp?EditorialsID=9105)

Windows Internal Database

Windows Internal Database (codenamed WYukon, sometimes referred to as SQL Server Embedded Edition) is a variant of SQL Server Express 2005-2012 that is included with Windows Server 2008 (SQL 2005), Windows Server 2008 R2 (SQL 2005) and Windows Server 2012 (SQL 2012), and is included with other free Microsoft products released after 2007 that require an SQL Server database backend. Windows SharePoint Services 3.0 and Windows Server Update Services 3.0 both include Windows Internal Database, which can be used as an alternative to using a retail edition of SQL Server. WID was a 32-bit application, even as component of Windows Server 2008 64-bit, which installs in the path C:\Program Files (x86)\Microsoft SQL Server. In Windows Server 2012, it is a 64-bit application, installed in C:\Windows\WID.

Windows Internal Database is not available as a standalone product for use by end-user applications; Microsoft provides SQL Server Express and Microsoft SQL Server for this purpose. Additionally, it is designed to only be accessible to Windows Services running on the same machine.

Several components of Windows Server 2008 and 2012 use Windows Internal Database for their data storage: Active Directory Rights Management Services, Windows System Resource Manager, UDDI Services, Active Directory Federation Services 2.0, IPAM and Windows SharePoint Services. On Windows Server 2003, SharePoint and Windows Server Update Services will install Windows Internal Database and use it as a default data store if a retail SQL Server database instance is not provided. A Knowledge Base article published by Microsoft states that Windows Internal Database does not identify itself as a removable component, and provides instructions how it may be uninstalled by calling Windows Installer directly.

SQL Server Management Studio Express can be used to connect to an instance of Windows Internal Database using `\\.\pipe\MSSQL$MICROSOFT##SSEE\sql\query` (2003-2008) or `\\.\pipe\MICROSOFT##WID\tsql\query` (2012) as instance name. But this will only work locally, as Remote Connections cannot be enabled for this edition of SQL Server. *Also note that "Windows Authentication" should be used (as opposed to SQL Server Authentication), and administrators seem to have the best results of authenticating successfully when logged on using the same administrative account that was created when Windows was installed.*

References

External links

- Planning and Architecture for Windows SharePoint Services 3.0 Technology (http://go.microsoft.com/fwlink/?LinkId=79600)
- Release Notes for Microsoft Windows Server Update Services 3.0 (http://go.microsoft.com/fwlink/?LinkId=71220)
- http://www.mssqltips.com/tip.asp?tip=1577 (http://www.mssqltips.com/tip.asp?tip=1577)

SQL Server Agent

SQL Server Agent is a component of Microsoft SQL Server which schedules jobs and handles other automated tasks. It runs as a Windows service so can start automatically when the system boots or it can be started manually. It is .

Typical system tasks performed include scheduling maintenance plans (such as backups), handling Reporting Services subscriptions and performing log shipping sub-tasks (backup, copy, restore & check). User tasks, such as scheduling some T-SQL or command line statement are also common.

SQLAgent has support for operators and alerts, so that administrators can be notified, e.g. by email.

SQL Server Compact

SQL Server Compact

Filename extension	`.sdf`
Developed by	Microsoft
Type of format	Relational database

Microsoft SQL Server Compact (**SQL CE**) is a compact relational database produced by Microsoft for applications that run on mobile devices and desktops. Prior to the introduction of the desktop platform, it was known as *SQL Server for Windows CE* and *SQL Server Mobile Edition*. The latest release is the SQL Server Compact 4.0 supporting .NET Framework 4.0, and dropping support for Windows Mobile in this release. It includes both 32-bit and 64-bit native support. SQL CE targets occasionally connected applications and applications with an embedded database. It is free to download and redistribute. An ODBC driver for SQL CE does not exist, nor is one planned. Native applications may use SQL CE via OLE DB.

Overview

SQL Server Compact shares a common API with the other Microsoft SQL Server editions. It also includes ADO.NET providers for data access using ADO.NET APIs, and built-in synchronization capabilities, as well as support for LINQ and Entity Framework. Future releases will unify the synchronization capabilities with Microsoft Synchronization Services. Unlike other editions of Microsoft SQL Server, SQL CE runs in-process with the application which is hosting it. It has a disk footprint of less than 2 MB and a memory footprint of approximately 5 MB. SQL CE is optimized for an architecture where all applications share the same memory pool. Windows Store apps for Windows 8 cannot use SQL Server Compact edition, or any other edition of SQL Server.

Support

SQL CE databases can support ACID-compliance, but do not meet the durability requirement by default because AutoFlush buffers changes in memory (including enlisted ambient transactions and explicit SQL CE transactions that do not override the Commit() call with an CommitMode.Immediate value). Therefore committed transaction changes can be lost. To meet the durability requirement the commit call on the transaction must specify the immediate flag. Like Microsoft SQL Server, SQL CE supports transactions, referential integrity constraints, locking as well as multiple connections to the database store. However, nested transactions are not supported, even though parallel transactions (on different tables) are. The current release does not support stored procedures or native XML data type either. It uses a subset of T-SQL for querying and due to lack of XML support, XQuery is not supported either. Queries are processed by an optimizing query processor. SQL CE databases also support indexing, as well as support remote data replication (local caching of data in remote databases) and merge replication (bidirectional synchronization with master databases).

SQL CE databases can be created and managed from Microsoft Visual Studio and some older versions of SQL Server Management Studio as well.

File Format

SQL CE databases reside in a single ***.sdf*** file, which can be up to 4 GB in size. The *.sdf* file can be encrypted with 128-bit encryption for data security. SQL CE runtime mediates concurrent multi-user access to the *.sdf* file. The *.sdf* file can simply be copied to the destination system for deployment, or be deployed through ClickOnce. SQL CE runtime has support for **DataDirectories**. Applications using an SQL CE database need not specify the entire path to an *.sdf* file in the ADO.NET connection string, rather it can be specified as |*DataDirectory*|\\<*database_name*>.*sdf*, defining the data directory (where the *.sdf* database file resides) being defined in the assembly manifest for the application.

SQL Server Management Studio 2005 can read and modify CE 3.0 and 3.1 database files (with the latest service pack), but SQL Server Management Studio 2008 (or later) is required to read version 3.5 files. Microsoft Visual Studio Express 2008 SP1 can create, modify and query CE 3.5 SP1 database files. SQL Server Management Studio cannot read CE 4.0 files. Visual Studio 2010 SP1 can handle CE 4.0 database files.

The .sdf (Sqlce Database File) naming convention is optional and any extension can be used.

Setting a password for the database file is optional. The database can be compressed and repaired with the option of the compacted/repaired database to be placed into a new database file.

References

External links

- SQL Server Compact 3.5 (http://www.microsoft.com/sqlserver/2008/en/us/compact.aspx)
- SQL Server Compact Release Versions (http://blogs.msdn.com/sqlservercompact/archive/2008/02/08/sql-server-compact-release-versions.aspx)
- Microsoft's Embedded Database - SQL Server Compact - Team Blog (http://blogs.msdn.com/b/sqlservercompact)

SQL CLR

SQL CLR or **SQLCLR** (SQL Common Language Runtime) is technology for hosting of the Microsoft .NET common language runtime engine within SQL Server. The SQLCLR allows managed code to be hosted by, and run in, the Microsoft SQL Server environment.

This technology, introduced in Microsoft SQL Server 2005, allow users for example to create the following types of managed code objects in SQL Server in .NET languages such as C# or VB.NET.

- Stored procedures (SPs) which are analogous to *procedures* or *void functions* in procedural languages like VB or C,
- triggers which are stored procedures that fire in response to Data Manipulation Language (DML) or Data Definition Language (DDL) events,
- User-defined functions (UDFs) which are analogous to functions in procedural languages,
- User-defined aggregates (UDAs) which allow developers to create custom aggregates that act on sets of data instead of one row at a time,
- User-defined types (UDTs) that allow users to create simple or complex data types which can be serialized and deserialized within the database.

The SQL CLR relies on the creation, deployment, and registration of CLI assemblies, which are physically stored in managed code dynamic load libraries (DLLs). These assemblies may contain CLI namespaces, classes, functions and properties.

External links

- MSDN: Using CLR Integration in SQL Server 2005 [1]
- MSDN Forum on .NET Framework in SQL Server [2]
- SqlClr.net Independent site [3]
- SQL CLR Team Blog (No posts since 2006, might be dead) [4]

References

[1] http://msdn2.microsoft.com/en-us/library/ms345136.aspx
[2] http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=86&SiteID=1
[3] http://www.sqlclr.net/
[4] http://blogs.msdn.com/sqlclr/

Microsoft Transaction Server

Microsoft Transaction Server (**MTS**) was software that provided services to Component Object Model (COM) software components, to make it easier to create large distributed applications. The major services provided by MTS were automated transaction management, instance management (or *just-in-time activation*) and role-based security. MTS is considered to be the first major software to implement aspect-oriented programming.

MTS was first offered in the Windows NT 4.0 Option Pack. In Windows 2000, MTS was enhanced and better integrated with the operating system and COM, and was renamed COM+. COM+ added object pooling, loosely-coupled events and user-defined simple transactions (compensating resource managers) to the features of MTS.

COM+ is still provided with Windows Server 2003 and Windows Server 2008, and the Microsoft .NET Framework provides a wrapper for COM+ in the EnterpriseServices namespace. The Windows Communication Foundation (WCF) provides a way of calling COM+ applications with web services. However, COM+ is based on COM, and Microsoft's strategic software architecture is now web services and .NET, not COM. There are pure .NET-based alternatives for many of the features provided by COM+, and in the long term it is likely COM+ will be phased out.

Architecture

A basic MTS architecture comprises:

- the MTS Executive (mtxex.dll)
- the Factory Wrappers and Context Wrappers for each component
- the MTS Server Component
- MTS clients
- auxiliary systems like:
  - COM runtime services
  - the Service Control Manager (SCM)
  - the Microsoft Distributed Transaction Coordinator (MS-DTC)
  - the Microsoft Message Queue (MSMQ)
  - the COM-Transaction Integrator (COM-TI)
  - etc.

COM components that run under the control of the MTS Executive are called MTS components. In COM+, they are referred to as COM+ Applications. MTS components are in-process DLLs. MTS components are deployed and run in the MTS Executive which manages them. As with other COM components, an object implementing the IClassFactory interface serves as a Factory Object to create new instances of these components.

MTS inserts a Factory Wrapper Object and an Object Wrapper between the actual MTS object and its client. This interposing of wrappers is called *interception*. Whenever the client makes a call to the MTS component, the wrappers (Factory and Object) intercept the call and inject their own instance-management algorithm called the Just-In-Time Activation (JITA) into the call. The wrapper then makes this call on the actual MTS component. Interception was considered difficult at the time due to a lack of extensible metadata.

In addition, based on the information from the component's deployment properties, transaction logic and security checks also take place in these wrapper objects.

For every MTS-hosted object, there also exists a Context Object, which implements the IObjectContext interface. The Context Object maintains specific information about that object, such as its transactional information, security information and deployment information. Methods in the MTS component call into the Context Object through its IObjectContext interface.

MTS does not create the actual middle-tier MTS object until the call from a client reaches the container. Since the object is not running all the time, it does not use up a lot of system resources (even though an object wrapper and skeleton for the object do persist).

As soon as the call comes in from the client, the MTS wrapper process activates its Instance Management algorithm called JITA. The actual MTS object is created "just in time" to service the request from the wrapper. And when the request is serviced and the reply is sent back to the client, the component either calls SetComplete()/SetAbort(), or its transaction ends, or the client calls Release() on the reference to the object, and the actual MTS object is destroyed. In short, MTS uses a stateless component model.

Generally, when a client requests services from a typical MTS component, the following sequence occurs on the server :

1. acquire a database connection
2. read the component's state from either the Shared Property Manager or from an already existing object or from the client
3. perform the business logic
4. write the component's changed state, if any, back to the database
5. close and release the database connection
6. *vote* on the result of the transaction. MTS components do not directly commit transactions, rather they communicate their success or failure to MTS.

It is thus possible to implement high-latency resources as asynchronous resource pools, which should take advantage of the stateless JIT activation afforded by the middleware server.

References

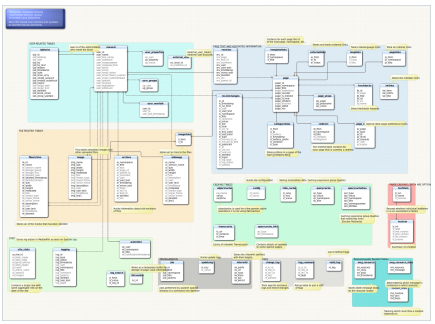
External links and references

- More details about MTS (http://my.execpc.com/~gopalan/mts/mts.html)
- Quick Tour of Microsoft Transaction Server (http://www.microsoft.com/technet/archive/transsrv/quicktr.mspx)
- Interpreting the MTS events in the event log (http://support.microsoft.com/kb/q262187/)

Oracle

Database schema

A **database schema** (/ˈski.mə/ ***SKEE**-mə*) of a database system is its structure described in a formal language supported by the database management system (DBMS) and refers to the organization of data as a blueprint of how a database is constructed (divided into database tables in case of Relational Databases). The formal definition of database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real world entities are modeled in the database.



A depiction of MediaWiki database schema.

"A database schema specifies, based on the database administrator's knowledge of possible applications, the facts that can enter the database, or those of interest to the possible end-users." The notion of a database schema plays the same role as the notion of theory in predicate calculus. A model of this “theory” closely corresponds to a database, which can be seen at any instant of time as a mathematical object. Thus a schema can contain formulas representing integrity constraints specifically for an application and the constraints specifically for a type of database, all expressed in the same database language. In a relational database, the schema defines the tables, fields, relationships, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views, synonyms, database links, directories, XML schemas, and other elements.

Schemas are generally stored in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure. In other words, schema is the structure of the database that defines the objects in the database.

In an Oracle Database system, the term "schema" has a slightly different connotation.

Ideal requirements for schema integration

Completeness

All information in the source data should be included in the database schema.

Overlap preservation

Each of the overlapping elements specified in the input mapping is also in a database schema relation.

Extended overlap preservation

Source-specific elements that are associated with a source's overlapping elements are passed through to the database schema.

Normalization

Independent entities and relationships in the source data should not be grouped together in the same relation in the database schema. In particular, source specific schema elements should not be grouped with overlapping schema elements, if the grouping co-locates independent entities or relationships.

Minimality

If any elements of the database schema are dropped then the database schema is not ideal.

These requirements influence the detailed structure of schemas that are produced. Certain applications will not require that all of these conditions are met, but these five requirements are the most ideal.

Example of two schema integrations

Example: Suppose we want a mediated (database) schema to integrate two travel databases, Go-travel and Ok-travel.

***Go-travel* has three relations:**

Go-flight(f-num, time, meal)
Go-price(f-num, date, price)

f-num is the flight number and meal is a boolean.

***Ok-travel* has just one relation:**

Ok-flight(f-num, date, time, price, nonstop)

'nonstop' is a boolean.

The overlapping information in Ok-travel's and Go-travel's schemas could be represented in a mediated schema:

Flight(f-num, date, time, price)

Oracle database specificity

In the context of Oracle databases, a **schema object** is a logical data storage structure.

In an Oracle database, associated with each database **user** is a schema. A schema comprises a collection of schema objects. Examples of schema objects include:

- tables
- views
- sequences
- synonyms
- indexes
- clusters
- database links
- snapshots
- procedures
- functions
- packages

On the other hand, non-schema objects may include:

- users
- roles
- contexts
- directory objects

Schema objects do not have a one-to-one correspondence to physical files on disk that store their information. However, Oracle databases store schema objects logically within a tablespace of the database. The data of each object is physically contained in one or more of the tablespace's datafiles. For some objects (such as tables, indexes, and clusters) a database administrator can specify how much disk space the Oracle RDBMS allocates for the object within the tablespace's datafiles.

There is no necessary relationship between schemas and tablespaces: a tablespace can contain objects from different schemas, and the objects for a single schema can reside in different tablespaces.

References

External links

- http://www.databaseanswers.org/data_models/
- http://weblogs.asp.net/scottgu/archive/2006/07/12/Tip_2F00_Trick_3A00_-Online-Database-Schema-Samples-Library.aspx
- http://msdn.microsoft.com/en-us/library/bb187299%28SQL.80%29.aspx
- http://www.ciobriefings.com/Publications/WhitePapers/DesigningtheStarSchemaDatabase/tabid/101/Default.aspx

Oracle Database

Oracle Database

ORACLE®	
Developer(s)	Oracle Corporation
Development status	Active
Written in	Assembly language, C, C++
Available in	Multilingual
Type	ORDBMS
License	Proprietary
Website	Oracle RDBMS [1]

The **Oracle Database** (commonly referred to as **Oracle RDBMS** or simply as **Oracle**) is an object-relational database management system produced and marketed by Oracle Corporation.

Larry Ellison and his friends, former co-workers Bob Miner and Ed Oates, started the consultancy Software Development Laboratories (SDL) in 1977. SDL developed the original version of the Oracle software. The name *Oracle* comes from the code-name of a CIA-funded project Ellison had worked on while previously employed by Ampex.

Physical and logical structures

An Oracle database system—identified by an alphanumeric system identifier or SID—comprises at least one instance of the application, along with data storage. An instance—identified persistently by an instantiation number (or activation id: SYS.V_$DATABASE.ACTIVATION#)—comprises a set of operating-system processes and memory-structures that interact with the storage. (Typical processes include PMON (the process monitor) and SMON (the system monitor).) Oracle documentation can refer to an active database instance as a "shared memory realm".

Users of Oracle databases refer to the server-side memory-structure as the SGA (System Global Area). The SGA typically holds cache information such as data-buffers, SQL commands, and user information. In addition to storage, the database consists of online redo logs (or logs), which hold transactional history. Processes can in turn archive the online redo logs into archive logs (offline redo logs), which provide the basis (if necessary) for data recovery and for the physical-standby forms of data replication using Oracle Data Guard.

If the Oracle database administrator has implemented Oracle RAC (Real Application Clusters), then multiple instances, usually on different servers, attach to a central storage array. This scenario offers advantages such as better performance, scalability and redundancy. However, support becomes more complex, and many sites do not use RAC. In version 10*g*, grid computing introduced shared resources where an instance can use (for example) CPU resources from another node (computer) in the grid.

The Oracle DBMS can store and execute stored procedures and functions within itself. PL/SQL (Oracle Corporation's proprietary procedural extension to SQL), or the object-oriented language Java can invoke such code objects and/or provide the programming structures for writing them.

Storage

The Oracle RDBMS stores data logically in the form of tablespaces and physically in the form of data files ("datafiles"). Tablespaces can contain various types of memory segments, such as Data Segments, Index Segments, etc. Segments in turn comprise one or more extents. Extents comprise groups of contiguous data blocks. Data blocks form the basic units of data storage.

A DBA can impose maximum quotas on storage per user within each tablespace.

Partitioning

Newer versions of the database can also include a partitioning feature: this allows the partitioning of tables based on different set of keys. Specific partitions can then be easily added or dropped to help manage large data sets.

Monitoring

Oracle database management tracks its computer data storage with the help of information stored in the `SYSTEM` tablespace. The `SYSTEM` tablespace contains the data dictionary—and often (by default) indexes and clusters. A data dictionary consists of a special collection of tables that contains information about all user-objects in the database. Since version 8*i*, the Oracle RDBMS also supports "locally managed" tablespaces which can store space management information in bitmaps in their own headers rather than in the `SYSTEM` tablespace (as happens with the default "dictionary-managed" tablespaces). Version 10*g* and later introduced the `SYSAUX` tablespace which contains some of the tables formerly stored in the `SYSTEM` tablespace, along with objects for other tools such as OEM which previously required its own tablespace.

Disk files

Disk files primarily represent one of the following structures:

- Data and index files: These files provide the physical storage of data, which can consist of the data-dictionary data (associated to the tablespace SYSTEM), user data, or index data. These files can be managed manually or managed by Oracle itself ("Oracle-managed files"). Note that a datafile has to belong to exactly one tablespace, whereas a tablespace can consist of multiple datafiles.
- Redo log files, consisting of all changes to the database, used to recover from an instance failure. Note that often a database will store these files multiple times, for extra security in case of disk failure. The identical redo log files are said to belong to the same group.
- Undo files: These special datafiles, which can only contain undo information, aid in recovery, rollbacks, and read-consistency.
- Archive log files: These files, copies of the redo log files, are usually stored at different locations. They are necessary (for example) when applying changes to a standby database, or when performing recovery after a media failure. It is possible to archive to multiple locations.
- Tempfiles: These special datafiles serve exclusively for temporary storage data (used for example for large sorts or for global temporary tables)
- Control file, necessary for database startup. "A binary file that records the physical structure of a database and contains the names and locations of redo log files, the time stamp of the database creation, the current log sequence number, checkpoint information, and so on."[2]

At the physical level, data files comprise one or more data blocks, where the block size can vary between data files.

Data files can occupy pre-allocated space in the file system of a computer server, utilize raw disk directly, or exist within ASM logical volumes.

Database schema

Most Oracle database installations traditionally came with a default schema called `SCOTT`. After the installation process has set up the sample tables, the user can log into the database with the username `scott` and the password `tiger`. The name of the `SCOTT` schema originated with Bruce Scott, one of the first employees at Oracle (then Software Development Laboratories), who had a cat named Tiger.[3]

Oracle Corporation has de-emphasized the use of the `SCOTT` schema, as it uses few of the features of the more recent releases of Oracle. Most recent[4] examples supplied by Oracle Corporation reference the default HR or OE schemas.

Other default schemas include:

- `SYS` (essential core database structures and utilities)
- `SYSTEM` (additional core database structures and utilities, and privileged account)
- `OUTLN` (utilized to store metadata for stored outlines for stable query-optimizer execution plans.)
- `BI`, `IX`, `HR`, `OE`, `PM`, and `SH` (expanded sample schemas containing more data and structures than the older `SCOTT` schema).

System Global Area

Each Oracle instance uses a System Global Area or SGA—a shared-memory area—to store its data and control-information.

Each Oracle instance allocates itself an SGA when it starts and de-allocates it at shut-down time. The information in the SGA consists of the following elements, each of which has a fixed size, established at instance startup:

- Datafiles

Every Oracle database has one or more physical datafiles, which contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the datafiles allocated for a database.

Datafiles have the following characteristics:

- One or more datafiles form a logical unit of database storage called a tablespace.
- A datafile can be associated with only one tablespace.
- Datafiles can be defined to extend automatically when they are full.

Data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle Database. For example, if a user wants to access some data in a table of a database, and if the requested information is not already in the memory cache for the database, then it is read from the appropriate datafiles and stored in memory.

Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and to increase performance, data is pooled in memory and written to the appropriate datafiles all at once

- the redo log buffer: this stores redo entries—a log of changes made to the database. The instance writes redo log buffers to the redo log as quickly and efficiently as possible. The redo log aids in instance recovery in the event of a system failure.
- the shared pool: this area of the SGA stores shared-memory structures such as shared SQL areas in the library cache and internal information in the data dictionary. An insufficient amount of memory allocated to the shared pool can cause performance degradation.
- the Large pool Optional area that provides large memory allocations for certain large processes, such as Oracle backup and recovery operations, and I/O server processes
- Database buffer cache: Caches blocks of data retrieved from the database
- KEEP buffer pool: A specialized type of database buffer cache that is tuned to retain blocks of data in memory for long periods of time

- RECYCLE buffer pool: A specialized type of database buffer cache that is tuned to recycle or remove block from memory quickly
- nK buffer cache: One of several specialized database buffer caches designed to hold block sizes different than the default database block size
- Java pool:Used for all session-specific Java code and data in the Java Virtual Machine (JVM)
- Streams pool: Used by Oracle Streams to store information required by capture and apply

When you start the instance by using Enterprise Manager or SQL*Plus, the amount of memory allocated for the SGA is displayed.[5]

Library cache

The library cache stores shared SQL, caching the parse tree and the execution plan for every unique SQL statement. If multiple applications issue the same SQL statement, each application can access the shared SQL area. This reduces the amount of memory needed and reduces the processing-time used for parsing and execution planning.

Data dictionary cache

The data dictionary comprises a set of tables and views that map the structure of the database.

Oracle databases store information here about the logical and physical structure of the database. The data dictionary contains information such as:

- user information, such as user privileges
- integrity constraints defined for tables in the database
- names and datatypes of all columns in database tables
- information on space allocated and used for schema objects

The Oracle instance frequently accesses the data dictionary in order to parse SQL statements. The operation of Oracle depends on ready access to the data dictionary: performance bottlenecks in the data dictionary affect all Oracle users. Because of this, database administrators should make sure that the data dictionary cache has sufficient capacity to cache this data. Without enough memory for the data-dictionary cache, users see a severe performance degradation. Allocating sufficient memory to the shared pool where the data dictionary cache resides precludes these particular performance problem.

Program Global Area

The Program Global Area[6] or PGA memory-area of an Oracle instance contains data and control-information for Oracle's server-processes.

The size and content of the PGA depends on the Oracle-server options installed. This area consists of the following components:

- stack-space: the memory that holds the session's variables, arrays, and so on
- session-information: unless using the multithreaded server, the instance stores its session-information in the PGA. (In a multithreaded server, the session-information goes in the SGA.)
- private SQL-area: an area which holds information such as bind-variables and runtime-buffers
- sorting area: an area in the PGA which holds information on sorts, hash-joins, etc.

DBAs can monitor PGA usage via the `V$SESSTAT` system view.

Dynamic performance views

The dynamic performance views (also known as "fixed views") within an Oracle database present information from virtual tables (X$ tables) built on the basis of database memory. Database users can access the V$ views (named after the prefix of their synonyms) to obtain information on database structures and performance.

Process architectures

Oracle processes

The Oracle RDBMS typically relies on a group of processes running simultaneously in the background and interacting to monitor and expedite database operations. Typical operating environments might include some of the following individual processes (shown along with their abbreviated nomenclature):

- advanced queueing processes (Qnnn)
- archiver processes (ARCn)
- checkpoint process (CKPT) *REQUIRED*
- coordinator-of-job-queues process (CJQn): dynamically spawns slave processes for job-queues
- database writer processes (DBWn) *REQUIRED*
- dispatcher processes (Dnnn): multiplex server-processes on behalf of users
- main Data Guard Broker monitor process (DMON)
- job-queue slave processes (Jnnn)
- log-writer process (LGWR) *REQUIRED*
- log-write network-server (LNSn): transmits redo logs in Data Guard environments
- logical standby coordinator process (LSP0): controls Data Guard log-application
- media-recovery process (MRP): detached recovery-server process
- memory-manager process (MMAN): used for internal database tasks such as Automatic Shared Memory Management
- memory-monitor process (MMON): process for automatic problem-detection, self-tuning and statistics-gathering[7]
- memory-monitor light process (MMNL): gathers and stores Automatic Workload Repository (AWR) data
- mmon slaves (Mnnnn—M0000, M0001, etc.): background slaves of the MMON process[8]
- process-monitor process (PMON) *REQUIRED*
- process-spawner (PSP0): spawns Oracle processes
- queue-monitor coordinator process (QMNC): dynamically spawns queue monitor slaves
- queue-monitor processes (QMNn)
- recoverer process (RECO)
- remote file-server process (RFS)
- shared server processes (Snnn): serve client-requests
- system monitor process (SMON) *REQUIRED*

User processes, connections and sessions

Oracle Database terminology distinguishes different computer-science terms in describing how end-users interact with the database:

- user processes involve the invocation of application software
- a connection refers to the pathway linking a user process to an Oracle instance
- sessions consist of specific connections to an Oracle instance. Each session within an instance has a session identifier or "SID" (distinct from the system-identifier SID).

Concurrency and locking

Oracle databases control simultaneous access to data resources with locks (alternatively documented as "enqueues"). The databases also utilize "latches" - low-level serialization mechanisms to protect shared data structures in the System Global Area.

Configuration

Database administrators control many of the tunable variations in an Oracle instance by means of values in a parameter file. This file in its ASCII default form ("pfile") normally has a name of the format `init<SID-name>.ora`. The default binary equivalent server parameter file ("spfile") (dynamically reconfigurable to some extent) defaults to the format `spfile<SID-name>.ora`. Within an SQL-based environment, the views `V$PARAMETER` and `V$SPPARAMETER` give access to reading parameter values.

Administration

The "Scheduler" (from Oracle 10g) and the Job subsystem permit the automation of predictable processing.

Oracle Resource Manager aims to allocate CPU resources between users and groups of users when such resources become scarce.

Oracle Corporation stated in product announcements that manageability for DBAs had improved from Oracle9i to 10g. Lungu and Vătuiu (2008) assessed the relative manageability by performing common DBA tasks and measuring timings. They performed their tests on a single Pentium CPU (1.7 GHz) with 512 MB RAM,running Windows Server 2000. From Oracle9i to 10g, installation improved 36%,day-to-day administration 63%, backup and recovery 63%, and performance diagnostics and tuning 74%, for a weighted total improvement of 56%. The researchers concluded that "Oracle10g represents a giant step forward from Oracle9i in making the database easier to use and manage".

Internationalization

Oracle Database software comes in 63 language-versions (including regional variations such as British English and American English). Variations between versions cover the names of days and months, abbreviations, time-symbols (such as A.M. and A.D.), and sorting.

Oracle Corporation has translated Oracle Database error-messages into Arabic, Catalan, Chinese, Czech, Danish, Dutch, English, Finnish, French, German, Greek, Hebrew, Hungarian, Italian, Japanese, Korean, Norwegian, Polish, Portuguese, Romanian, Russian, Slovak, Spanish, Swedish, Thai and Turkish.

Oracle Corporation provides database developers with tools and mechanisms for producing internationalized database applications: referred to internally as "Globalization".

History

Corporate/technical timeline

- 1977: Larry Ellison and friends founded Software Development Laboratories (SDL).
- 1978: Oracle Version 1, written in assembly language, runs on PDP-11 under RSX, in 128K of memory. Implementation separates Oracle code and user code. Oracle V1 is never officially released.[9]
- 1979: SDL changed its company-name to "Relational Software, Inc." (RSI) and introduced its product Oracle V2 as an early relational database system - often citedWikipedia:Manual of Style/Words to watch#Unsupported attributions as the first commercially sold RDBMS. The version did not support transactions, but implemented the basic SQL functionality of queries and joins. (RSI never released a version 1 - instead calling the first version *version 2* as a marketing gimmick.)[10]
- 1982: RSI in its turn changed its name, becoming known as "Oracle Corporation",[11] to align itself more closely with its flagship product.
- 1983: The company released Oracle version 3, which it had re-written using the C programming language and which supported `COMMIT` and `ROLLBACK` functionality for transactions. Version 3 extended platform support from the existing Digital VAX/VMS systems to include Unix environments.
- 1984: Oracle Corporation released Oracle version 4, which supported read-consistency. In October it also released the first Oracle for the IBM PC.
- 1985: Oracle Corporation released Oracle version 5, which supported the client–server model—a sign of networks becoming more widely available in the mid-1980s.
- 1986: Oracle version 5.1 started supporting distributed queries.
- 1988: Oracle RDBMS version 6 came out with support for PL/SQL embedded within Oracle Forms v3 (version 6 could not store PL/SQL in the database proper), row-level locking and hot backups.[12]
- 1989: Oracle Corporation entered the application-products market and developed its ERP product, (later to become part of the Oracle E-Business Suite), based on the Oracle relational database.
- 1990: the release of Oracle Applications release 8
- 1992: Oracle version 7 appeared with support for referential integrity, stored procedures and triggers.
- 1997: Oracle Corporation released version 8, which supported object-oriented development and multimedia applications.
- 1999: The release of Oracle8*i* aimed to provide a database inter-operating better with the Internet (the *i* in the name stands for "Internet"). The Oracle8*i* database incorporated a native Java virtual machine (Oracle JVM, also known as "Aurora").
- 2000: Oracle E-Business Suite 11*i* pioneers integrated enterprise application software
- 2001: Oracle9*i* went into release with 400 new features, including the ability to read and write XML documents. 9*i* also provided an option for Oracle RAC, or "Real Application Clusters", a computer-cluster database, as a replacement for the Oracle Parallel Server (OPS) option.
- 2002: the release of Oracle 9i Database Release 2 (9.2.0)
- 2003: Oracle Corporation released Oracle Database 10*g*, which supported regular expressions. (The *g* stands for "grid"; emphasizing a marketing thrust of presenting 10*g* as "grid computing ready".)
- 2005: Oracle Database 10.2.0.1—also known as Oracle Database 10*g* Release 2 (10*g*R2)—appeared.
- 2006: Oracle Corporation announces Unbreakable Linux and acquires i-flex
- 2007: Oracle Database 10*g* release 2 sets a new world record TPC-H 3000 GB benchmark result
- 2007: Oracle Corporation released Oracle Database 11*g* for Linux and for Microsoft Windows.
- 2008: Oracle Corporation acquires BEA Systems.
- 2010: Oracle Corporation acquires Sun Microsystems.
- 2011: Oracle Corporation acquires web content management system FatWire Software.

- 2011: On October 18, Oracle Corporation acquires Endeca Technologies Inc. faceted search engine software vendor.
- 2013: Oracle Corporation released Oracle Database 12*c* for Linux, Solaris and Windows. (The *c* stands for "cloud".)

Patch Updates and Security Alerts

Oracle Corporation releases Critical Patch Updates (CPUs) or Security Patch Updates (SPUs) and Security Alerts to close security holes through which data theft may occur. Critical Patch Updates (CPUs) and Security Alerts [13] come out quarterly on the Tuesday closest to 17th day of the month.

- Customers may receive release notification by email [14].
- White Paper: Critical Patch Update Implementation Best Practices [15]

Version numbering

Oracle products follow a custom release numbering and naming convention. With the Oracle RDBMS 10*g* release, Oracle Corporation began using the "10*g*" label in all versions of its major products, although some sources refer to Oracle Applications Release 11*i* as Oracle 11*i*.Wikipedia:Please clarify The suffixes "i", "g" and "c" do not actually represent a low-order part of the version number, as letters typically represent in software industry version numbering; that is, there is no predecessor version of Oracle 10*g* called Oracle 10*f*. Instead, the letters stand for "internet", "grid" and "cloud", respectively.[16] Consequently many simply drop the "g" or "i" suffix when referring to specific versions of an Oracle product.

Major database-related products and some of their versions include:

- Oracle Application Server 10*g* (also known as "Oracle AS 10*g*"): a middleware product;
- Oracle Applications Release 11*i* (aka Oracle e-Business Suite, Oracle Financials or Oracle 11i): a suite of business applications;
- Oracle Developer Suite 10*g* (9.0.4);
- Oracle JDeveloper 10*g*: a Java integrated development environment;

Since version 2, Oracle's RDBMS release numbering has used the following codes:

- Oracle v2 : 2.3
- Oracle v3 : 3.1.3
- Oracle v4 : 4.1.4.0-4.1.4.4
- Oracle v5 : 5.0.22, 5.1.17, 5.1.22
- Oracle v6 : 6.0.17-6.0.36 (no OPS code), 6.0.37 (with OPS)
- Oracle7: 7.0.12–7.3.4
- Oracle8 Database: 8.0.3–8.0.6
- Oracle8*i* Database Release 1: 8.1.5.0–8.1.5.1
- Oracle8*i* Database Release 2: 8.1.6.0–8.1.6.3
- Oracle8*i* Database Release 3: 8.1.7.0–8.1.7.4
- Oracle9*i* Database Release 1: 9.0.1.0–9.0.1.5 (Patchset as of December 2003[4])
- Oracle9*i* Database Release 2: 9.2.0.1–9.2.0.8 (Patchset as of April 2007[4])
- Oracle Database 10*g* Release 1: 10.1.0.2–10.1.0.5 (Patchset as of February 2006[4])
- Oracle Database 10*g* Release 2: 10.2.0.1–10.2.0.5 (Patchset as of April 2010[4])
- Oracle Database 11*g* Release 1: 11.1.0.6–11.1.0.7 (Patchset as of September 2008[4])
- Oracle Database 11*g* Release 2: 11.2.0.1–11.2.0.4 (Patchset as of August 2013[4])
- Oracle Database 12*c* Release 1: 12.1 (Patchset as of June 2013[4])
- Oracle Database 12*c* Release 1: 12.1.0.2.0 (Patchset 16994047as of February 2014[4])

The version-numbering syntax within each release follows the pattern: major.maintenance.application-server.component-specific.platform-specific.

For example, "10.2.0.1 for 64-bit Solaris" means: 10th major version of Oracle, maintenance level 2, Oracle Application Server (OracleAS) 0, level 1 for Solaris 64-bit.

The *Oracle Database Administrator's Guide* [17] offers further information on Oracle release numbers.

Marketing editions

Over and above the different versions of the Oracle database management software developed over time, Oracle Corporation subdivides its product into varying "editions" - apparently for marketing and license-tracking reasons. (Do not confuse the marketing "editions" with the internal virtual versioning "editions" introduced with Oracle 11.2). In approximate order of decreasing scale:

- Enterprise Edition[18] (EE) includes more features than the 'Standard Edition', especially in the areas of performance and security. Oracle Corporation licenses this product on the basis of users or of processors, typically for servers running 4 or more CPUs. EE has no memory limits, and can utilize clustering using Oracle RAC software.
- Standard Edition[19] (SE) contains base database functionality. Oracle Corporation licenses this product on the basis of users or of processors, typically for servers running from one to four CPUs. If the number of CPUs exceeds 4 CPUs, the user must convert to an Enterprise license. SE has no memory limits, and can utilize clustering with Oracle RAC at no additional charge.
- Standard Edition One,[20] (SE1 or SEO) introduced with Oracle 10*g*, has some additional feature-restrictions. Oracle Corporation markets it for use on systems with one or two CPUs. It has no memory limitations.
- Express Edition ("Oracle Database XE")
  - The first Express Edition, introduced in 2005, offered Oracle 10*g* free to distribute on Windows and Linux platforms. It had a footprint of only 150 MB, had a limitation to a maximum of 4 GB of user data and could use only a single CPU. Although it could install on a server with any amount of memory, it used a maximum of 1 GB. Support for this version came exclusively through on-line forums and not through Oracle support.
  - Oracle 11g Express Edition, released by Oracle Corporation on 24 September 2011, can support 11 GB of user data.
- Oracle Database Lite, intended for running on mobile devices. The embedded mobile database located on the mobile device can synchronize with a server-based installation. Includes support for Win32, Windows CE, Palm OS, and EPOC database clients, integration with Oracle's Advanced Queuing (AQ) mechanism, and data and application synchronization software (to enterprise Oracle databases). Supports 100% Java development (through JDBC drivers and the database's native support for embedded SQLJ and Java stored procedures).

Host platforms

Prior to releasing Oracle 9*i* in 2001, Oracle Corporation ported its database product to a wide variety of platforms. Subsequently Oracle Corporation consolidated on a smaller range of operating-system platforms.

As of November 2011[4], Oracle Corporation supported the following operating systems and hardware platforms for Oracle Database 11*g* (11.2.0.2.0):[21]

- zLinux64
- Microsoft Windows (32-bit)
- Microsoft Windows (x64)
- Linux x86
- Linux x86-64
- Solaris (SPARC)

- Solaris (x86-64)
- HP-UX Itanium
- HP-UX PA-RISC (64-bit)
- AIX (PPC64)
- OpenVMS (IA64)

In 2011, Oracle Corporation announced the availability of Oracle Database Appliance, a pre-built, pre-tuned, highly available clustered database server built using two SunFire X86 servers and direct attached storage.

Some Oracle Enterprise edition databases running on certain Oracle-supplied hardware can utilize Hybrid Columnar Compression for more efficient storage.

Related software

Oracle products

- Oracle Database Firewall analyzes database traffic on a network to prevent threats such as SQL injection.

Database options

Oracle Corporation refers to some extensions to the core functionality of the Oracle database as "database options".[22] As of 2013[4] such options include:

- Active Data Guard [23] (extends Oracle Data Guard physical standby functionality in 11*g*)
- Advanced Compression (compresses tables, backups and redo-data)
- Advanced Security [24] (adds data encryption methods for both data at rest and on the network)
- Content database [25] (provides a centralized repository for unstructured information)
- Data Mining [26] (ODM) (mines for patterns in existing data)
- Database Vault [27] (enforces extra security on data access)
- In-Memory Database Cache [28] (utilizes TimesTen technology)
- Label Security [29] (enforces row-level security)
- Management Packs [30] (various). For example:
  - Oracle Database Change Management Pack (tracks and manages schema changes)
- Oracle Answers [31] (for *ad-hoc* analysis and reporting)
- Oracle Application Express, a no-cost environment for database-oriented software-development
- Oracle GoldenGate 11*g* [32] (distributed real-time data acquisition)
- Oracle Multitenant - a container database holding pluggable databases (PDBs) (from 12c)
- Oracle OLAP [33] (adds analytical processing)
- Oracle Programmer (provides programmatic access to Oracle databases via precompilers, interfaces and bindings)[34]
- Oracle Real Application Testing (new at version 11*g*)—including Database Replay (for testing workloads) and SQL Performance Analyzer (SPA) (for preserving SQL efficiency in changing environments)
- Oracle Spatial and Graph (includes 2D,3D and Raster geospatial data types, indexes, and spatial analytics and data models used in business applications and geographic information systems (GIS)) as well as World Wide Web Consortium Resource Description Framework (RDF) graph management and analysis
- Oracle Text [35] (standard SQL to index, search, and analyze text and documents stored in the Oracle database)
- Oracle XML DB [36], a no-cost component in each edition of the database which provides high-performance technology for storing and retrieving native XML
- Oracle Warehouse Builder (in various forms and sub-options)
- Partitioning [37] (granularizes tables and indexes for efficiency)
- Real Application Clusters (RAC) (coordinates multiple database servers, together accessing the same database)

- Records database [38] (a records management application)
- Transparent Gateway [39] for connecting to non-Oracle systems. Offers optimized solution, with more functionality and better performance than Oracle Generic Connectivity.
- Total Recall [40] (optimizes long-term storage of historical data)

This list is incomplete; you can help by expanding it [4].

In most cases, using these options entails extra licensing costs.[41]

Suites

In addition to its RDBMS, Oracle Corporation has released several related suites of tools and applications relating to implementations of Oracle databases. For example:

- Oracle Application Server, a J2EE-based application server, aids in developing and deploying applications which utilise Internet technologies and a browser.
- Oracle Collaboration Suite contains messaging, groupware and collaboration applications.
- Oracle Developer Suite contains software development tools, including JDeveloper.
- Oracle E-Business Suite collects together applications for enterprise resource planning (including Oracle Financials), customer relationship management and human resources management (Oracle HR).
- Oracle Enterprise Manager (OEM) used by database administrators (DBAs) to manage the DBMS, and recently[4] in version 10*g*, a web-based rewrite of OEM called "Oracle Enterprise Manager Database Control". Oracle Corporation has dubbed the super-Enterprise-Manager used to manage a grid of multiple DBMS and Application Servers "Oracle Enterprise Manager Grid Control".
- Oracle Programmer/2000, a bundling of interfaces for 3GL programming languages, marketed with Oracle7 and Oracle8.
- Oracle WebCenter Suite

Database "features"

Apart from the clearly defined database options, Oracle databases may include many semi-autonomous software sub-systems, which Oracle Corporation sometimes refers to as "features" in a sense subtly different from the normal usage of the word. For example, Oracle Data Guard counts officially as a "feature", but the command-stack within SQL*Plus, though a usability feature, does not appear in the list of "features" in Oracle's list [42].Wikipedia:No original research Such "features" may include (for example):

- Active Session History (ASH), the collection of data for immediate monitoring of very recent database activity.
- Automatic Workload Repository (AWR) [43], providing monitoring services to Oracle database installations from Oracle version 10. Prior to the release of Oracle version 10, the Statspack facility provided similar functionality.
- Clusterware
- Data Aggregation and Consolidation [44]
- Data Guard [45] for high availability
- Generic Connectivity [39] for connecting to non-Oracle systems.
- Data Pump utilities, which aid in importing and exporting data and metadata between databases
- Database Resource Manager (DRM), which controls the use of computing resources.
- Fast-start parallel rollback
- Fine-grained auditing (FGA) (in Oracle Enterprise Edition) supplements standard security-auditing features
- Flashback for selective data recovery and reconstruction[46]
- iSQL*Plus [47], a web-browser-based graphical user interface (GUI) for Oracle database data-manipulation (compare SQL*Plus)
- Oracle Data Access Components (ODAC), tools which consist of:
  - Oracle Data Provider for .NET (ODP.NET)

  - Oracle Developer Tools (ODT) for Visual Studio
  - Oracle Providers for ASP.NET
  - Oracle Database Extensions for .NET
  - Oracle Provider for OLE DB
  - Oracle Objects for OLE
  - Oracle Services for Microsoft Transaction Server
- Oracle-managed files [48] (OMF) -- a feature allowing automated naming, creation and deletion of datafiles at the operating-system level.
- Oracle Multimedia (known as "Oracle *inter*Media" before Oracle 11g) for storing and integrating multimedia data within a database
- Recovery Manager [49] (rman) for database backup, restoration and recovery
- SQL*Plus, a program that allows users to interact with Oracle database(s) via SQL and PL/SQL commands on a command-line. Compare iSQL*Plus.
- Universal Connection Pool (UCP), a connection pool based on Java and supporting JDBC, LDAP, and JCA
- Virtual Private Database[50] (VPD), an implementation of fine-grained access control.

This list is incomplete; you can help by expanding it [4].

Tools

Users can develop their own applications in Java and PL/SQL using tools such as:

- Oracle Forms
- Oracle JDeveloper
- Oracle Reports

As of 2007[4] Oracle Corporation had startedWikipedia:Please clarify a drive toward "wizard"-driven environments with a view to enabling non-programmers to produce simple data-driven applications.

The **Database Upgrade Assistant** (**DBUA**) provides a GUI for the upgrading of an Oracle dastabase.

JAccelerator (NCOMP) - a native-compilation Java "accelerator", integrates hardware-optimized Java code into an Oracle 10g database.

Oracle SQL Developer, a free graphical tool for database development, allows developers to browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. It incorporates standard and customized reporting.

Oracle's **OPatch** provides patch management for Oracle databases.

The **SQLTXPLAIN** tool (or **SQLT**) provides tuning assistance for Oracle SQL queries.

Other databases marketed by Oracle Corporation

By acquiring other technology in the database field, Oracle Corporation can also offer:

- TimesTen, a memory-resident database that can cache transactions and synchronize data with a centralized Oracle database server. It functions as a real-time infrastructure software product intended for the management of low-latency, high-volume data, of events and of transactions.
- BerkeleyDB, a simple, high-performance, embedded database
- Oracle Rdb, a legacy relational database for the OpenVMS operating-system
- MySQL a relational database purchased as part of Oracle Corporation's takeover of its immediate previous owner, Sun Microsystems
- Oracle NoSQL Database, a scalable, distributed key-value NoSQL database

Use

The Oracle RDBMS has had a reputation among novice users as difficult to install on Linux systems.[*citation needed*] Oracle Corporation has packaged recent[4] versions for several popular Linux distributions in an attempt to minimize installation challenges beyond the level of technical expertise required to install a database server.[*citation needed*]

Official support

Users who have Oracle support contracts can use Oracle's "My Oracle Support" web site. The "My Oracle Support" site was known as MetaLink until a re-branding exercise completed in October 2010. The support site provides users of Oracle Corporation products with a repository of reported problems, diagnostic scripts and solutions. It also integrates with the provision of support tools, patches and upgrades.

The *Remote Diagnostic Agent* or *RDA* can operate as a command-line diagnostic tool executing a script. The data captured provides an overview of the Oracle Database environment intended for diagnostic and trouble-shooting. Within RDA, the *HCVE* (Health Check Validation Engine) can verify and isolate host system environmental issues that may affect the performance of Oracle software.

Database-related guidelines

Oracle Corporation also endorses certain practices and conventions as enhancing the use of its database products. These include:

- Oracle Maximum Availability Architecture (MAA) guidelines on developing high-availability systems
- Optimal Flexible Architecture (OFA), blueprints for mapping Oracle-database objects to file-systems

Oracle Certification Program

The Oracle Certification Program, a professional certification program, includes the administration of Oracle Databases as one of its main certification paths. It contains three levels:

1. Oracle Certified Associate (OCA)
2. Oracle Certified Professional (OCP)
3. Oracle Certified Master (OCM)

User groups

A variety of official (Oracle-sponsored) and unofficial Oracle User Groups has grown up of users and developers of Oracle databases. They include:

- Geographical/regional user groups
- Independent Oracle Users Group
- Industry-centric user groups
- Oracle Technology Network
- Product-centric user groups
- The OakTable Network
- Usenet newsgroups

Market position

Competition

In the market for relational databases, Oracle Database competes against commercial products such as IBM's DB2 UDB and Microsoft SQL Server. Oracle and IBM tend to battle for the mid-range database market on UNIX and Linux platforms, while Microsoft dominates the mid-range database market on Microsoft Windows platforms. However, since they share many of the same customers, Oracle and IBM tend to support each other's products in many middleware and application categories (for example: WebSphere, PeopleSoft, and Siebel Systems CRM), and IBM's hardware divisions work closely[*citation needed*] with Oracle on performance-optimizing server-technologies (for example, Linux on zSeries). The two companies have a relationship perhapsWikipedia:No original research best described as "coopetition". Niche commercial competitors include Teradata (in data warehousing and business intelligence), Software AG's ADABAS, Sybase, and IBM's Informix, among many others.

In 2007, competition with SAP AG occasioned litigation from Oracle Corporation.[51]

Increasingly, the Oracle database products compete against such open-source software relational database systems as PostgreSQL, Firebird, and MySQL. Oracle acquired Innobase, supplier of the InnoDB codebase to MySQL, in part to compete better against open source alternatives, and acquired Sun Microsystems, owner of MySQL, in 2010. Database products licensed as open source are, by the legal terms of the Open Source Definition, free to distribute and free of royalty or other licensing fees.

Pricing

Oracle Corporation offers term licensing for all Oracle products. It bases the list price for a term-license on a specific percentage of the perpetual license price. Prospective purchasers can obtain licenses based either on the number of processors in their target server machines or on the number of potential seats ("named users").

Enterprise Edition (DB EE)

> As of July 2010[4], the database that costs the most per machine-processor among Oracle database editions, at $47,500 per processor. The term "per processor" for Enterprise Edition is defined with respect to physical cores and a processor core multiplier (common processors = 0.5*cores). e.g. An 8-processor, 32-core server using Intel Xeon 56XX CPUs would require 16 processor licenses.

Standard Edition (DB SE)

> Cheaper: it can run on up to four processors but has fewer features than Enterprise Edition—it lacks proper parallelization,[52] etc.; but remains quite suitable for running medium-sized applications. There are not additional cost for Oracle RAC on the latest Oracle 11g R2 standard edition release.

Standard ONE (DB SE1 or DB SEO)

> Sells even more cheaply, but remains limited to two CPUs. Standard Edition ONE sells on a per-seat basis with a five-user minimum. Oracle Corporation usually sells the licenses with an extra 22% cost for support and upgrades (access to My Oracle Support—Oracle Corporation's support site) which customers need to renew annually.

Oracle Express Edition (DB XE) (Oracle XE)

> An addition to the Oracle database product family (beta version released in 2005, production version released in February 2006), offers a free version of the Oracle RDBMS, but one limited to 11 GB of user data and to 1 GB of memory used by the database (SGA+PGA).[53] XE will use no more than one CPU and lacks an internal JVM. XE runs only on 32-bit Windows and 64-bit Linux, but not on AIX, Solaris, HP-UX and the other operating systems available for other editions. Support is via a free Oracle Discussion Forum [54] only.

As computers running Oracle often have many multi-core processors (resulting in many cores, all to be licensed), the software price can rise into the hundreds of thousands of dollars. The total cost of ownership often exceeds this, as large Oracle installations usually require experienced and trained database administrators to do the set-up properly. Furthermore, further components must be licensed and paid for, for instance the Enterprise Options used with the databases. Many licensing pitfalls let even rise the costs of ownership. Because of the product's large installed base and available training courses, Oracle specialists in some areas have become a more abundant resource than those for more exotic databases. Oracle frequently provides special training offers for database-administrators.

On Linux, Oracle's *certified configurations* include Oracle's own Oracle Linux and other commercial Linux distributions (Red Hat Enterprise Linux 3, 4 and 5, SuSE SLES 8, 9, 10 and 11, Asianux) which can cost in a range from a few hundred to a few thousand USD per year (depending on processor architecture and the support package purchased).

The Oracle database system can also install and run on freely available Linux distributions such as the Red Hat-based CentOS, or Debian-based systems.

References

[1] http://www.oracle.com/us/products/database/overview/index.html

[2] Oracle Corporation, Oracle Database Concepts 11g Release 1 (11.2), http://download.oracle.com/docs/cd/E11882_01/server.112/e25789/glossary.htm#CHDDFGEC, 2011

[3] Oracle FAQ (http://web.archive.org/web/20080116210119/http://www.orafaq.com/faqora.htm#SCOTT)

[4] http://en.wikipedia.org/w/index.php?title=Oracle_Database&action=edit

[5] Karlsson André. "Oracle DB Architecture - The Basics" (http://blog.protractus.se/?page_id=34),

[6] PGA Definition (http://download.oracle.com/docs/cd/B19306_01/mix.102/b14388/gloss-p.htm#index-PGA), Oracle Database Master Glossary

[7] Safaribooksonline.com (http://my.safaribooksonline.com/9780072263053/new_background_processes_in_10)

[8] Safaribooksonline.com (http://my.safaribooksonline.com/9780072263053/new_background_processes_in_10)

[9] http://www.oracle.com/us/corporate/profit/p27anniv-timeline-151918.pdf

[10] As Larry Ellison said in an Oracle OpenWorld keynote presentation (http://news.cnet.com/8301-10784_3-9814858-7.html) on 11 November 2007: "Who'd buy a version 1.0 from four guys in California?"

[11] Oracle.com (http://www.oracle.com/oramag/profit/07-may/p27anniv_timeline.pdf)

[12] Compare Oracle.com (http://www.oracle.com/oramag/profit/07-may/p27anniv_timeline.pdf)

[13] http://www.oracle.com/technetwork/topics/security/alerts-086861.html

[14] http://www.oracle.com/technetwork/topics/security/securityemail-090378.html

[15] http://www.oracle.com/us/support/assurance/leveraging-cpu-wp-164638.pdf?ssSourceSiteId=otnen

[16] theregister.co.uk: Oracle gears up for infrastructure cloud and 12c database launches • The Register (http://www.theregister.co.uk/2012/09/20/oracle_openworld_preview_q1_f2012_numbers/)

[17] http://docs.oracle.com/cd/B14117_01/server.101/b10739/dba.htm#sthref35

[18] Enterprise Edition (http://www.oracle.com/database/enterprise_edition.html)

[19] Standard Edition (http://www.oracle.com/database/standard_edition.html)

[20] Standard Edition One (http://www.oracle.com/database/std_one.html)

[21] Oracle Database Online Documentation 11g Release 2 (11.2) (http://www.oracle.com/pls/db112/portal.portal_db?selected=11)

[22] Oracle database options (http://www.oracle.com/us/products/database/options/index.html)

[23] http://www.oracle.com/database/active-data-guard.html

[24] http://www.oracle.com/us/products/database/options/advanced-security/index.html

[25] http://www.oracle.com/database/contentdb.html

[26] http://www.oracle.com/technology/products/bi/odm/index.html

[27] http://www.oracle.com/us/products/database/options/database-vault/index.html

[28] http://www.oracle.com/database/in-memory-database-cache.html

[29] http://www.oracle.com/us/products/database/options/label-security/index.html

[30] http://www.oracle.com/technology/products/oem/extensions/index.html

[31] http://www.oracle.com/appserver/business-intelligence/standard-edition-one.html

[32] http://www.oracle.com/us/products/middleware/data-integration/goldengate/index.html

[33] http://www.oracle.com/solutions/business_intelligence/olap.html

[34] See download.oracle.com/docs/cd/B28359_01/license.111/b28287/options.htm#CIHBDGAD

[35] http://www.oracle.com/technetwork/database/enterprise-edition/index-098492.html

[36] http://www.oracle.com/technetwork/database/features/xmldb/index.html
[37] http://www.oracle.com/technology/products/oracle9i/datasheets/partitioning.html
[38] http://www.oracle.com/database/recordsdb.html
[39] http://www.oracle.com/technology/products/oracle9i/datasheets/gateways/gateway_rel2_ds.html
[40] http://www.oracle.com/database/total-recall.html
[41] See "Term licenses" at http://oraclestore.oracle.com/for various markets/countries.
[42] http://download.oracle.com/docs/cd/B19306_01/license.102/b14199/editions.htm#CJACGHEB
[43] http://www.oracle.com/technology/pub/articles/10gdba/week6_10gdba.html
[44] http://www.oracle.com/solutions/business_intelligence/warehouse-builder.html
[45] http://www.oracle.com/technology/deploy/availability/htdocs/DataGuardOverview.html
[46] Oracle.com (http://www.oracle.com/technology/deploy/availability/htdocs/Flashback_Overview.htm)
[47] http://www.oracle.com/technology/tech/sql_plus/index.html
[48] http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/omf.htm#i1007206
[49] http://web.archive.org/web/20051208092120/http://www.oracle.com/technology/deploy/availability/htdocs/rman_overview.htm
[50] "Virtual Private Database" appears listed as a feature available as part of Oracle Enterprise Edition in:
[51] About the case (http://hirek.prim.hu/cikk/61917/) in Hungarian
[52] Oracle Database Licensing Information Database Editions (http://download.oracle.com/docs/cd/B28359_01/license.111/b28287/editions.htm#BABDJGGI)
[53] Licensing Restrictions (http://download.oracle.com/docs/cd/E17781_01/install.112/e18802/toc.htm#BABIECJA)
[54] https://forums.oracle.com/forums/forum.jspa?forumID=251&start=0

Bibliography

- Loney, Kevin (17 December 2008). *Oracle Database 11g The Complete Reference* (http://www.mhprofessional.com/product.php?isbn=0071598758) (1st ed.). McGraw-Hill. p. 1368. ISBN 0-07-159875-8. Retrieved 2009-09-05

External links

- Overview provided by Oracle Corporation (http://www.oracle.com/technology/software/products/database/oracle10g/).
- Oracle Licensing Knowledge Net (http://omtco.eu/references/oracle/).

Oracle Exadata

Oracle Exadata is a database appliance with support for both OLTP (transactional) and OLAP (analytical) database systems. It was initially designed in collaboration between Oracle Corporation and Hewlett Packard. Oracle designed the database, operating system (based on the Oracle Linux distribution), and storage software whereas HP designed the hardware for it. After Oracle's acquisition of Sun Microsystems, in 2010 Oracle announced the Exadata Version 2 with improved performance and Sun storage systems.

History

Exadata was announced by Larry Ellison at the 2009 Oracle OpenWorld conference in San Francisco for immediate delivery. The main headline was that Oracle was entering the hardware business with a pre-built database machine, engineered by Oracle. The hardware at this time was manufactured, delivered and supported by HP. Since the acquisition of Sun Microsystems by Oracle circa January 2010, Exadata used Sun-based hardware. In August 2011, Oracle announced that Exadata database machines would be orderable with Solaris 11 Express (in addition to Oracle Linux).[1]

Technical details

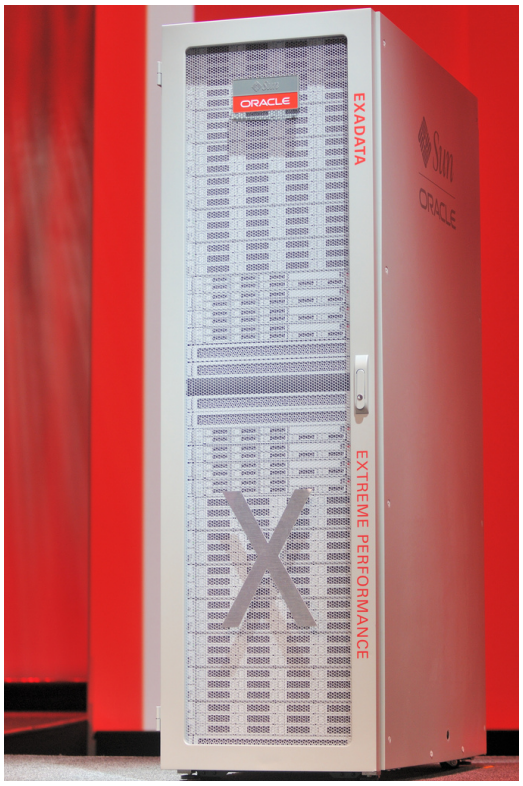
Database servers X3-2

A third generation was announced in 2012.[2] Each Database Server with

- 2 x Eight-Core Intel Xeon E5-2690 Processors (2.9 GHz)
- 128 GB Memory (expandable to 256GB)
- Disk Controller HBA with 512MB Battery Backed Write Cache
- 4 x 300 GB 10,000 RPM Disks
- 2 x QDR (40Gbit/s) Ports
- 4 x 1/10 Gb Ethernet Ports (copper)
- 2 x 10 Gb Ethernet Ports (optical)
- 1 x ILOM Ethernet Port
- 2 x Redundant Hot-Swappable Power Supplies

The Exadata X3-2 comes in 4 different sizes

- Eighth Rack - 2 Database Servers (half of the cores used in each server [8 cores per server]), 3 Storage Servers (half of the disks enabled, half of the Flash Cache enabled)
- Quarter Rack - 2 Database Servers, 3 Storage Servers
- Half Rack - 4 Database Servers, 7 Storage Servers
- Full Rack - 8 Database Servers, 14 Storage Servers



Exadata X2-2 at Oracle OpenWorld 2009

Database servers X3-8

2 Servers, each with

- 8 x Ten-Core Intel Xeon E7-8870 Processors (2.40 GHz)
- 2 TB Memory
- Disk Controller HBA with 512MB Battery Backed Write Cache
- 8 x 300 GB 10,000 RPM Disks
- 8 x InfiniBand QDR (40Gbit/s) Ports
- 8 x 10 Gb Ethernet Ports based on the Intel 82599 10GbE Controller
- 8 x 1 Gb Ethernet Ports
- 1 x ILOM Ethernet Port
- 4 x Redundant Hot-Swappable Power Supplies

Storage Servers X3-2

Each Storage Server has

- Processors: 2 x Six-Core Intel Xeon E5-2630L (2.0 GHz) Processors
- Exadata Smart Flash: Cache 1.6 TB (4 PCI cards)
- System Memory: 64 GB
- Disk Controller: Disk Controller HBA with 512MB Battery Backed Write Cache
- InfiniBand Connectivity: Dual-Port QDR (40Gbit/s) InfiniBand Host Channel Adapter
- Power Supplies: Dual-redundant, hot-swappable power supply
- Disk Drives:
  - 12 x 600 GB 15,000 RPM High Performance or
  - 12 x 3 TB 7,200 RPM High Capacity
  - For raw disk capacity, 1 GB = 1 billion bytes. Actual formatted capacity is less.
- Remote Management: Integrated Lights Out Manager (ILOM) Ethernet port

Database servers X2-2

Sun Fire X4170 M2[3]

- 2 × Six-Core Intel Xeon X5675 Processors (3.06 GHz)
- 96 GB Memory (expandable to 144 GB with optional memory expansion kit)
- Disk Controller HBA with 512MB Battery Backed Write Cache
- 4 × 300 GB 10,000 RPM SAS Disks
- 2 × QDR (40Gbit/s) Ports
- 2 × 10 Gb Ethernet Ports based on the Intel 82599 10GbE Controller
- 4 × 1 Gb Ethernet Ports
- 1 × ILOM Ethernet Port
- 2 × Redundant Hot-Swappable Power Supplies
- 3 × 36 port QDR (40 Gbit/s) InfiniBand Switches (2 x in Quarter Rack configuration)

Database servers X2-8

Sun Fire X4800[4]

- 8 × Ten-Core Intel Xeon E7-8870 Processors (2.40 GHz)
- 2 TB Memory.
- Disk Controller HBA with 512MB Battery Backed Write Cache.
- 8 × 300 GB 10,000 RPM SAS Disks
- 8 × QDR (40Gbit/s) Ports
- 8 × 10 Gb Ethernet Ports based on the Intel 82599 10GbE Controller
- 8 × 1 Gb Ethernet Ports
- 1 × ILOM Ethernet Port
- 4 × Redundant Hot-Swappable Power Supplies
- 3 × 36 port QDR (40 Gbit/s) InfiniBand Switches

Storage Servers X2-2

- 2 × Six-Core Intel Xeon L5640 (2.26 GHz) Processors
- Exadata Smart Flash Cache 384 GB
- System Memory 24 GB
- Disk Controller HBA with 512MB Battery Backed Write Cache
- InfiniBand Connectivity Dual-Port QDR (40Gbit/s) InfiniBand Host Channel Adapter
- Power Supplies Dual-redundant, hot-swappable power supply
- Remote Management Sun Embedded Integrated Lights Out Manager (ILOM)
- Disk Drives 12 × 600 GB 15,000 RPM High Performance SAS or
- 12 × 3 TB 7,200 RPM High Capacity SAS
- Integrated Lights Out Manager (ILOM) Ethernet port[5]

There was also an expansion rack.[6]

Software

Software caches database objects in flash memory, replacing slow, mechanical I/O operations to disk with rapid flash memory operations. Software also provides logging feature to speed database log I/O. Exadata storage cellsWikipedia:Please clarify determine which rows contain values that are being queried. Smart Scan only returns blocks that are relevant to the compute nodes. Storage cells can take over data intensive processing from compute nodes.

Storage cells keep track on maximum and minimum values stored in different areas and use those values to determine where predicates can not exist. This allows the storage cell to not have to read the area at all thus saving time and processing cycles.

InfiniBand switches

The 2010 version of the product used Sun Microsystems datacenter InfiniBand switches with 36 ports.

	Database Machine Full Rack	Database Machine Half Rack	Database Machine Quarter Rack
Database Servers	8	4	2
Exadata Storage Servers	14	7	3
InfiniBand Switches	3	3	2
Upgradability	Connect multiple Full Racks via the included InfiniBand fabric	Field upgrade to Full Rack	Field upgrade from Quarter Rack to Half Rack

Source Oracle Corporation

References

[1] http://www.oracle.com/us/corporate/press/454114
[2] http://www.oracle.com/us/products/database/exadata-db-machine-x3-2-1851253.pdf
[3] http://www.oracle.com/technetwork/database/exadata/dbmachine-x2-2-datasheet-175280.pdf?ssSourceSiteId=ocomen
[4] http://www.oracle.com/technetwork/database/exadata/dbmachine-x2-8-datasheet-173705.pdf
[5] http://www.oracle.com/technetwork/database/exadata/exadata-x2-2-datasheet-175368.pdf?ssSourceSiteId=ocomen
[6] http://www.oracle.com/technetwork/database/exadata/exadata-storage-exp-rack-ds-v1-425092.pdf

External links

- Oracle Exadata Database Machine (http://www.oracle.com/us/products/database/exadata-database-machine/index.html)
- Oracle & Sun Unveil Exadata Version 2 (https://www.youtube.com/watch?v=3WPOrdUGteE) on YouTube - Larry Ellison's Exadata V2 announcement
- History of Exadata (http://flashdba.com/history-of-exadata/)

Oracle Coherence

In computing, **Oracle Coherence** is a proprietary[1] Java-based in-memory data grid designed to improve reliability, scalability and performance compared to traditional relational database management systems.

Coherence provides several core services:

- Replicated and partitioned data management and caching services - At its core Oracle Coherence is a highly scalable and fault-tolerant distributed cache engine. Coherence uses a specialized scalable protocol and many inexpensive computers to create a cluster which can be seamlessly expanded to add more memory, processing power or both. As a result Coherence has no single point of failure and transparently fails over if a cluster member fails. When a Coherence server is added or removed the cluster automatically re-balances to share the workload. As a result Coherence provides a highly available and predictably horizontally scalable infrastructure for managing application data. [2]
- Replicated data processing engine - In addition to caching Coherence provides a rich data processing model so processing can be farmed out to where the data is, and results returned to the client. By moving the processing to the data, processing too is highly scalable. This is to some extent similar to a MapReduce framework, but lacks the option of parallel reductions. [3]
- Event model allowing developers to interact with data as it changes.
- Support for clients written in Java, C++, .NET as well as other languages using Representational State Transfer (REST).

In addition Coherence provides a variety of mechanisms to integrate with other services using TopLink, Java Persistence API, Oracle Golden Gate or almost any other platform using Coherence provided APIs.

Coherence can be used to manage HTTP sessions via Coherence*Web. With Coherence*Web, application services such as Oracle WebLogic Server, IBM WebSphere, Apache Tomcat and others can reap the same benefits of performance, fault tolerance, and scalability as data.

Some Coherence usage patterns are open source and are listed and supported through the Oracle Coherence incubator.[4] These patterns implement features such as messaging, work distribution and data replication across wide area networks with Coherence.

Tangosol Inc. developed the original Coherence product. Oracle Corporation acquired Tangosol in April 2007, when Coherence had about 120 direct customers.[5] It was also embedded in a few products from companies that included some of Oracle's competitors.[6]

References

[1] Oracle Fusion Middleware Licensing Information, Release 12c (12.1.2) - Release 12c (12.1.2) (http://docs.oracle.com/middleware/1212/core/FMWLC/title.htm). Docs.oracle.com (2013-08-02). Retrieved on 2013-09-18.

[2] Oracle Coherence Product Page (http://www.oracle.com/technetwork/middleware/coherence/overview/index.html)

[3] Oracle Coherence and MapReduce (http://blog.cfelde.com/2012/12/oracle-coherence-and-mapreduce/)

[4] The Coherence Incubator (https://java.net/projects/cohinc/)

[5] Oracle acquires Tangosol (http://www.oracle.com/corporate/press/2007_mar/tangosol.html)

[6] Oracle to Acquire a Lead in Extreme Transaction Processing (http://www.gartner.com/resources/147600/147673/oracle_to_acquire_a_lead_in__147673.pdf)

External links

- Oracle Coherence Product page (http://www.oracle.com/technetwork/middleware/coherence/overview/index.html)
- Oracle Coherence User Forum (http://forums.oracle.com/forums/forum.jspa?forumID=480)
- Weblogic Coherence (http://weblogicserveradministration.blogspot.in/)
- The Oracle Coherence Knowledge Base (http://coherence.oracle.com/display/COH/Oracle+Coherence+Knowledge+Base+Home)
- The Oracle Coherence v10 incubator page (http://coherence.oracle.com/display/INC10/Home)
- Oracle Coherence 3.5 by Aleksander Seovic, Packt Press (http://www.packtpub.com/oracle-coherence-35/book)

NVL

In Oracle's dialect of SQL (and in PL/SQL), the **NVL** function lets you substitute a value when a null value is encountered.

The syntax for the NVL function is:

```
NVL( string1, replace_with_if_null )
```

string1 is the string to test for a null value. replace_with_if_null is the value returned if string1 is null.

If the first argument is a character type the function returns a varchar2 value. If the first argument is numeric the function returns a numeric value. You cannot use this function to replace a null integer by a string unless you call the TO_CHAR function on that value:

```
NVL(TO_CHAR(numeric_column), 'some string')
```

In comparison to COALESCE NVL evaluates all arguments before execution:

With NVL:

```
SQL> DECLARE
  2    a NUMBER;
  3
  4    FUNCTION err RETURN NUMBER IS
  5    BEGIN
  6      RAISE program_error;
  7    END err;
  8  BEGIN
  9    a := NVL(1, err);
 10  END;
 11  /
DECLARE
  a NUMBER;

  FUNCTION err RETURN NUMBER IS
  BEGIN
    RAISE program_error;
```

```
  END err;
BEGIN
  a := NVL(1, err);
END;
ORA-06501: PL/SQL: program error
ORA-06512: at line 6
ORA-06512: at line 9
```

With COALESCE:

```
SQL> DECLARE
  2    a NUMBER;
  3
  4    FUNCTION err RETURN NUMBER IS
  5    BEGIN
  6      RAISE program_error;
  7    END err;
  8  BEGIN
  9    a := COALESCE(1, err);
 10  END;
 11  /
PL/SQL procedure successfully completed
```

References

http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions105.htm

Pro*C

Pro*C (also known as **Pro*C/C++**) is an embedded SQL programming language used by Oracle Database database management systems. Pro*C uses either C or C++ as its host language. During compilation, the embedded SQL statements are interpreted by a precompiler and replaced by C or C++ function calls to their respective SQL library. The output from the Pro*C precompiler is standard C or C++ code that is then compiled by any one of several C or C++ compilers into an executable.

External links

- Introduction to Pro*C Embedded SQL [1]
- Embedded SQL with Pro*C [2]
- Oracle 11.2 Pro*C/C++ Programmer's Guide [3]

References

[1] http://infolab.stanford.edu/%7Eullman/fcdb/oracle/or-proc.html
[2] http://www.oreillynet.com/pub/a/databases/2006/12/07/embedded-sql-with-pro-c.html
[3] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e10825/toc.htm

Tools

Toad Data Modeler

Toad Data Modeler is a database design tool allowing users to visually create, maintain and document new or existing database systems. It was previously called "CASE Studio 2" before it was acquired from Charonware by Quest Software in 2006. [1] Quest Software was acquired by Dell as of September 28, 2012.[2]

Visual creation of Entity Relationship Diagrams

Toad Data Modeler allows users to draw logical, physical and universal entity relationship diagrams (ERD). This tool supports various database specific items and therefore it is necessary to select a target database system for a physical ERD.

Physical ERD can be created for:

- DB2
- Ingres
- MS Access
- MS SQL Azure
- MS SQL Server
- MySQL
- Oracle
- PostgreSQL
- Sybase ASE
- Sybase SQL Anywhere

SQL Code Generation

Complex SQL/DDL code can be generated automatically for all above mentioned database systems. It is possible to select types of objects for SQL code generation as well as define options how to generate the output.

Reverse engineering and database visualization

Existing databases can be reverse engineered and displayed in form of a diagram. For all supported database systems it is possible to select particular connection method - native connection, ODBC etc. In addition, ER Diagrams can be created from SQL files for Oracle, Microsoft SQL Azure, Microsoft SQL Server, DB2 and MySQL databases.

Documentation

Toad Data Modeler users can generate reports in HTML, RTF or PDF format. It is also possible to generate output based on XSL transformations - XML, TXT, CSV etc. Pictures of ER Diagrams can be generated for HTML reports only.

Verification

Models can be verified and list of errors, warnings and hints can be displayed as a result of model verification process.

Advanced features

- Synchronization of models with physically existing database (using Alter Script Generation and Model Merge features)
- Version Manager
- Version Control System - Apache Subversion for TDM 4, integrated with Projects to achieve the best possible cooperative functionality
- Creation and maintenance of tasks related to a model or particular part of a model using To-Do list feature.
- Customization, including customization of forms
- Gallery of objects, which can be shared
- Refactoring Utility

Other features

- Support for unicode
- Undo/Redo
- Integration with Toad for Oracle

References

[1] TDM - Statement of Future Direction (http://www.casestudio.com/enu/tdm_statement_of_direction.aspx)

[2] Dell Completes Acquisition of Quest Software (http://www.dell.com/Learn/us/en/uscorp1/secure/2012-09-28-dell-acquisition-quest-software?c=us&l=en&s=corp&delphi:gr=true)

External links

- Official website (http://www.quest.com/toad-data-modeler/)
- Modeling community (http://modeling.inside.quest.com/)

SQuirreL SQL Client

SQuirreL SQL Client

Developer(s)	Colin Bell, Gerd Wagner, Rob Manning and others
Stable release	3.5.0 / May 5, 2013
Operating system	Cross-platform
Platform	Java
Type	Database administration tool
License	LGPL
Website	www.squirrelsql.org [1]

The **SQuirreL SQL Client** is a database administration tool. It uses JDBC to allow users to explore and interact with databases via a JDBC driver. It provides an editor that offers code completion and syntax highlighting for standard SQL. It also provides a plugin architecture that allows plugin writers to modify much of the application's behavior to provide database-specific functionality or features that are database-independent. As this desktop application is written entirely in Java with Swing UI components, it should run on any platform that has a JVM.

SQuirreL SQL Client is free as open source software that is distributed under the GNU Lesser General Public License.

Feature Summary

- Object Tree allows for browsing database objects such as catalogs, schemas, tables, triggers, views, sequences, procedures, UDTs, etc.
- The SQL Editor is based on RSyntaxTextArea by fifesoft.com to provide syntax highlighting. It can be used to open, create, save and execute files containing SQL statements.
- Supports simultaneous sessions with multiple databases. This allows to compare data and share SQL statements between databases.
- It runs on any platform that has a JVM.
- Plugin architecture to facilitate database vendor-specific extensions (Information or actions not available using standard JDBC - see SQuirreL SQL Client Plugin API for more details)
- Translations for the user interface are available in: (Bulgarian, Brazilian Portuguese, Chinese, Czech, French, German, Italian, Japanese, Polish, Spanish, Russian).
- Graph capabilities to create charts showing table relationships.
- Bookmarks, which are user-defined code templates. SQuirreL comes with predefined example bookmarks for the most common SQL and DDL statements.

History

The SQuirreL SQL project was developed by a team of Java developers around the world and led by Colin Bell. It has been hosted as a SourceForge project since 2001, and is still being developed today.

Supported databases

- Axion Java RDBMS.
- Apache Derby
- Daffodil (One$DB)
- Fujitsu Siemens SESAM/SQL-Server with the SESAM/SQL JDBC driver
- Firebird with the JayBird JCA/JDBC Driver
- FrontBase
- Hypersonic SQL
- H2 (DBMS)
- IBM DB2 for Linux, OS/400 and Windows
- Informix
- Ingres (and OpenIngres)
- InstantDB
- InterBase
- Mckoi SQL Database
- Microsoft Access with the JDBC/ODBC bridge.
- Microsoft SQL Server
- Mimer SQL
- MySQL
- Netezza
- Oracle Database 8i, 9i, 10g, 11g
- Pointbase
- PostgreSQL 7.1.3 and higher
- SAPDB
- Sybase
- Sunopsis XML Driver (JDBC Edition)
- Teradata Warehouse
- ThinkSQL RDBMS
- Vertica Analytic Database

References

[1] http://www.squirrelsql.org/

SQuirreL SQL Client Plugin API

Plugin Basics

Introduction

SQuirreL SQL Client uses the JDBC API to interact with a database, which for the most part makes it database implementation-agnostic. However, it can be extended to support implementation-specific features by writing a plugin that uses the Plugin API. For example, JDBC doesn't specify a generic way to obtain the source code that can be used to re-create database objects such as triggers, stored procedures, functions, views, etc. Yet this information is usually available in some data dictionary view that can be queried by object name. Although the SQL standard specifies a special optional schema (INFORMATION_SCHEMA) to store this information in, only a few database vendors have implemented this, while others have chosen a different schema to store this information in. A plugin can be written with the implementation-specific queries that are required to retrieve the source code for each type of database object. Additionally, other plugins that are not implementation-specific have been written using the Plugin API to implement features not available in the base SQuirreL software installation (for example, code completion, syntax highlighting, Look and Feel, query favorites, etc.)

Plugin Loading

A plugin is made available to SQuirreL as a set of classes that are packaged into a jar file which is located in the plugins directory in the installation directory. SQuirrel uses a custom classloader to find and load jars containing plugins. The PluginManager is responsible for reading prefs.xml (as an XML-Bean) and for each plugin found, noting whether or not to load it according to the attribute called "loadAtStartup". If a plugin is added to the plugins directory, SQuirreL must be restarted to pick up the new plugin (Updated plugin definitions require a restart as well).

Session Lifecycle

Plugins can register to listen for session start/end events by implementing the ISessionPlugin interface and/or DefaultSessionPlugin base class. Sessions can be started and stopped by the user at will. Therefore, it is important for plugins that implement the ISessionPlugin interface (including those that extend DefaultSessionPlugin) to use care when storing references to Sessions. Specifically, if a plugin stores a reference to an ISession instance that it received in a call to sessionStarted (ISession session), it should remove that reference when it's sessionEnding (ISession session) is called. Otherwise, the Session can not be garbage collected, which is a memory leak.

Structural Customizations

Providing Source Tabs

Source tabs can be added for specific node types by using the IObjectTreeAPI interface. What follows is example of an implementation of the ISessionPlugin interface method called sessionStarted (ISession session)

```
    public PluginSessionCallback sessionStarted(final ISession session)
 {
        IObjectTreeAPI treeApi =
session.getSessionInternalFrame().getObjectTreeAPI();
        _treeAPI.addDetailTab(DatabaseObjectType.VIEW, new
ViewSourceTab("Source");
    }
```

A reference to every newly created Session can be obtained by implementing the sessionStarted method in the ISessionPlugin interface. Notice that the "ViewSourceTab" is only installed for DatabaseObjectType.VIEW objects. This is so that our implementation of a "view" source tab can rely on the fact that it's always a view that the user has selected. In this way, different source tabs can be implemented for each type of object (table, view, trigger, UDT, sequence, procedure, function, etc.)

A source tab can then be implemented by extending the classes:

```
net.sourceforge.squirrel_sql.client.session.mainpanel.objecttree.tabs.BaseSourceTab

net.sourceforge.squirrel_sql.client.session.mainpanel.objecttree.tabs.BaseSourcePanel
```

In particular, BaseSourceTab has an abstract method called createStatement which is implemented to return a PreparedStatement by using BaseObjectTab.getSession() and BaseObjectTab.getDatabaseObjectInfo(). The method getDatabaseObjectInfo returns the node that is selected from which can be found the name, schema and catalog of the object to write an SQL statement that queries the data dictionary for the source for that object. The BaseSourcePanel.load method is then called with the ResultSet obtained by executing the PreparedStatement.

Custom data type implementations

Data Type classes are used by SQuirreL to read, render and display data stored in a column of a database table. Type codes are assigned (by JDBC as well as vendors) to represent standard and vendor-specific SQL types. The standard types are defined in the various SQL standards (SQL92, SQL:2003). These include types such as DATE, VARCHAR, CHAR, etc.

SQuirreL defines the IDataTypeComponent interface which encapsulates behavior necessary to read, write and render data of a given type. The SQuirreL framework module (located in sql12/fw/src) includes many implementations of this interface. Some examples are DataTypeBlob, DataTypeChar, DataTypeFloat and DataTypeDate. These implementations can be used as examples when a plugin writer wishes to support a data types that are only found in one database.

The CellComponentFactory class has a static method called registerDataType that allows plugin writers to install custom IDataTypeComponent implementations when the plugin is loaded.

Behavioral Customizations

Changing How SQuirreL Tokenizes a Script into Individual Statements

Native database tools that are used to run scripts must have a way of breaking a script into statements. Since statements can span lines, the end of line character (eol) normally isn't used for this purpose. For example, Oracle uses the semi-colon (;), whereas Sybase and MS SQL-Server use the word "GO" to distinguish multiple statements in a script. Users don't want to change their scripts to work with SQuirreL, so it is important for a plugin writer to consider customizing SQuirreL's default expectation of ";" if that is indeed not the statement separator used by the native database script interpreter. Other examples that require custom handling are:

- Stored Procedures (These can have embedded statement separators and may use other characters for delimiters - for example Oracle uses slash (/)
- References to external scripts (Oracle uses "@<script_location" to pull in the contents of another script during execution)

Plugins should implement the interface IQueryTokenizer and call the method ISession.setQuerytokenizer to add this capability to SQuirreL.

Formatting Vendor-Extended SQLExceptions

With some JDBC drivers, SQLException is extended to provide "enhanced" reporting of errors. In this case, the SQLException that is thrown by the driver may be cast to the vendor-specific class and non-standard methods may be called to get more specific information about the problem that caused the SQLException to be thrown. For example, the DB2 Universal Database Driver provides a DB2Diagnosable exception which extends SQLException. In some cases, the database will provide information that is only available using the DB2Diagnosable interface methods. According to the documentation [1] SQLException must be cast to a DB2Diagnosable, then call getSqlca() to get a DB2Sqlca object upon which a call to getMessage() will give the error message such as "Table foo does not exist" - when the exception results from selecting from a non-existent table.

In order for SQuirreL to facilitate this vendor-specific API, a new plugin API was introduced, known as the ExceptionFormatter. Each Session can have exactly one custom ExceptionFormatter (default and one custom ExceptionFormatter). The ExceptionFormatter is installed by calling ISession.setExceptionFormatter. The ExceptionFormatter interface looks as follows:

```
    /**
     * Returns a boolean indicating whether or not this formatter can
format the
     * specified exception.
     *
     * @param t the exception to determine formatting compatibility
     *
     * @return a boolean value to indicate whether format should be
called on the
     *         given throwable
     */
    boolean formatsException(Throwable t);

    /**
     *
     * @param t the exception to be formatted
     *
     * @return the message
     */
    String format(Throwable t);
```

A custom ExceptionFormatter implementation should be reflective in that it shouldn't reference any database vendor-specific classes directly, but indirectly using Class.forName(). This is required so that merely installing the plugin without the requisite driver classes won't cause the app to experience ClassNotFoundExceptions while loading the plugin. It is not required that the ExceptionFormatter handle all Throwables. Whichever Throwables aren't handled by the custom ExceptionFormatter, will be handled by the default ExceptionFormatter called DefaultExceptionFormatter.

Component Access

Getting the Currently Selected ResultTab

1. When a session starts, save off a reference to the SessionManager:

```
    IApplication application = ISession.getApplication();
    SessionManager sessionManager = application.getSessionManager();
```

2. At some later point to get a reference to the currently selected result tab do the following:

```
    ISession currentSession = sessionManager.getActiveSession();
    ISQLPanelAPI sqlPanelApi =
currentSession.getSQLPanelAPIOfActiveSessionWindow();
    ISQLResultExecuter executer = sqlPanelApi.getSQLResultExecuter();
    IResultTab selectedResultTab = executer .getSelectedResultTab();
```

References

[1] http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/ad/tjvjcerr.htm

SQLPro SQL Client

SQLPro

Original author(s)	Vive Corp.
Stable release	2.0 / November 26, 2013
Operating system	Windows
Type	SQL Programming Tool, Programming tool, Database administration and automation
Website	[1] [1]

SQLPro is a proprietary, visual database management and development tool for multiple databases. SQLPro comes with a three-pane interface and has integrated SQL editor with many useful SQL editing features. SQLPro connects directly (using native drivers) to most popular database servers using one single interface. It supports native data source connection for Oracle, PostgreSQL, MySQL, Microsoft SQL Server, Microsoft Access and SQLite. Provides quick view of the database schema, table schema and results of the query execution.

Amenities include color-coding of the SQL, drag-and-drop of objects into the SQL pane to save you from typing their name, retrieval of SQL for things like stored procedures and triggers from the underlying database, and one-click creation of SELECT and INSERT statements. You can save SQL files and print result sets.

Feature Summary

- SQLPro runs on all 32-bit Windows platforms, including Windows 95, 98, NT, 2000, XP, and Vista.
- No specific hardware requirements.
- Color coding of SQL keywords
- Retrieve stored procedures, triggers and functions
- No other software or third party drivers to install. SQLPro comes with full set of drivers.
- Multiple host/database connections.

Supported Databases

- Microsoft Access
- Microsoft SQL Server
- MySQL
- Oracle 8i, 9i, 10g and 11g
- PostgreSQL
- SQLite

External links

- Official Web Site [2]
- Review: SQLPro [3]

References

[1] http://www.vive.net/products/sqlpro.htm
[2] http://www.vive.net/
[3] http://www.larkware.com/Reviews/sqlpro.html

Navicat

Navicat

Original author(s)	PremiumSoft CyberTech Ltd.
Developer(s)	PremiumSoft CyberTech Ltd.
Initial release	2002
Operating system	Cross-platform
Available in	Multilingual (8)
Type	SQL database management and development system
License	Proprietary / Shareware
Website	www.navicat.com [1]

Navicat is a series of graphical database management and development software produced by PremiumSoft CyberTech Ltd. for MySQL, MariaDB, Oracle, SQLite, PostgreSQL and Microsoft SQL Server. It has an Explorer-like graphical user interface and supports multiple database connections for local and remote databases. Its design is made to meet the needs of a variety of audiences, from database administrators and programmers to various businesses/companies that serve clients and share information with partners.[2][3]

History

The initial version of Navicat was developed in 2001. The main target of the initial version was to simplify the management of MySQL installations. In 2008, Navicat for MySQL was the winner of the Hong Kong ICT 2008 *Award of the Year*, *Best Business Grand Award* and *Best Business (Product) Gold Award*.[4]

Supported platforms and languages

Navicat is a cross-platform tool and works on Microsoft Windows, Mac OS X and Linux platforms. Upon purchase, users are able to select a language for the software from eight available languages: English, French, Spanish, Japanese, Korean, Polish, Simplified Chinese and Traditional Chinese.

Development

Navicat for MySQL

Officially released in March 2002, the Windows version of Navicat for MySQL became the first product offered to the public by PremiumSoft. Subsequently, the company released two additional versions of Navicat for MySQL on the Mac OS X and Linux operating system in June and October 2003 respectively. In November 2013, added the support of MariaDB.[5][6]

Navicat for PostgreSQL

PremiumSoft continued to expand their Navicat series by releasing Navicat for PostgreSQL for Windows in October 2005 and then for Mac OS X in June 2006. The Linux version of Navicat for PostgreSQL would not be released until 3 years later in August 2009.[7]

Navicat for Oracle

In August 2008 Navicat decided to further continue their product line and branch out into the Oracle community, creating Navicat for Oracle for Windows and Mac. In August of the following year they followed up with a version for the Linux Platform.[8] The Oracle version of Navicat supports most of the latest Oracle objects features including Directory, Tablespace, Synonym, Materialized View, Trigger, Sequence, and Type, etc.[9][10]

Navicat for SQLite

Navicat for SQLite was released for Windows and Mac OS X simultaneously in April 2009, and the Linux version soon followed two months later in June of the same year.[11] In April 2010, Navicat Premium began including Navicat for SQLite starting from version 9 to expand the usability of Navicat Premium.[12]

Navicat Premium

In 2009, PremiumSoft released Navicat Premium, a series of Navicat software that combines all previous Navicat versions into a single version and can connect to different database types including MySQL, Oracle, and PostgreSQL simultaneously, allowing users to do data migration between cross databases. Navicat Premium version also supports cross-platform administration, serving Windows, Mac OS X and Linux. In April 2010, version 9 of Navicat Premium was released, which added the connectivity of SQLite database to Navicat Premium, allowing Navicat Premium to connect to MySQL, Oracle, PostgreSQL and SQLite in a single application. In November 2010, support for Microsoft SQL Server was added. In January 2011, SQL Azure was included. In November 2013, added the support of MariaDB.[13]

Navicat for SQL Server

Navicat for SQL Server was released in November 2010 for the Windows platform and Mac OS X. Also at the release, the SQL server version was included in the Premium version of Navicat. In January 2011, support for SQL Azure was added.[14]

Navicat Essentials

Navicat Essentials was officially released in November 2011. This is a simple Navicat version for commercial use. The Essentials editions of Navicat lack several features found in the Standard/Enterprise editions, including form view, record filtering, visual query building, data modeling and options for import, export and backup of data, etc.[15]

Navicat Data Modeler

Navicat Data Modeler Windows version was officially released in March 2012. Then, Mac OS X and Linux version were released in May 2012 and June 2012. This is a standalone product for developers to create data models for MySQL, SQL Server, Oracle, PostgreSQL and SQLite databases. Navicat Data Modeler allows users to visually design database structures, perform reverse/forward engineer process, import table structures from ODBC data sources, generate SQL files and print models to files, etc.

Navicat for MariaDB

MariaDB is currently the newest addition to the list of database Navicat supports. The new line of product, called Navicat for MariaDB, was released in November 2013 for the Windows, Mac OS X and Linux. It provides a native environment for MariaDB database management and supports the extra features like new storage engines, microsecond, virtual columns. Also at the release, the MariaDB version was included in both Navicat Premium and Navicat for MySQL.

Features

Navicat's features include:

- visual query-builder
- SSH and HTTP tunneling
- data and structure migration and synchronization[16]
- import and export and backup of data[17]
- report builder
- task scheduling and wizards tool

There are differences in the features available across operating systems.[18]

Navicat also supports forks of MySQL such as Drizzle, OurDelta, and Percona

Version History

Version	Platforms & Release dates							Some notable features
	MySQL	PostgreSQL	Oracle	SQLite	SQL Server	MariaDB	Premium	
4.x	• Windows: March 2002 • Mac OS X: June 2003 • Linux: No release	N/A	N/A	N/A	N/A	N/A	N/A	• Analyze, check, and repair tables • Visual user manager to administer users and privilege • Schedule backup file • Import and Export data from MS Excel and MS Access
5.x	• Windows: January 2003 • Mac OS X: January 2004 • Linux: September 2004	N/A	N/A	N/A	N/A	N/A	N/A	• Back up database and tables to SQL scripts • Execute SQL script • Multiple SQL editors with syntax highlighted feature
6.x	• Windows: April 2004 • Mac OS X: November 2005 • Linux: October 2005	• Mac OS X: June 2006	N/A	N/A	N/A	N/A	N/A	• Import data from ODBC • Connection to MySQL Server through SSH • Able to set schedule on profiles

7.x	• Windows: November 2005 • Mac OS X: November 2007	• Windows: October 2005 • Mac OS X: December 2007	• Mac OS X: August 2008	N/A	N/A	N/A	• Data and structure Synchronization • Able to drag and drop tables • Create parameter queries
8.x	• Windows: January 2008 • Mac OS X: February 2009 • Linux: January 2008	• Windows: January 2008 • Mac OS X: April 2009 • Linux: August 2009	• Windows: August 2008 • Mac OS X: April 2009 • Linux: August 2009	N/A	N/A	N/A	• Windows: June 2009 • Mac OS X: June 2009 • Linux: August 2009	• Virtual grouping • Server Monitor to change system variables • Allows form view • Able to preview SQL before execution
9.x	• Windows: April 2010 • Mac OS X: April 2010 • Linux: June 2010	• Windows: April 2010 • Mac OS X: April 2010 • Linux: June 2010	• Windows: April 2010 • Mac OS X: April 2010 • Linux: June 2010	• Windows: April 2010 • Mac OS X: April 2010 • Linux: June 2010	• Windows: November 2010 • Mac OS X: November 2010 • Linux: N/A	N/A	• Windows: April 2010 • Mac OS X: April 2010 • Linux: June 2010	• History log viewer • Allows unicode character reports • Import and export connections • Word completion for SQL editors
10.0	• Windows: September 2011 • Mac OS X: September 2011 • Linux: September 2011	• Windows: September 2011 • Mac OS X: September 2011 • Linux: September 2011	• Windows: September 2011 • Mac OS X: September 2011 • Linux: September 2011	• Windows: September 2011 • Mac OS X: September 2011 • Linux: September 2011	• Windows: September 2011 • Mac OS X: September 2011 • Linux: N/A	N/A	• Windows: September 2011 • Mac OS X: September 2011 • Linux: September 2011	• Data Modeling Tool • ER Diagram view • Database wide search • Minify SQL

10.1	• Windows: June 2012 • Mac OS X: June 2012 • Linux: June 2012	• Windows: June 2012 • Mac OS X: June 2012 • Linux: June 2012	• Windows: June 2012 • Mac OS X: June 2012 • Linux: June 2012	• Windows: June 2012 • Mac OS X: June 2012 • Linux: June 2012	• Windows: June 2012 • Mac OS X: June 2012 • Linux: N/A	N/A	• Windows: June 2012 • Mac OS X: June 2012 • Linux: June 2012	• Multiple schemas in one model • Enhanced Import from Database Wizard • Enhanced Synchronize to Database Wizard • Align/Distribute Diagram Objects • Search table in Model Designer
11.0	• Windows: April 2013 • Mac OS X: April 2013 • Linux: April 2013	• Windows: April 2013 • Mac OS X: April 2013 • Linux: April 2013	• Windows: April 2013 • Mac OS X: April 2013 • Linux: April 2013	• Windows: April 2013 • Mac OS X: April 2013 • Linux: April 2013	• Windows: April 2013 • Mac OS X: April 2013 • Linux: N/A	• Windows: November 2013 • Mac OS X: November 2013 • Linux: November 2013	• Windows: April 2013 • Mac OS X: April 2013 • Linux: April 2013	• New SSH Manager • Connection Tree Filter • Find and Replace data • Search Columns • Enhanced Import and Export Wizard • Native Windows and Mac 64-bit version

Navicat Community

Navicat Community provides a flexible environment for Navicat users to post questions, share experiences and solutions. It includes forum discussions, blog articles, tutorial videos and Wiki FAQ.

References

[1] http://www.navicat.com

[2] http://onlamp.com/pub/a/onlamp/2004/12/22/navicat.html

[3] http://www.lasplash.com/publish/Software_Reviews_and_News_130/navicat_mysql_gui_review.php

[4] Winner of the HKICT Awards 2008: Best Business Grand Award selected winner of HKICT Award of the Year (http://www.hkictawards.hk/ClientFolder/hkictawards/Library/Tree/doc_pdf/2008wl/BB.pdf)

[5] Navicat for MySQL release notes (http://www.navicat.com/en/products/navicat-for-mysql-release-note)

[6] http://www.techmixer.com/navicat-mysql-mysql-database-management-tools-for-windows

[7] Navicat for PostgreSQL release notes (http://navicat.com/en/products/navicat-for-postgresql-release-note)

[8] Navicat for Oracle release notes (http://navicat.com/en/products/navicat-for-oracle-release-note)

[9] Navicat for Oracle overview (http://navicat.com/en/products/navicat-for-oracle)

[10] http://www.techmixer.com/navicat-for-oracle-windows-oracle-administration-tools/

[11] Navicat for SQLite release notes (http://www.navicat.com/products/navicat-for-sqlite-release-note)

[12] Navicat Premium release notes (http://www.navicat.com/en/products/navicat-premium-release-note)

[13] http://www.techmixer.com/navicat-premium-cross-database-administrator-management-tool

[14] Navicat for SQL server overview (http://navicat.com/en/products/navicat-for-sqlserver)

[15] Navicat Essentials feature (http://navicat.com/en/products/navicat-essentials)

[16] http://www.webdotdev.com/nvd/content/view/438/

[17] http://ted.onflash.org/2005/04/navicat-review-mysql-administration.php

[18] Navicat Feature Matrix, for MySQL (http://www.navicat.com/products/navicat-for-mysql-feature-matrix), MariaDB (http://www.navicat.com/products/navicat-for-mariadb-feature-matrix), PostegreSQL (http://www.navicat.com/products/navicat-for-postgresql-feature-matrix), Oracle (http://www.navicat.com/products/navicat-for-oracle-feature-matrix), SQLite (http://www.navicat.com/products/navicat-for-sqlite-feature-matrix), SQL Server (http://www.navicat.com/products/navicat-for-sqlserver-feature-matrix), Premium (http://www.navicat.com/products/navicat-premium-feature-matrix)

External links

- Official website (http://www.navicat.com/)
- Navicat Community (https://community.navicat.com/)

ModelRight

ModelRight is a database design and data modeling tool developed by ModelRight Inc. It is used by data modelers, database developers and database architects to create, visualize, and document their databases as an Entity Relationship Diagram (ERD).

Features

- Reverse Engineer an existing database
- Generate SQL DDL to create a database
- Compare a data model and a database
- Compare a data model with another data model
- Document a database with notes, definitions, and revision notes
- Automation and scripting support
- Model subset management
- Generate HTML reports
- Supports large, complex data modeling and data warehousing projects
- Generate model contents as XML

Support

- IDEF1X
- Information Engineering (IE)/Crow's Foot Notation
- UML Class Diagrams
- Barker Notation

External links

- modelright.com [1]

References

[1] http://www.modelright.com/

Other DBMS

MySQL

MySQL

Screenshot of the default MySQL command line

Original author(s)	MySQL AB
Developer(s)	Oracle Corporation
Initial release	23 May 1995
Stable release	5.6.16 / 31 January 2014
Preview release	5.7.3 / 3 December 2013
Development status	Active
Written in	C, C++
Operating system	Windows, Linux, Solaris, OS X, FreeBSD
Available in	English
Type	RDBMS
License	GPL (version 2) or commercial[1]
Website	www.mysql.com [2]

MySQL (/maɪHelp:IPA for English#Key ˌɛskjuːˈɛl/ "My S-Q-L", officially, but also called /maɪHelp:IPA for English#Key ˈsiːkwəl/ "My Sequel") is (as of July 2013) the world's second most[3] In the second quarter of 2013 alone, 213 million smartphones shipped, of which 200 million were Android and iOS.</ref> widely used open-source relational database management system (RDBMS). It is named after co-founder Michael Widenius's daughter, My. The SQL phrase stands for Structured Query Language.

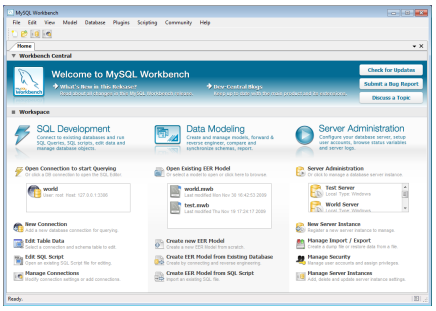
The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr, and YouTube.

Interfaces

MySQL is a relational database management system (RDBMS), and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools,[4][5] or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records.[6][7][8][9] The official set of MySQL front-end tools, MySQL Workbench is actively developed by Oracle, and is freely available for use.[10]



MySQL Workbench in Windows

Graphical

The official MySQL Workbench is a free integrated environment developed by MySQL AB, that enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL front end, MySQL Workbench lets users manage database design & modeling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).

MySQL Workbench is available in two editions, the regular free and open source *Community Edition* which may be downloaded from the MySQL website, and the proprietary *Standard Edition* which extends and improves the feature set of the Community Edition.

Third-party proprietary and free graphical administration applications (or "front ends") are available that integrate with MySQL and enable users to work with database structure and data visually. Some well-known front ends, in alphabetical order, are:

- Adminer – a free MySQL front end written in one PHP script, capable of managing multiple databases, with many CSS skins available.
- DaDaBIK – a customizable CRUD front-end to MySQL. Written in PHP. Commercial.
- DBEdit – a free front end for MySQL and other databases.
- HeidiSQL – a full featured free front end that runs on Windows, and can connect to local or remote MySQL servers to manage databases, tables, column structure, and individual data records. Also supports specialised GUI features for date/time fields and enumerated multiple-value fields.
- LibreOffice Base – LibreOffice Base allows the creation and management of databases, preparation of forms and reports that provide end users easy access to data. Like Microsoft Access, it can be used as a front-end for various database systems, including Access databases (JET), ODBC data sources, and MySQL or PostgreSQL.
- Navicat – a series of proprietary graphical database management applications, developed for Windows, Macintosh and Linux.
- OpenOffice.org – OpenOffice.org Base can manage MySQL databases if the entire suite is installed. Free and open-source.
- phpMyAdmin – a free Web-based front end widely installed[*citation needed*] by web hosts, since it is developed in PHP and is included in the LAMP stack, MAMP, XAMPP and WAMP software bundle installers.
- SQLBuddy – a free Web-based front end, developed in PHP.
- SQLyog – commercial, but there is also a free 'community' edition available.
- Toad for MySQL – a free development and administration front end for MySQL from Quest Software

Other available proprietary MySQL front ends include dbForge Studio for MySQL, DBStudio, Epictetus, Microsoft Access, Oracle SQL Developer, SchemaBank, SQLPro SQL Client, Toad Data Modeler.

Command line

MySQL ships with many command line tools, from which the main interface is 'mysql' client. Third-parties have also developed tools to manage, optimize, monitor and backup a MySQL server, some listed below. All these tools work on *NIX type operating systems, and some of them also on Windows.

- Maatkit – a cross-platform toolkit for MySQL, PostgreSQL and Memcached, developed in Perl. Maatkit can be used to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up servers. Maatkit is included with several Linux distributions such as CentOS and Debian and packages are available for Fedora and Ubuntu as well. As of late 2011, Maatkit is no longer developed, but Percona has continued development under the Percona Toolkit brand.[11]
- XtraBackup – Open Source MySQL hot backup software. Some notable features include hot, non-locking backups for InnoDB storage, incremental backups, streaming, parallel-compressed backups, throttling based on the number of IO operations per second, etc.
- MySQL::Replication – a replacement for MySQL's built-in replication, developed in Perl. MySQL::Replication can be used to create a peer-to-peer, multi-master MySQL replication network.
- SymmetricDS – asynchronous MySQL multi-master replication capable of multi-tier replication and designed for a large number of databases across low-bandwidth connections. Licensed as open source (GPL) with commercial options from JumpMind.

Programming

MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the JDBC driver for Java. In addition, an ODBC interface called MyODBC allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The HTSQL – URL-based query method also ships with a MySQL adapter, allowing direct interaction between a MySQL database and any web client via structured URLs.

Features

As of April 2009[12], MySQL offered MySQL 5.1 in two different variants: the open source MySQL Community Server and the commercial Enterprise Server. MySQL 5.5 is offered under the same licenses. They have a common code base and include the following features:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM
- Triggers
- Cursors
- Updatable views
- Information schema
- Strict mode (ensures MySQL does not truncate or otherwise modify data to conform to an underlying data type, when an incompatible value is inserted into that type)
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using Oracle's InnoDB engine

- Independent storage engines (MyISAM for read speed, InnoDB for transactions and referential integrity, MySQL Archive for storing historical data in little space)
- Transactions with the InnoDB and NDB Cluster storage engines; savepoints with InnoDB
- SSL support
- Query caching
- Sub-SELECTs (i.e. nested SELECTs)
- Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master. Multi-master replication is provided in MySQL Cluster, and multi-master support can be added to unclustered configurations using Galera Cluster.
- Full-text indexing and searching using MyISAM engine
- Embedded database library
- Unicode support (however prior to 5.5.3 UTF-8 and UCS-2 encoded strings are limited to the BMP, in 5.5.3 and later use utf8mb4 for full unicode support)
- ACID compliance when using transaction capable storage engines (InnoDB and Cluster)
- Partitioned tables with pruning of partitions in optimizer
- Shared-nothing clustering through MySQL Cluster
- Hot backup (via `mysqlhotcopy`) under certain conditions
- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application (in MySQL 5.0, storage engines must be compiled in; in MySQL 5.1, storage engines can be dynamically loaded at run time):
  - Native storage engines (MyISAM, Falcon, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, Cluster, EXAMPLE, Aria, and InnoDB, which was made the default as of 5.5)
  - Partner-developed storage engines (solidDB, Infobright (formerly Brighthouse), Kickfire, XtraDB, IBM DB2). InnoDB used to be a partner-developed storage engine, but with recent acquisitions, Oracle now owns both MySQL core and InnoDB.
  - Community-developed storage engines (memcache engine, httpd, PBXT, Revision Engine)
  - Custom storage engines
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second. (PostgreSQL has an advanced form of this functionality)

The developers release monthly versions of the MySQL Server. The sources can be obtained from MySQL's website or from MySQL's Bazaar repository, both under the GPL license.

Limitations

Like other SQL databases, MySQL does not currently comply with the full SQL standard for some of the implemented functionality, including foreign key references when using some storage engines other than the 'standard' InnoDB (or third-party engines which supports foreign keys).

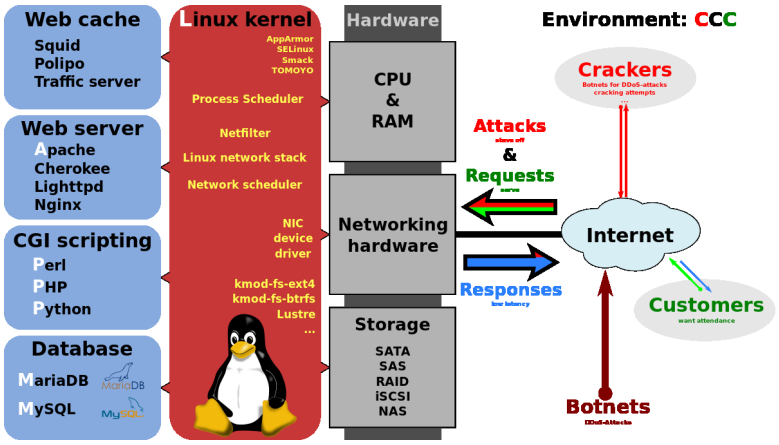
Triggers are currently limited to one per action / timing, i.e. maximum one after insert and one before insert on the same table. There are no triggers on views.

MySQL, like most other transactional relational databases, is strongly limited by hard disk performance. This is especially true in terms of write latency. Given the recent appearance of very affordable consumer grade SATA interface solid-state drives that offer zero mechanical latency, a fivefold speedup over even an eight drive RAID array can be had for a smaller investment.[13]

MySQL database's inbuilt functions like UNIX_TIMESTAMP() will return 0 after 03:14:07 UTC on 19 January 2038.

Deployment

MySQL can be built and installed manually from source code, but this can be tedious so it is more commonly installed from a binary package unless special customizations are required. On most Linux distributions the package management system can download and install MySQL with minimal effort, though further configuration is often required to adjust security and optimization settings.



The LAMP software bundle (here additionally with Squid). A high performance and high-availability solution for a hostile environment

Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well. It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP-based web application or as a standalone database server. Much of MySQL's appeal originates in its relative simplicity and ease of use, which is enabled by an ecosystem of open source tools such as phpMyAdmin. In the medium range, MySQL can be scaled by deploying it on more powerful hardware, such as a multi-processor server with gigabytes of memory.

There are however limits to how far performance can scale on a single server ('scaling up'), so on larger scales, multi-server MySQL ('scaling out') deployments are required to provide improved performance and reliability. A typical high-end configuration can include a powerful master database which handles data write operations and is replicated to multiple slaves that handle all read operations. The master server synchronizes continually with its slaves so in the event of failure a slave can be promoted to become the new master, minimizing downtime. Further improvements in performance can be achieved by caching the results from database queries in memory using memcached, or breaking down a database into smaller chunks called shards which can be spread across a number of distributed server clusters.

High availability

Ensuring high availability requires a certain amount of redundancy in the system. For database systems, the redundancy traditionally takes the form of having a primary server acting as a master, and using replication to keep secondaries available to take over in case the primary fails. This means that the "server" that the application connects to is in reality a collection of servers, not a single server. In a similar manner, if the application is using a sharded database, it is in reality working with a collection of servers, not a single server. In this case, a collection of servers is usually referred to as a *farm*.

One of the projects aiming to provide high availability for MySQL is *MySQL Fabric*, an integrated system for managing a collection of MySQL servers, and a framework on top of which high availability and database sharding is built. MySQL Fabric is open-source and is intended to be extensible, easy to use, and to support procedure execution even in the presence of failure, providing an execution model usually called *resilient execution*. MySQL client libraries are extended so they are hiding the complexities of handling failover in the event of a server failure, as well as correctly dispatching transactions to the shards. As of September 2013, there is currently support for Fabric-aware versions of Connector/J, Connector/PHP, Connector/Python, as well as some rudimentary support for Hibernate and Doctrine. As of March 2014, MySQL Fabric is in a beta stage of development.

Cloud deployment

MySQL can also be run on cloud computing platforms such as Amazon EC2. Listed below are some common deployment models for MySQL on the cloud:

- Virtual Machine Image – cloud users can upload a machine image of their own with MySQL installed, or use a ready-made machine image with an optimized installation of MySQL on it, such as the one provided by Amazon EC2.
- MySQL as a Service – some cloud platforms offer MySQL "as a service". In this configuration, application owners do not have to install and maintain the MySQL database on their own. Instead, the database service provider takes responsibility for installing and maintaining the database, and application owners pay according to their usage. Notable cloud-based MySQL services are the Amazon Relational Database Service; the Xeround Cloud Database, which runs on EC2; Rackspace; HP Converged Cloud; Heroku and Jelastic.
- Managed MySQL cloud hosting – the database is not offered as a service, but the cloud provider hosts the database and manages it on the application owner's behalf. As of 2011, of the major cloud providers, only Terremark and Rackspace offer managed hosting for MySQL databases.

Community

The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, beginning from 28 June 2000 (which in 2009 has been extended with a FLOSS License Exception) or to use a proprietary license. In 2013, the Linux man pages for MySQL switched to a proprietary license temporarily due to a build system bug. The original GPL license has since been restored.

Support can be obtained from the official manual.[14] Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and in price. Additionally, a number of third party organisations exist to provide support and services, including SkySQL Ab and Percona.

MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case." and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good".[15] It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server".[16]

Related projects

- Drizzle – a fork targeted at the web-infrastructure and cloud computing markets. The developers of the product describe it as a "smaller, slimmer and (hopefully) faster version of MySQL". As a result, many common MySQL features will be stripped out, including stored procedures, query cache, prepared statements, views, and triggers. This is a partial rewrite of the server that does not maintain compatibility with MySQL.
- MariaDB – a community-developed fork of the MySQL database source code, the impetus being the community maintenance of its free status under GPL as opposed to any uncertainty of MySQL license status under its current ownership by Oracle. The intent also being to maintain high fidelity with MySQL, ensuring a "drop-in" replacement capability with library binary equivalency and exact matching with MySQL APIs and commands. It includes the XtraDB storage engine as a replacement for InnoDB.
- Percona Server – a fork of MySQL that includes the XtraDB storage engine. Its policy is to deviate as little as possible from MySQL and remain fully compatible, while providing new features, better performance, and additional instrumentation for analysis of performance and usage.
- OurDelta – No longer maintained. It was a fork compiled with various patches, including patches from MariaDB, Percona, and Google, and a storage engine called OQGRAPH.

History

MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from mSQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as mSQL. By keeping the API consistent with the mSQL system, many developers were able to use MySQL instead of the (commercially licensed) mSQL antecedent.[*citation needed*]Wikipedia:Disputed statement

Legal and acquisition impacts

On 15 June 2001, NuSphere sued MySQL AB, TcX DataKonsult AB and its original authors Michael ("Monty") Widenius and David Axmark in U.S District Court in Boston for "breach of contract, tortious interference with third party contracts and relationships and unfair competition".

In 2002, MySQL AB sued Progress NuSphere for copyright and trademark infringement in United States district court. NuSphere had allegedly violated MySQL's copyright by linking MySQL's GPL'ed code with NuSphere Gemini table without being in compliance with the license. After a preliminary hearing before Judge Patti Saris on 27 February 2002, the parties entered settlement talks and eventually settled. After the hearing, FSF commented that "Judge Saris made clear that she sees the GNU GPL to be an enforceable and binding license."

In October 2005, Oracle Corporation acquired Innobase OY, the Finnish company that developed the third-party InnoDB storage engine that allows MySQL to provide such functionality as transactions and foreign keys. After the acquisition, an Oracle press release mentioned that the contracts that make the company's software available to MySQL AB would be due for renewal (and presumably renegotiation) some time in 2006. During the MySQL Users Conference in April 2006, MySQL issued a press release that confirmed that MySQL and Innobase OY agreed to a "multi-year" extension of their licensing agreement.

In February 2006, Oracle Corporation acquired Sleepycat Software, makers of the Berkeley DB, a database engine providing the basis for another MySQL storage engine. This had little effect, as Berkeley DB was not widely used, and was dropped (due to lack of use) in MySQL 5.1.12, a pre-GA release of MySQL 5.1 released in October 2006.

In January 2008, Sun Microsystems bought MySQL for $1 billion.

In April 2009, Oracle Corporation entered into an agreement to purchase Sun Microsystems, then owners of MySQL copyright and trademark. Sun's board of directors unanimously approved the deal, it was also approved by Sun's shareholders, and by the U.S. government on 20 August 2009. On 14 December 2009, Oracle pledged to continue to enhance MySQL as it had done for the previous four years.

A movement against Oracle's acquisition of MySQL, to "Save MySQL" from Oracle was started by one of the MySQL founders, Monty Widenius. The petition of 50,000+ developers and users called upon the European Commission to block approval of the acquisition. At the same time, several Free Software opinion leaders (including Eben Moglen, Pamela Jones of Groklaw, Jan Wildeboer and Carlo Piana, who also acted as co-counsel in the merger regulation procedure) advocated for the unconditional approval of the merger.[*citation needed*] As part of the negotiations with the European Commission, Oracle committed that MySQL server will continue until at least 2015 to use the dual-licensing strategy long used by MySQL AB, with commercial and GPL versions available. The antitrust of the EU had been "pressuring it to divest MySQL as a condition for approval of the merger". But, as revealed by Wikileaks, the US Department of Justice and Antitrust, at the request of Oracle, pressured the EU to unconditionally approve the merger. The European Commission eventually unconditionally approved Oracle's acquisition of MySQL on 21 January 2010.

In January 2009, prior to Oracle's acquisition of MySQL, Monty Widenius started a GPL-only fork, MariaDB. MariaDB is based on the same code base as MySQL server 5.1 and strives to maintain compatibility with Oracle-provided versions.

In August 2012, TechCrunch's Alex Williams reported that Oracle was holding back MySQL Server test cases, a move that he concluded indicated that Oracle is attempting to kill the product. Percona also reported that Oracle is no longer synchronizing their changes with the public source repositories. Widenius called this a breach of the agreement that Oracle entered into with the EU as a condition of their acquisition of Sun.

Since the final quarter of 2012, several Linux distributions and some important users (like Wikipedia[17] and Google[18]) started to replace MySQL with MariaDB.[19][20][21]Wikipedia:Disputed statement

Milestones

Notable milestones in MySQL development include:

- Original development of MySQL by Michael Widenius and David Axmark beginning in 1994
- First internal release on 23 May 1995
- Version 3.19: End of 1996, from www.tcx.se
- Version 3.20: January 1997
- Windows version was released on 8 January 1998 for Windows 95 and NT
- Version 3.21: production release 1998, from www.mysql.com
- Version 3.22: alpha, beta from 1998
- Version 3.23: beta from June 2000, production release 22 January 2001
- Version 4.0: beta from August 2002, production release March 2003 (unions)
- Version 4.01: beta from August 2003, JyotiWikipedia:Please clarify[*citation needed*] adopts MySQL for database tracking
- Version 4.1: beta from June 2004, production release October 2004 (R-trees and B-trees, subqueries, prepared statements)
- Version 5.0: beta from March 2005, production release October 2005 (cursors, stored procedures, triggers, views, XA transactions)

  The developer of the Federated Storage Engine states that "The Federated Storage Engine is a proof-of-concept storage engine", but the main distributions of MySQL version 5.0 included it and turned it on by default. Documentation of some of the short-comings appears in "MySQL Federated Tables: The Missing Manual".
- Sun Microsystems acquired MySQL AB in 2008.
- Version 5.1: production release 27 November 2008 (event scheduler, partitioning, plugin API, row-based replication, server log tables)

  Version 5.1 contained 20 known crashing and wrong result bugs in addition to the 35 present in version 5.0 *(almost all fixed as of release 5.1.51)*.

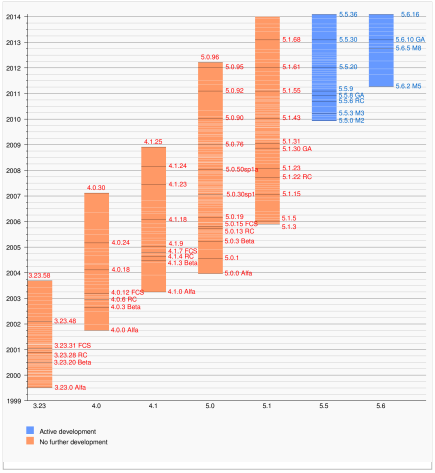
  MySQL 5.1 and 6.0 showed poor performance when used for data warehousing — partly due to its inability to utilize multiple CPU cores for processing a single query.
- Oracle acquired Sun Microsystems on 27 January 2010.
- MySQL Server 5.5 is currently generally available (as of December 2010[12]). Enhancements and features include:
  - The default storage engine is InnoDB, which supports transactions and referential integrity constraints.
  - Improved InnoDB I/O subsystem
  - Improved SMP support
  - Semisynchronous replication.
  - SIGNAL and RESIGNAL statement in compliance with the SQL standard.
  - Support for supplementary Unicode character sets utf16, utf32, and utf8mb4.
  - New options for user-defined partitioning.

MySQL Server 6.0.11-alpha was announced on 22 May 2009 as the last release of the 6.0 line. Future MySQL Server development uses a New Release Model. Features developed for 6.0 are being incorporated into future releases.

MySQL 5.6, a development milestone release, was announced at the MySQL users conference 2011. New features include performance improvements to the query optimizer, higher transactional throughput in InnoDB, new NoSQL-style memcached APIs, improvements to partitioning for querying and managing very large tables, improvements to replication and better performance monitoring by expanding the data available through the PERFORMANCE_SCHEMA. In July further previews with a BINLOG API, group commit, and InnoDB full text searching were released.

Versions

The following chart provides an overview of various MySQL versions and their current development statuses:



Notes

[1] http://www.mysql.com/downloads/

[2] http://www.mysql.com/

[3] Following Sqllite, which is deployed with every iPhone and Android device along with the Chrome and Firefox browsers.<ref name="sqllite">

[4] mysql — The MySQL Command-Line Tool (http://dev.mysql.com/doc/refman/5.5/en/mysql.html), MySQL Reference Manual

[5] mysqladmin – the MySQL command-line tool (http://dev.mysql.com/doc/refman/5.5/en/mysqladmin.html), MySQL Reference Manual

[6] MySQL Client Programs (http://dev.mysql.com/doc/refman/5.0/en/programs-client.html), MySQL Reference Manual

[7] MySQL Tools Family (http://www.sqlmaestro.com/products/mysql/), SQLMaestro Group

[8] MySQL GUI Tools (http://www.webyog.com/en/), WebYog

[9] HeidiSQL (http://www.heidisql.com/), HeidiSQL MySQL GUI

[10] MySQL Workbench (http://dev.mysql.com/downloads/workbench/), MySQL Downloads

[11] Percona Toolkit (http://www.percona.com/software/percona-toolkit), Percona Software

[12] http://en.wikipedia.org/w/index.php?title=MySQL&action=edit

[13] Matsunobu, Yoshinori. "SSD Deployment Strategies for MySQL." (http://www.slideshare.net/matsunobu/ssd-deployment-strategies-for-mysql) *Sun Microsystems*, 15 April 2010.

[14] MySQL Support Manual (http://dev.mysql.com/doc/refman/5.5/en/index.html), MySQL Developers

[15] Review of MySQL Server 5.0 (http://review.techworld.com/applications/346/mysql-50-open-source-database/), Techworld.com, November 2005

[16] MySQL Server Review (http://community.linuxmint.com/software/view/mysql-server), LinuxMint.com

[17] Wikipedia Adopts MariaDB (https://blog.wikimedia.org/2013/04/22/wikipedia-adopts-mariadb/)

[18] http://www.theregister.co.uk/2013/09/12/google_mariadb_mysql_migration/

[19] Monty has last laugh as distros abandon MySQL (http://www.itwire.com/opinion-and-analysis/open-sauce/58571-monty-has-last-laugh-as-distros-abandon-mysql)

[20] MariaDB living in interesting times (http://monty-says.blogspot.it/2013/02/mariadb-living-in-interesting-times.html)

[21] Distributions Which Include MariaDB (https://kb.askmonty.org/en/distributions-which-include-mariadb/)

References

External links

- Official website (http://www.mysql.com)
- MySQL site at Oracle.com (http://www.oracle.com/us/products/mysql/index.html)
- Interview with David Axmark, MySQL co-founder (http://intruders.tv/en-tech/mysql-co-founder-david-axmark-on-sun-s-billion-dollar-acquisition/) Video
- MySQL (http://www.dmoz.org//Computers/Software/Databases/MySQL/) on the Open Directory Project

SQL Anywhere

SAP SQL Anywhere is a relational database management system (RDBMS) product from SAP. SQL Anywhere was known as Sybase SQL Anywhere prior to the acquisition of Sybase by SAP.

Features

SQL Anywhere can be run on Windows, Windows CE, Mac OS X, and various UNIX platforms, including Linux, AIX, HP-UX and Solaris. Database files are independent of the operating system, allowing them to be copied between supported platforms. The product provides several standard interfaces (ODBC, JDBC, and ADO.NET) and a number of special interfaces such as PHP and Perl. The engine supports stored procedures, user functions (using Watcom SQL, T-SQL, Java, or C/C++), triggers, referential integrity, row-level locking, replication, high availability, proxy tables, and events (scheduled and system events). Strong encryption is supported for both database files and client-server communication.

Uses

SQL Anywhere is used in several contexts, including as an embedded database, particularly as an application data store. For example, it is used in Intuit QuickBooks, in network management products, and in backup products. Its ability to be used with minimal administration is a distinguishing feature in this role. It can be used as a database server for work groups or for small or medium-sized businesses. It can also function as a mobile database. It includes scalable data synchronization technology that provides change-based replication between separate databases, including large server-based RDBMS systems.

Technologies

SQL Anywhere Server is a high performing and embeddable relational database-management system (RDBMS) that scales from thousands of users in server environments down to desktop and mobile applications used in widely deployed, zero-administration environments.

Ultralite: UltraLite is a database-management system designed for small-footprint mobile devices such as PDAs and smart phones.

Mobilink: MobiLink is a highly-scalable, session-based synchronization technology for exchanging data among relational databases and other non-relational data sources.

QAnywhere: QAnywhere facilitates the development of robust and secure store-and-forward mobile messaging applications.

SQL Remote: SQL Remote technology is based on a store and forward architecture that allows occasionally connected users to synchronize data between SQL Anywhere databases using a file or message transfer mechanism.

History

- Initially created by Watcom as Watcom SQL.
- Version 3: 1992
- Watcom acquired by Powersoft in 1993; Watcom SQL shipped with their visual programming environment PowerBuilder
- Version 4: 1994 (Stored procedures, triggers)
- PowerSoft and Sybase merged in 1995: Watcom SQL was renamed **SQL Anywhere**.
- Version 5: 1995 (SQL Remote data replication; graphical administration tools)
- Version 6: 1998. Renamed **Adaptive Server Anywhere**. (multi-processor support, Java objects in the database)
- Version 6.0.2: 1999 (MobiLink data synchronization, UltraLite mobile database for Palm OS and Windows CE)
- Version 7: 2000 (dynamic cache, task scheduling and event handling, cross-platform administration tools)
- Version 8: 2001 (Volcano query optimizer, encrypted data storage and transmission)
- Version 9: 2003 (Index consultant, embedded HTTP server)
- Version 10: 2006 - renamed **SQL Anywhere** (high availability, intra-query parallelism, materialized views)
- Version 11: 2008 (full text search, BlackBerry support)
- Version 12: 2010 (support for spatial data)[1]
- Version 16: April 18, 2013 - (faster synchronization and improved security) [2]

References

[1] http://www.sybase.com/files/White_Papers/SQLAnywhere_12_New-Features_WP.pdf
[2] http://www.kessler.de/prd/sybase/Ianywhere16.pdf

External links

- SQL Anywhere official product page (http://www.sybase.com/products/databasemanagement/sqlanywhere)
- Sybase iAnywhere main page (http://www.sybase.com/ianywhere)

SQLite

SQLite

Developer(s)	D. Richard Hipp
Initial release	August 2000
Stable release	3.8.4 (March 10, 2014) [±] [1]
Written in	C
Operating system	Cross-platform
Size	658 KiB
Type	RDBMS (embedded)
License	Public domain
Website	sqlite.org [2]

SQLite (/ˌɛskjuːɛlˈlaɪt/ or /ˈsiːkwəl.laɪt/) is a relational database management system contained in a C programming library. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SQLite is a popular choice as embedded database for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others. SQLite has many bindings to programming languages.

The source code for SQLite is in the public domain.

Design

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. (In this, SQLite follows the precedent of Informix SE of c. 1984 [3]) The library can also be called dynamically. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication. SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

History

D. Richard Hipp designed SQLite in the spring of 2000 while working for General Dynamics on contract with the United States Navy. Hipp was designing software used on board guided missile destroyers, which were originally based on HP-UX with an IBM Informix database back-end. The design goals of SQLite were to allow the program to be operated without installing a database management system or requiring a database administrator. In August 2000, version 1.0 of SQLite was released, based on gdbm (GNU Database Manager). SQLite 2.0 replaced gdbm with a custom B-tree implementation, adding support for transactions. SQLite 3.0, partially funded by America Online, added internationalization, manifest typing, and other major improvements.

In 2011 Hipp announced his plans to add an UnQL interface to SQLite databases and to develop *UnQLite*, an embeddable document-oriented database. Howard Chu ported SQLite 3.7.7.1 to use Openldap MDB instead of the original Btree code and called it sqlightning. One cited insert test of 1000 records was 20 times faster.[4][5]

Features

SQLite implements most of the SQL-92 standard for SQL but it lacks some features. For example it has partial support for triggers, and it can't write to views (however it supports INSTEAD OF triggers that provide this functionality). While it supports complex queries, it still has limited ALTER TABLE support, as it can't modify or delete columns.

SQLite uses an unusual type system for an SQL-compatible DBMS; instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is *dynamically typed*. Moreover, it is *weakly typed* in some of the same ways that Perl is: one can insert a string into an integer column (although SQLite will try to convert the string to an integer first, if the column's preferred type is integer). This adds flexibility to columns, especially when bound to a dynamically typed scripting language. However, the technique is not portable to other SQL products. A common criticism is that SQLite's type system lacks the data integrity mechanism provided by statically typed columns in other products. The SQLite web site describes a "strict affinity" mode, but this feature has not yet been added. However, it can be implemented with constraints like `CHECK(typeof(x)='integer')`.

Several computer processes or threads may access the same database concurrently. Several read accesses can be satisfied in parallel. A write access can only be satisfied if no other accesses are currently being serviced. Otherwise, the write access fails with an error code (or can automatically be retried until a configurable timeout expires). This concurrent access situation would change when dealing with temporary tables. This restriction is relaxed in version 3.7 when WAL is turned on enabling concurrent reads and writes.

A standalone program called `sqlite3` is provided that can be used to create a database, define tables within it, insert and change rows, run queries and manage an SQLite database file. This program is a single executable file on the host machine. It also serves as an example for writing applications that use the SQLite library.

SQLite full Unicode support is optional.

SQLite has automated regression testing prior to each release. Over 2 million tests are run as part of a release's verification. Starting with the August 10, 2009 release of SQLite 3.6.17, SQLite releases have 100% branch test coverage, one of the components of code coverage.

As of version 3.8.2 it's possible to create tables without rowid.[6]

Development

SQLite development stores revisions of its source code in Fossil, a distributed version control system that is itself built upon an SQLite database.

Adoption

Programming languages

SQLite has bindings for a large number of programming languages, including :

• BASIC	• Haskell	• PureBasic
• Delphi	• Java	• Python[7]
• C	• JavaScript[8]	• R
• C#	• Livecode	• REALbasic
• C++	• Lua	• REBOL
• Clipper//Harbour	• newLisp	• Ruby[9]
• Common Lisp	• Objective-C (on OS X and iOS)	• Scheme
• Curl	• OCaml	• Smalltalk
• D	• Perl[10]	• Tcl
• Free Pascal	• PHP	• Visual Basic
	• Pike	

Middleware

- ADO.NET adapter, initially developed by Robert Simpson, is maintained jointly with the SQLite developers since April 2010.[11]
- ODBC driver has been developed and is maintained separately by Christian Werner.[12] Werner's ODBC driver is the recommended connection method for accessing SQLite from OpenOffice.org.[13]
- COM (ActiveX) wrapper making SQLite accessible on Windows to scripted languages such as JScript and VBScript. This adds database capabilities to HTML Applications (HTA).

Web browsers

- Mozilla Firefox and Mozilla Thunderbird store a variety of configuration data (bookmarks, cookies, contacts etc.) in internally managed SQLite databases, and even offer an add-on to manage SQLite databases.
- Google's Chrome browser
- The Opera Internet suite and browser uses SQLite 3.7.9 for managing WebSQL databases. This is noted in `opera:about`, although without the mention of WebSQL (databases can be managed through `opera:webdatabases`).

Web application frameworks

- Django, a Python web framework, supports SQLite3 by default.
- As of version 7, Drupal, a PHP-based content management system for making websites and blogs, has an option to install using SQLite.
- Ruby on Rails' default database management system is also SQLite.
- web2py, a Python web framework, default database management system is also SQLite.

Various

- Skype is a widely deployed application that uses SQLite.
- Adobe Systems uses SQLite as its file format in Adobe Photoshop Lightroom, a standard database in Adobe AIR, and internally within Adobe Reader.
- The Service Management Facility, used for service management within the Solaris and OpenSolaris operating systems, uses SQLite internally.
- Flame, a malware program used for cyberespionage, used SQLite to store the data it collects.
- The Xojo Programming Language has SQLite support built in to both the desktop and web frameworks.

Operating systems

Because of its small size, SQLite is well suited to embedded systems, and is also included in:

- Blackberry's BlackBerry 10 OS
- Microsoft's Windows Phone 8
- Apple's iOS
- Symbian OS
- Nokia's Maemo
- Google's Android
- Linux Foundation's MeeGo
- LG's webOS
- NetBSD
- OpenBSD
- FreeBSD where starting with 10-RELEASE version it is the core package management system.

However, it is also suitable for desktop operating systems; Apple adopted it as an option in OS X's Core Data API from the original implementation in Mac OS X 10.4 onwards, and also for administration of videos and songs on the iPhone.

Citations

[1] http://en.wikipedia.org/w/index.php?title=Template:Latest_stable_software_release/SQLite&action=edit
[2] http://sqlite.org/
[3] http://www.iiug.org/faqs/informix-faq/ifaq01.htm.1#1.2
[4] MDB: A Memory-Mapped Database and Backend for OpenLDAP (http://ldapcon.org/downloads/chu-paper.pdf), Howard Chu, [MDB: A Memory-Mapped Database and Backend for OpenLDAP LDAPCon 2011].
[5] sqlightning source code (http://gitorious.org/mdb/sqlightning#more).
[6] ReleaseLog (http://sqlite.org/news.html) SQLite.org, visited 8th December 2013
[7] PySQLite (http://trac.edgewall.org/wiki/PySqlite): Python bindings for SQLite
[8] JSPDO (http://code.google.com/p/v8-juice/wiki/JSPDO) JavaScript database access abstraction API
[9] SQLite/Ruby (http://rubyforge.org/projects/sqlite-ruby): Ruby bindings for SQLite
[10] DBD::SQLite (https://metacpan.org/module/DBD::SQLite): Perl DBI Interface to SQLite
[11] http://system.data.sqlite.org/index.html/doc/trunk/www/index.wiki
[12] http://www.ch-werner.de/sqliteodbc/
[13] http://documentation.openoffice.org/HOW_TO/data_source/SQLite.pdf

References

- Allen, Grant; Owens, Mike (November 5, 2010). *The Definitive Guide to SQLite* (http://apress.com/book/view/1430232250) (2nd ed.). Apress. p. 368. ISBN 1-4302-3225-0.
- Kreibich, Jay A. (August 17, 2010). *Using SQLite* (http://oreilly.com/catalog/9780596521196) (1st ed.). O'Reilly Media. p. 528. ISBN 0-596-52118-9.
- van der Lans, Rick F. (September 7, 2009). *The SQL Guide to SQLite* (1st ed.). lulu.com. p. 542. ISBN 0-557-07676-5.
- Newman, Chris (November 9, 2004). *SQLite (Developer's Library)* (http://www.informit.com/store/product.aspx?isbn=067232685X) (1st ed.). Sams. p. 336. ISBN 0-672-32685-X.

External links

- Official website (http://www.sqlite.org/)
- SQLite (http://www.dmoz.org/Computers/Software/Databases/SQLite) on the Open Directory Project
- An Introduction to SQLite (https://www.youtube.com/watch?v=giAMt8Tj-84) on YouTube

SESAM (database)

SESAM / SQL Server is a relational database system developed by Fujitsu Technology Solutions. It runs on the BS2000/OSD mainframe. Endpoint clients running on BS2000/OSD, UNIX systems, Solaris, Linux and Microsoft Windows are possible.

Features

- Support for SQL3
- Data can be stored in EBCDIC and Unicode
- Cost-based optimizer
- Maintenance is possible during operation
- Table partitioning
- Multi-platform support
- Apache web-server integration

References

- SESAM/SQL Product Page [1]
- SESAM/SQL Presentation [2]

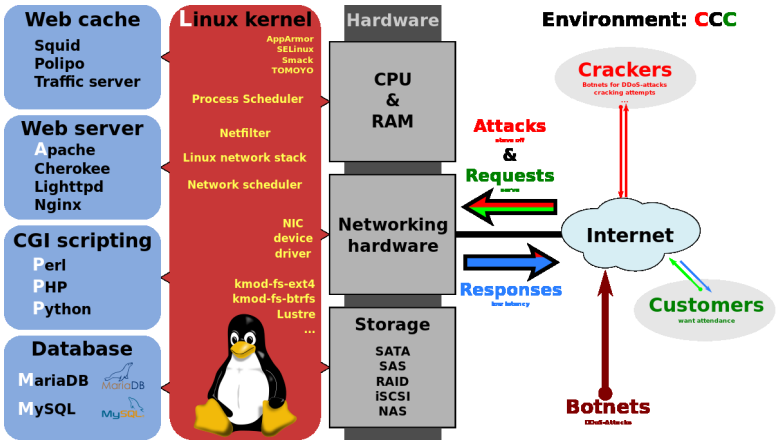
References

[1] http://ts.fujitsu.com/products/software/database_systems/sesamsql.html

[2] https://globalsp.ts.fujitsu.com/dmsp/docs/ps_sesam-sql-v6.pdf

LAMP

The acronym **LAMP** refers to first letters of the four components of a solution stack, composed entirely of free and open-source software, suitable for building high-availability heavy-duty dynamic web sites, and capable of serving tens of thousands of requests simultaneously.



The LAMP, LYME and LYCE software bundles, displayed here together with Squid. High-performance and high-availability web server solutions providing security in hostile environments.

The meaning of the LAMP acronym depends on which specific components are used as part of the actual bundle:

- **L**inux, the operating system (i.e. not just the Linux kernel, but also glibc and some other essential components of an operating system)
- **A**pache HTTP Server, the web server
- **M**ySQL, **M**ariaDB or **M**ongoDB, the database management system
- **P**HP, **P**erl, or **P**ython, the scripting languages (respectively programming languages) used for dynamic web pages and web development.

The exact combination of the software included in a LAMP stack is prone to variation, for example Apache web server can be replaced by some other web server software. Though the original authors of these programs did not design them to work as a component of the LAMP stack, the development philosophy and tool sets are shared and were developed in close conjunction, so they work and scale very well together. The software combination has become popular because it is entirely free and open-source software, which means that each component can be adapted to the underlying hardware and customized to meet the specification as exactly as possible, without the slightest vendor lock-in. The complete software stack is also free of cost, maximizing the available budget for tailoring the hardware and software.

Due to the nature of free and open-source software and the ubiquity of its components, each component of the LAMP stack is very well tested regarding performance and security. At the same time, there is an abundance of experienced contractors to do the tailoring required for various customizations, or for complex setups. There is also constant development going on.

The components of the LAMP stack are present in the software repositories of most (if not all) Linux distributions, giving any end-user a simple way to install, set up and operate an initial LAMP stack out of the box. The web presence of a small company that does not have a high hit count and is not prone to frequent attacks, can therefore be administered by another small company, by a one man company or even by a student.

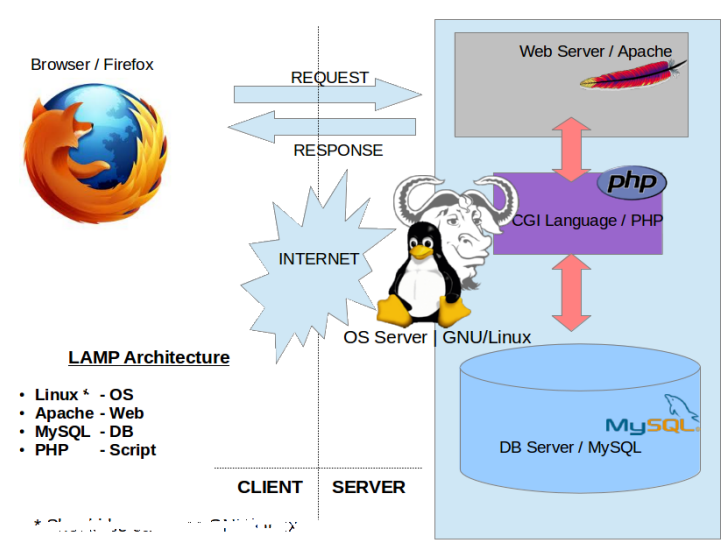
The LAMP bundle can be and often is combined with many other free and open-source software packages such as, for example:

- netsniff-ng for security testing and hardening
- Snort, an intrusion detection (IDS) and intrusion prevention system (IPS)
- RRDtool for diagrams
- Nagios, Collectd or Cacti, for monitoring.

Software components

Linux

Linux is a Unix-like and POSIX-compliant computer operating system assembled under the model of free and open source software development and distribution. The main form of distribution are Linux distributions, usually providing complete LAMP setups out of the box through their package management systems. Of the most widespread Linux distributions, as of 1 October 2013, 58.5% of web server market share is shared between Debian and Ubuntu, while RHEL, Fedora and CentOS together share 37.3%.



High-level overview of LAMP components

Many options are available for customizing and securing Linux installations, for example by using SELinux, or by employing chroot environments.

Apache

Apache is a web server, the most popular in use. As of June 2013[1], Apache was estimated to serve 54.2% of all active websites and 53.3% of the top servers across all domains.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Released under the Apache License, Apache is open-source software. A wide variety of features is supported, and many of them are implemented as compiled modules which extend the core functionality of Apache. These can range from server-side programming language support to authentication schemes.

MySQL, MariaDB, MongoDB

MySQL is a multithreaded, multi-user, SQL database management system (DBMS) now owned by Oracle Corporation.[2] MySQL has been owned by Oracle Corporation since January 27, 2010 through the purchase of Sun Microsystems.[3][4] Sun had originally acquired MySQL on February 26, 2008. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements.

MariaDB is a fork of MySQL. MongoDB is a widely used open-source NoSQL database. MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (calling the format BSON), making the integration of data in certain types of applications easier and faster.

Other RDBM systems such as PostgreSQL (forming up the LAPP bundle) are also viable.[5]

PHP, Perl, Python

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP code is interpreted by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone graphical applications.

PHP is free software released under the PHP License, which is incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term *PHP*.

Perl is a family of high-level, general-purpose, interpreted, dynamic programming languages. The languages in this family include Perl 5 and Perl 6. The Perl languages borrow features from other programming languages including C, shell scripting (sh), AWK, and sed. They provide powerful text processing facilities without the arbitrary data-length limits of many contemporary Unix commandline tools, facilitating easy manipulation of text files. Perl 5 gained widespread popularity in the late 1990s as a CGI scripting language, in part due to its parsing abilities.

Python is a widely used general-purpose, high-level programming language. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.[6] Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts.

Variants and equivalents on other platforms

With the growing use of LAMP, variations and retronyms appeared for other combinations of operating system, web server, database, and software language. For example the equivalent installation on a Microsoft Windows operating system is known as WAMP. An alternative running IIS in place of Apache called WIMP. Variants involving other operating systems include MAMP (Macintosh), SAMP (Solaris), FAMP (FreeBSD) and iAMP (iSeries).

The web server or database management system also vary. LEMP is a version where Apache has been replaced with the more lightweight web server Nginx. A version where MySQL has been replaced by PostgreSQL is called LAPP, or sometimes by keeping the original acronym, LAMP (Linux / Apache / Middleware (Perl, PHP, Python, Ruby) / PostgreSQL).

A server running LAMP may be colloquially known as a lamp box, punning on the type of post box. The GNU project is advocating people to use the term "GLAMP" since many distributions of what is known as "Linux" include the GNU tools as well as the Linux kernel.

High availability and load balancing

Specific solutions are required for web sites serving large numbers of requests, or providing services demanding no downtimes. Usual approach for the LAMP stack involves multiple web and database servers, with additional components providing logical aggregation of resources provided by each of the servers, and distribution of the workload across multiple servers. Such aggregation for web servers is usually provided by a form of load balancer placed in front of them, such as Linux Virtual Server (LVS). For the database servers, MySQL provides internal replication mechanisms, implementing a master/slave relationship between the original database (master) and its copies (slaves).

Such setups are improving the availability of LAMP instances by providing various forms of redundancy, making it possible for a certain number of instance's components (separate servers) to go down without causing interruptions to provided services. Also, such redundant setups allow for hardware failures resulting in data loss on separate servers, without the stored data actually becoming lost. Besides higher availability, such LAMP setups are providing almost linear improvements in performance for services where the number of internal database read operations is much higher than the number of write/update operations.

References

[1] http://en.wikipedia.org/w/index.php?title=LAMP_(software_bundle)&action=edit
[2] Top Reasons for Product Managers to Embed MySQL (http://www.mysql.com/why-mysql/topreasons_pm.html) on [mySQL.co]
[3] Robin Schumacher & Arjen Lentz Dispelling the Myths (http://dev.mysql.com/tech-resources/articles/dispelling-the-myths.html)
[4] Charles Babcock, InformationWeek Sun Locks Up MySQL, Looks To Future Web Development (http://www.informationweek.com/news/showArticle.jhtml?articleID=206900327)
[5] A LAPP appliance (http://www.turnkeylinux.org/lapp) on [turnkeylinux.org]
[6] , second section "Fans of Python use the phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files."

External links

- Install a LAMP server on Ubuntu Linux (https://help.ubuntu.com/community/ApacheMySQLPHP)
- Install a LAMP server on Debian GNU/Linux (http://wiki.debian.org/LaMp)
- Install a LAMP server on SUSE Linux (http://en.opensuse.org/SDB:Linux_Apache_MySQL_PHP)
- Install a LAMP server on Fedora Linux (http://fedorasolved.org/server-solutions/lamp-stack)
- Install a LAMP server on CentOS Linux (http://jesseforrest.name/setup-a-production-centos-lamp-web-server/212)
- Install a LAMP server on FRITZ!Box (http://blog.videgro.net/2013/11/fritzbox-lamp-server/)

Watcom SQL

Watcom SQL was a relational database for PC platforms released by Watcom in 1992. It was renamed to SQL Anywhere Studio after Watcom joined Powersoft which was subsequently acquired by Sybase.

The Watcom SQL 4.0 family of databases are based on scalable technology that combines a full-featured SQL database engine with unparalleled simplicity, economy and performance. Watcom SQL version 4.0 is designed for environments ranging from large departmental networks with a diverse range of PC client systems to peer-to-peer workgroups to standalone PCs. Watcom SQL Version 4.0 is available immediately in standalone versions for Windows and multi-user network server versions for NetWare. Standalone and multi-user versions for OS/2, Windows NT (Windows New Technology) A 32-bit operating system from Microsoft for Intel x86 CPUs. NT is the core technology in Windows 2000 and Windows XP (see Windows). Available in separate client and server versions, it includes built-in networking and preemptive multitasking. and DOS will be available within sixty days.

"To meet the needs of the 'shrink-wrapped' server marketplace, Watcom SQL was designed with a small 'footprint' for the desktop PC and LAN server environments," said Ian McPhee, president of Watcom. "Watcom SQL requires minimal database administration, takes only a few minutes to install and offers excellent performance out of the box without tuning."

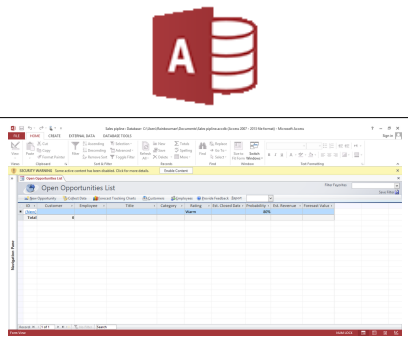
"The addition of stored procedures and triggers to Watcom SQL 4.0 brings industrial-strength features to the workgroup database market," said Herb Edelstein, principal and founder of Euclid Associates, a Maryland-based database and image management consulting firm. "Workgroup users will appreciate the functionality added to Watcom SQL, a database that remains easy to set up and run."

"The Watcom SQL database fits perfectly into our plans," said Jim Braun, project manager at Kansas State University, main campus at Manhattan; coeducational; land-grant and state supported; chartered and opened 1863. There is an additional campus at Salina. Among the university's research facilities are the J. R. . "We needed a database that could distribute key systems across our organization, and Watcom SQL 4.0 gave us this, in addition to stored procedures and triggers. Watcom SQL is an impressive, economical package of performance and functionality."

References

Microsoft Access

Microsoft Access



Microsoft Office Access 2013 running on Windows 8

Developer(s)	Microsoft Corporation
Initial release	November 1992
Stable release	2013 (15.0.4420.1017) / October 2, 2012
Development status	Active
Operating system	Microsoft Windows
Type	DBMS
License	Trialware
Website	office.microsoft.com/access [1]

Microsoft Access, also known as **Microsoft Office Access**, is a database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately.

Microsoft Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.

Software developers and data architects can use Microsoft Access to develop application software, and "power users" can use it to build software applications. Like other Office applications, Access is supported by Visual Basic for Applications, an object-oriented programming language that can reference a variety of objects including DAO (Data Access Objects), ActiveX Data Objects, and many other ActiveX components. Visual objects used in forms and reports expose their methods and properties in the VBA programming environment, and VBA code modules may declare and call Windows operating-system functions.

History

Project Omega

Microsoft's first attempt to sell a relational database product was during the mid-1980s, when Microsoft obtained a license to sell R:Base. In the late 1980s Microsoft developed its own solution codenamed Omega. It was confirmed in 1988 that a database product for Windows and OS/2 was in development. It was going to include "EB" Embedded Basic language, which was going to be the language for writing macros in all Microsoft applications, but the unification of macro languages did not happen until the introduction of VBA. Omega was also expected to provide a front end to the Microsoft SQL Server. The application was very resource-hungry, and there were reports that it was working slowly on the 386 processors that were available at the time. It was scheduled to be released in the 1st quarter of 1990, but in 1989 the development of the product was reset and it was rescheduled to be delivered no sooner than in January 1991. Parts of the project were later used for other Microsoft projects: Cirrus (codename for Access) and Thunder (codename for Visual Basic, where the Embedded Basic engine was used). After Access's premiere, the Omega project was demonstrated in 1992 to several journalists and included features that were not available in Access.

Project Cirrus

After the Omega project was scrapped, some of its developers were assigned to the Cirrus project (most were assigned to the team which created Visual Basic). Its goal was to create a competitor for applications like Paradox or dBase that would work on Windows. After Microsoft acquired FoxPro, there were rumors that the Microsoft project might get replaced with it, but the company decided to develop them in parallel. It was assumed that the project would make use of Extensible Storage Engine (Jet Blue) but, in the end, only support for Microsoft Jet Database Engine (Jet Red) was provided. The project used some of the code from both the Omega project and a pre-release version of Visual Basic. In July 1992, betas of Cirrus shipped to developers and the name Access became the official name of the product.

Timeline

1992: Microsoft released Access version 1.0 on 13 November 1992, and an Access 1.1 release in May 1993 to improve compatibility with other Microsoft products and to include the Access Basic programming language.

1994: Microsoft specified the minimum hardware requirements for Access v2.0 as: Microsoft Windows v3.1 with 4 MB of RAM required, 6 MB RAM recommended; 8 MB of available hard disk space required, 14 MB hard disk space recommended. The product shipped on seven 1.44 MB diskettes. The manual shows a 1994 copyright date.

Originally, the software worked well with relatively small databases but testing showed that some circumstances caused data corruption. For example, file sizes over 10 MB proved problematic (note that most hard disks held less than 500 MB at the time this was in wide use), and the *Getting Started* manual warns about a number of circumstances where obsolete device drivers or incorrect configurations can cause data loss. With the phasing out of Windows 95, 98 and ME, improved network reliability, and Microsoft having released 8 service packs for the Jet Database Engine, the reliability of Access databases has improvedWikipedia:Manual of Style/Dates and numbers#Chronological items and it supports both more data and a larger number of users.

With Office 95, Microsoft Access 7.0 (a.k.a. "Access 95") became part of the Microsoft Office Professional Suite, joining Microsoft Excel, Word, and PowerPoint and transitioning from Access Basic to Visual Basic for Applications (VBA). Since then, Microsoft has released new versions of Microsoft Access with each release of Microsoft Office. This includes Access 97 (version 8.0), Access 2000 (version 9.0), Access 2002 (version 10.0), Access 2003 (version 11.5), Access 2007 (version 12.0), and Access 2010 (version 14.0).

Versions 3.0 and 3.5 of Microsoft Jet database engine (used by Access 7.0 and the later-released Access 97 respectively) had a critical issue which made these versions of Access unusable on a computer with more than 1 GB

of memory.[2] While Microsoft fixed this problem for Jet 3.5/Access 97 post-release, it never fixed the issue with Jet 3.0/Access 95.

The native Access database format (the Jet MDB Database) has also evolved over the years. Formats include Access 1.0, 1.1, 2.0, 7.0, 97, 2000, 2002, 2007, and 2010. The most significant transition was from the Access 97 to the Access 2000 format; which is not backward compatible with earlier versions of Access. As of 2011[3] all newer versions of Access support the Access 2000 format. New features were added to the Access 2002 format which can be used by Access 2002, 2003, 2007, and 2010.

Microsoft Access 2000 increased the maximum database size to 2GB from 1GB in Access 97.

Microsoft Access 2007 introduced a new database format: ACCDB. ACCDB supports links to SharePoint lists and complex data types such as multivalue and attachment fields. These new field types are essentially recordsets in fields and allow the storage of multiple values or files in one field. Microsoft Access 2007 also introduced File Attachment field, which stored data more efficiently than the OLE (Object Linking and Embedding) field.

Microsoft Access 2010 introduced a new version of the ACCDB format supported hosting Access Web solutions on a SharePoint 2010 server. For the first time, this allowed Access solutions to be run without having to install Access on their PC and was the first support of Mac users. Any user on the SharePoint site with sufficient rights could use the Access Web solution. A copy of Access was still required for the developer to create the Access Web solution, and the desktop version of Access remained part of Access 2010. The Access Web solutions were not the same as the desktop solutions. Automation was only through the macro language (not VBA) which Access automatically converted to JavaScript. The data was no longer in an Access database but SharePoint lists. An Access desktop database could link to the SharePoint data, so hybrid applications were possible so that SharePoint users needing basic views and edits could be supported while the more sophisticated, traditional solutions could remain in the desktop Access database.

Microsoft Access 2013 offers traditional Access desktop solutions plus a significantly updated SharePoint 2013 web solution.[4] The Access Web model in Access 2010 was replaced by a new architecture that stores its data in actual SQL Server databases. Unlike SharePoint lists, this offers true relational database design with referential integrity, scalability, extensibility and performance one would expect from SQL Server.[5] The database solutions that can be created on SharePoint 2013 offer a modern user interface designed to display multiple levels of relationships that can be viewed and edited, along with resizing for different devices and support for touch. The Access 2013 desktop is similar to Access 2010 but several features were discontinued including support for Access Data Projects (ADPs), pivot tables, pivot charts, Access data collections, source code control, replication, and other legacy features.[6] Access desktop database maximum size remained 2GB (as it has been since the 2000 version).

Prior to the introduction of Access, Borland (with Paradox and dBase) and Fox (with FoxPro) dominated the desktop database market. Microsoft Access was the first mass-market database program for Windows. With Microsoft's purchase of FoxPro in 1992 and the incorporation of Fox's Rushmore query optimization routines into Access, Microsoft Access quickly became the dominant database for Windows - effectively eliminating the competition which failed to transition from the MS-DOS world.[7]

Access's initial codename was Cirrus; the forms engine was called Ruby. This was before Visual Basic. Bill Gates saw the prototypes and decided that the BASIC language component should be co-developed as a separate expandable application, a project called Thunder. The two projects were developed separately.

Access was also the name of a communications program from Microsoft, meant to compete with ProComm and other programs. This proved a failure and was dropped.[8] Years later, Microsoft reused the name for its database software.

Uses

In addition to using its own database storage file, Microsoft Access also may be used as the 'front-end' with other products as the 'back-end' tables, such as Microsoft SQL Server and non-Microsoft products such as Oracle and Sybase. Multiple backend sources can be used by a Microsoft Access Jet Database (accdb and mdb formats). Similarly, some applications such as Visual Basic, ASP.NET, or Visual Studio .NET will use the Microsoft Access database format for its tables and queries. Microsoft Access may also be part of a more complex solution, where it may be integrated with other technologies such as Microsoft Excel, Microsoft Outlook, Microsoft Word, Microsoft PowerPoint and ActiveX Controls.

Access tables support a variety of standard field types, indices, and referential integrity including cascading updates and deletes. Access also includes a query interface, forms to display and enter data, and reports for printing. The underlying Jet database, which contains these objects, is multiuser-aware and handles record-locking.

Repetitive tasks can be automated through macros with point-and-click options. It is also easy to place a database on a network and have multiple users share and update data without overwriting each other's work. Data is locked at the record level which is significantly different from Excel which locks the entire spreadsheet.

There are template databases within the program and for download from their website [9]. These options are available upon starting Access and allow users to enhance a database with predefined tables, queries, forms, reports, and macros. Database templates support VBA code but Microsoft's templates do not include VBA code.

Programmers can create solutions using the programming language Visual Basic for Applications (VBA), which is similar to Visual Basic 6.0 (VB6) and used throughout the Microsoft Office programs such as Excel, Word, Outlook and PowerPoint. Most VB6 code, including the use of Windows API calls, can be used in VBA. Power users and developers can extend basic end-user solutions to a professional solution with advanced automation, data validation, error trapping, and multi-user support.

The number of simultaneous users that can be supported depends on the amount of data, the tasks being performed, level of use, and application design. Generally accepted limits are solutions with 1 GB or less of data (Access supports up to 2 GB) and performs quite well with 100 or fewer simultaneous connections (255 concurrent users are supported). This capability is often a good fit for department solutions. If using an Access database solution in a multi-user scenario, the application should be "split". This means that the tables are in one file called the back end (typically stored on a shared network folder) and the application components (forms, reports, queries, code, macros, linked tables) are in another file called the front end. The linked tables in the front end point to the back end file. Each user of the Access application would then receive his or her own copy of the front end file.

Applications that run complex queries or analysis across large datasets would naturally require greater bandwidth and memory. Microsoft Access is designed to scale to support more data and users by linking to multiple Access databases or using a back-end database like Microsoft SQL Server. With the latter design, the amount of data and users can scale to enterprise-level solutions.

Microsoft Access's role in web development prior to version 2010 is limited. User interface features of Access, such as forms and reports, only work in Windows. In versions 2000 through 2003 an Access object type called Data Access Pages created publishable web pages. Data Access Pages are no longer supported. The Microsoft Jet Database Engine, core to Access, can be accessed through technologies such as ODBC or OLE DB. The data (i.e., tables and queries) can be accessed by web-based applications developed in ASP.NET, PHP, or Java. With the use of Microsoft's Terminal Services and Remote Desktop Application in Windows Server 2008 R2, organizations can host Access applications so they can be run over the web.[10] This technique does not scale the way a web application would but is appropriate for a limited number of users depending on the configuration of the host.

Access 2010 allows databases to be published to SharePoint 2010 web sites running Access Services. These web-based forms and reports run in any modern web browser. The resulting web forms and reports, when accessed via a web browser, don't require any add-ins or extensions (e.g. ActiveX, Silverlight).

Access 2013 can create web applications directly in SharePoint 2013 sites running Access Services. Access 2013 web solutions store its data in an underlying SQL Server database which is much more scalable and robust than the Access 2010 version which used SharePoint lists to store its data.

A compiled version of an Access database (File extensions: .MDE /ACCDE or .ADE; ACCDE only works with Access 2007 or later) can be created to prevent user from accessing the design surfaces to modify module code, forms, and reports. An MDE/ACCDE file is a Microsoft Access database file with all modules compiled and all editable source code removed. An ADE file is an Access project file with all modules compiled and all editable source code removed. Both the .MDE/ACCDE and .ADE versions of an Access database are used when end-user modifications are not allowed or when the application's source code should be kept confidential.

Microsoft also offers developer extensions [11] for download to help distribute Access 2007 applications, create database templates, and integrate source code control with Microsoft Visual SourceSafe.

Features

Users can create tables, queries, forms and reports, and connect them together with macros. Advanced users can use VBA to write rich solutions with advanced data manipulation and user control. Access also has report creation features that can work with any data source that Access can "access".

The original concept of Access was for end users to be able to "access" data from any source. Other features include: the import and export of data to many formats including Excel, Outlook, ASCII, dBase, Paradox, FoxPro, SQL Server, Oracle, ODBC, etc. It also has the ability to link to data in its existing location and use it for viewing, querying, editing, and reporting. This allows the existing data to change while ensuring that Access uses the latest data. It can perform heterogeneous joins between data sets stored across different platforms. Access is often used by people downloading data from enterprise level databases for manipulation, analysis, and reporting locally.

There is also the Jet Database format (MDB or ACCDB in Access 2007) which can contain the application and data in one file. This makes it very convenient to distribute the entire application to another user, who can run it in disconnected environments.

One of the benefits of Access from a programmer's perspective is its relative compatibility with SQL (structured query language) — queries can be viewed graphically or edited as SQL statements, and SQL statements can be used directly in Macros and VBA Modules to manipulate Access tables. Users can mix and use both VBA and "Macros" for programming forms and logic and offers object-oriented possibilities. VBA can also be included in queries.

Microsoft Access offers parameterized queries. These queries and Access tables can be referenced from other programs like VB6 and .NET through DAO or ADO. From Microsoft Access, VBA can reference parameterized stored procedures via ADO.

The desktop editions of Microsoft SQL Server can be used with Access as an alternative to the Jet Database Engine. This support started with MSDE (Microsoft SQL Server Desktop Engine), a scaled down version of Microsoft SQL Server 2000, and continues with the SQL Server Express versions of SQL Server 2005 and 2008.

Microsoft Access is a file server-based database. Unlike client–server relational database management systems (RDBMS), Microsoft Access does not implement database triggers, stored procedures, or transaction logging. Access 2010 includes table-level triggers and stored procedures built into the ACE data engine. Thus a Client-server database system is not a requirement for using stored procedures or table triggers with Access 2010. Tables, queries, forms, reports and macros can now be developed specifically for web base application in Access 2010. Integration with Microsoft SharePoint 2010 is also highly improved.

Access Services and Web database

ASP.NET web forms can query a Microsoft Access database, retrieve records and display them on the browser.

SharePoint Server 2010 via Access Services allows for Access 2010 databases to be published to SharePoint, thus enabling multiple users to interact with the database application from any standards-compliant Web browser. Access Web databases published to SharePoint Server can use standard objects such as tables, queries, forms, macros, and reports. Access Services stores those objects in SharePoint.

Access 2013 offers the ability to publish Access web solutions on SharePoint 2013. Rather than using SharePoint lists as its data source, Access 2013 uses an actual SQL Server database hosted by SharePoint or SQL Azure. This offers a true relational database with referential integrity, scalability, maintainability, and extensibility compared to the SharePoint views Access 2010 used.. The macro language is enhanced to support more sophisticated programming logic and database level automation.[12]

Import or Link sources

Microsoft Access can also import or link directly to data stored in other applications and databases. Microsoft Office Access 2007 and newer can import from or link to:

- Microsoft Access
- Excel
- SharePoint lists
- Plain text
- XML
- Outlook
- HTML
- dBase (dropped in Access 2013)
- Paradox (with Access 2007; dropped in Access 2010)
- Lotus 1-2-3 (dropped in Access 2010)
- ODBC-compliant data containers, including:
  - Microsoft SQL Server
  - Oracle
  - MySQL
  - PostgreSQL
  - IBM Lotus Notes
  - AS 400 DB2

Microsoft Access Runtime

Microsoft offers free runtime versions of Microsoft Access: Access 2013 Runtime [13], Access 2010 Runtime [14], Access 2007 Runtime [15], which allow users to run an Access desktop application without needing to purchase or install a full version of Microsoft Access. This allows Access developers to create databases that can be freely distributed to an unlimited number of end-users. The runtime version allows users to view, edit and delete data, along with running queries, forms, reports, macros and VBA module code. But the runtime version does not allow users to change the design of Microsoft Access objects or code. The runtime versions are similar to their corresponding full version of Access and usually compatible with earlier versions; for example Access Runtime 2010 allows a user to run an Access application made with the 2010 version as well as 2007 through 2000. Due to deprecated features in Access 2013, its runtime version is also unable to support those older features.

Development

Access stores all database tables, queries, forms, reports, macros, and modules in the Access Jet database as a single file.

For query development, Access offers a "Query Designer", a graphical user interface that allows users to build queries without knowledge of structured query language. In the Query Designer, users can "show" the datasources of the query (which can be tables or queries) and select the fields they want returned by clicking and dragging them into the grid. One can set up joins by clicking and dragging fields in tables to fields in other tables. Access allows users to view and manipulate the SQL code if desired. Any Access table, including linked tables from different data sources, can be used in a query.

Access also supports the creation of "pass-through queries". These snippets of SQL code can address external data sources through the use of ODBC connections on the local machine. This enables users to interact with data stored outside the Access program without using linked tables or Jet. Users construct the pass-through queries using the SQL syntax supported by the external data source.

When developing reports (in "Design View") additions or changes to controls cause any linked queries to execute in the background and the designer is forced to wait for records to be returned before being able to make another change. This feature cannot be turned off.

Non-programmers can use the macro feature to automate simple tasks through a series of drop-down selections. Macros allow users to easily chain commands together such as running queries, importing or exporting data, opening and closing forms, previewing and printing reports, etc. Macros support basic logic (IF-conditions) and the ability to call other macros. Macros can also contain sub-macros which are similar to subroutines. In Access 2007, enhanced macros included error-handling and support for temporary variables. Access 2007 also introduced embedded macros that are essentially properties of an object's event. This eliminated the need to store macros as individual objects. However, macros were limited in their functionality by a lack of programming loops and advanced coding logic until Access 2013. With significant further enhancements introduced in Access 2013, the capabilities of macros became fully comparable to VBA. They made feature rich web-based application deployments practical, via a greatly enhanced Microsoft SharePoint interface and tools, as well as on traditional Windows desktops.

In common with other products in the Microsoft Office suite, the other programming language used in Access is Microsoft VBA Visual Basic for Applications. VBA is similar to Visual Basic 6.0 (VB6) and code can be stored in modules, classes, and code behind forms and reports. To create a richer, more efficient and maintainable finished product with good error handling, most professional Access applications are developed using the VBA programming language rather than macros, except where web deployment is a business requirement.

To manipulate data in tables and queries in VBA or macros, Microsoft provides two database access libraries of COM components:

1. Data Access Objects (DAO) (32-bit only), which is included in Access and Windows and evolved to ACE in Microsoft Access 2007 for the ACCDE database format
2. ActiveX Data Objects ActiveX Data Objects (ADO) (both 32-bit and 64-bit versions)

As well as DAO and ADO, developers can also use OLE DB and ODBC for developing native C/C++ programs for Access. For ADPs and the direct manipulation of SQL Server data, ADO is required. DAO is most appropriate for managing data in Access/Jet databases, and the only way to manipulate the complex field types in ACCDB tables.

In the database container or navigation pane in Access 2007 and later versions, the system automatically categorizes each object by type (e.g., table, query, macro). Many Access developers use the Leszynski naming convention, though this is not universal; it is a programming convention, not a DBMS-enforced rule.[16][17] It is particularly helpful in VBA where references to object names may not indicate its data type (e.g. tbl for tables, qry for queries).

Developers deploy Microsoft Access most often for individual and workgroup projects (the Access 97 speed characterization was done for 32 users).[18] Since Access 97, and with Access 2003 and 2007, Microsoft Access and

hardware have evolved significantly. Databases under 1 GB in size (which can now fit entirely in RAM) and 50 simultaneous users are well within the capabilities of Microsoft Access. Of course, performance depends on the database design and tasks. Disk-intensive work such as complex searching and querying take the most time.

As data from a Microsoft Access database can be cached in RAM, processing speed may substantially improve when there is only a single user or if the data are not changing. In the past, the effect of packet latency on the record-locking system caused Access databases to run slowly on a Virtual Private Network (VPN) or a Wide Area Network (WAN) against a Jet database. As of 2010[3] broadband connections have mitigated this issue. Performance can also be enhanced if a continuous connection [19] is maintained to the back-end database throughout the session rather than opening and closing it for each table access. If Access database performance over VPN or WAN suffers, then a client using Remote Desktop Protocol (such as Microsoft Terminal Services) can provide an effective solution. Access databases linked to SQL Server or to Access Data Projects work well[*citation needed*] over VPNs and WANs.

In July 2011, Microsoft acknowledged an intermittent query performance problem with all versions of Access and Windows 7 and Windows Server 2008 R2 due to the nature of resource management being vastly different in newer operating systems.[20] This issue severely affects query performance on both Access 2003 and earlier with the Jet Database Engine code, as well as Access 2007 and later with the Access Database Engine (ACE). Microsoft has issued hotfixes KB2553029 [21] for Access 2007 and KB2553116 [22] for Access 2010, but will not fix the issue with Jet 4.0 as it is out of mainstream support.

In previous versions of Microsoft Access the ability to distribute applications required the purchase of the Developer Toolkit; in Access 2010 and Access 2013 the "Runtime Only" version is offered as a free download, making the distribution of royalty-free applications possible on Windows 7 and Windows 8.x.[23]

Split Database Architecture

Microsoft Access applications can adopt a split-database architecture. The single database can be divided into a separate "back-end" file that contains the data tables (shared on a file server) and a "front-end" (containing the application's objects such as queries, forms, reports, macros, and modules). The "front-end" Access application is distributed to each user's desktop and linked to the shared database. Using this approach, each user has a copy of Microsoft Access (or the runtime version) installed on their machine along with their application database. This reduces network traffic since the application is not retrieved for each use. The "front-end" database can still contain local tables for storing a user's settings or temporary data. This split-database design also allows development of the application independent of the data. One disadvantage is that users may make various changes to their own local copy of the application and this makes it hard to manage version control. When a new version is ready, the front-end database is replaced without impacting the data database. Microsoft Access has two built-in utilities, Database Splitter and Linked Table Manager, to facilitate this architecture.

Linked tables in Access use absolute paths rather than relative paths, so the development environment either has to have the same path as the production environment or a "dynamic-linker" routine can be written in VBA.

For very large Access databases, this may have performance issues and a SQL backend should be considered in these circumstances. This is less of an issue if the entire database can fit in the PC's RAM since Access caches data and indexes.

Access to SQL Server Upsizing (Upgrading) (SQL as a backend)

To scale Access applications to enterprise or web solutions, one possible technique involves migrating to Microsoft SQL Server or equivalent server database. A client–server design significantly reduces maintenance and increases security, availability, stability, and transaction logging.

Access 2010 included a feature called the Upsizing Wizard that allowed users to upgrade their databases to Microsoft SQL Server, an ODBC client–server database. This feature was removed from Access 2013. An additional solution, the SQL Server Migration Assistant for Access (SSMA), continues to be available for free download from Microsoft.

A variety of upgrading options are available.[24] After migrating the data and queries to SQL Server, the MDB/ACCDB Access database can be linked to the database. However, certain data types are problematic, most notably "Yes/No". In MS Access there are three states for the Yes/No (True/False) data type: empty, No/False (zero) and Yes/True (-1). The corresponding SQL Server data type is binary, with only two states, permissible values, zero and 1. Regardless, SQL Server is still the easiest migration, and most appropriate especially if the user does not have rights to create objects such as stored procedures on SQL Server. Retrieving data from linked tables is optimized to just the records needed, but this scenario may operate less efficiently than what would otherwise be optimal for SQL Server. For example, in instances where multi-table joins still require copying the whole table across the network.

In previous versions of Access, including Access 2010, databases can also be converted to Access Data Projects (ADP) which are tied directly to one SQL Server database. This feature was removed from Access 2013. ADP's support the ability to directly create and modify SQL Server objects such as tables, views, stored procedures, and SQL Server constraints. The views and stored procedures can significantly reduce the network traffic for multi-table joins. Fortunately, SQL Server supports temporary tables and links to other data sources beyond the single SQL Server database.

Finally, some Access databases are completely replaced by another technology such as ASP.NET or Java once the data is converted. However any migration may dictate major effort since the Access SQL language is a more powerful superset of standard SQL. Further, Access application procedures, whether VBA and macros, are written at a relatively higher level versus the currently available alternatives that are both robust and comprehensive. Note that the Access macro language, allowing an even higher level of abstraction than VBA, was significantly enhanced in Access 2010 and again in Access 2013.

In many cases, developers build direct web-to-data interfaces using ASP.NET, while keeping major business automation processes, administrative and reporting functions that don't need to be distributed to everyone in Access for information workers to maintain.

While all most Access data can migrate to SQL Server directly, some queries cannot migrate successfully. In some situations, you may need to translate VBA functions and user defined functions into T–SQL or .NET functions / procedures. Crosstab queries can be migrated to SQL Server using the PIVOT command.

Protection

Microsoft Access offers several ways to secure the application while allowing users to remain productive.

The most basic is a database password. Once entered, the user has full control of all the database objects. This is a relatively weak form of protection which can be easily cracked.

A higher level of protection is the use of workgroup security requiring a user name and password. Users and groups can be specified along with their rights at the object type or individual object level. This can be used to specify people with read-only or data entry rights but may be challenging to specify. A separate workgroup security file contains the settings which can be used to manage multiple databases. Workgroup security is not supported in the Access 2007 and Access 2010 ACCDB database format, although Access 2007 and Access 2010 still support it for MDB databases.

Databases can also be encrypted. The ACCDB format offers significantly advanced encryption from previous versions.[25]

Additionally, if the database design needs to be secured to prevent changes, Access databases can be locked/protected (and the source code compiled) by converting the database to a .MDE file. All changes to the VBA project (modules, forms, or reports) need to be made to the original MDB and then reconverted to MDE. In Access 2007 and Access 2010, the ACCDB database is converted to an ACCDE file. Some tools are available for unlocking and "decompiling", although certain elements including original VBA comments and formatting are normally irretrievable.

File extensions

Microsoft Access saves information under the following file formats:

File format	Extension
Protected Access Data Project (not supported in Access 2013)	.ade
Access Data Project (not supported in Access 2013)	.adp
Access Blank Project Template	.adn
Access Database (2007)	.accdb
Access Database Runtime (2007)	.accdr
Access Database Template (2007)	.accdt
Access Database (2003 and earlier)	.mdb
Access Database, used for addins (2,95,97), previously used for workgroups (2).	.mda
Access Blank Database Template (2003 and earlier)	.mdn
Access Add-in Data (2003 and earlier)	.mdt
Access Workgroup, database for user-level security.	.mdw
Access (SQL Server) detached database (2000)	.mdf
Protected Access Database, with compiled VBA and/or macros (2003 and earlier)	.mde
Protected Access Database, with compiled VBA and/or macros (2007)	.accde
Windows Shortcut: Access Macro	.mam
Windows Shortcut: Access Query	.maq
Windows Shortcut: Access Report	.mar
Windows Shortcut: Access Table	.mat
Windows Shortcut: Access Form	.maf
Access lock files (associated with .mdb)	.ldb
Access lock files (associated with .accdb)	.laccdb

Versions

Version	Version number	Release Date	Jet version	Supported OS	Office suite version
Access 1.1	1.0	1992	1.1	Windows 3.0[*citation needed*]	
Access 2.0	2.0	1993	2.0	Windows 3.1x[*citation needed*]	Office 4.3 Pro
Access for Windows 95	7.0	Aug. 24, 1995	3.0	Windows 95[*citation needed*]	Office 95 Professional
Access 97	8.0	Jan. 16, 1997	3.5	Windows 95, Windows NT 3.51 SP5, Windows NT 4.0 SP2	Office 97 Professional and Developer
Access 2000	9.0	June 7, 1999	4.0 SP1	Windows 95, Windows NT 4.0, Windows 98, Windows 2000	Office 2000 Professional, Premium and Developer
Access 2002	10.0	May 31, 2001	4.0 SP1	Windows NT 4.0 SP6, Windows 98, Windows 2000, Windows Me	Office XP Professional and Developer
Access 2003	11.0	Nov. 27, 2003	4.0 SP1	Windows 2000 SP3 or later, Windows XP, Windows Vista, Windows 7	Office 2003 Professional and Professional Enterprise
Access 2007	12.0	Jan. 27, 2007	12	Windows XP SP2, Windows Server 2003 SP1, or newer operating system	Office 2007 Professional, Professional Plus, Ultimate and Enterprise
Access 2010	14.0	July 15, 2010	14	Windows XP SP3, Windows Server 2003 SP2, Windows Server 2003 R2, Windows Vista SP1, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows Server 2012, Windows 8	Office 2010 Professional, Professional Academic and Professional Plus
Access 2013	15.0	Jan. 29, 2013	15	Windows 7, Windows Server 2008 R2, Windows Server 2012, Windows 8	Office 2013

Notes

There are no Access versions between 2.0 and 7.0 because the Office 95 version was launched with Word 7. All of the Office 95 products have OLE 2 capabilities, and Access 7 shows that it was compatible with Word 7.

Version number 13 was skipped.

References

[1] http://office.microsoft.com/access

[2] "Out of memory" error starting Microsoft Access (http://support.microsoft.com/kb/161255)

[3] http://en.wikipedia.org/w/index.php?title=Microsoft_Access&action=edit

[4] Changes in Access 2013 (http://msdn.microsoft.com/en-us/library/office/jj618413.aspx)

[5] What's new for Access 2013 developers (http://msdn.microsoft.com/en-us/library/office/jj250134.aspx)

[6] Discontinued features and modified functionality in Access 2013 (http://office.microsoft.com/en-us/access-help/discontinued-features-and-modified-functionality-in-access-2013-HA102749226.aspx)

[7] Microsoft Access History (http://www.fmsinc.com/MicrosoftAccess/history/index.html)

[8] Where did the name for Microsoft Access come from? (http://blogs.msdn.com/oldnewthing/archive/2006/04/13/575739.aspx)

[9] http://office.microsoft.com/en-us/templates/default.aspx

[10] Using Terminal Services and RemoteApp to Extend Your Microsoft Access and other Windows Applications Over the Internet (http://www.fmsinc.com/microsoftaccess/terminal-services/remoteapp.htm)

[11] http://www.microsoft.com/en-us/download/details.aspx?id=24569

[12] What's new for Access 2013 developers (http://msdn.microsoft.com/en-us/library/office/jj250134.aspx)

[13] https://www.microsoft.com/en-us/download/details.aspx?id=39358

[14] http://www.microsoft.com/en-us/download/details.aspx?id=10910

[15] http://www.microsoft.com/en-us/download/details.aspx?id=4438

[16] Naming Conventions for Microsoft Access (http://technet.microsoft.com/en-us/library/bf926bek(v=VS.80).aspx)

[17] Naming Conventions for Visual Basic (http://support.microsoft.com/kb/110264)

[18] Kevin Collins (Microsoft Jet Program Management), "Microsoft Jet 3.5 Performance Overview and Optimization Techniques", MSDN. Retrieved July 19, 2005.
[19] http://www.fmsinc.com/MicrosoftAccess/Performance/LinkedDatabase.html
[20] Very slow Access 2002 query with Windows 7 (http://social.msdn.microsoft.com/Forums/en-US/accessdev/thread/86e59bc0-3524-45be-89d0-3528cfea842b)
[21] http://support.microsoft.com/kb/2553029
[22] http://support.microsoft.com/kb/2553116
[23] Microsoft Access 2010 Runtime (http://www.microsoft.com/en-us/download/details.aspx?id=10910)
[24] Access Upsizing Project (http://www.fmsinc.com/FMSUpsize/docs/EvolvingMicrosoftAccessApplications.pdf)
[25] Security Considerations and Guidance for Access 2007 (http://msdn.microsoft.com/en-us/library/bb421308.aspx)

External links

- Official website (http://office.microsoft.com/access)
- Access Blog (http://blogs.office.com/b/microsoft-access)

VMDS

VMDS abbreviates the relational database technology called **Version Managed Data Store** provided by GE Energy as part of its Smallworld technology platform and was designed from the outset to store and analyse the highly complex spatial and topological networks typically used by enterprise utilities such as power distribution and telecommunications.

VMDS was originally introduced in 1990 as has been improved and updated over the years. Its current version is 6.0.

VMDS has been designed as a spatial database. This gives VMDS a number of distinctive characteristics when compared to conventional attribute only relational databases.

Distributed server processing

VMDS is composed of two parts: a simple, highly scalable data block server called **SWMFS** (Smallworld Master File System) and an intelligent client API written in C and Magik. Spatial and attribute data are stored in data blocks that reside in special files called data store files on the server. When the client application requests data it has sufficient intelligence to work out the optimum set of data blocks that are required. This request is then made to SWMFS which returns the data to the client via the network for processing.

This approach is particularly efficient and scalable when dealing with spatial and topological data which tends to flow in larger volumes and require more processing then plain attribute data (for example during a map redraw operation). This approach makes VMDS well suited to enterprise deployment that might involve hundreds or even thousands of concurrent clients.

Support for long transactions

Relational databases support short transactions in which changes to data are relatively small and are brief in terms in duration (the maximum period between the start and the end of a transaction is typically a few seconds or less).

VMDS supports long transactions in which the volume of data involved in the transaction can be substantial and the duration of the transaction can be significant (days, weeks or even months). These types of transaction are common in advanced network applications used by, for example, power distribution utilities.

Due to the time span of a long transaction in this context the amount of change can be significant (not only within the scope of the transaction, but also within the context of the database as a whole). Accordingly, it is likely that the

same record might be changed more than once. To cope with this scenario VMDS has inbuilt support for automatically managing such conflicts and allows applications to review changes and accept only those edits that are correct.

Spatial and topological capabilities

As well as conventional relational database features such as attribute querying, join fields, triggers and calculated fields, VMDS has numerous spatial and topological capabilities. This allows spatial data such as points, texts, polylines, polygons and raster data to be stored and analysed.

Spatial functions include: find all features within a polygon, calculate the Voronoi polygons of a set of sites and perform a cluster analysis on a set of points.

Vector spatial data such as points, polylines and polygons can be given topological attributes that allow complex networks to be modelled. Network analysis engines are provided to answer questions such as find the shortest path between two nodes or how to optimize a delivery route (the travelling salesman problem). A topology engine can be configured with a set of rules that define how topological entities interact with each other when new data is added or existing data edited.

Data abstraction

In VMDS all data is presented to the application as objects. This is different from many relational databases that present the data as rows from a table or query result using say JDBC. VMDS provides a data modelling tool and underlying infrastructure as part of the Smallworld technology platform that allows administrators to associate a table in the database with a Magik exemplar (or class). Magik get and set methods for the Magik exemplar can be automatically generated that expose a table's field (or column). Each VMDS *row* manifests itself to the application as an instance of a Magik object and is known as an **RWO** (or real world object). Tables are known as collections in Smallworld parlance.

```
# all_rwos hold all the rwos in the database and is heterogeneous
all_rwos << my_application.rwo_set()

# valve_collection holds the valve collection
valves << all_rwos.select(:collection, {:valve})
number_of_valves << valves.size
```

Queries are built up using predicate objects:

```
# find 'open' valves.
open_valves << valves.select(predicate.eq(:operating_status, "open"))
number_of_open_valves << open_valves.size
```

```
_for valve _over open_valves.elements()
_loop
  write(valve.id)
_endloop
```

Joins are implemented as methods on the parent RWO. For example a manager might have several employees who report to him:

```
# get the employee collection.
employees << my_application.database.collection(:gis, :employees)
```

```
# find a manager called 'Steve' and get the first matching element
steve << employees.select(predicate.eq(:name, "Steve").and(predicate.eq(:role, "manager")).an_element()
```

```
# display the names of his direct reports. name is a field (or column)
# on the employee collection (or table)
_for employee _over steve.direct_reports.elements()
_loop
   write(employee.name)
_endloop
```

Performing a transaction:

```
# each key in the hash table corresponds to the name of the field (or column) in
# the collection (or table)
valve_data << hash_table.new_with(
  :asset_id, 57648576,
  :material, "Iron")
```

```
# get the valve collection directly
valve_collection << my_application.database.collection(:gis, :valve)
```

```
# create an insert transaction to insert a new valve record into the collection a
# comment can be provide that describes the transaction
transaction << record_transaction.new_insert(valve_collection, valve_data, "Inserted a new valve")
transaction.run()
```

External links

- Smallworld Product Suite Technology [1]
- Version Management in GIS - Applications and Techniques [2]
- GIS Databases are Different [3]
- The why and how of the long transaction [4]

References

[1] http://www.ge-energy.com/products_and_services/products/geospatial_systems_and_mobile_workforce/smallworld_geospatial_solutions.jsp
[2] http://cfis.savagexi.com/pages/technical_paper_4
[3] http://cfis.savagexi.com/pages/technical_paper_8
[4] http://cfis.savagexi.com/pages/technical_paper_9

Superbase database

Superbase is an end-user desktop database program that started on the Commodore 64 and was ported from that to various operating systems over the course of more than 20 years. It also has generally included a programming language to automate database-oriented tasks, and with later versions included WYSIWYG form and report designers as well as more sophisticated programming capabilities.

History

It was originally created in 1983 by Precision Software for the Commodore 64 and 128 and later the Amiga and Atari ST. In 1989, it was the first database management system to run on a Windows computer.

Precision Software, a UK-based company, was the original creator of the product Superbase. Superbase was and still is used by a large number of people on various platforms. It was often used only as an end-user database but a very large number of applications were built throughout industry, government, and academia and these were often of significant complexity. Some of these applications continue in use to the current day, mostly in small businesses.

The initial versions were text mode only, but with the release of the Amiga version, a completely new paradigm for accessing databases was created. Superbase was the first product to use the now common VCR control panel for browsing through records. It also provided a number of different media formats that it directly supported, including images, sounds, and video. Superbase was often referred to as the *multimedia* database in early years, when such features were uncommon. Amiga version also featured an internal language and the capability to generate front end "masks" for queries and reports, years before Microsoft Access.

This version was a huge success and that resulted in a version being created for a number of platforms using the same approach. Eventually a Microsoft Windows version was released and a couple of years later the company was sold by its founders to Software Publishing Corporation. SPC sold off the non-Windows versions of the product and after releasing version 2 and in the late alpha stages of version 3, sold the product to a company called Computer Concepts Corporation.

This relatively unknown company created a subsidiary called Superbase, Inc. and after finishing off the late stage alpha of version 3 and launching it as Superbase 95, eventually appeared to have lost interest in the product, at which point it was bought by a small group of former customers and brought back to the UK. This company, Superbase Developers plc, continued to extend and support the product through Superbase Classic.

A new, next-generation rewrite of the product initially called Superbase Next Generation (SBNG) which included a new object oriented programming language called SIMPOL was begun in 1999-2000. It had primarily been an alpha product; although it was billed as a beta release in 2005 with promises that a true release would be around the corner.

In 2006, SIMPOL was sold to RealBasics Ltd which was later renamed Simpol Ltd (www.simpol.com).

In April 2009 this company launched SIMPOL Professional, which is the next generation product, as a cross-platform language and database tool set.

In February 2009, it was announced that Superbase Developers plc was in liquidation.

In March 2010 Papatuo Holdings Ltd. purchased the Superbase family of products from the official receivers of Superbase Developers plc.[1]

Uses

Superbase has been used for very basic end-user tasks, but its real strength lies in the ability of relatively untrained programmers to create complex applications. These are typically built up over time as need arises. The types of applications run the gamut from accounting systems, ERP/MRP packages, business information systems, production control systems, and similar complex products down to very basic membership list or contact management systems.

Features

It contains a high-speed versatile ISAM database engine and its own powerful dialect of BASIC, as well as sophisticated forms and report engines. It also includes powerful support for acting as the front-end for one or more SQL databases. Its biggest drawback is the fact that it was written to the 16-bit Windows API and was not easily portable to the 32-bit version. The rewritten version is intended to cure that and to make the package even easier to use and more powerful.

From a casual programmer's perspective, the fact that the database is not based on SQL is a significant advantage, since the level of complexity is far less and it is easier for the user to grasp the concepts of how to manage and traverse the database.

There are numerous powerful features in the product, a few of them are:

- Virtual Database Tables — these only exist in memory
- Virtual Database Columns — these are calculated at access time
- Peer-to-peer Client/Server (PPCS) — this technology allows any version of Superbase to act as either a database server, a client, or both. The database tables are accessed via UDP/IP.
- Small footprint — Superbase runs on every version of Windows except the 64-bit versions and requires only a minimum of 6 MB of system RAM.

Versions

- 1983 Superbase 64 for the Commodore 64
- 1983 Superbase 700 for the Commodore CBM-II
- 1983 Superbase version 2.0 for the Apple II
- 1984 Superbase for the Commodore Plus/4
- 1985 Superbase for the Amiga
- 1985 Superbase 128 for the Commodore 128
- 1986 Superbase for the Atari ST
- 1987 Superbase for GEM on the PC
- 1988 Superbase 4 version 1.0 for Windows
- 1991 Superbase 4 version 1.31 for Windows
- 1991 Superbase 4 version 1.31 for Amiga
- 1992 Superbase version 2.0 for Windows
- 1994 Superbase 95 (version 3.0) for Windows
- 1997 Superbase version 3.2 for Windows
- 1998 Superbase version 3.5 for Windows
- 1999 Superbase version 3.6i for Windows
- 2000 SuperBase 4 Pro version 1.36 for Amiga
- 2001 Superbase 2001 for Windows
- 2003 Superbase Classic for Windows

External links

- Superbase Developers plc official website [2]
- Simpol Limited official website [3]
- SBase4Pro Amiga Upgrade [4]

References

[1] http://www.superbase.com/
[2] http://www.superbase.com
[3] http://www.simpol.com
[4] http://www.mrhardwarecomputers.com/pages/sb4upgrd.htm

Rocket U2

Rocket U2 Product Family

Developer(s)	Rocket Software
Stable release	UniData 7.3, UniVerse 11.1, SB/XA 6.2.2, Web DE 5.0
Operating system	Cross-platform
Type	MultiValue database
License	Proprietary
Website	www.rocketsoftware.com/u2 [1]

Rocket U2 is a suite of database management (DBMS) and supporting software now owned by Rocket Software. It includes two MultiValue database platforms: *UniData* and *UniVerse*.[2] Both of these products are operating environments which run on current Unix, Linux and Windows operating systems.[3][4] They are both derivatives of the Pick operating system.[5][*citation needed*] The family also includes developer and web-enabling technologies including *SystemBuilder/SB+*, *SB/XA*, *U2 Web DE*, *UniObjects* and *wIntegrate*.

History

UniVerse was originally developed by VMark Software and UniData was originally developed by the Unidata Corporation. Both Universe and Unidata are used for vertical application development and are embedded into the vertical software applications. In 1997, the Unidata Corporation merged with VMark Systems to form Ardent Software.[6] In March 2000, Ardent Software was acquired by Informix. IBM subsequently acquired the database division of Informix in April 2001,[7] making UniVerse and UniData part of IBM's DB2 product family. IBM subsequently created the Information Management group of which Data Management is one of the sub-areas under which the IBM U2 family comprised UniData and UniVerse along with the tools, SystemBuilder Extensible Architecture (SB/XA), U2 Web Development Environment (U2 Web DE) and wIntegrate.

On 1 October 2009 it was announced that Rocket Software had purchased the entire U2 portfolio from IBM.[8][9] The U2 portfolio is grouped under the name RocketU2.

System structure

Accounts

Systems are made of one or more accounts. Accounts are directories stored on the host operating system that initially contain the set of files needed for the system to function properly. This includes the system's VOC (vocabulary) file that contains every command, filename, keyword, alias, script, and other pointers. Each of these classes of VOC entries can also be created by a user.

Files

Files are similar to tables in a relational database in that each file has a unique name to distinguish it from other files and zero to multiple unique records that are logically related to each other.

Files are made of two parts: a data file and a file dictionary (DICT). The data file contains records that store the actual data. The file dictionary may contain metadata to describe the contents or to output the contents of a file.

Hashed files

For hashed files, a U2 system uses a hashing algorithm to allocate the file's records into groups based on the record IDs. When searching for data in a hashed file, the system only searches the group where the record ID is stored, making the search process more efficient and quicker than searching through the whole file.

Nonhashed files

Nonhashed files are used to store data with little or no logical structure such as program source code, XML or plain text. This type of file is stored as a subdirectory within the account directory on the host operating system and may be read or edited using external tools.

Records

Files are made of records, which are similar to rows within tables of a traditional relational database. Each record has a unique key (called a "record ID" in U2) to distinguish it from other records in the file. These record IDs are typically hashed so that data can be retrieved quickly and efficiently.

Records (including record IDs) store the actual data as pure ASCII strings; there is no binary data stored in U2. For example, the hardware representation of a floating-point number would be converted to its ASCII equivalent before being stored. Usually these records are divided into fields (which are sometimes called "attributes" in U2). Each field is separated by a "field mark" (hexadecimal character FE).

Thus this string:

```
123-45-6789^JOHN JONES^jjones@example.com^432100^...
```

might represent a record in the EMPLOYEE file with 123-45-6789 as the Record ID, JOHN JONES as the first field, jjones@company.com as the second field and $4321.00 as a monthly salary stored in the third field. (The up-arrow (^) above is the standard Pick notation of a field mark; that is, xFE).

Thus the first three fields of this record, including the record ID and trailing field mark, would use 49 bytes of storage. A given value uses only as many bytes as needed. For example, in another record of the same file, JOHN JONES (10 bytes) may be replaced by MARJORIE Q. HUMPERDINK (21 bytes) yet each name uses only as much storage as it needs, plus one for the field mark.

Fields may be broken down into values and even subvalues. Values are separated by value marks (character xFD); subvalues are separated by subvalue marks (character xFC). Thus, if John Jones happened to get a second email address, the record may be updated to:

```
123-45-6789^JOHN JONES^jjones@example.com]johnnyjones@example.net^432100^...
```

where the close bracket (]) represents a value mark.

Since each email address can be the ID of a record in separate file (in SQL terms, an outer join; in U2 terms, a "translate"), this provides the reason why U2 may be classified as a MultiValued database.

Data

Raw information is called Data. A record is a set of logical grouped data. e.g. an employee record will have data stored in the form of fields/attributes like his name, address etc.

Programmability

Both UniVerse and UniData have a structured BASIC language (UniVerse Basic and UniBasic, respectively), similar to Pick/BASIC which naturally operates on the structures of the MultiValue database. They also have a structured database query language (RetrieVe and UniQuery) used to select records for further processing and for adhoc queries and reports.

RocketU2 provides a set of Client Tools to allow software developers to access U2 databases from other software languages.[10]

Client Tool interfaces include:

- ODBC / JDBC
- UniOLEDB - OLEDB Driver
- UniObjects (COM)
- UniObjects (.NET)
- UniObjects (Java)
- Native XML
- U2 Web Services

Professional certification

RocketU2 offers five professional certification designations related to the U2 product family. All carry the title *Certified Solutions Expert*.

- U2 Family Application Development
- U2 UniData V7.1 Administration
- U2 UniData V7.2 Administration
- U2 UniVerse V10.2 Administration
- U2 UniVerse V10.3 Administration

Notes

[1] http://www.rocketsoftware.com/u2
[2] 'U2 Product Family' (http://www.rocketsoftware.com/u2/), Rocket Software
[3] 'U2 Product Matrix' (http://u2tc.rocketsoftware.com/matrix.asp) Rocket Software
[4] 'UniVerse System Description, Version 10.3' (http://www.rocketsoftware.com/u2/epubs/pdf/22922140.pdf) page 1-3, Rocket Software
[5] 'UniVerse Guide for Pick Users, Version 10.3' (http://www.rocketsoftware.com/u2/epubs/pdf/22922260.pdf) page 1-3, Rocket Software
[6] 'Ardent Definition' (http://www.pcmag.com/encyclopedia_term/0,2542,t=Ardent&i=37967,00.asp), PC Magazine
[7] 'Informix Definition' (http://www.pcmag.com/encyclopedia_term/0,2542,t=Informix&i=44972,00.asp) PC Magazine
[8] 'Rocket Software U2 Acquisition Announcement' (http://www.rocketsoftware.com/news/news_item/205), Rocket Software
[9] 'Rocket Software to purchase U2 from IBM' (http://www.intl-spectrum.com/article/426/Rocket_Software_to_purchase_U2_from_IBM.aspx), International Spectrum
[10] 'Client Tools' (http://www.rocketsoftware.com/u2/middleware/), Rocket Corporation

External links

- Official website (http://www.rocketsoftware.com/u2)
- U2UG (http://u2ug.org), a recognized international user group

PointBase

PointBase

Developer(s)	PointBase Inc.
Written in	Java
Operating system	Cross-platform
Type	RDBMS

PointBase is relational database management system (RDBMS) written in the Java programming language.

History

In 1998, Bruce Scott, a co-founder of the Oracle Corporation (with Larry Ellison, Bob Miner and Ed Oates), started PointBase Inc. with Jeff Richey (an architect of Sybase) and Daren Race. It was written in pure Java, and supported DCOM and CORBA, and is an object-relational database. It was designed to integrate the internet and databases. PointBase Inc. was established in San Mateo, California, then moved to Mountain View, California. Like Java, PointBase was aimed at portable devices.

In the early 2000s it was the database that was shipped for free with the Java platform.

In 2003, the database was acquired by DataMirror of Markham, Ontario.

In September 2007, IBM acquired DataMirror.

Today (2012) PointBase's SQL Engine is part of Oracle's WebLogic platform.

Applications

It has been shipped with the Oracle WebLogic Server, a Java EE server.

PointBase is supported only for the design, development, and verification of applications; it is not supported for enterprise-quality deployment. The evaluation license of PointBase has a database size limit of 30 MB.

Versions

- PointBase Server Edition
- PointBase Mobile Edition

References

External links

- History of PointBase (http://www2.sys-con.com/itsg/virtualcd/java/archives/0406/davison/index.html)
- Guide to PointBase (http://www.cs.toronto.edu/~nn/csc309/guide/pointbase/docs/html/htmlfiles/pbsystemTOC.html)
- Oracle BEA WebLogic Pointbase Reference (http://docs.oracle.com/cd/E13218_01/wlp/docs92/db/pointbase.html)

R:BASE System

R:BASE (or RBASE) was the first relational database program for the PC. Created by Wayne Erickson in 1981, on November 13, 1981, Erickson and his brother, Ron Erickson, incorporated the company, MicroRim, Inc. to sell the database, MicroRIM. In June 1998, A. Razzak Memon, President & CEO of R:BASE Technologies, Inc. (a privately held company in Murrysville, Pennsylvania, USA) acquired the R:BASE products from Abacus Software Group. Since 1998, R:BASE is available as R:BASE for Windows v6.1a, v7.1, v7.5, v7.6, Turbo V-8, v9.1, and now v9.5 (32/64) for Windows.

History

Founding

Created by Wayne Erickson in 1981, the original R:Base database was written on a Heathkit CPM computer that Erickson built at home. On November 13, 1981, Erickson and his brother, Ron Erickson, incorporated the company, MicroRim, Inc. to sell the database, MicroRIM. (RIM was an acronym for Relational Information Management, a mainframe database developed by Erikson at Boeing Computer Services, as part of NASA's IPAD project for which he and NASA colleagues received a NASA award, was used by NASA to track Space shuttle heat shield tiles).

The earliest version released by Microrim was called R:Base 4000 and was released in 1983. It worked with early version of Microsoft MS-DOS or IBM PC DOS (version 2 or above). It shipped with a binder-type manual and the program on 360K floppy disks. The system being DOS-based, the interface was entirely text with the exception of DOS line-draw characters.[*citation needed*]

Privately funded and ultimately venture backed, the MicroRim database products achieved significant market share in the mid-1980s in what was dubbed by some, the "database wars" between R:Base and the market share leader, Ashton-Tate's D-Base. One clever MicroRim ad stated "R-way versus D-hardway," a jab at the inferior D-Base architecture. MicroRim adhered to the rules of the father of relational database technology, Edgar F. Codd and prided itself on the elegance of its code.[*citation needed*]

In the mid-1980s, when Microsoft did not have their own database, they obtained a license to resell R:BASE in Europe so they could have a full suite of software products.[*citation needed*]

1990s

In June 1998, R:BASE Technologies, Inc. (a privately held company in Murrysville, Pennsylvania, USA) acquired the R:BASE products from Abacus Software Group.[*citation needed*]

Recent years

Some of the features included, and continue to include, a programming-free application development wizard, automatic multi-user capabilities, a full-featured 4GL programming language, form, report and label designers, and a fully ANSI SQL compliant relational language capability.[*citation needed*]

Since September 2007, R:BASE is available as R:BASE for Windows v7.6, R:BASE for DOS v7.6 and R:BASE Turbo V-8 for Windows. The Version 8.0 has an extended address management for file handling and is able to cover databases up to 2.3 million TB versus V7.6 which covers databases up to 2 GB. A German kernel has existed since R:Base V7.6.[*citation needed*]

Legacy R:BASE Products

R:BASE 4000

The earliest version released by Microrim was called R:Base 4000 and was released in 1983. It worked with early version of Microsoft MS-DOS or IBM PC DOS (version 2 or above). It shipped with a binder-type manual and the program on 360K floppy disks. The system being DOS-based, the interface was entirely text with the exception of DOS line-draw characters.

In spite of its relative ease of use and ability to create useful forms and reports, the first R:Base did not have a conventional programming language, but instead relied on SQL statements to accept input and produce output. The lack of a complete programming language meant that the product was not well received by some portions of the market. This may have helped the early, barely relational, dBase products to become dominant. The product was quickly upgraded to include Added Variables and a conventional programming Language (IF, WHILE, etc.) to the original SQL based language The update was released as R:Base 4000 Version 1.1 in March 1984. R:Base became the second most popular DOS database in the PC market (behind dBase).[*citation needed*]

Portions of the program allowed the user to design screens, called "Forms" in R:Base. Line-draw characters could implement buttons or boxes that would group text on screen. A separate utility allowed the design of printed output formats and was called "Reports." The report design system allowed a user to define and edit fields included in database reports on screen. Limited printer support was included as DOS programs each had their own unique printer driver for similar printer engines. A markup language allowed italics and bold output if the corresponding printer had a capability. Reports could be piped to the display or a serial port for testing if one were so inclined. Database names were constrained to seven characters. The actual data were contained in three files. In an example database named *Sales*, files name SALES1.RBF SALES2.RBF, AND SALES3.RBF would contain the database. Forms and reports were stored in files external to the database file.[*citation needed*]

By default, the application would start with a menu asking which database file you wanted to open. Using a startup switch, R:Base could be run entirely from a command prompt, called the "R-prompt," in system documentation. The application command prompt was an "R>" although this could be modified to other characters by editing a configuration file. In an example database named *Sales*, to query the database, you would first open it by typing "OPEN SALES" at the R-prompt. Using SQL-style queries, one could pull on-screen displays of data from tables. "SELECT FNAME LNAME CITY ZIPCODE FROM MAIN" would display one screen of data from the fields FNAME LNAME CITY ZIPCODE from the table named MAIN. Pressing the space bar would scroll to the next 24 records. A built-in help system produced text after the R> prompt if your query was invalid or the syntax was not understood by the database engine.[*citation needed*]

A feature of the program was its ability to create applications that ran scripts generated by an internal scripting system. Scripts were stored in files with an extension .APP. The system would first ask for type of menu desired, (one option was pull-down, for example,) then asked you to fill out the pull down headings. Next, you were stepped through a list of actions for each menu choice. At the end, the procedures that had been stepped through were recorded in the database file and could be called from an automatically generated menu system. To prevent a user from tampering with the generated script, an encoded version was created. The user could password protect the encoded version for configuration management.

A utility called *File Gateway* allowed import and export of common file formats of the era such as Data Interchange Format (DIF), SYLK, Lotus 1-2-3, and dBase files. Another utility, called *Recover*, was intended to recover

damaged R:Base databases.[*citation needed*]

R:BASE 5000, R:BASE 2.0

R:Base 4000 was followed by R:Base 5000, which substantially improved features and gained wider acceptance.

R:BASE 2.0 rolled out a new file format and introduced the ability to use memory above 640K. There was support for the Intel 80286 processor. The system had substantially better documentation. This version continued the evolution toward full ANSI SQL compliance. Forms, scripts, and reports were rolled into the database files. Three files with extensions .RB1, .RB2, and .RB3 contained everything for a single database.[*citation needed*]

R:BASE 3.x

R:Base 3.0 was ANSI SQL (1989?) compliant and utilized the DOS4GW memory manager. This memory manager was also seen in many DOS games of the era. R:Base 3.1 introduced a **multi-user network capability**. A version was also rolled out for the Convergent Technologies Operating System operating system, this was apparently a follow-on to Burroughs Teechnologies Operating System (BTOS).[*citation needed*]

By purchasing license packs, the database gained a multi-user capability in five-user increments. This included a sophisticated (for a DOS application in the day) record-level locking scheme. To work properly, the multi-user database had to be on a file server with all users accessing the database through a network. It was not true client-server because processing occurred in the clients. The configuration file expanded to allow language support and user-defined re-mapping of characters. For example, German characters such as the letter "ö" (o with an umlaut) could be remapped to the string *oe*. There were character fold tables and sort orders could be adjusted by the user. An "unlimited number of licenses" runtime version was offered, allowing developers to sell applications and include the run-time R:Base engine.

Example of an R:Base 3.1 command prompt transaction asking the application to list the structure of a database table of California cities, (CALIFCY):[*citation needed*]

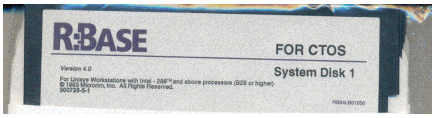
```
R> LIST CALIFCY
  # Name               Type      Index Expression
  1 STATE              TEXT    2
  2 FEATURE            TEXT   85
  3 FEATURET           TEXT    9
  4 COUNTY1            TEXT   15
  5 FIPSST             TEXT    2
  6 FIPSCO             TEXT    3
  7 LATITUDE           TEXT    7
  8 LONGITUD           TEXT    8
  9 LAT_DEC            TEXT    8
 10 LON_DEC            TEXT   10
 11 SOURCELA           TEXT    7
 12 SOURCELO           TEXT    8
 13 SOUR_lat           TEXT    8
 14 SOUR_lon           TEXT   10
 15 ELEVATIO           TEXT    5
 16 FIELD16            TEXT    8
 17 MAPNAME            TEXT   27
 18 LAT1               DOUBLE
 19 LON                DOUBLE
 20 ITEM_NO            DOUBLE
```

```
  Current number of rows:   7070

R>
```

R:BASE 4.x

R:Base 4.0 rolled out Intel 80386 support and a newer DOS4GW memory manager. It included a newer file format, replacing the format used with Version 3.1. To support legacy customers, Version 4.0 included a copy of Version 3.1 with a lot of warnings about new file format and features of 4 that were not supported in 3.1. While the documentation claimed 2GB data files were supported, there were data integrity problems with some very large tables over 1 million records. Still, the software was designed to accommodate up to 750 tables and easily handled tables with tens of thousands of records. It was faster than 3.1 and a reliable and practical application for many users.[*citation needed*]



R:Base for CTOS disk label

R:Base 4.5 rolled out another new file format and greatly improved capacity. OBDC drivers were rolled out to allow interchange of data with Microsoft Windows based applications without running the DOS-based File Gateway utility. While number of records in a database was "limited only by disk space," in practice, some users found there were problems with databases which contained over about 1.1 million records.[*citation needed*]

First R:BASE for Windows

The first product produced by Microrim for use in Microsoft Windows was named R:Base for Windows. This rolled out in 1994. This version was compatible with R:Base 4.5 files and was fully ANSI SQL Level II 1989 compliant. The application was partially ANSI SQL 1992 Level II compliant. The screen capture images in documentation look like Windows 3.1, but documentation claimed it would also run on Windows 95 or the more trustworthy Windows inside OS/2 Warp version 3. A variety of run-time licensing schemes were available to developers.[*citation needed*]

Current Generation R:BASE Products

- R:BASE 7.6 for Windows
- R:BASE 7.6 for DOS
- R:BASE Turbo V-8 for Windows
- R:BASE 9.1 for DOS
- R:BASE eXtreme 9.1 (32) for Windows
- R:BASE eXtreme 9.1 (64) for Windows
- **R:BASE eXtreme 9.5 (32) for Windows**
- **R:BASE eXtreme 9.5 (64) for Windows**

References

External links

- R:BASE Technologies, Inc. (http://www.rbase.com)
- R:BASE Technical Articles (http://www.razzak.com/fte/)
- R:BASE Sample Applications (http://www.razzak.com/sampleapplications/)
- RBase Box Picture 1 (http://www.rbase.com/history/boxes/microsoft/microsoft_1.jpg)
- RBase Box Picture 2 (http://www.rbase.com/history/boxes/microsoft/microsoft_2.jpg)
- RBase Box Picture 3 (http://www.rbase.com/history/boxes/microsoft/microsoft_3.jpg)
- RBase Box Picture 4 (http://www.rbase.com/history/boxes/microsoft/microsoft_4.jpg)
- R:Base Support and Distribution for German spoken countries (http://www.rbase.ch)

REAL Server

REAL Server is a relational database management system (RDBMS) built on top of the sqlite database engine.

History

REAL Server evolved from the SQLiteServer originally developed by SQLabs in 2004. In May 2005 Real Software, Inc., creator of Realbasic, purchased the source code and the copyrights of the SQLiteServer and invested in its development. In 2007/2008 the first version of the REAL SQL Server was released. A new version was released in April 2009 and renamed REAL Server. In September 2010 SQLabs repurchased all the server's Intellectual Properties and a new major release is now under development by the SQLabs team.[*citation needed*]

Features

- Event based,
- Asynchronous sockets,
- Multi-core and multiprocessor aware,
- Strong AES encryption (128, 192 and 256 bit),
- Supports unlimited connections (For each supported operating system, REAL Server uses a state of the art event API, kqueue on Mac OS X, epool on Linux and I/O Completion Ports on Windows),
- Full ACID (Atomic, Consistent, Isolated, Durable) compliant,
- Platform independent storage engine,
- Full support of triggers and transactions,
- Journal engine for crash recovery,
- Supports databases of 2 terabytes,
- Supports sqlite 3 databases,
- Automatic logging,
- Automatic compression,
- Multiversion concurrency control (MVCC),
- Plugins for extending the SQL language and the custom commands supported by the server,
- Restore and backup support,
- Mac OS X, Windows and Linux support.

Connectivity

REAL Server can be used with the following:

- Realbasic
- PHP
- C SDK
- ODBC

External links

- SQLabs. [1]
- REAL Server [2], Official REAL Server website

References

[1] http://www.sqlabs.com

[2] http://www.sqlabs.com/sqlabsserver.php

[1]

[1] REAL Software (June 12, 2009). "REAL Server". REAL Software. Retrieved on 2009-06-12.

MaxDB

MaxDB

Developer(s)	SAP AG
Stable release	7.6.06.10/7.7.07.16 / March, 2010
Development status	Active
Written in	C++
Operating system	Cross-platform
Available in	English
Type	RDBMS
License	SAP freeware license agreement for MaxDB (closed source)
Website	http://maxdb.sap.com/

MaxDB is an ANSI SQL-92 (entry level) compliant relational database management system (RDBMS) from SAP AG, which was delivered also by MySQL AB from 2003 to 2007. MaxDB is targeted for large SAP environments e.g. mySAP Business Suite and other applications that require enterprise-level database functionality. It is able to run terabyte-range data in continuous operation.[*citation needed*]

History

The database development started in 1977 as a research project at the Technical University of Berlin headed by Rudolf Munz. In the early 1980s it became a database product that subsequently was owned by Nixdorf Computer, Siemens-Nixdorf, Software AG and today by SAP AG. Along this line it has been named VDN, RDS, Reflex, Supra 2, DDB/4, Entire SQL-DB-Server and Adabas D. In 1997 SAP acquired the software from Software AG and developed it as SAP DB, releasing the source code under the GNU General Public License in October 2000.

In 2003 SAP AG and MySQL AB joined a partnership and re-branded the database system to MaxDB. In October 2007 this reselling was terminated and sales and support of the database reverted to SAP.[1] SAP AG is now managing MaxDB development, distribution, and support. Source code of MaxDB is no longer available under the GNU General Public License. SAP also stated that "Further commercial support concepts to cover mission critical use requirements outside of SAP scenarios are currently subject to discussion."[2]

Version 7.5 of MaxDB is a direct advancement of the SAP DB 7.4 code base. Therefore, the MaxDB software version 7.5 can be used as a direct upgrade of previous SAP DB versions starting 7.2.04 and higher.

Features

MaxDB is delivered with a set of administration and development tools. Most tools are GUI based and have CLI (Command Line Interface) based counterparts. It offers bindings for JDBC; ODBC; SQLDBC (native C/C++ interface); Precompiler; PHP; Perl; Python; WebDAV; OLE DB, ADO, DAO, RDO and .NET via ODBC; Delphi and Tcl via Third Party Programming Interfaces. MaxDB is Cross-platform, offering releases for HP-UX, IBM AIX, Linux, Solaris, Microsoft Windows 2000, Microsoft Windows Server 2003, and Microsoft Windows XP. SAP users should check the details of the platform availability on the SAP product pages for the product that will be used together with MaxDB. MaxDB is subjected to SAP AG's quality assurance process before it is shipped.

Distinguishing features

MaxDB offers built-in hot backup, does not need any online reorganizations and claims to be SQL 92 Entry-Level compatible. One current development goal is "zero administration" and by this "low TCO". It is also fair to expect good online transaction processing (OLTP) performance combined with relatively low hardware requirements.

High availability mode and small fault record is another historically observed feature of MaxDB, which holds some appraisal from the open source software and db communities.

Future releases

The next release will have the name MaxDB 7.7.00. One possible future feature for 7.7.00 is the use of Multiversion Concurrency Control (MVCC) instead of the current lock based implementation.

Licensing

MaxDB was licensed under the GNU GPL from versions 7.2 through 7.6. Programming interfaces were licensed under the GPL with exceptions for projects released under other Open Source licenses.

SAP DB 7.3 and 7.4 were licensed as GPL with LGPL drivers. MaxDB 7.5 was offered under dual licensing.i.e. licensed as GPL with GPL drivers or a commercial license.

From version 7.5 through version 7.6 onwards distribution of MaxDB (previously SAP DB) to the open source community was provided by MySQL AB, the same company that develops the open-source software database, MySQL. Development was done by SAP AG, MySQL AB and the open-source software community.

In October 2007, SAP assumed full sales and commercial support for MaxDB. MaxDB 7.6 is now closed source, available free-of-charge (without support, and with usage restrictions) for use with non-SAP applications. Commercial support models for using MaxDB outside of SAP environments are under consideration.

References

[1] MySQL AB :: MySQL AB to Optimize its Open Source Database for SAP NetWeaver (http://www.mysql.com/news-and-events/press-release/release_2007_40.html)

[2] *MaxDB back under the SAP roof!* (https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/7514)

External links

- Official website (http://maxdb.sap.com/)
- MaxDB Wiki (http://wiki.sdn.sap.com/wiki/display/MaxDB/) on SAP Community Network
- SAP MaxDB - The SAP Database System (http://scn.sap.com/community/maxdb) discussions, blogs, documents and videos on the SAP Community Network (SCN) (http://scn.sap.com/welcome)

Adaptive Server Enterprise

Adaptive Server Enterprise

Developer(s)	Sybase - An SAP Company
Initial release	May 23, 1980
Written in	C, C++
Operating system	Cross-platform
Available in	English
Type	RDBMS
License	Proprietary EULA
Website	ASE [1]

Adaptive Server Enterprise (ASE) is a relational model database server product for businesses developed by Sybase Corporation which became part of SAP AG. ASE is predominantly used on the Unix platform, but is also available for Microsoft Windows.

History

Originally for Unix platforms in 1987, Sybase Corporation's primary relational database management system product was initially marketed under the name Sybase SQL Server. In 1988, SQL Server for OS/2 was co-developed for the PC by Sybase, Microsoft, and Ashton-Tate. Ashton-Tate divested its interest and Microsoft became the lead partner after porting SQL Server to Windows NT. Microsoft and Sybase sold and supported the product through version 4.2.1.

Sybase released SQL Server 4.8 in 1992. This release included internationalization and localization and support for symmetric multiprocessing systems.

In 1993, the co-development licensing agreement between Microsoft and Sybase ended, and the companies parted ways while continuing to develop their respective versions of the software. Sybase released SQL Server 10.0, which was part of the System 10 product family, which also included Back-up Server, Open Client/Server APIs, SQL Monitor, SA Companion and OmniSQL Gateway.

Sybase provides native low-level programming interfaces to its database server which uses a protocol called Tabular Data Stream. Prior to version 10, DBLIB (DataBase LIBrary) was used. Version 10 and onwards uses CTLIB (Client LIBrary).

In 1995, Sybase released SQL Server 11.0. Starting with version 11.5 released in 1996, Sybase moved to differentiate its product from Microsoft SQL Server by renaming it to Adaptive Server Enterprise.

In 1998, ASE 11.9.2 was rolled out with support for row-level locking, distributed joins and improved SMP performance. ASE 12.0 was released in 1999, providing support for Java, high availability and distributed transaction management. In 2001, ASE 12.5 was released, providing features such as dynamic memory allocation, an EJB container, and support for XML and SSL.

In 2005, Sybase released ASE 15.0. It included support for partitioning table rows in a database across individual disk devices, and "virtual columns" which are computed only when required. In ASE 15.0, many parameters that had been static (which required server reboot for the changes to take place) were made dynamic (changes take effect immediately). This improved performance and reduced downtime. For example, one parameter that was made dynamic was the "tape retention in days" (the number of days that the backup is kept on the tape media without

overwriting the existing contents in the production environment).

Structure

A single stand alone installation of ASE typically comprises one "dataserver" and one corresponding "backup server". In multi server installation many dataservers can share one single backup server though. A dataserver consists of system databases and user's databases. Minimum system databases that are mandatory for normal working of dataserver are 'master', 'tempdb', 'model', 'sybsystemdb' and 'sybsystemprocs'. 'master' database holds critical system related information that includes, logins, passwords, and dataserver configuration parameters. 'tempdb' is used for storage of data that are required for intermediate processing of queries, and temporary data. 'model' is used a template for creating new databases. 'sybsystemprocs' consists of system supplied stored procedures that queries system tables and manipulates data in them.

ASE is a single process multithreaded dataserver application, it means when server is up and running there is one single OS process.

Versions

SAP also has a developer edition that can be used for free to develop against (but not for production use).[2] It only allows 1 engine and 25 connections.[3] There is also an SAP Sybase ASE, express edition available which is limited to 1 server engine, 2 Gb of memory and 5 Gb of disk space per server. This edition is free for production purposes.

References

[1] http://www.sybase.com/products/databasemanagement/adaptiveserverenterprise
[2] http://www.sybase.com/ase_1500devel
[3] http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.asewqig/html/asewqig/asewqig3.htm

External links

- SAP Sybase ASE official website (http://www.sap.com/pc/tech/database/software/adaptive-server-enterprise/index.html)
- SAP Sybase ASE online documentation (http://sybooks.sybase.com/nav/summary.do?prod=9938&lang=en&prodName=Adaptive+Server+Enterprise)
- SAP Sybase ASE Wiki (http://sybaseblog.com/sybasewiki/index.php?title=Sybase_Adaptive_Server_Enterprise_(ASE))

Advantage Database Server

Advantage Database Server

Developer(s)	Sybase
Initial release	August, 1993
Written in	C, C++
Operating system	Cross-platform
Available in	English
Type	RDBMS
Website	advantagedatabase.com [1]

Advantage Database Server is a relational database management system (RDBMS) for small to medium sized businesses by Sybase iAnywhere. Database author Cary Jensen describes Advantage as follows: "Advantage has been around since 1993, when it was introduced to provide a stable solution for Clipper developers who were tired of slow performance and corrupt indexes inherent to file server-based databases. Over the years, ADS has grown in both popularity and features. Advantage is now a mature product with an impressive collection of features that rival many of the more expensive and complicated database servers".[2]

“In short, the Advantage Database Server is a high-performance, low-maintenance, remote database server that permits you to easily build and deploy client/server applications and web-based applications”.[3]

Uses

Advantage Database Server is most popular among application developers as a client/server backend solution for shared, networked, standalone, mobile and Internet database applications. ADS is unique among other database offerings because it provides both ISAM table-based and SQL based data access.

Features

- Replication
- Online Backup
- ISAM access (Navigational Access)
- SQL-92 compliant
- SQL Query Optimizer
- User Defined Functions
- Triggers
- Stored Procedures
- Views
- Server Side Aliases
- Encrypted indexes and communications
- DBF tables and memos greater than 4 gigabytes
- Transactions
- Events / Notifications
- 64-bit support
- Full Text Search
- Multiple Processor support

- Small footprint
- Odata Web Service
- Native DBF data format support
- Unicode Support

Programming Languages

- Popular development environments including CodeGear Delphi, CodeGear C++Builder, Microsoft Visual Basic, Microsoft Visual C++, CA-Visual Objects, CA-Clipper and Microsoft Visual FoxPro.
- Supports standard interfaces such as ODBC, OLE DB, JDBC, PHP, and ADO.NET.

History

- August 1993 – Initially released as Advantage xBase server by Extended Systems.
- December 1995 – version 4.0 release, now called Advantage Database Server. This release included the first ODBC driver.
- February 1997 – Advantage Internet Server released.
- April 2000 – Advantage ODBC driver with StreamlineSQL released.
- November 2005 – Advantage Database Server: The Official Guide published.[4]
- November 2005 – Extended Systems acquired by Sybase, rolled in to iAnywhere Solutions subsidiary.
- November 2008 – Advantage Database Server 9.1.
- June 2010 – Advantage Database Server 10.0
- December 2010 - Advantage Database Server 10.1
- July 2012 – Advantage Database Server 11.0

References

[1] http://www.advantagedatabase.com/

[2] Cary Jensen, Ph.D.Loy Anderson, Ph.D, Advantage Database Server: A Developer's Guide, February 5, 2007, 616 pages, AuthorHouse, 978-1425-7726-9

[3] Cary Jensen, Ph.D.Loy Anderson, Ph.D, Advantage Database Server: A Developer's Guide, February 5, 2007, page 7, AuthorHouse, 978-1425-7726-9

[4] Cary Jensen, Ph.D.Loy Anderson, Ph.D, Advantage Database Server: The Official Guide, October 29, 2003, 496 pages, McGraw-Hill Osborne Media, 0-07-223084-1

External links

- Advantage Database Server website (http://www.sybase.com/products/databasemanagement/advantagedatabaseserver/)
- iAnywhere website on Sybase.com (http://www.sybase.com/iAnyhwhere)
- Chris Franz's Advantage blog (http://blog.advantageevangelist.com/)
- Cary Jensen's website (http://www.jensendatasystems.com/)
- Advantage Developer Zone (http://devzone.advantagedatabase.com/)

Ingres (database)

Ingres

Ingres Corporation logo from 2007	
Original author(s)	University of California, Berkeley
Developer(s)	Actian Corporation
Stable release	Ingres Database 10 / October 12, 2010
Written in	C
Operating system	Cross-platform
Type	RDBMS
License	GNU General Public License or proprietary
Website	www.actian.com [1]

Ingres Database (/ɪŋˈgrɛs/ *ing-**GRESS***) is a commercially supported, open-source SQL relational database management system intended to support large commercial and government applications. Ingres Database is fully open source with a global community of contributors. However, Actian Corporation controls the development of Ingres and makes certified binaries available for download, as well as providing worldwide support.

Ingres began as a research project at the University of California, Berkeley, starting in the early 1970s and ending in 1985. The original code, like that from other projects at Berkeley, was available at minimal cost under a version of the BSD license. Ingres spawned a number of commercial database applications, including Sybase, Microsoft SQL Server, NonStop SQL and a number of others. Postgres (**Post** I**ngres**), a project which started in the mid-1980s, later evolved into PostgreSQL.

Ingres is ACID and is fully transactional (including all DDL statements).

Ingres is part of the Lisog open-source stack initiative.

History

In 1973 when the System R project was getting started at IBM, the research team released a series of papers describing the system they were building. Two scientists at Berkeley, Michael Stonebraker and Eugene Wong, became interested in the concept after reading the papers, and started a relational database research project of their own, named *University INGRES*.

They had already raised money for researching a geographic database system for Berkeley's economics group, which they called **Ingres**, for **IN**teractive **G**raphics **RE**trieval **S**ystem. They decided to use this money to fund their relational project instead, and used this as a seed for a new and much larger project. For further funding, Stonebraker approached the DARPA, the obvious funding source for computing research and development at the time, but both the DARPA and the Office of Naval Research (ONR) turned them down as they were already funding database research elsewhere. Stonebraker then introduced his idea to other agencies, and, with help from his colleagues he eventually obtained modest support from the NSF and three military agencies: the Air Force Office of Scientific Research, the Army Research Office, and the Navy Electronic Systems Command.

Thus funded, Ingres was developed during the mid-1970s by a rotating team of students and staff. Ingres went through an evolution similar to that of System R, with an early prototype in 1974 followed by major revisions to make the code maintainable. Ingres was then disseminated to a small user community, and project members rewrote the prototype repeatedly to incorporate accumulated experience, feedback from users, and new ideas. The research

projected ended in 1985. Ingres remained largely similar to IBM's System R in concept, but it was based on "low-end" systems, namely Unix on DEC machines.

Commercialization

Unlike System R, the Ingres source code was available (on tape) for a modest fee. By 1980 some 1,000 copies had been distributed, primarily to universities. Many students from U.C. Berkeley and other universities who used the Ingres source code, worked on various commercial database software systems.

Berkeley students Jerry Held and later Karel Youseffi moved to Tandem Computers, where they built a system that evolved into NonStop SQL. The Tandem database system was a re-implementation of the Ingres technology. It evolved into a system that ran effectively on parallel computers; that is, it included functionality for distributed data, distributed execution, and distributed transactions (the last being fairly difficult). Components of the system were first released in the late 1970s. By 1989, the system could run queries in parallel and the product became fairly famous for being one of the few systems that scales almost linearly with the number of processors in the machine: adding a second CPU to an existing NonStop SQL server will almost exactly double its performance. Tandem was later purchased by Compaq, which started a re-write in 2000, and now the product is at Hewlett-Packard.

In the early 1980s, Ingres competed head-to-head with Oracle. The two products were widely regarded as the leading hardware-independent relational database implementations; they had comparable functionality, performance, market share, and pricing, and many commentators considered Ingres to be a (perhaps marginally) superior product. From around 1985, however, Ingres steadily lost market share. One reason was Oracle's aggressive marketing; another was the increasing recognition of SQL as the preferred relational query language. Ingres originally had provided a different language, Quel, and the conversion to SQL (delivered in Ingres version 6) took about three years, losing valuable time in the race.

Robert Epstein, the chief programmer on the project while he was at Berkeley, formed Britton Lee, Inc. along with other students from the Ingres Project, Paula Hawthorn and Michael Ubell; they were joined later by Eric Allman. Later, Epstein founded Sybase. Sybase had been the #2 product (behind Oracle) for some time through the 1980s and into the 1990s, before Informix came "out of nowhere" and took over in 1997. Sybase's product line had also been licensed to Microsoft in 1992, who rebranded it as Microsoft SQL Server. This relationship soured in the late 1990s, and today SQL Server outsells Sybase by a wide margin.

Several companies used the Ingres source code to produce products. The most successful was a company named Relational Technology, Inc. (RTI), founded in 1980 by Stonebraker and Wong, and another Berkeley professor, Lawrence A. Rowe. RTI was renamed Ingres Corporation in the late 1980s. The company converted the code to DEC VAX/VMS, which was the commercial operating system for DEC VAX computers. They also developed a collection of front-end tools for creating and manipulating databases (e.g., reporterwriters, forms entry and update, etc.) and application development tools. Over time, much of the source was rewritten to add functionality (for example, multiple-statement transactions, SQL, B-tree access method, date/time datatypes, etc.) and improve performance (for example, compiled queries, multithreaded server). The company was purchased by ASK Corporation in November 1990. The founders left the company over the next several months. In 1994, ASK/Ingres was purchased by Computer Associates, who continued to offer Ingres under a variety of brand names (for example, OpenIngres, Ingres II, or Advantage Ingres).

In 2004, Computer Associates released Ingres r3 under an open source license. The code includes the DBMS server and utilities and the character-based front-end and application-development tools. In essence, the code has everything except OpenROAD, the Windows 4GL GUI-based development environment. In November 2005, Garnett & Helfrich Capital, in partnership with Computer Associates, created a new company called Ingres Corporation, which provided support and services for Ingres, OpenROAD, and the connectivity products.

Recent years

In February 2006, Ingres Corporation released Ingres 2006 under the GNU General Public Licence. Ingres 9.3 was released on October 7, 2009. It was a limited release targeted at new application development on Linux and Windows only.[2]

Ingres 10 was released on October 12, 2010, as a full release, supporting upgrade from earlier versions of the product. It was available on 32-bit and 64-bit Linux, and 32-bit Microsoft Windows.

Open-source community initiatives with Ingres included:

Community Bundles – Alliances with other open-source providers and projects, such as Alfresco, JasperSoft, Hibernate, Apache Tomcat, and Eclipse, enable Ingres to provide its platform and technology with other open-source technologies.

Established by Ingres and Carleton University, a series of Open Source Boot Camps were held in 2008 to work with other open-source communities and projects to introduce university and college students and staff to the concepts and realities of open source.

Other involvement includes: Global Ingres University Alliances, Ingres Engineering Summit, Ingres Janitors Project and several memberships in open-source initiatives.

Ingres Icebreaker is an appliance that combines the Ingres Database with the Linux operating system, enabling people to simultaneously deploy and manage a database and operating system.

Ingres CAFÉ (Consolidated Application Foundation for Eclipse), created by a team of developers at Carleton University, is an integrated environment that helps software architects accelerate and simplify Java application development.[3]

Ingres Geospatial was community-based project to create industry-standards-compliant geospatial storage features in the Ingres DBMS. In other words, for storing map data and providing powerful analysis functions within the DBMS.[4]

In November 2010 Garnett & Helfrich Capital acquired the last 20% of equity in Ingres Corp that it did not already own. On September 22, 2011, Ingres Corporation became Actian Corporation. It focused on Action Apps, which use Ingres or Vectorwise RDBMS systems.

Postgres

The Postgres project was started to address limitations of existing database-management implementations of the relational model. Primary among these was their inability to let the user define new domains (or "types") which are combinations of simpler domains (see relational model for an explanation of the term "domain"). The project explored other ideas including the incorporation of write-once media (e.g., optical disks), the use of massive storage (e.g., never delete data), inferencing, and object-oriented data models. The implementation also experimented with new interfaces between the database and application programs (e.g., "portals", which are sometimes referred to as "fat cursors").

The resulting project, named "Postgres", aimed at introducing the minimum number of features needed to add complete types support. These included the ability to define types, but also the ability to fully describe relationships – which up until this time had been widely used but maintained entirely by the user. In Postgres, the database "understood" relationships, and could retrieve information in related tables in a natural way using *rules*.

In the 1990s, Stonebraker started a new company to commercialize Postgres, under the name **Illustra**. The company and technology were later purchased by Informix.

Installation

Ingres may be installed as a *Client Installation* or as a *Server Installation*, the difference being that the Client has no databases associated with it, but allows access to databases created in Server Installations.

A typical site would install Ingres Client Installations on its employees' PCs, and these would communicate with the Ingres Server installations on the site's core computing facility.

Note that the expression "instance" is a synonym for "installation".

An installation can be viewed as a collection of server processes, shared memory and semaphores for interprocess communication, as well as disk-based files used for transaction processing and (in the event of a failure of the host or of the installation) for database recovery.

Installation identifier

An installation is often referred to by its installation identifier. This is a two-character case-sensitive identifier, beginning with a letter. The default identifier is II. The installation identifier is used internally to compute what ports the Ingres servers will listen on. For example "II" indicates that the servers will listen on port 21064 plus the 7 port numbers after that.

Any host (machine or virtual machine) may have multiple Ingres installations on it, but each installation must use unique identifiers to ensure that its clients and components communicate with the correct installation.

A single installation may use multiple installation identifiers. The classic example is when wishing to run more than eight server processes. Furthermore, although Ingres database servers (iidbms) and Ingres communication servers (iigcc) conventionally use the same installation identifier, there is no requirement to do so.

Installation paths

At the point of creating the installation, several critical paths need to be assigned. Once created, these cannot be changed without re-installing, hence care should be taken in their choice.

These paths appear in the following table. Note that the 'II_' prefix does not indicate that these are for the 'II' installation. Each installation, regardless of its identifier, will have its own set of these variables.

Name	Purpose
II_SYSTEM	The installations binaries, utilities, text files used for configuration etc. are kept under this path.
II_DATABASE	The primary data location for the installation.
II_CHECKPOINT	The location used when creating backups of the installations databases.
II_JOURNAL	The transaction journaling location for the installations databases. Journals are used by the recovery system to provide point-of-failure recovery. They may also be used for auditing purposes.
II_DUMP	The location of the installations 'dump' files. These may be generated during a databases 'on-line' backup and are essential for the databases recovery.
II_WORK	Used to hold work files generated by the server when performing queries on the database.

Patching

The installation is created by a privileged user of the host (i.e. username "root"). However, the addition of software patches to the installation is performed by the installation owner (typically the user: ingres).

In Ingres, software patches are cumulative and sequentially numbered. Hence installing patch *N+1* will automatically include all the additions by patch *N*.

To determine the current installation version and patch level, it is simply a matter of inspecting the text file: II_SYSTEM/ingres/version.rel.

The text file "II_SYSTEM/ingres/version.dat" provides extra information on the date of installation.

Note that both files are cumulative, and the top entry is the current version and patch.

Databases

An Ingres installation (or instance) may support many databases, each being owned by any user known to the installation. The installation will allow many databases to be available concurrently. The number available is a configurable quantity. Note this simply restricts the number of databases available at any instant, and many more databases may be created.

On creation of an Ingres Server installation, the databases named "iidbdb" and "imadb" are created. These databases are owned by the user "$ingres". The database iidbdb is also known as the "Master Catalog database", and it contains many tables specific to the management of the installation itself. The database imadb is the Ingres Management Architecture database, and it also contains many registered objects useful for management of the installation.

Of particular note is that databases do not need to be "pre-sized". Each database in the installation is permitted to grow as large as the available disk space will permit.

Multiple data locations

Each database may be created on any data location known to the installation. If no data location is specified, then the primary data location indicated by the installation default of II_DATABASE is assumed. Once created, the database may then be extended to use any (or all) of the other data locations known to the installation.

A database with multiple locations has the advantage of allowing parallel backups, and hence it can potentially reduce the backup time.

Public or private

Databases may be marked as public or private, at the point of creation, or afterwards. A public database is accessible to all known Ingres users in the installation – unless they have been specifically denied access. A private database is accessible only to specified permitted users, groups and roles.

Unicode

A database may be created with a specific Unicode collation. This attribute can also be added after creation. Ingres supports the Unicode collation algorithm; optional Unicode support allows Ingres to minimize its resource requirements.

Distributed databases

Ingres provides a distributed database system via the IngresSTAR server.

A database must be created as distributed by suffixing the database name with the "/star" service class. Once they have been created, the tables, views and procedures from other databases may be registered within the distributed database. The distributed database may also have its own tables, views and procedures.

The IngresNET server allows the source databases to be on any other Ingres installation as well as on the installation which holds the distributed database. The IngresBRIDGE server allows the source databases to be non-Ingres databases as well.

User access to the distributed database is exactly as per regular databases. User grants to the registered tables and views are determined by the database from which they are registered.

Queries may then be run across the tables as per normal, although there are some restrictions on query types. Furthermore a user transparent two-phase commit is inbuilt to the system.

Database objects

Catalogs

Regardless of ownership, each database is created with a set of tables and views owned by the user "$ingres". These are referred to as *catalogs* and are used to control many aspects of the databases interaction with the world.

The Master Catalog Database "iidbdb" has a specific set of catalogs which will not be loaded into any other database.

Catalogs are publicly readable, but cannot be altered by anyone other than a privileged user.

Tables

The database owner and permitted users are allowed to create tables as they wish, within the database and may share access to these as they wish. Note that regardless of the database access mode (public/private), a table is private until the owner of the table grants other users some access to it.

Tables are not "pre-sized" at the point of creation. Ingres makes no restrictions and will allow any table to grow as far as disk space permits.

The same table name may be used by multiple table owners. When a distinction needs to be made in the application code, it may specify the full schema name of "table_owner.table_name". If the schema has not been specified, then the system will check to see if the current user has a table of this name, and if not, it will then check if the database owner has a table of this name.

Ingres supports four table types, and has compressed subtypes available for each. These types are: Heap, Hash, ISAM (indexed sequential), B-tree (binary tree). The Heap type is unstructured; all others are structured tables where a "Primary Key" is designated. These table types allow tables to be tailored to suit the needs of queries and considerably improve query performance.

The table type dictates the way in which data is stored within the table, and the tables response to insert, update, delete or select requests. The frequency of such activity dictates the occasional maintenance requirement of restructuring the table to ensure optimum query response.

A table may be located on any of the data locations that the database has been permitted to use. The table may be spread across multiple locations -a feature of particular use for large tables and for parallel backups. Ingres will attempt to spread the data evenly across all locations the table is permitted to use.

A table is composed of pages. The data and the keying details for the table structure are all stored on these pages. Each table is permitted to grow to approx. 8.4 million pages. All the pages for the table are of a fixed size, specified at creation or when last restructured. The six available page sizes are: 2K, 4K, 8K, 16K, 32K, 64K. The installation

must be configured to support the chosen size. Typically an installation defaults to provide 2K, 4K and 8K pages.

Once a table has been created with a specific size, it may be subsequently restructured to a different page size. The correct choice of page size for a table can be beneficial in allowing both increased size in the table and in allowing the possibility of row-level locking (available on page sizes of 4K and above).

Each page may hold a maximum of 512 rows of data. No row may span a page. A certain amount of each page is reserved for system purposes, hence the entire space is not available to data. For example, a 2K page has only 2,008 bytes (of the total 2,048) available for data.

If larger tables are required, the table may be partitioned. Each partition of the table is effectively a separate table, and each may grow to 8.4 million pages. The set of partitions then makes a logical table, completely transparently to the users accessing the table. The partitions may also be partitioned, effectively providing a limitless table size. This feature allows Ingres databases to seamlessly grow from a few megabytes to several terabytes.

Indexes

Each table may have zero, one or more indexes created upon it. An index may be of any structured type i.e. HASH, ISAM or B-TREE. The addition of a secondary index on the table can give improved access to the table data for specific queries.

Indexes may be queried directly. In most respects, they behave just like tables. An index may be created with a different page size to its base table.

Both primary key and secondary-index keys may be designated as unique or non-unique.

Temporary tables

Ingres supports the creation of "lightweight" or temporary tables, which exist purely for the lifetime of the connected session which creates them. These tables can be structured as per regular tables, but may not be shared. The temporary table exists within the server, until it grows too large, at which point its details will be transparently written to a disk. If this occurs, the details will be removed as soon as the session disconnects.

They are useful in holding temporary data for reports and for simplifying complicated queries.

Views

A view is a logical object with no physical disk presence other than its definition. A view is like a predefined select query on one or more tables or other views. A view may be treated like any table, but cannot have an index or structure imposed upon it.

Constraints

Ingres supports the following table constraints, as well as propagation constraint and ON UPDATE CASCADE on foreign keys.

- Check Constraint, where a column value is mandated to be a specific value or within a range of values on the basis of a simple calculation.
- Unique Constraint, where a column value will have uniqueness enforced.
- Foreign Key Constraint, where a column value must exist in another table.
- Primary Key Constraint, where nominated columns within the table are grouped into a unique primary key. This is an adjunct to the normal Ingres Primary Key which may be defined upon the table.

Most of these constraints require a secondary index to perform their function. If such an index is not nominated, then Ingres will automatically create an appropriate index on the table.

Constraints may be created when the table is created, or added afterwards.

Database procedures

A Database Procedure (DBP) is a named routine consisting of SQL and procedural statements that is stored in the database, close to the data. When a DBP is created, Ingres optimizes and compiles the procedure and caches the generated code. The database procedure can then be invoked directly from a client application program or from another database procedure, or it can be triggered by a rule (see below).

Most of the usual SQL statements are available, supplemented by procedural code features, such as variable creation and assignment, flow of control and event and error-control statements.

Some advantages of Database Procedures

- **Performance:** The DBP code typically only needs to be compiled once by Ingres and a DBP can reduce the data traffic between an application and Ingres by performing calculations in situ.
- **Integrity:** Similarly to "getter" and "setter" methods, DBPs can be used to control access to the tables. Users can also program Ingres to execute a DBP when a table undergoes a specific change.
- **Security:** DBPs can be used to restrict the operations available to the tables. The SQL GRANT statement can be used to provide execute permission for a DBP accessing the tables, even though the tables provide no access permission.
- **Control**: DBPs can be used as a central place to maintain the data manipulation logic used by a whole variety of applications. In this way the DBP logic can be updated without needing to change each application (the DBP signature/contract must be maintained), and it is even possible to update a DBP on a live system (if careful).
- **Portability**: An Ingres DBP will work unchanged in any Ingres DBMS regardless of the underlying operating system or platform.

Database rules

Database Rules may be created on tables, also called "triggers". Rules are typically used to enforce integrity checks which would be too complicated for simple constraints. However, they may be employed to perform other tasks, such as raise events, etc.

Rules are triggered *before* or *after* nominated action(s) on the associated table. Note that older versions of Ingres allowed only *after* rules to be defined. There is no restriction on the number of rules a table may have. If an action causes multiple rules to fire, then the order of firing is undefined.

The rules will cause an associated database procedure to be executed. That procedure is referred to as a Rules Fired Procedure or RFP. In most respects RFPs and DEPs are similar in capability, however there are some restrictions on the RFPs:

- They must not return a value or rows.
- They cannot take a temporary table as a parameter.
- They may not issue either the *commit* or *rollback* statement.
- If the rule is triggered by a *before* action then it may not directly perform insert, update or delete activity on the database. It is believed that this restriction may be removed in a future release. In the mean time a *work-around* is for the RFP to call another procedure to perform any required insert, update or delete activity.

There are also differences in the effects of errors being raised by RFPs and DEPs. In an RFP, raising an error will cause the procedure to stop, all statements executed by the procedure will be rolled back and the statement which caused the rule to fire will also be rolled back.

Parameters to an RFP may be passed by value or reference. For example *before* fired rules may use a parameter passed by reference to install a desired value in a column of the row of data which initially caused the rule to be fired.

Data types

Ingres supports the conventional data types such as:

- integers (1 byte, 2 byte, 4 byte and 8 byte)
- floats (4 byte, 8 byte)
- fixed precision numbers (numeric/decimal)
- characters (fixed and variable length)
- binary (fixed and variable length)
- dates and times (ANSI date, time, and timestamp)

Ingres supports user defined types though the Object Management Extension

- You can use a user-defined data type in any context in which you can use a standard Ingres data type
- You can use user-defined SQL functions in queries to manipulate both user-defined data types
- To support new data types and functions, you can add new capabilities to existing SQL comparison and arithmetic operators.

Ingres supports Unicode with types:

- nchar
- nvarchar

Ingres supports large objects with:

- long varchar
- long byte

Ingres supports proprietary types such as:

- ingres date
- money

Ingres supports geospatial data types (version 10S and later):

- point, multipoint
- linestring, multilinestring
- polygon, multipolygon
- geometry, geometrycollection

Backup and recovery

Journaling

Ingres is a fully transactional DBMS. These transactions may be recorded as **journals** associated with the database under the II_JOURNAL path. The journals created by the DBMS may then be examined as part of auditing activity or used in a database recovery.

To enable journaling on a database is a two step process. Namely:

- Enable journaling on the database as a whole. This is done by specifying the "+j" flag to the **ckpdb** command.
- Nominate tables within the database for journaling. Note that indexes do not need to be nominated, views cannot be journaled.

Note that the configuration of the Ingres DBMS allows for the new tables to be automatically journal-enabled via the *default journaling* parameter. Some care should be taken with this facility as not all tables should be journaled. For example, a work table which is constantly emptied and refreshed should not be journaled as it places extra data in the journal system, data which are generally irrelevant to auditing and not required for database recovery.

A table's journaling status may be easily altered. However if journaling is enabled on the table the journaling will not commence until after the next occasion the database is backed up using the **ckpdb** command. If the table is created journal-enabled, the journaling will commence immediately. If journaling is disabled the effect is also immediate.

To examine the journals for a database in a human readable form is simply a matter of using the **auditdb** command. The command is option-rich and has many features for auditing transactions committed within a given time frame, by specified users on a nominated set of tables, etc. The command has other options which can be used as part of an *audit trail recovery* for a database.

Backup

The principal backup utility provided in the dbms is the command: **ckpdb**

Ingres backups may be taken *On-line* where some user activity is permitted on the database, or *Off-line* where no user activity is permitted on the database. During On-line checkpoints users may still select, insert, update or delete from the database but are not permitted to drop tables, modify existing structures or other DDL statements.

Backups will capture the entire database by default, however the ckpdb command may be directed to restrict itself to specific tables.

This ckpdb utility would typically create a tar file snapshot of the database. These snapshots are referred to as *checkpoints*. The files created are stored in the databases II_CHECKPOINT location. To allow for changes being made to the databases tables during the lifetime of the backup, the system will also create **dump** files. These reflect the changes being made and are used to ensure the database will be restored to a consistent state as at the start of the checkpoint if a database recovery is required. The dump files created are stored in the databases II_DUMP area.

A databases backup history may be examined using the **infodb** command. It will print a human readable summary of the databases backup history ... as well as other datum. For example *infodb iidbdb* would generate the backup history of the master catalog database iidbdb.

Other archiving tools may be used. More recent versions of ingres also provide a *cpio*-based version. Furthermore, some customisation of the backup is achievable by editing the *Checkpoint template file*. A typical user customisation is to direct tar to use compression.

Multiple template files may be created and a specific one selected by pointing to it with the environment variable II_CKTMPL_FILE. The default template file is: II_SYSTEM/ingres/files/cktmpl.def.

The alternatives to ckpdb are the utilities: **copydb** or **unloaddb**. These provide static snapshots only. If these utilities are used, care should be taken to ensure the correct representation of floating point numbers and dates.

It is not a good idea to back up an Ingres database with an OS dump of the database's data areas.

Recovery

The principal means of recovering an Ingres database from a checkpoint is the utility: "rollforwarddb".

By default rollforwarddb will restore the database from its most recent valid backup and then apply all the databases journals and thus restore the database as completely as possible. Furthermore, the command is option rich, and it may be directed to:

- use an older checkpoint,
- not apply journals,
- apply journals up to a specified end time. The time ensuring all transaction **committed** at or before this time are restored.
- apply journals from a specified begin time. The time ensuring that all transactions **committed** on or after this time are restored. This is a rarely employed option.

Note that for time-based recoveries the critical feature is when the transaction was **committed**, not when it was started. If it becomes necessary to see what transactions will be included in the recovery the time parameters can be used in the **auditdb** utility. That utility will display the included transactions and their details.

Customisation of the rollforwarddb utility may also be performed by editing the *checkpoint template file*. For example, if the checkpoints were compressed the rollforwarddb command will need a customisation installed to allow it to process the compressed tar file.

References

[1] http://www.actian.com/
[2] Ingres Database 9.3 (http://esd.ingres.com/product/Ingres_Database/9.3)
[3] Ingres CAFÉ (http://community.ingres.com/wiki/CAFÉ)
[4] Ingres Geospatial (http://community.ingres.com/wiki/IngresGeospatial)

External links

- The Design and Implementation of INGRES (http://www.cs.wisc.edu/~nil/764/root/3_p189-stonebraker.pdf)
- Retrospection on a Database System (http://portal.acm.org/citation.cfm?doid=320141.320158)
- Ingres FAQ (http://www.bizyx.com/ingres/faq.htm) (from 1997)
- Actian Corp. (http://www.actian.com/)
- University INGRES,Version 8.9 (http://s2k-ftp.cs.berkeley.edu/ingres/)

Normal Form

Data redundancy

Data redundancy occurs in database systems which have a field that is repeated in two or more tables. For instance, when customer data is duplicated and attached with each product bought then redundancy of data is a known source of inconsistency, since customer might appear with different values for given attribute. Data redundancy leads to data anomalies and corruption and generally should be avoided by design. Database normalization prevents redundancy and makes the best possible usage of storage. Proper use of foreign keys can minimize data redundancy and chance of destructive anomalies. However, concerns of efficiency and convenience can sometimes result in redundant data design despite the risk of corrupting the data.

Notes and references

Database normalization

Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships.

Edgar F. Codd, the inventor of the relational model, introduced the concept of normalization and what we now know as the First Normal Form (1NF) in 1970. Codd went on to define the Second Normal Form (2NF) and Third Normal Form (3NF) in 1971,[1] and Codd and Raymond F. Boyce defined the Boyce-Codd Normal Form (BCNF) in 1974.[2] Informally, a relational database table is often described as "normalized" if it is in the Third Normal Form.[3] Most 3NF tables are free of insertion, update, and deletion anomalies.

A standard piece of database design guidance is that the designer should first create a fully normalized design; then selective denormalization can be performed for performance reasons.[4]

Objectives

A basic objective of the first normal form defined by Edgar Frank "Ted" Codd in 1970 was to permit data to be queried and manipulated using a "universal data sub-language" grounded in first-order logic.[5] (SQL is an example of such a data sub-language, albeit one that Codd regarded as seriously flawed.)[6]

The objectives of normalization beyond 1NF (First Normal Form) were stated as follows by Codd:

1. To free the collection of relations from undesirable insertion, update and deletion dependencies;

2. To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs;

3. To make the relational model more informative to users;

4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

— E.F. Codd, *"Further Normalization of the Data Base Relational Model"*[7]

The sections below give details of each of these objectives.

Free the database of modification anomalies

When an attempt is made to modify (update, insert into, or delete from) a table, undesired side-effects may follow. Not all tables can suffer from these side-effects; rather, the side-effects can only arise in tables that have not been sufficiently normalized. An insufficiently normalized table might have one or more of the following characteristics:

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

An **update anomaly**. Employee 519 is shown as having different addresses on different records.

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

An **insertion anomaly**. Until the new faculty member, Dr. Newsome, is assigned to teach at least one course, his details cannot be recorded.

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

DELETE

A **deletion anomaly**. All information about Dr. Giddens is lost if he temporarily ceases to be assigned to any courses.

- The same information can be expressed on multiple rows; therefore updates to the table may result in logical inconsistencies. For example, each record in an "Employees' Skills" table might contain an Employee ID, Employee Address, and Skill; thus a change of address for a particular employee will potentially need to be applied to multiple records (one for each skill). If the update is not carried through successfully—if, that is, the employee's address is updated on some records but not others—then the table is left in an inconsistent state. Specifically, the table provides conflicting answers to the question of what this particular employee's address is. This phenomenon is known as an **update anomaly**.
- There are circumstances in which certain facts cannot be recorded at all. For example, each record in a "Faculty and Their Courses" table might contain a Faculty ID, Faculty Name, Faculty Hire Date, and Course Code—thus we can record the details of any faculty member who teaches at least one course, but we cannot record the details of a newly hired faculty member who has not yet been assigned to teach any courses except by setting the Course Code to null. This phenomenon is known as an **insertion anomaly**.
- Under certain circumstances, deletion of data representing certain facts necessitates deletion of data representing completely different facts. The "Faculty and Their Courses" table described in the previous example suffers from this type of anomaly, for if a faculty member temporarily ceases to be assigned to any courses, we must delete the last of the records on which that faculty member appears, effectively also deleting the faculty member. This phenomenon is known as a **deletion anomaly**.

Minimize redesign when extending the database structure

When a fully normalized database structure is extended to allow it to accommodate new types of data, the pre-existing aspects of the database structure can remain largely or entirely unchanged. As a result, applications interacting with the database are minimally affected.

Make the data model more informative to users

Normalized tables, and the relationship between one normalized table and another, mirror real-world concepts and their interrelationships.

Avoid bias towards any particular pattern of querying

Normalized tables are suitable for general-purpose querying. This means any queries against these tables, including future queries whose details cannot be anticipated, are supported. In contrast, tables that are not normalized lend themselves to some types of queries, but not others.

For example, consider an online bookseller whose customers maintain wishlists of books they'd like to have. For the obvious, anticipated query—what books does this customer want?—it's enough to store the customer's wishlist in the table as, say, a homogeneous string of authors and titles.

With this design, though, the database can answer only that one single query. It cannot by itself answer interesting but unanticipated queries: What is the most-wished-for book? Which customers are interested in WWII espionage? How does Lord Byron stack up against his contemporary poets? Answers to these questions must come from special adaptive tools completely separate from the database. One tool might be software written especially to handle such queries. This special adaptive software has just one single purpose: in effect to normalize the non-normalized field.

Unforeseen queries can be answered trivially, and entirely within the database framework, with a normalized table.

Example

Querying and manipulating the data within a data structure which is not normalized, such as the following non-1NF representation of customers' credit card transactions, involves more complexity than is really necessary:

Customer Jones Wilkinson Stevens Transactions

Tr. ID	Date	Amount
12890	14-Oct-2003	−87
12904	15-Oct-2003	−50

Tr. ID	Date	Amount
12898	14-Oct-2003	−21

Tr. ID	Date	Amount
12907	15-Oct-2003	−18
14920	20-Nov-2003	−70
15003	27-Nov-2003	−60

To each customer corresponds a *repeating group* of transactions. The automated evaluation of any query relating to customers' transactions therefore would broadly involve two stages:

1. Unpacking one or more customers' groups of transactions allowing the individual transactions in a group to be examined, and
2. Deriving a query result based on the results of the first stage

For example, in order to find out the monetary sum of all transactions that occurred in October 2003 for all customers, the system would have to know that it must first unpack the *Transactions* group of each customer, then sum the *Amounts* of all transactions thus obtained where the *Date* of the transaction falls in October 2003.

One of Codd's important insights was that this structural complexity could always be removed completely, leading to much greater power and flexibility in the way queries could be formulated (by users and applications) and evaluated (by the DBMS). The normalized equivalent of the structure above would look like this:

Customer	Tr. ID	Date	Amount
Jones	12890	14-Oct-2003	−87
Jones	12904	15-Oct-2003	−50
Wilkins	12898	14-Oct-2003	−21
Stevens	12907	15-Oct-2003	−18
Stevens	14920	20-Nov-2003	−70
Stevens	15003	27-Nov-2003	−60

Now each row represents an individual credit card transaction, and the DBMS can obtain the answer of interest, simply by finding all rows with a Date falling in October, and summing their Amounts. The data structure places all of the values on an equal footing, exposing each to the DBMS directly, so each can potentially participate directly in queries; whereas in the previous situation some values were embedded in lower-level structures that had to be handled specially. Accordingly, the normalized design lends itself to general-purpose query processing, whereas the unnormalized design does not.

Background to normalization: definitions

Functional dependency

> In a given table, an attribute *Y* is said to have a functional dependency on a set of attributes *X* (written $X \rightarrow Y$) if and only if each *X* value is associated with precisely one *Y* value. For example, in an "Employee" table that includes the attributes "Employee ID" and "Employee Date of Birth", the functional dependency {Employee ID} $\rightarrow$ {Employee Date of Birth} would hold. It follows from the previous two sentences that each {Employee ID} is associated with precisely one {Employee Date of Birth}.

Full functional dependency

> An attribute is fully functionally dependent on a set of attributes X if it is:

- functionally dependent on X, and
- not functionally dependent on any proper subset of X. {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a *full* functional dependency, because it is also dependent on {Employee

ID}.Even by the removal of {Skill} functional dependency still holds between {Employee Address} and {Employee ID}.

Transitive dependency

A transitive dependency is an indirect functional dependency, one in which $X \to Z$ only by virtue of $X \to Y$ and $Y \to Z$.

Trivial functional dependency

A trivial functional dependency is a functional dependency of an attribute on a superset of itself. {Employee ID, Employee Address} → {Employee Address} is trivial, as is {Employee Address} → {Employee Address}.

Multivalued dependency

A multivalued dependency is a constraint according to which the presence of certain rows in a table implies the presence of certain other rows.

Join dependency

A table *T* is subject to a join dependency if *T* can always be recreated by joining multiple tables each having a subset of the attributes of *T*.

Superkey

A superkey is a combination of attributes that can be used to uniquely identify a database record. A table might have many superkeys.

Candidate key

A candidate key is a special subset of superkeys that do not have any extraneous information in them: it is a minimal superkey.

Example:

A table with the fields <Name>, <Age>, <SSN> and <Phone Extension> has many possible superkeys. Three of these are <SSN>, <Phone Extension, Name> and <SSN, Name>. Of those, only <SSN> is a candidate key as the others contain information not necessary to uniquely identify records ('SSN' here refers to Social Security Number, which is unique to each person).

Non-prime attribute

A non-prime attribute is an attribute that does not occur in any candidate key. Employee Address would be a non-prime attribute in the "Employees' Skills" table.

Prime attribute

A prime attribute, conversely, is an attribute that does occur in some candidate key.

Primary key

One candidate key in a relation may be designated the primary key. While that may be a common practice (or even a required one in some environments), it is strictly notational and has no bearing on normalization. With respect to normalization, all candidate keys have equal standing and are treated the same.

Normal forms

The **normal forms** (abbrev. **NF**) of relational database theory provide criteria for determining a table's degree of immunity against logical inconsistencies and anomalies. The higher the normal form applicable to a table, the less vulnerable it is. Each table has a "**highest normal form**" (**HNF**): by definition, a table always meets the requirements of its HNF and of all normal forms lower than its HNF; also by definition, a table fails to meet the requirements of any normal form higher than its HNF.

The normal forms are applicable to individual tables; to say that an entire database is in normal form *n* is to say that all of its tables are in normal form *n*.

Newcomers to database design sometimes suppose that normalization proceeds in an iterative fashion, i.e. a 1NF design is first normalized to 2NF, then to 3NF, and so on. This is not an accurate description of how normalization typically works. A sensibly designed table is likely to be in 3NF on the first attempt; furthermore, if it is 3NF, it is overwhelmingly likely to have an HNF of 5NF. Achieving the "higher" normal forms (above 3NF) does not usually require an extra expenditure of effort on the part of the designer, because 3NF tables usually need no modification to meet the requirements of these higher normal forms.

The main normal forms are summarized below.

	Normal form	Defined by	In	Brief definition
1NF	First normal form	Two versions: E.F. Codd (1970), C.J. Date (2003)	1970 and 2003[8]	A relation is in first normal form if the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.
2NF	Second normal form	E.F. Codd	1971	No non-prime attribute in the table is functionally dependent on a proper subset of any candidate key
3NF	Third normal form	Two versions: E.F. Codd (1971), C. Zaniolo (1982)	1971 and 1982[9]	Every non-prime attribute is non-transitively dependent on every candidate key in the table. The attributes that do not contribute to the description of the primary key are removed from the table. In other words, no transitive dependency is allowed.
EKNF	Elementary Key Normal Form	C. Zaniolo	1982	Every non-trivial functional dependency in the table is either the dependency of an elementary key attribute or a dependency on a superkey
BCNF	Boyce–Codd normal form	Raymond F. Boyce and E.F. Codd	1974[10]	Every non-trivial functional dependency in the table is a dependency on a superkey
4NF	Fourth normal form	Ronald Fagin	1977	Every non-trivial multivalued dependency in the table is a dependency on a superkey
5NF	Fifth normal form	Ronald Fagin	1979[11]	Every non-trivial join dependency in the table is implied by the superkeys of the table
DKNF	Domain/key normal form	Ronald Fagin	1981[12]	Every constraint on the table is a logical consequence of the table's domain constraints and key constraints
6NF	Sixth normal form	C.J. Date, Hugh Darwen, and Nikos Lorentzos	2002[13]	Table features no non-trivial join dependencies at all (with reference to generalized join operator)

Denormalization

Databases intended for online transaction processing (OLTP) are typically more normalized than databases intended for online analytical processing (OLAP). OLTP applications are characterized by a high volume of small transactions such as updating a sales record at a supermarket checkout counter. The expectation is that each transaction will leave the database in a consistent state. By contrast, databases intended for OLAP operations are primarily "read mostly" databases. OLAP applications tend to extract historical data that has accumulated over a long period of time. For such databases, redundant or "denormalized" data may facilitate business intelligence applications. Specifically, dimensional tables in a star schema often contain denormalized data. The denormalized or redundant data must be carefully controlled during extract, transform, load (ETL) processing, and users should not be permitted to see the data until it is in a consistent state. The normalized alternative to the star schema is the snowflake schema. In many cases, the need for denormalization has waned as computers and RDBMS software have become more powerful, but since data volumes have generally increased along with hardware and software performance, OLAP databases often still use denormalized schemas.

Denormalization is also used to improve performance on smaller computers as in computerized cash-registers and mobile devices, since these may use the data for look-up only (e.g. price lookups). Denormalization may also be used when no RDBMS exists for a platform (such as Palm), or no changes are to be made to the data and a swift response is crucial.

Non-first normal form (NF² or N1NF)

Denormalization is the opposite of normalization. In recognition that denormalization can be deliberate and useful, the non-first normal form is a definition of database designs which do not conform to first normal form, by allowing "sets and sets of sets to be attribute domains" (Schek 1982). The languages used to query and manipulate data in the model must be extended accordingly to support such values.

One way of looking at this is to consider such structured values as being specialized types of values (domains), with their own domain-specific languages. However, what is usually meant by non-1NF models is the approach in which the relational model and the languages used to query it are extended with a general mechanism for such structure; for instance, the nested relational model supports the use of relations as domain values, by adding two additional operators (*nest* and *unnest*) to the relational algebra that can create and flatten nested relations, respectively.

Consider the following table:

First Normal Form

Person	Favourite Colour
Bob	blue
Bob	red
Jane	green
Jane	yellow
Jane	red

Assume a person has several favourite colours. Obviously, favourite colours consist of a set of colours modeled by the given table. To transform a 1NF into an NF² table a "nest" operator is required which extends the relational algebra of the higher normal forms. Applying the "nest" operator to the 1NF table yields the following NF² table:

Non-First Normal Form

Person	Favourite Colours
Bob	**Favourite Colour** blue red
Jane	**Favourite Colour** green yellow red

To transform this NF² table back into a 1NF an "unnest" operator is required which extends the relational algebra of the higher normal forms. The unnest, in this case, would make "colours" into its own table.

Although "unnest" is the mathematical inverse to "nest", the operator "nest" is not always the mathematical inverse of "unnest". Another constraint required is for the operators to be bijective, which is covered by the Partitioned Normal Form (PNF).

Notes and references

[1] Codd, E.F. "Further Normalization of the Data Base Relational Model". (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems", New York City, May 24–25, 1971.) IBM Research Report RJ909 (August 31, 1971). Republished in Randall J. Rustin (ed.), *Data Base Systems: Courant Computer Science Symposia Series 6*. Prentice-Hall, 1972.

[2] Codd, E. F. "Recent Investigations into Relational Data Base Systems". IBM Research Report RJ1385 (April 23, 1974). Republished in *Proc. 1974 Congress* (Stockholm, Sweden, 1974). , N.Y.: North-Holland (1974).

[3] C.J. Date. *An Introduction to Database Systems*. Addison-Wesley (1999), p. 290

[4] Chris Date, for example, writes: "I believe firmly that anything less than a fully normalized design is *strongly contraindicated* ... [Y]ou should *"denormalize" only as a last resort*. That is, you should back off from a fully normalized design only if all other strategies for improving performance have somehow failed to meet requirements." Date, C.J. *Database in Depth: Relational Theory for Practitioners*. O'Reilly (2005), p. 152.

[5] "The adoption of a relational model of data ... permits the development of a universal data sub-language based on an applied predicate calculus. A first-order predicate calculus suffices if the collection of relations is in first normal form. Such a language would provide a yardstick of linguistic power for all other proposed data languages, and would itself be a strong candidate for embedding (with appropriate syntactic modification) in a variety of host Ianguages (programming, command- or problem-oriented)." Codd, "A Relational Model of Data for Large Shared Data Banks" (http://www.acm.org/classics/nov95/toc.html), p. 381

[6] Codd, E.F. Chapter 23, "Serious Flaws in SQL", in *The Relational Model for Database Management: Version 2*. Addison-Wesley (1990), pp. 371–389

[7] Codd, E.F. "Further Normalization of the Data Base Relational Model", p. 34

[8] Date, C. J. "What First Normal Form Really Means" in *Date on Database: Writings 2000–2006* (Springer-Verlag, 2006), pp. 127–128.

[9] Zaniolo, Carlo. "A New Normal Form for the Design of Relational Database Schemata." *ACM Transactions on Database Systems* 7(3), September 1982.

[10] Codd, E. F. "Recent Investigations into Relational Data Base Systems". IBM Research Report RJ1385 (April 23, 1974). Republished in *Proc. 1974 Congress* (Stockholm, Sweden, 1974). New York, N.Y.: North-Holland (1974).

[11] Ronald Fagin. "Normal Forms and Relational Database Operators". ACM SIGMOD International Conference on Management of Data, May 31-June 1, 1979, Boston, Mass. Also IBM Research Report RJ2471, Feb. 1979.

[12] Ronald Fagin (1981) *A Normal Form for Relational Databases That Is Based on Domains and Keys* (http://www.almaden.ibm.com/cs/people/fagin/tods81.pdf), *Communications of the ACM*, vol. 6, pp. 387–415

[13] C.J. Date, Hugh Darwen, Nikos Lorentzos. *Temporal Data and the Relational Model*. Morgan Kaufmann (2002), p. 176

- Paper: "Non First Normal Form Relations" by G. Jaeschke, H. -J Schek ; IBM Heidelberg Scientific Center. -> Paper studying normalization and denormalization operators nest and unnest as mildly described at the end of this wiki page.

Further reading

- Litt's Tips: Normalization (http://www.troubleshooters.com/littstip/ltnorm.html)
- Date, C. J. (1999), *An Introduction to Database Systems* (http://www.aw-bc.com/catalog/academic/product/0,1144,0321197844,00.html) (8th ed.). Addison-Wesley Longman. ISBN 0-321-19784-4.
- Kent, W. (1983) *A Simple Guide to Five Normal Forms in Relational Database Theory* (http://www.bkent.net/Doc/simple5.htm), Communications of the ACM, vol. 26, pp. 120–125
- H.-J. Schek, P. Pistor Data Structures for an Integrated Data Base Management and Information Retrieval System

External links

- Database Normalization Basics (http://databases.about.com/od/specificproducts/a/normalization.htm) by Mike Chapple (About.com)
- Database Normalization Intro (http://www.databasejournal.com/sqletc/article.php/1428511), Part 2 (http://www.databasejournal.com/sqletc/article.php/26861_1474411_1)
- An Introduction to Database Normalization (http://mikehillyer.com/articles/an-introduction-to-database-normalization/) by Mike Hillyer.
- A tutorial on the first 3 normal forms (http://phlonx.com/resources/nf3/) by Fred Coulson
- DB Normalization Examples (http://www.dbnormalization.com/)
- Description of the database normalization basics (http://support.microsoft.com/kb/283878) by Microsoft
- Database Normalization and Design Techniques (http://www.barrywise.com/2008/01/database-normalization-and-design-techniques/) by Barry Wise, recommended reading for the Harvard MIS.
- A Simple Guide to Five Normal Forms in Relational Database Theory (http://www.bkent.net/Doc/simple5.htm)

Functional dependency

In relational database theory, a **functional dependency** is a **constraint** between two sets of attributes in a relation from a database.

Given a relation *R*, a set of attributes *X* in *R* is said to **functionally determine** another set of attributes *Y*, also in *R*, (written $X \rightarrow Y$) if, and only if, each *X* value is associated with precisely one *Y* value; *R* is then said to *satisfy* the functional dependency $X \rightarrow Y$. Equivalently, the projection $\pi_{X,Y}R$ is a function, i.e. *Y* is a function of *X*. In simple words, if the values for the *X* attributes are known (say they are *x*), then the values for the *Y* attributes corresponding to *x* can be determined by looking them up in *any* tuple of *R* containing *x*. Customarily *X* is called the *determinant* set and *Y* the *dependent* set. A functional dependency FD: $X \rightarrow Y$ is called *trivial* if *Y* is a subset of *X*.

The determination of functional dependencies is an important part of designing databases in the relational model, and in database normalization and denormalization. A simple application of functional dependencies is **Heath's theorem**; it says that a relation *R* over an attribute set *U* and satisfying a functional dependency $X \rightarrow Y$ can be safely split in two relations having the lossless-join decomposition property, namely into $\pi_{XY}(R) \bowtie \pi_{XZ}(R) = R$ where $Z = U - XY$ are the rest of the attributes. (Unions of attribute sets are customarily denoted by mere juxtapositions in database theory.) An important notion in this context is a candidate key, defined as a minimal set of attributes that functionally determine all of the attributes in a relation. The functional dependencies, along with the attribute domains, are selected so as to generate constraints that would exclude as much data inappropriate to the user domain from the system as possible.

A notion of logical implication is defined for functional dependencies in the following way: a set of functional dependencies $\sum$ logically implies another set of dependencies $\prod$, if any relation *R* satisfying all dependencies from $\sum$ also satisfies all dependencies from $\prod$; this is usually written $\Sigma \models \Gamma$. The notion of logical implication for functional dependencies admits a sound and complete finite axiomatization, known as **Armstrong's axioms**.

Examples

Cars

Suppose one is designing a system to track vehicles and the capacity of their engines. Each vehicle has a unique vehicle identification number (VIN). One would write **VIN** → **EngineCapacity** because it would be inappropriate for a vehicle's engine to have more than one capacity. (Assuming, in this case, that vehicles only have one engine.) However, *EngineCapacity* → *VIN*, is incorrect because there could be many vehicles with the same engine capacity.

This functional dependency may suggest that the attribute EngineCapacity be placed in a relation with candidate key VIN. However, that may not always be appropriate. For example, if that functional dependency occurs as a result of the transitive functional dependencies VIN → VehicleModel and VehicleModel → EngineCapacity then that would not result in a normalized relation.

Lectures

This example illustrates the concept of functional dependency. The situation modelled is that of college students visiting one or more lectures in each of which they are assigned a teaching assistant (TA). Let's further assume that every student is in some semester and is identified by a unique integer ID.

StudentID	Semester	Lecture	TA
1234	6	Numerical Methods	Azhar
2380	4	Numerical Methods	Peter
1234	6	Visual Computing	Ahmed
1201	4	Numerical Methods	Peter
1201	4	Physics II	Simone

We notice that whenever two rows in this table feature the same StudentID, they also necessarily have the same Semester values. This basic fact can be expressed by a functional dependency:

- StudentID → Semester.

Other nontrivial functional dependencies can be identified, for example:

- {StudentID, Lecture} → TA
- {StudentID, Lecture} → {TA, Semester}

The latter expresses the fact that the set {StudentID, Lecture} is a superkey of the relation.

Properties and axiomatization of functional dependencies

Given that *X*, *Y*, and *Z* are sets of attributes in a relation *R*, one can derive several properties of functional dependencies. Among the most important are the following, usually called Armstrong's axioms:

- **Reflexivity**: If *Y* is a subset of *X*, then $X \to Y$
- **Augmentation**: If $X \to Y$, then $XZ \to YZ$
- **Transitivity**: If $X \to Y$ and $Y \to Z$, then $X \to Z$

"Reflexivity" can be weakened to just $X \to \varnothing$, i.e. it is an actual axiom, where the other two are proper inference rules, more precisely giving rise to the following rules of syntactic consequence:[1]

$\vdash X \to \varnothing$
$X \to Y \vdash XZ \to YZ$
$X \to Y, Y \to Z \vdash X \to Z$.

These three rules are a sound and complete axiomatization of functional dependencies. This axiomatization is sometimes described as finite because the number of inference rules is finite, with the caveat that the axiom and rules of inference are all schemata, meaning that the *X*, *Y* and *Z* range over all ground terms (attribute sets).

From these rules, we can derive these secondary rules:

- **Union**: If $X \to Y$ and $X \to Z$, then $X \to YZ$
- **Decomposition**: If $X \to YZ$, then $X \to Y$ and $X \to Z$
- **Pseudotransitivity**: If $X \to Y$ and $WY \to Z$, then $WX \to Z$

The union and decomposition rules can be combined in a logical equivalence stating that $X \to YZ$, holds iff $X \to Y$ and $X \to Z$. This is sometimes called the splitting/combining rule.

Another rule that is sometimes handy is:

- **Composition**: If $X \to Y$ and $Z \to W$, then $XZ \to YW$

Equivalent sets of functional dependencies are called *covers* of each other. Every set of functional dependencies has a canonical cover.

Applications to normalization

Heath's theorem

An important property (yielding an immediate application) of functional dependencies is that if *R* is a relation with columns named from some set of attributes *U* and *R* satisfies some functional dependency $X \to Y$ then $R = \pi_{XY}(R) \bowtie \pi_{XZ}(R)$ where $Z = U - XY$. Intuitively, if a functional dependency $X \to Y$ holds in *R*, then the relation can be safely split in two relations alongside the column *X* (which is a key for $\pi_{XY}(R) \bowtie \pi_{XZ}(R)$) ensuring that when the two parts are joined back no data is lost, i.e. a functional dependency provides a simple way to construct a lossless-join decomposition of *R* in two smaller relations. This fact is sometimes called **Heath's theorem**; it is one of the early results in database theory.[2]

Heath's theorem effectively says we can pull out the values of *Y* from the big relation *R* and store them into one, $\pi_{XY}(R)$, which has no value repetitions in the row for *X* and is effectively a lookup table for *Y* keyed by *X* and consequently has only one place to update the *Y* corresponding to each *X* unlike the "big" relation *R* where there are potentially many copies of each *X*, each one with its copy of *Y* which need to be kept synchronized on updates. (This elimination of redundancy is an advantage in OLTP contexts, where many changes are expected, but not so much in OLAP contexts, which involve mostly queries.) Heath's decomposition leaves only *X* to act as a foreign key in the remainder of the big table $\pi_{XZ}(R)$.

Functional dependencies however should not be confused with inclusion dependencies, which are the formalism for foreign keys; even though they are used for normalization, functional dependencies express constraints over one relation (schema), whereas inclusion dependencies express constraints between relation schemas in a database schema.Furthermore, the two notions do not even intersect in the classification of dependencies: functional dependencies are equality-generating dependencies whereas inclusion dependencies are tuple-generating dependencies. Enforcing referential constraints after relation schema decomposition (normalization) requires a new formalism, i.e. inclusion dependencies. In the decomposition resulting from Heath's theorem, there's nothing preventing the insertion of tuples in $\pi_{XZ}(R)$ having some value of *X* not found in $\pi_{XY}(R)$.

Normal forms

Normal forms are database normalization levels which determine the "goodness" of a table. Generally, the third normal form is considered to be a "good" standard for a relational database.[*citation needed*]

Normalization aims to free the database from update, insertion and deletion anomalies. It also ensures that when a new value is introduced into the relation, it has minimal effect on the database, and thus minimal effect on the applications using the database.[*citation needed*]

Irreducible function depending set

A functional depending set S is irreducible if the set has the following three properties:

1. Each right set of a functional dependency of S contains only one attribute.
2. Each left set of a functional dependency of S is irreducible. It means that reducing any one attribute from left set will change the content of S (S will lose some information).
3. Reducing any functional dependency will change the content of S.

Sets of Functional Dependencies(FD) with these properties are also called *canonical* or *minimal*.

References

[1] M. Y. Vardi. Fundamentals of dependency theory (http://www.cs.rice.edu/~vardi/papers/ttcs87.pdf). In E. Borger, editor, Trends in Theoretical Computer Science, pages 171–224. Computer Science Press, Rockville, MD, 1987. ISBN 0881750840

[2] cited in:

External links

- Gary Burt (summer, 1999). "CS 461 (Database Management Systems) lecture notes" (http://www.cs.umbc.edu/courses/461/current/burt/lectures/lec14/). University of Maryland Baltimore County Department of Computer Science and Electrical Engineering.
- Jeffrey D. Ullman. "CS345 Lecture Notes" (http://www-db.stanford.edu/~ullman/cs345notes/slides01-1.ps) (PostScript). Stanford University.
- Osmar Zaiane (June 9, 1998). "Chapter 6: Integrity constraints" (http://www.cs.sfu.ca/CC/354/zaiane/material/notes/Chapter6/node10.html). *CMPT 354 (Database Systems I) lecture notes*. Simon Fraser University Department of Computing Science.

Armstrong's axioms

Armstrong's axioms are a set of axioms (or, more precisely, inference rules) used to infer all the functional dependencies on a relational database. They were developed by William W. Armstrong on his 1974 paper.[1] The axioms are sound in generating only functional dependencies in the closure of a set of functional dependencies (denoted as F^+) when applied to that set (denoted as F). They are also complete in that repeated application of these rules will generate all functional dependencies in the closure F^+.

More formally, let $< R (U), F >$ denote a relational scheme over the set of attributes U with a set of functional dependencies F. We say that a functional dependency f is logically implied by F,and denote it with $F \models f$ if and only if for every instance r of R that satisfies the functional dependencies in F, r also satisfies f. We denote by F^+ the set of all functional dependencies that are logically implied by F.

Furthermore, with respect to a set of inference rules A, we say that a functional dependency f is derivable from the functional dependencies in F by the set of inference rules A, and we denote it by $F \vdash_A f$ if and only if f is obtainable by means of repeatedly applying the inference rules in A to functional dependencies in F. We denote by F_A^* the set of all functional dependencies that are derivable from F by inference rules in A.

Then, a set of inference rules A is sound if and only if the following holds:

$F_A^* \subseteq F^+$

that is to say, we cannot derive by means of A functional dependencies that are not logically implied by F. The set of inference rules A is said to be complete if the following holds:

$F^+ \subseteq F_A^*$

more simply put, we are able to derive by A all the functional dependencies that are logically implied by F.

Axioms

Let $R (U)$ be a relation scheme over the set of attributes U. Henceforth we will denote by letters X, Y, Z any subset of U and, for short, the union of two sets of attributes X and Y by XY instead of the usual $X \cup Y$; this notation is rather standard in database theory when dealing with sets of attributes.

Axiom of Reflexivity

If $Y \subseteq X$, then $X \rightarrow Y$

Axiom of augmentation

If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z

Axiom of transitivity

If $X \to Y$ and $Y \to Z$, then $X \to Z$

Additional rules

These rules can be derived from above axioms.

Union

If $X \to Y$ and $X \to Z$ then $X \to YZ$

Decomposition

If $X \to YZ$ then $X \to Y$ and $X \to Z$

Pseudo transitivity

If $A \to B$ and $BC \to D$ then $AC \to D$

Armstrong relation

Given a set of functional dependencies F, the **Armstrong relation** is a relation which satisfies all the functional dependencies in the closure F^+ and only those dependencies. Unfortunately, the minimum-size Armstrong relation for a given set of dependencies can have a size which is an exponential function of the number of attributes in the dependencies considered.

External links

- UMBC CMSC 461 Spring '99 [2]
- CS345 Lecture Notes from Stanford University [3]

References

[1] William Ward Armstrong: *Dependency Structures of Data Base Relationships*, page 580-583. IFIP Congress, 1974.

[2] http://www.cs.umbc.edu/courses/461/current/burt/lectures/lec14/

[3] http://www-db.stanford.edu/~ullman/cs345notes/slides01-1.ps

Transitive dependency

In Database Management System, a **transitive dependency** is a functional dependency which holds by virtue of transitivity. A transitive dependency can occur only in a relation that has three or more attributes. Let A, B, and C designate three distinct attributes (or distinct collections of attributes) in the relation. Suppose all three of the following conditions hold:

1. $A \rightarrow B$
2. It is not the case that $B \rightarrow A$
3. $B \rightarrow C$

Then the functional dependency $A \rightarrow C$ (which follows from 1 and 3 by the axiom of transitivity) is a transitive dependency.

In database normalization, one of the important features of third normal form is that it excludes certain types of transitive dependencies. E.F. Codd, the inventor of the relational model, introduced the concepts of transitive dependence and third normal form in 1971.[1]

Example

A transitive dependency occurs in the following relation:

Book	Genre	Author	Author Nationality
Twenty Thousand Leagues Under the Sea	Science Fiction	Jules Verne	French
Journey to the Center of the Earth	Science Fiction	Jules Verne	French
Leaves of Grass	Poetry	Walt Whitman	American
Anna Karenina	Literary Fiction	Leo Tolstoy	Russian
A Confession	Religious Autobiography	Leo Tolstoy	Russian

The functional dependency {Book} → {Author Nationality} applies; that is, if we know the book, we know the author's nationality. Furthermore:

- {Book} → {Author}
- {Author} does not → {Book}
- {Author} → {Author Nationality}

Therefore {Book} → {Author Nationality} is a transitive dependency.

Transitive dependency occurred because a non-key attribute (Author) was determining another non-key attribute (Author Nationality).

Notes

[1] Codd, E.F. "Further Normalization of the Data Base Relational Model." (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems," New York City, May 24th-25th, 1971.) IBM Research Report RJ909 (August 31st, 1971). Republished in Randall J. Rustin (ed.), *Data Base Systems: Courant Computer Science Symposia Series 6*. Prentice-Hall, 1972. See pages 45-51, which cover third normal form and transitive dependence.

Superkey

A **superkey** is defined in the relational model of database organization as a set of attributes of a relation variable for which it holds that in all relations assigned to that variable, there are no two distinct tuples (rows) that have the same values for the attributes in this set. Equivalently a superkey can also be defined as a set of attributes of a relation schema upon which all attributes of the schema are functionally dependent.

Note that the set of **all** attributes is a trivial superkey, because in relational algebra duplicate rows are not permitted.

Also note that if attribute set *K* is a superkey of relation *R*, then at all times it is the case that the projection of *R* over *K* has the same cardinality as *R* itself.

Informally, a superkey is a set of attributes within a table whose values can be used to uniquely identify a tuple. A candidate key is a minimal set of attributes necessary to identify a tuple, this is also called a minimal superkey. For example, given an employee schema, consisting of the attributes employeeID, name, job, and departmentID, we could use the employeeID in combination with any or all other attributes of this table to uniquely identify a tuple in the table. Examples of superkeys in this schema would be {employeeID, Name}, {employeeID, Name, job}, and {employeeID, Name, job, departmentID}. The last example is known as trivial superkey, because it uses all attributes of this table to identify the tuple.

In a real database we do not need values for all of those attributes to identify a tuple. We only need, per our example, the set {employeeID}. This is a **minimal superkey –** that is, a minimal set of attributes that can be used to identify a single tuple. So, employeeID is a candidate key.

Example

English Monarchs

Monarch Name	Monarch Number	Royal House
Edward	II	Plantagenet
Edward	III	Plantagenet
Richard	III	Plantagenet
Henry	IV	Lancaster

First, list out all the (non-empty) sets of attributes:

- {Monarch Name}
- {Monarch Number}
- {Royal House}
- {Monarch Name, Monarch Number}
- {Monarch Name, Royal House}
- {Monarch Number, Royal House}
- {Monarch Name, Monarch Number, Royal House}

Second, eliminate all the sets which **do not** meet superkey's requirement. For example, {Monarch Name, Royal House} cannot be a superkey because for the same attribute values (Edward, Plantagenet), there are two distinct tuples:

- (Edward, **II**, Plantagenet)
- (Edward, **III**, Plantagenet)

Finally, after elimination, the remaining sets of attributes are the only possible superkeys in this example:

- {Monarch Name, Monarch Number} **(Candidate Key)**
- {Monarch Name, Monarch Number, Royal House}

In real situations, however, superkeys are normally not determined by this method, which is very tedious and time-consuming, but by analyzing functional dependencies.

References

- Silberschatz, Abraham (2011). *Database System Concepts (6th ed.)*. McGraw-Hill. pp. 45–46. ISBN 978-0-07-352332-3.

External links

- Relation Database terms of reference, Keys (http://rdbms.opengrass.net/2_Database Design/2.1_TermsOfReference/2.1.2_Keys.html): An overview of the different types of keys in an RDBMS

First normal form

First normal form (**1NF**) is a property of a relation in a relational database. A relation is in first normal form if the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

Edgar Codd, in a 1971 conference paper, defined a relation in first normal form to be one such that none of the domains of that relation should have elements which are themselves sets.

First normal form is an essential property of a relation in a relational database. Database normalization is the process of representing a database in terms of relations in standard normal forms, where first normal is a minimal requirement.

Examples

The following scenario illustrates how a database design might violate first normal form.

Domains and values

Suppose a designer wishes to record the names and telephone numbers of customers. He defines a customer table which looks like this:

Customer

Customer ID	First Name	Surname	Telephone Number
123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659
789	Maria	Fernandez	555-808-9633

The designer then becomes aware of a requirement to record **multiple** telephone numbers for some customers. He reasons that the simplest way of doing this is to allow the "Telephone Number" field in any given record to contain more than one value:

Customer

Customer ID	First Name	Surname	Telephone Number
123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659 555-776-4100
789	Maria	Fernandez	555-808-9633

Assuming, however, that the Telephone Number column is defined on some telephone number-like domain, such as the domain of 12-character strings, the representation above is not in first normal form. It is in violation of first normal form as a single field has been allowed to contain multiple values. A typical relational database management system will not allow fields in a table to contain multiple values in this way.

A design that complies with 1NF

A design that is unambiguously in first normal form makes use of two tables: a Customer Name table and a Customer Telephone Number table.

Customer ID	First Name	Surname
123	Robert	Ingram
456	Jane	Wright
789	Maria	Fernandez

Customer ID	Telephone Number
123	555-861-2025
456	555-403-1659
456	555-776-4100
789	555-808-9633

Repeating groups of telephone numbers do not occur in this design. Instead, each Customer-to-Telephone Number link appears on its own record. With Customer ID as key, a one-to-many relationship exists between the two tables. A record in the "parent" table, Customer Name, can have many telephone number records in the "child" table, Customer Telephone Number, but each telephone number belongs to one, and only one customer. It is worth noting that this design meets the additional requirements for second and third normal form.

Atomicity

Edgar F. Codd's definition of 1NF makes reference to the concept of 'atomicity'. Codd states that the "values in the domains on which each relation is defined are required to be atomic with respect to the DBMS."[1] Codd defines an atomic value as one that "cannot be decomposed into smaller pieces by the DBMS (excluding certain special functions)"[2] meaning a field should not be divided into parts with more than one kind of data in it such that what one part means to the DBMS depends on another part of the same field.

Hugh Darwen and Chris Date have suggested that Codd's concept of an "atomic value" is ambiguous, and that this ambiguity has led to widespread confusion about how 1NF should be understood.[3][4] In particular, the notion of a "value that cannot be decomposed" is problematic, as it would seem to imply that few, if any, data types are atomic:

- A character string would seem not to be atomic, as the RDBMS typically provides operators to decompose it into substrings.
- A fixed-point number would seem not to be atomic, as the RDBMS typically provides operators to decompose it into integer and fractional components.
- An ISBN would seem not to be atomic, as it includes language and publisher identifiers.

Date suggests that "the notion of atomicity *has no absolute meaning*":[5] a value may be considered atomic for some purposes, but may be considered an assemblage of more basic elements for other purposes. If this position is

accepted, 1NF cannot be defined with reference to atomicity. Columns of any conceivable data type (from string types and numeric types to array types and table types) are then acceptable in a 1NF table—although perhaps not always desirable; for example, it would be more desirable to separate a Customer Name field into two separate fields as First Name, Surname.

First normal form, as defined by Chris Date, permits relation-valued attributes (tables within tables). Date argues that relation-valued attributes, by means of which a field within a table can contain a table, are useful in rare cases.[6]

1NF tables as representations of relations

According to Date's definition, a table is in first normal form if and only if it is "isomorphic to some relation", which means, specifically, that it satisfies the following five conditions:[7]

1. There's no top-to-bottom ordering to the rows.
2. There's no left-to-right ordering to the columns.
3. There are no duplicate rows.
4. Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else).
5. All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps].

Violation of any of these conditions would mean that the table is not strictly relational, and therefore that it is not in first normal form.

Examples of tables (or views) that would not meet this definition of first normal form are:

- A table that lacks a unique key. Such a table would be able to accommodate duplicate rows, in violation of condition 3.
- A view whose definition mandates that results be returned in a particular order, so that the row-ordering is an intrinsic and meaningful aspect of the view.[8] This violates condition 1. The tuples in true relations are not ordered with respect to each other.
- A table with at least one nullable attribute. A nullable attribute would be in violation of condition 4, which requires every field to contain exactly one value from its column's domain. It should be noted, however, that this aspect of condition 4 is controversial. It marks an important departure from Codd's later vision of the relational model,[9] which made explicit provision for nulls.[10]

References

[1] Codd, E. F. *The Relational Model for Database Management Version 2* (Addison-Wesley, 1990).

[2] Codd, E. F. *The Relational Model for Database Management Version 2* (Addison-Wesley, 1990), p. 6.

[3] Darwen, Hugh. "Relation-Valued Attributes; or, Will the Real First Normal Form Please Stand Up?", in C. J. Date and Hugh Darwen, *Relational Database Writings 1989-1991* (Addison-Wesley, 1992).

[4] "[F]or many years," writes Date, "I was as confused as anyone else. What's worse, I did my best (worst?) to spread that confusion through my writings, seminars, and other presentations." Date, C. J. ["What First Normal Form Really Means"] in *Date on Database: Writings 2000-2006* (Springer-Verlag, 2006), p. 108

[5] Date, C. J. ["What First Normal Form Really Means"] p. 112.

[6] Date, C. J. ["What First Normal Form Really Means"] pp. 121–126.

[7] Date, C. J. ["What First Normal Form Really Means"] pp. 127–128.

[8] Such views cannot be created using SQL that conforms to the SQL:2003 standard.

[9] "Codd first defined the relational model in 1969 and didn't introduce nulls until 1979" Date, C. J. *SQL and Relational Theory* (O'Reilly, 2009), Appendix A.2.

[10] The third of Codd's 12 rules states that "Null values ... [must be] supported in a fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type." Codd, E. F. "Is Your DBMS Really Relational?" *Computerworld*, October 14, 1985.

Further reading

- Litt's Tips: Normalization (http://www.troubleshooters.com/littstip/ltnorm.html)
- Date, C. J., & Lorentzos, N., & Darwen, H. (2002). *Temporal Data & the Relational Model* (http://www.elsevier.com/wps/product/cws_home/680662) (1st ed.). Morgan Kaufmann. ISBN 1-55860-855-9.
- Date, C. J. (1999), *An Introduction to Database Systems* (http://www.aw-bc.com/catalog/academic/product/0,1144,0321197844,00.html) (8th ed.). Addison-Wesley Longman. ISBN 0-321-19784-4.
- Kent, W. (1983) *A Simple Guide to Five Normal Forms in Relational Database Theory* (http://www.bkent.net/Doc/simple5.htm), Communications of the ACM, vol. 26, pp. 120–125

Second normal form

Second normal form (**2NF**) is a normal form used in database normalization. 2NF was originally defined by E.F. Codd in 1971.[1]

A table that is in first normal form (1NF) must meet additional criteria if it is to qualify for second normal form. Specifically: a table is in 2NF if and only if it is in 1NF and no non-prime attribute is dependent on any proper subset of any candidate key of the table. A non-prime attribute of a table is an attribute that is not a part of any candidate key of the table.

Put simply, a table is in 2NF if and only if it is in 1NF and every non-prime attribute of the table is dependent on the whole of a candidate key.

Example

Consider a table describing employees' skills:

Employees' Skills

Employee	Skill	Current Work Location
Brown	Light Cleaning	73 Industrial Way
Brown	Typing	73 Industrial Way
Harrison	Light Cleaning	73 Industrial Way
Jones	Shorthand	114 Main Street
Jones	Typing	114 Main Street
Jones	Whittling	114 Main Street

Neither {Employee} nor {Skill} is a candidate key for the table. This is because a given Employee might need to appear more than once (he might have multiple Skills), and a given Skill might need to appear more than once (it might be possessed by multiple Employees). Only the composite key {Employee, Skill} qualifies as a candidate key for the table.

The remaining attribute, Current Work Location, is dependent on only part of the candidate key, namely Employee. Therefore the table is not in 2NF. Note the redundancy in the way Current Work Locations are represented: we are told three times that Jones works at 114 Main Street, and twice that Brown works at 73 Industrial Way. This redundancy makes the table vulnerable to update anomalies: it is, for example, possible to update Jones' work location on his "**Shorthand**" and "**Typing**" records and not update his "**Whittling**" record. The resulting data would imply contradictory answers to the question "What is Jones' current work location?"

A 2NF alternative to this design would represent the same information in two tables: an "Employees" table with candidate key {Employee}, and an "Employees' Skills" table with candidate key {Employee, Skill}:

Employee	Current Work Location
Brown	73 Industrial Way
Harrison	73 Industrial Way
Jones	114 Main Street

Employee	Skill
Brown	Light Cleaning
Brown	Typing
Harrison	Light Cleaning
Jones	Shorthand
Jones	Typing
Jones	Whittling

Neither of these tables can suffer from update anomalies.

Not all 2NF tables are free from update anomalies, however. An example of a 2NF table which suffers from update anomalies is:

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Des Moines Masters	1998	Chip Masterson	14 March 1977
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

Even though Winner and Winner Date of Birth are determined by the whole key {Tournament, Year} and not part of it, particular Winner / Winner Date of Birth combinations are shown redundantly on multiple records. This leads to an update anomaly: if updates are not carried out consistently, a particular winner could be shown as having two different dates of birth.

The underlying problem is the transitive dependency to which the Winner Date of Birth attribute is subject. Winner Date of Birth actually depends on Winner, which in turn depends on the key Tournament / Year.

This problem is addressed by third normal form (3NF).

2NF and candidate keys

A table for which there are no partial functional dependencies on the primary key is typically, but not always, in 2NF. In addition to the primary key, the table may contain other candidate keys; it is necessary to establish that no non-prime attributes have part-key dependencies on **any** of these candidate keys.

Multiple candidate keys occur in the following table:

Electric Toothbrush Models

Manufacturer	Model	Model Full Name	Manufacturer Country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

Even if the designer has specified the primary key as {Model Full Name}, the table is not in 2NF. {Manufacturer, Model} is also a candidate key, and Manufacturer Country is dependent on a proper subset of it: Manufacturer. To make the design conform to 2NF, it is necessary to have two tables:

Electric Toothbrush Manufacturers

Manufacturer	Manufacturer Country
Forte	Italy
Dent-o-Fresh	USA
Kobayashi	Japan
Hoch	Germany

Electric Toothbrush Models

Manufacturer	Model	Model Full Name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

References

[1] Codd, E.F. "Further Normalization of the Data Base Relational Model." (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems," New York City, May 24th-25th, 1971.) IBM Research Report RJ909 (August 31st, 1971). Republished in Randall J. Rustin (ed.), *Data Base Systems: Courant Computer Science Symposia Series 6*. Prentice-Hall, 1972.

Further reading

- Litt's Tips: Normalization (http://www.troubleshooters.com/littstip/ltnorm.html)
- Date, C. J., & Lorentzos, N., & Darwen, H. (2002). *Temporal Data & the Relational Model* (http://www.elsevier.com/wps/product/cws_home/680662) (1st ed.). Morgan Kaufmann. ISBN 1-55860-855-9.
- C.J.Date (2004). *Introduction to Database Systems* (8th ed.). Boston: Addison-Wesley. ISBN 978-0-321-19784-9.
- Kent, W. (1983) *A Simple Guide to Five Normal Forms in Relational Database Theory* (http://www.bkent.net/Doc/simple5.htm), Communications of the ACM, vol. 26, pp. 120–125

External links

- Database Normalization Basics (http://databases.about.com/od/specificproducts/a/normalization.htm) by Mike Chapple (About.com)
- An Introduction to Database Normalization (http://mikehillyer.com/articles/an-introduction-to-database-normalization/) by Mike Hillyer.
- A tutorial on the first 3 normal forms (http://phlonx.com/resources/nf3/) by Fred Coulson
- Description of the database normalization basics (http://support.microsoft.com/kb/283878) by Microsoft

Third normal form

The **third normal form** (**3NF**) is a normal form used in database normalization. 3NF was originally defined by E.F. Codd in 1971.[1] Codd's definition states that a table is in 3NF if and only if both of the following conditions hold:

- The relation R (table) is in second normal form (**2NF**)
- Every non-prime attribute of R is non-transitively dependent (i.e. directly dependent) on every superkey of R.

A **non-prime attribute** of R is an attribute that does not belong to any candidate key of R.[2] A transitive dependency is a functional dependency in which $X \to Z$ (X determines Z) indirectly, by virtue of $X \to Y$ and $Y \to Z$ (where it is not the case that $Y \to X$).[3]

A 3NF definition that is equivalent to Codd's, but expressed differently, was given by Carlo Zaniolo in 1982. This definition states that a table is in 3NF if and only if, for each of its functional dependencies $X \to A$, **at least one** of the following conditions holds:

- X contains A (that is, $X \to A$ is trivial functional dependency), or
- X is a superkey, or
- Every element of A-X, the set difference between A and X, is a **prime attribute** (i.e., each attribute in A-X is contained in some candidate key)[4][5]Wikipedia:Verifiability

Zaniolo's definition gives a clear sense of the difference between 3NF and the more stringent Boyce–Codd normal form (BCNF). BCNF simply eliminates the third alternative ("Every element of A-X, the set difference between A and X, is a prime attribute").

"Nothing but the key"

A memorable statement of Codd's definition of 3NF, paralleling the traditional pledge to give true evidence in a court of law, was given by Bill Kent: "[Every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key."[6] A common variation supplements this definition with the oath: "so help me Codd".[7]

Requiring existence of "the key" ensures that the table is in 1NF; requiring that non-key attributes be dependent on "the whole key" ensures 2NF; further requiring that non-key attributes be dependent on "nothing but the key" ensures 3NF.

Chris Date refers to Kent's summary as "an intuitively attractive characterization" of 3NF, and notes that with slight adaptation it may serve as a definition of the slightly stronger Boyce–Codd normal form: "Each attribute must represent a fact about the key, the whole key, and nothing but the key."[8] The 3NF version of the definition is weaker than Date's BCNF variation, as the former is concerned only with ensuring that *non-key* attributes are dependent on keys. Prime attributes (which are keys or parts of keys) must not be functionally dependent at all; they each represent a fact about the key in the sense of providing part or all of the key itself. (It should be noted here that this rule applies only to functionally dependent attributes, as applying it to all attributes would implicitly prohibit composite candidate keys, since each part of any such key would violate the "whole key" clause.)

An example of a 2NF table that fails to meet the requirements of 3NF is:

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

Because each row in the table needs to tell us who won a particular Tournament in a particular Year, the composite key {Tournament, Year} is a minimal set of attributes guaranteed to uniquely identify a row. That is, {Tournament, Year} is a candidate key for the table.

The breach of 3NF occurs because the non-prime attribute Winner Date of Birth is transitively dependent on the candidate key {Tournament, Year} via the non-prime attribute Winner. The fact that Winner Date of Birth is functionally dependent on Winner makes the table vulnerable to logical inconsistencies, as there is nothing to stop the same person from being shown with different dates of birth on different records.

In order to express the same facts without violating 3NF, it is necessary to split the table into two:

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Player	Date of Birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

Update anomalies cannot occur in these tables, which are both in 3NF.

Derivation of Zaniolo's conditions

The definition of 3NF offered by Carlo Zaniolo in 1982, and given above, is proved in the following way: Let $X \to A$ be a nontrivial FD (i.e. one where X does not contain A) and let A be a non-key attribute. Also let Y be a key of R. Then $Y \to X$.

Normalization beyond 3NF

Most 3NF tables are free of update, insertion, and deletion anomalies. Certain types of 3NF tables, rarely met with in practice, are affected by such anomalies; these are tables which either fall short of Boyce–Codd normal form (BCNF) or, if they meet BCNF, fall short of the higher normal forms 4NF or 5NF.

References

[1] Codd, E.F. "Further Normalization of the Data Base Relational Model." (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems," New York City, May 24th–25th, 1971.) IBM Research Report RJ909 (August 31st, 1971). Republished in Randall J. Rustin (ed.), *Data Base Systems: Courant Computer Science Symposia Series 6*. Prentice-Hall, 1972.

[2] Codd, p. 43.

[3] Codd, p. 45–46.

[4] Zaniolo, Carlo. "A New Normal Form for the Design of Relational Database Schemata." *ACM Transactions on Database Systems* 7(3), September 1982.

[5] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts* (http://www.db-book.com/) *(5th edition), p. 276-277*

[6] Kent, William. "A Simple Guide to Five Normal Forms in Relational Database Theory" (http://www.bkent.net/Doc/simple5.htm), *Communications of the ACM* **26** (2), Feb. 1983, pp. 120–125.

[7] The author of a 1989 book on database management credits one of his students with coming up with the "so help me Codd" addendum. Diehr, George. *Database Management* (Scott, Foresman, 1989), p. 331.

[8] Date, C.J. *An Introduction to Database Systems* (7th ed.) (Addison Wesley, 2000), p. 379.

Further reading

- Date, C. J. (1999), *An Introduction to Database Systems* (http://www.aw-bc.com/catalog/academic/product/0,1144,0321197844,00.html) (8th ed.). Addison-Wesley Longman. ISBN 0-321-19784-4.
- Kent, W. (1983) *A Simple Guide to Five Normal Forms in Relational Database Theory* (http://www.bkent.net/Doc/simple5.htm), Communications of the ACM, vol. 26, pp. 120–126

External links

- Litt's Tips: Normalization (http://www.troubleshooters.com/littstip/ltnorm.html)
- Database Normalization Basics (http://databases.about.com/od/specificproducts/a/normalization.htm) by Mike Chapple (About.com)
- An Introduction to Database Normalization (http://mikehillyer.com/articles/an-introduction-to-database-normalization/) by Mike Hillyer.
- A tutorial on the first 3 normal forms (http://phlonx.com/resources/nf3/) by Fred Coulson
- Description of the database normalization basics (http://support.microsoft.com/kb/283878) by Microsoft

Boyce–Codd normal form

Boyce–Codd normal form (or **BCNF** or **3.5NF**) is a normal form used in database normalization. It is a slightly stronger version of the third normal form (3NF). BCNF was developed in 1974 by Raymond F. Boyce and Edgar F. Codd to address certain types of anomaly not dealt with by 3NF as originally defined.[1]

If a relational schema is in BCNF then all redundancy based on functional dependency has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce–Codd normal form if and only if for every one of its dependencies $X \rightarrow Y$, at least one of the following conditions hold:

- $X \rightarrow Y$ is a trivial functional dependency ($Y \subseteq X$)
- X is a superkey for schema R

Chris Date has pointed out that a definition of what we now know as BCNF appeared in a paper by Ian Heath in 1971.[2] Date writes:

> "Since that definition predated Boyce and Codd's own definition by some three years, it seems to me that BCNF ought by rights to be called *Heath* normal form. But it isn't."[3]

Edgar F.Codd released his original paper 'A Relational Model of Data for Large Shared Databanks' in June 1970. This was the first time the notion of a relational database was published. All work after this, including the Boyce-Codd normal form method was based on this relational model.

3NF tables not meeting BCNF (Boyce–Codd normal form)

Only in rare cases does a 3NF table not meet the requirements of BCNF. A 3NF table which does not have multiple overlapping candidate keys is guaranteed to be in BCNF.[4] Depending on what its functional dependencies are, a 3NF table with two or more overlapping candidate keys may or may not be in BCNF.

An example of a 3NF table that does not meet BCNF is:

Today's Court Bookings

Court	Start Time	End Time	Rate Type
1	09:30	10:30	SAVER
1	11:00	12:00	SAVER
1	14:00	15:30	STANDARD
2	10:00	11:30	PREMIUM-B
2	11:30	13:30	PREMIUM-B
2	15:00	16:30	PREMIUM-A

- Each row in the table represents a court booking at a tennis club that has one hard court (Court 1) and one grass court (Court 2)
- A booking is defined by its Court and the period for which the Court is reserved
- Additionally, each booking has a Rate Type associated with it. There are four distinct rate types:
  - SAVER, for Court 1 bookings made by members
  - STANDARD, for Court 1 bookings made by non-members
  - PREMIUM-A, for Court 2 bookings made by members
  - PREMIUM-B, for Court 2 bookings made by non-members

The table's superkeys are:

- S_1 = {Court, Start Time}

- S_2 = {Court, End Time}
- S_3 = {Rate Type, Start Time}
- S_4 = {Rate Type, End Time}
- S_5 = {Court, Start Time, End Time}
- S_6 = {Rate Type, Start Time, End Time}
- S_7 = {Court, Rate Type, Start Time}
- S_8 = {Court, Rate Type, End Time}
- S_T = {Court, Rate Type, Start Time, End Time}, the trivial superkey

Note that even though in the above table *Start Time* and *End Time* attributes have no duplicate values for each of them, we still have to admit that in some other days two different bookings on court 1 and court 2 could *start at the same time* or *end at the same time*. This is the reason why {Start Time} and {End Time} cannot be considered as the table's superkeys.

However, only S_1, S_2, S_3 and S_4 are candidate keys (that is, minimal superkeys for that relation) because e.g. $S_1 \subset S_5$, so S_5 cannot be a candidate key.

Recall that 2NF prohibits partial functional dependencies of non-prime attributes (i.e. an attribute that does not occur in ANY candidate key) on candidate keys, and that 3NF prohibits transitive functional dependencies of non-prime attributes on candidate keys.

In **Today's Court Bookings** table, there are no non-prime attributes: that is, all attributes belong to some candidate key. Therefore the table adheres to both 2NF and 3NF.

The table does not adhere to BCNF. This is because of the dependency Rate Type → Court, in which the determining attribute (Rate Type) is neither a candidate key nor a superset of a candidate key.

Dependency Rate Type → Court is respected as a Rate Type should only ever apply to a single Court.

The design can be amended so that it meets BCNF:

Rate Types

Rate Type	Court	Member Flag
SAVER	1	Yes
STANDARD	1	No
PREMIUM-A	2	Yes
PREMIUM-B	2	No

Today's Bookings

Rate Type	Start Time	End Time
SAVER	09:30	10:30
SAVER	11:00	12:00
STANDARD	14:00	15:30
PREMIUM-B	10:00	11:30
PREMIUM-B	11:30	13:30
PREMIUM-A	15:00	16:30

The candidate keys for the Rate Types table are {Rate Type} and {Court, Member Flag}; the candidate keys for the Today's Bookings table are {Rate Type, Start Time} and {Rate Type, End Time}. Both tables are in BCNF. When {Rate Type} is a key in the Rate Types table, having one Rate Type associated with two different Courts is

impossible, so by using {Rate Type} as a key in the Rate Types table, the anomaly affecting the original table has been eliminated.

Achievability of BCNF

In some cases, a non-BCNF table cannot be decomposed into tables that satisfy BCNF and preserve the dependencies that held in the original table. Beeri and Bernstein showed in 1979 that, for example, a set of functional dependencies {AB → C, C → B} cannot be represented by a BCNF schema.[5] Thus, unlike the first three normal forms, BCNF is not always achievable.

Consider the following non-BCNF table whose functional dependencies follow the {AB → C, C → B} pattern:

Nearest Shops

Person	Shop Type	Nearest Shop
Davidson	Optician	Eagle Eye
Davidson	Hairdresser	Snippets
Wright	Bookshop	Merlin Books
Fuller	Bakery	Doughy's
Fuller	Hairdresser	Sweeney Todd's
Fuller	Optician	Eagle Eye

For each Person / Shop Type combination, the table tells us which shop of this type is geographically nearest to the person's home. We assume for simplicity that a single shop cannot be of more than one type.

The candidate keys of the table are:

- {Person, Shop Type}
- {Person, Nearest Shop}

Because all three attributes are prime attributes (i.e. belong to candidate keys), the table is in 3NF. The table is not in BCNF, however, as the Shop Type attribute is functionally dependent on a non-superkey: Nearest Shop.

The violation of BCNF means that the table is subject to anomalies. For example, Eagle Eye might have its Shop Type changed to "Optometrist" on its "Fuller" record while retaining the Shop Type "Optician" on its "Davidson" record. This would imply contradictory answers to the question: "What is Eagle Eye's Shop Type?" Holding each shop's Shop Type only once would seem preferable, as doing so would prevent such anomalies from occurring:

Shop Near Person

Person	Shop
Davidson	Eagle Eye
Davidson	Snippets
Wright	Merlin Books
Fuller	Doughy's
Fuller	Sweeney Todd's
Fuller	Eagle Eye

Shop

Shop	Shop Type
Eagle Eye	Optician
Snippets	Hairdresser
Merlin Books	Bookshop
Doughy's	Bakery
Sweeney Todd's	Hairdresser

In this revised design, the "Shop Near Person" table has a candidate key of {Person, Shop}, and the "Shop" table has a candidate key of {Shop}. Unfortunately, although this design adheres to BCNF, it is unacceptable on different grounds: it allows us to record multiple shops of the same type against the same person. In other words, its candidate keys do not guarantee that the functional dependency {Person, Shop Type} → {Shop} will be respected.

A design that eliminates all of these anomalies (but does not conform to BCNF) is possible. This design introduces a new normal form, known as Elementary Key Normal Form.[6] This design consists of the original "Nearest Shops" table supplemented by the "Shop" table described above. The table structure generated by Bernstein's schema generation algorithm [7] is actually EKNF, although that enhancement to 3NF had not been recognized at the time the algorithm was designed

Nearest Shops

Person	Shop Type	Nearest Shop
Davidson	Optician	Eagle Eye
Davidson	Hairdresser	Snippets
Wright	Bookshop	Merlin Books
Fuller	Bakery	Doughy's
Fuller	Hairdresser	Sweeney Todd's
Fuller	Optician	Eagle Eye

Shop

Shop	Shop Type
Eagle Eye	Optician
Snippets	Hairdresser
Merlin Books	Bookshop
Doughy's	Bakery
Sweeney Todd's	Hairdresser

If a referential integrity constraint is defined to the effect that {Shop Type, Nearest Shop} from the first table must refer to a {Shop Type, Shop} from the second table, then the data anomalies described previously are prevented.

References

[1] Codd, E. F. "Recent Investigations into Relational Data Base Systems." IBM Research Report RJ1385 (April 23, 1974). Republished in *Proc. 1974 Congress* (Stockholm, Sweden, 1974). New York, N.Y.: North-Holland (1974).

[2] Heath, I. "Unacceptable File Operations in a Relational Database." *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access, and Control*, San Diego, Calif. (November 11th–12th, 1971).

[3] Date, C.J. *Database in Depth: Relational Theory for Practitioners*. O'Reilly (2005), p. 142.

[4] Vincent, M.W. and B. Srinivasan. "A Note on Relation Schemes Which Are in 3NF But Not in BCNF." *Information Processing Letters* 48(6), 1993, pp. 281–83.

[5] Beeri, Catriel and Bernstein, Philip A. "Computational problems related to the design of normal form relational schemas." *ACM Transactions on Database Systems* 4(1), March 1979, p. 50.

[6] Zaniolo, Carlo. "A New Normal Form for the Design of Relational Database Schemata." *ACM Transactions on Database Systems* 7(3), September 1982, pp. 493.

[7] Bernstein, P.A. "Synthesizing Third Normal Form relations from functional dependencies." *ACM Transactions on Database Systems* 1(4), December 1976 pp. 277-298.

Bibliography

- Date, C. J. (1999). *An Introduction to Database Systems* (8th ed.). Addison-Wesley Longman. ISBN 0-321-19784-4.

External links

- Rules Of Data Normalization (http://web.archive.org/web/20080805014412/http://www.datamodel.org/NormalizationRules.html)
- Advanced Normalization (http://web.archive.org/web/20080423014733/http://www.utexas.edu/its/archive/windows/database/datamodeling/rm/rm8.html) by ITS, University of Texas.

Lossless-Join Decomposition

In computer science the concept of a **Lossless-Join Decomposition** is central in removing redundancy safely from databases while preserving the original data.

Lossless-join Decomposition

Can also be called Nonadditive. If you decompose a relation R into relations R_1 and R_2 you will guarantee a Lossless-Join if $R_1 \bowtie R_2 = R$.

If R is split into R1 and R2, for the decomposition to be lossless then at least one of the two should hold true.

Projecting on R1 and R2, and joining back, results in the relation you started with.[1] Let R be a relation schema.

Let F be a set of functional dependencies on R .

Let R_1 and R_2 form a decomposition of R .

The decomposition is a lossless-join decomposition of R if at least one of the following functional dependencies are in F^+ (where F^+ stands for the closure for every attribute in F):

- $R_1 \cap R_2 \rightarrow R_1 - R_2$
- $R_1 \cap R_2 \rightarrow R_2 - R_1$

Example

- Let $R = (A, B, C, D)$ be the relation schema, with A , B , C and D attributes.
- Let $F = \{A \rightarrow BC\}$ be the set of functional dependencies.
- Decomposition into $R_1 = (A, B, C)$ and $R_2 = (A, D)$ is lossless under F because $R_1 \cap R_2 = (A)$ and $A \rightarrow BC$ so $R_1 \cap R_2 \rightarrow R_1 - R_2$.

[2] [3]

References

[1] http://stackoverflow.com/questions/5771810/lossless-join-property
[2] http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter7/node7.html
[3] http://www.data-e-education.com/E121_Lossless_Join_Decomposition.html

Join dependency

In dependency theory, a **join dependency** is a constraint on the set of legal relations over a database scheme. A table T is subject to a join dependency if T can always be recreated by joining multiple tables each having a subset of the attributes of T. If one of the tables in the join has all the attributes of the table T, the join dependency is called trivial.

The join dependency plays an important role in the Fifth normal form, also known as *project-join normal form*, because it can be proven that if you decompose a scheme R in tables R_1 to R_n , the decomposition will be a lossless-join decomposition if you restrict the legal relations on R to a join dependency on R called $*(R_1, R_2, ...R_n)$.

Another way to describe a join dependency is to say that the set of relationships in the join dependency is independent of each other.

Unlike in the case of functional dependencies, there is no sound and complete axiomatization for join dependencies,[1] though axiomatization exist for more expressive dependency languages such as full typed dependencies.[2] However, implication of join dependencies is decidable.[3]

Formal definition

> Let R be a relation schema and let $R_1, R_2, ..., R_n$ be a decomposition of R .
> The relation $r(R)$ *satisfies* the join dependency $*(R_1, R_2, ...R_n)$ if $\bowtie_{i=1}^{n} \Pi_{R_i}(r) = r$.
> A join dependency is trivial if one of the R_i is R itself.
> —Silberschatz, Korth. *Database System Concepts*, 1st Edition[4]

2-ary join dependencies are called multivalued dependency as a historical artifact of the fact that they were studied before the general case. More specifically if *U* is a set of attributes and *R* a relation over it, then *R* satisfies $X \twoheadrightarrow Y$ iff *R* satisfies $*(X \cup Y, X \cup (U - Y))$

Example

Given a pizza-chain that models purchases in table Customer = { order-number, customer-name, pizza-name, courier}. It is obvious that you can derive the following relations:

- customer-name depends on order-number
- pizza-name depends on order-number
- courier depends on order-number

Since the relationships are independent you can say there is a join dependency as follows: *((order-number, customer-name), (order-number, pizza-name), (order-number,courier)).

If each customer has his own courier however, you could have a join-dependency like this: *((order-number, customer-name), (order-number, courier), (customer-name, courier), (order-number,pizza-name)), but *((order-number, customer-name, courier), (order-number,pizza-name)) would be valid as well. This makes it obvious that just having a join dependency is not enough to normalize a database scheme.

References

[1] S. V. Petrov. Finite axiomatization of languages for representation of system properties. *Information Sciences*, 47:339-372, 1989.
[2] Abiteboul, Hull, Vianu. Foundations of databases. Chapter 8, Bibliographic notes
[3] ibid, Theorem 8.4.12
[4] Silberschatz, Korth. *Database System Concepts*, 1st Edition

Multivalued dependency

In database theory, **multivalued dependency** *is a full constraint* between two sets of attributes in a relation.

In contrast to the functional dependency, the **multivalued dependency** requires that certain tuples be present in a relation. Therefore, a multivalued dependency is a special case of *tuple-generating dependency*. The multivalued dependency plays a role in the 4NF database normalization.

A multivalued dependency is a special case of a join dependency, with only two sets of values involved, i.e. it is a 2-ary join dependency.

Formal definition

The formal definition is given as follows.

Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$ (subsets). The multivalued dependency

$\alpha \twoheadrightarrow \beta$

(which can be read as α multidetermines β) holds on R if, in any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that

$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$

$t_3[\beta] = t_1[\beta]$

$t_3[R - \beta] = t_2[R - \beta]$

$t_4[\beta] = t_2[\beta]$

$t_4[R - \beta] = t_1[R - \beta]$

In more simple words the above condition can be expressed as follows: if we denote by (x, y, z) the tuple having values for α, β, $R - \alpha - \beta$ collectively equal to x, y, z, correspondingly, then whenever the tuples (a, b, c) and (a, d, e) exist in r , the tuples (a, b, e) and (a, d, c) should also exist in r .

Example

Consider this example of a database of teaching courses, the books recommended for the course, and the lecturers who will be teaching the course:

Course	Book	Lecturer
AHA	Silberschatz	John D
AHA	Nederpelt	William M
AHA	Silberschatz	William M
AHA	Nederpelt	John D
AHA	Silberschatz	Christian G
AHA	Nederpelt	Christian G
OSO	Silberschatz	John D
OSO	Silberschatz	William M

Because the lecturers attached to the course and the books attached to the course are independent of each other, this database design has a multivalued dependency; if we were to add a new book to the AHA course, we would have to add one record for each of the lecturers on that course, and vice versa.

Put formally, there are two multivalued dependencies in this relation: {course} $\twoheadrightarrow$ {book} and equivalently {course} $\twoheadrightarrow$ {lecturer}.

Databases with multivalued dependencies thus exhibit redundancy. In database normalization, fourth normal form requires that either every multivalued dependency $X \twoheadrightarrow Y$ is trivial or for every nontrivial multivalued dependency $X \twoheadrightarrow Y$, X is a superkey.

Interesting properties

- If $\alpha \twoheadrightarrow \beta$, Then $\alpha \twoheadrightarrow R - \beta$
- If $\alpha \twoheadrightarrow \beta$ and $\gamma \subseteq \delta$, Then $\alpha\delta \twoheadrightarrow \beta\gamma$
- If $\alpha \twoheadrightarrow \beta$ and $\beta \twoheadrightarrow \gamma$, then $\alpha \twoheadrightarrow \gamma - \beta$

The following also involve functional dependencies:

- If $\alpha \rightarrow \beta$, then $\alpha \twoheadrightarrow \beta$
- If $\alpha \twoheadrightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \twoheadrightarrow \gamma - \beta$

The above rules are sound and complete.

- A decomposition of R into (X, Y) and (X, $R - Y$) is a lossless-join decomposition if and only if $X \twoheadrightarrow Y$ holds in R.

Definitions

full constraint

> A constraint which expresses something about *all* attributes in a database. (In contrast to an **embedded constraint**.) That a multivalued dependency is a *full constraint* follows from its definition,as where it says something about the attributes $R - \beta$.

tuple-generating dependency

> A dependency which explicitly requires certain tuples to be present in the relation.

trivial multivalued dependency 1

A multivalued dependency which involves all the attributes of a relation i.e. $R = \alpha \cup \beta$. A trivial multivalued dependency implies, for tuples t_1 and t_2, tuples t_3 and t_4 which are equal to t_1 and t_2.

trivial multivalued dependency 2

A multivalued dependency for which $\beta \subseteq \alpha$.

References

External links

- Multivalued dependencies and a new Normal form for Relational Databases (http://www.almaden.ibm.com/cs/people/fagin/tods77.pdf) (PDF) - Ronald Fagin, IBM Research Lab

Fourth normal form

Fourth normal form (**4NF**) is a normal form used in database normalization. Introduced by Ronald Fagin in 1977, 4NF is the next level of normalization after Boyce–Codd normal form (BCNF). Whereas the second, third, and Boyce–Codd normal forms are concerned with functional dependencies, 4NF is concerned with a more general type of dependency known as a multivalued dependency. A Table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies $X \twoheadrightarrow Y$, X is a superkey—that is, X is either a candidate key or a superset thereof.[1]

Multivalued dependencies

If the column headings in a relational database table are divided into three disjoint groupings X, Y, and Z, then, in the context of a particular row, we can refer to the data beneath each group of headings as x, y, and z respectively. A multivalued dependency $X \twoheadrightarrow Y$ signifies that if we choose any x actually occurring in the table (call this choice x_c), and compile a list of all the $x_c yz$ combinations that occur in the table, we will find that x_c is associated with the same y entries regardless of z. So essentially the presence of z provides no useful information to constrain the possible values of y.

A **trivial multivalued dependency** $X \twoheadrightarrow Y$ is one where either Y is a subset of X, or X and Y together form the whole set of attributes of the relation.

A functional dependency is a special case of multivalued dependency. In a functional dependency $X \to Y$, every x determines *exactly one* y, never more than one.

Example

Consider the following example:

Pizza Delivery Permutations

Restaurant	Pizza Variety	Delivery Area
A1 Pizza	Thick Crust	Springfield
A1 Pizza	Thick Crust	Shelbyville
A1 Pizza	Thick Crust	Capital City
A1 Pizza	Stuffed Crust	Springfield
A1 Pizza	Stuffed Crust	Shelbyville
A1 Pizza	Stuffed Crust	Capital City
Elite Pizza	Thin Crust	Capital City
Elite Pizza	Stuffed Crust	Capital City
Vincenzo's Pizza	Thick Crust	Springfield
Vincenzo's Pizza	Thick Crust	Shelbyville
Vincenzo's Pizza	Thin Crust	Springfield
Vincenzo's Pizza	Thin Crust	Shelbyville

Each row indicates that a given restaurant can deliver a given variety of pizza to a given area.

The table has no non-key attributes because its only key is {Restaurant, Pizza Variety, Delivery Area}. Therefore it meets all normal forms up to BCNF. If we assume, however, that pizza varieties offered by a restaurant are not affected by delivery area, then it does not meet 4NF. The problem is that the table features two non-trivial multivalued dependencies on the {Restaurant} attribute (which is not a superkey). The dependencies are:

- {Restaurant} $\twoheadrightarrow$ {Pizza Variety}
- {Restaurant} $\twoheadrightarrow$ {Delivery Area}

These non-trivial multivalued dependencies on a non-superkey reflect the fact that the varieties of pizza a restaurant offers are independent from the areas to which the restaurant delivers. This state of affairs leads to redundancy in the table: for example, we are told three times that A1 Pizza offers Stuffed Crust, and if A1 Pizza starts producing Cheese Crust pizzas then we will need to add multiple rows, one for each of A1 Pizza's delivery areas. There is, moreover, nothing to prevent us from doing this incorrectly: we might add Cheese Crust rows for all but one of A1 Pizza's delivery areas, thereby failing to respect the multivalued dependency {Restaurant} $\twoheadrightarrow$ {Pizza Variety}.

To eliminate the possibility of these anomalies, we must place the facts about varieties offered into a different table from the facts about delivery areas, yielding two tables that are both in 4NF:

Varieties By Restaurant

Restaurant	Pizza Variety
A1 Pizza	Thick Crust
A1 Pizza	Stuffed Crust
Elite Pizza	Thin Crust
Elite Pizza	Stuffed Crust
Vincenzo's Pizza	Thick Crust
Vincenzo's Pizza	Thin Crust

Delivery Areas By Restaurant

Restaurant	Delivery Area
A1 Pizza	Springfield
A1 Pizza	Shelbyville
A1 Pizza	Capital City
Elite Pizza	Capital City
Vincenzo's Pizza	Springfield
Vincenzo's Pizza	Shelbyville

In contrast, if the pizza varieties offered by a restaurant sometimes did legitimately vary from one delivery area to another, the original three-column table would satisfy 4NF.

Ronald Fagin demonstrated that it is always possible to achieve 4NF.[2] Rissanen's theorem is also applicable on multivalued dependencies.

4NF in practice

A 1992 paper by Margaret S. Wu notes that the teaching of database normalization typically stops short of 4NF, perhaps because of a belief that tables violating 4NF (but meeting all lower normal forms) are rarely encountered in business applications. This belief may not be accurate, however. Wu reports that in a study of forty organizational databases, over 20% contained one or more tables that violated 4NF while meeting all lower normal forms.

References

[1] "A relation schema R* is in fourth normal form (4NF) if, whenever a nontrivial multivalued dependency $X \twoheadrightarrow Y$ holds for R*, then so does the functional dependency $X \rightarrow A$ for every column name A of R*. Intuitively all dependencies are the result of keys."

[2] Fagin, p. 268

Further reading

- Date, C. J. (1999), *An Introduction to Database Systems* (http://www.aw-bc.com/catalog/academic/product/0,1144,0321197844,00.html) (8th ed.). Addison-Wesley Longman. ISBN 0-321-19784-4.
- Kent, W. (1983) *A Simple Guide to Five Normal Forms in Relational Database Theory* (http://www.bkent.net/Doc/simple5.htm), Communications of the ACM, vol. 26, pp. 120–125
- Advanced Normalization (http://www.utexas.edu/its/windows/database/datamodeling/rm/rm8.html) by ITS, University of Texas.

Fifth normal form

Fifth normal form (**5NF**), also known as **project-join normal form** (**PJ/NF**) is a level of database normalization designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships. A table is said to be in the 5NF if and only if every non-trivial join dependency in it is implied by the candidate keys.

A join dependency *{A, B, … Z} on R is implied by the candidate key(s) of R if and only if each of A, B, …, Z is a superkey for R.[1]

Example

Consider the following example:

Traveling Salesman Product Availability By Brand

Traveling Salesman	Brand	Product Type
Jack Schneider	Acme	Vacuum Cleaner
Jack Schneider	Acme	Breadbox
Willy Loman	Robusto	Pruning Shears
Willy Loman	Robusto	Vacuum Cleaner
Willy Loman	Robusto	Breadbox
Willy Loman	Robusto	Umbrella Stand
Louis Ferguson	Robusto	Vacuum Cleaner
Louis Ferguson	Robusto	Telescope
Louis Ferguson	Acme	Vacuum Cleaner
Louis Ferguson	Acme	Lava Lamp
Louis Ferguson	Nimbus	Tie Rack

The table's predicate is: Products of the type designated by *Product Type*, made by the brand designated by *Brand*, are available from the traveling salesman designated by *Traveling Salesman*.

In the absence of any rules restricting the valid possible combinations of Traveling Salesman, Brand, and Product Type, the three-attribute table above is necessary in order to model the situation correctly.

Suppose, however, that the following rule applies: *A Traveling Salesman has certain Brands and certain Product Types in his repertoire. If Brand B1 and Brand B2 are in his repertoire, and Product Type P is in his repertoire, then (assuming Brand B1 and Brand B2 both make Product Type P), the Traveling Salesman must offer products of Product Type P those made by Brand B1 and those made by Brand B2.*

In that case, it is possible to split the table into three:

Product Types By Traveling Salesman

Traveling Salesman	Product Type
Jack Schneider	Vacuum Cleaner
Jack Schneider	Breadbox
Willy Loman	Pruning Shears
Willy Loman	Vacuum Cleaner
Willy Loman	Breadbox
Willy Loman	Umbrella Stand
Louis Ferguson	Telescope
Louis Ferguson	Vacuum Cleaner
Louis Ferguson	Lava Lamp
Louis Ferguson	Tie Rack

Brands By Traveling Salesman

Traveling Salesman	Brand
Jack Schneider	Acme
Willy Loman	Robusto
Louis Ferguson	Robusto
Louis Ferguson	Acme
Louis Ferguson	Nimbus

Product Types By Brand

Brand	Product Type
Acme	Vacuum Cleaner
Acme	Breadbox
Acme	Lava Lamp
Robusto	Pruning Shears
Robusto	Vacuum Cleaner
Robusto	Breadbox
Robusto	Umbrella Stand
Robusto	Telescope
Nimbus	Tie Rack

In this case, it's impossible for Louis Ferguson to refuse to offer Vacuum Cleaners made by ACME (assuming ACME makes Vacuum Cleaners) if he sells anything else made by Acme (Lava Lamp) and he also sells Vacuum Cleaners made by any other brand (Robusto).

Note how this setup helps to remove redundancy. Suppose that Jack Schneider starts selling Robusto's products Breadboxes and Vacuum Cleaners. In the previous setup we would have to add two new entries one for each product type (<Jack Schneider, Robusto, Breadboxes>, <Jack Schneider, Robusto, Vacuum Cleaners>). With the new setup we need to add only a single entry (<Jack Schneider, Robusto>)in Brands By Traveling Salesman.

Usage

Only in rare situations does a 4NF table not conform to 5NF. These are situations in which a complex real-world constraint governing the valid combinations of attribute values in the 4NF table is not implicit in the structure of that table. If such a table is not normalized to 5NF, the burden of maintaining the logical consistency of the data within the table must be carried partly by the application responsible for insertions, deletions, and updates to it; and there is a heightened risk that the data within the table will become inconsistent. In contrast, the 5NF design excludes the possibility of such inconsistencies.

References

[1] Analysis of normal forms for anchor-tables (http://www.anchormodeling.com/wp-content/uploads/2010/08/6nf.pdf)

Further reading

- Kent, W. (1983) *A Simple Guide to Five Normal Forms in Relational Database Theory* (http://www.bkent.net/Doc/simple5.htm), Communications of the ACM, vol. 26, pp. 120–125
- Date, C.J., & Darwen, H., & Pascal, F. *Database Debunkings* (http://www.dbdebunk.com)
- Darwen, H.; Date, C. J.; Fagin, R. (2012). "A normal form for preventing redundant tuples in relational databases" (http://www.almaden.ibm.com/cs/people/fagin/icdt12.pdf). *Proceedings of the 15th International Conference on Database Theory - ICDT '12*. pp. 114–126. doi: 10.1145/2274576.2274589 (http://dx.doi.org/10.1145/2274576.2274589). ISBN 9781450307918.

Sixth normal form

Sixth normal form (**6NF**) is a term in relational database theory, used in two different ways.

6NF (C. Date's definition)

A book by Christopher J. Date and others on temporal databases,[1] defined sixth normal form as a normal form for databases based on an extension of the relational algebra.

In this work, the relational operators, such as *join*, are generalized to support a natural treatment of interval data, such as sequences of dates or moments in time.[2] Sixth normal form is then based on this generalized join, as follows:

> A relvar R [table] is in **sixth normal form** (abbreviated 6NF) if and only if it satisfies no nontrivial join dependencies at all — where, as before, a join dependency is trivial if and only if at least one of the projections (possibly U_projections) involved is taken over the set of all attributes of the relvar [table] concerned.[Date et al.][3]

Any relation in 6NF is also in 5NF.

Sixth normal form is intended to decompose relation variables to irreducible components. Though this may be relatively unimportant for non-temporal relation variables, it can be important when dealing with temporal variables or other interval data. For instance, if a relation comprises a supplier's name, status, and city, we may also want to add temporal data, such as the time during which these values are, or were, valid (e.g., for historical data) but the three values may vary independently of each other and at different rates. We may, for instance, wish to trace the history of changes to Status.

For further discussion on Temporal Aggregation in SQL, see also Zimanyi.[4] For a different approach, see TSQL2.[5]

DKNF

Some authors use the term **sixth normal form** differently, namely, as a synonym for Domain/key normal form (DKNF). This usage predates Date et al.'s work.[6]

Usage

The sixth normal form is currently being used in some data warehouses where the benefits outweigh the drawbacks,[7] for example using Anchor Modeling. Although using 6NF leads to an explosion of tables, modern databases can prune the tables from select queries (using a process called 'table elimination') where they are not required and thus speed up queries that only access several attributes.

References

[1] Date et al., 2003

[2] op. cit., chapter 9: *Generalizing the relational operators*

[3] op. cit., section 10.4, p. 176

[4] Zimanyi 2005

[5] Snodgrass, Richard T. TSQL2 Temporal Query Language (http://www.cs.arizona.edu/~rts/tsql2.html). Describes history, gives references to standard and original book.

[6] See www.dbdebunk.com for a discussion on this topic (http://www.dbdebunk.com/page/page/621935.htm)

[7] See the Anchor Modeling website (http://www.anchormodeling.com) for a website that describes a data warehouse modelling method based on the sixth normal form

Further reading

- Date, C.J. (2006). *The relational database dictionary: a comprehensive glossary of relational terms and concepts, with illustrative examples*. O'Reilly Series Pocket references. O'Reilly Media, Inc. p. 90. ISBN 978-0-596-52798-3.
- Date, Chris J.; Hugh Darwen, Nikos A. Lorentzos (January 2003). *Temporal Data and the Relational Model: A Detailed Investigation into the Application of Interval and Relation Theory to the Problem of Temporal Database Management*. Oxford: Elsevier LTD. ISBN 1-55860-855-9.
- Zimanyi,, E. (June 2006). "Temporal Aggregates and Temporal Universal Quantification in Standard SQL" (http://www.sigmod.org/publications/sigmod-record/0606/sigmod-record.june2006.pdf) (PDF). *ACM SIGMOD Record, volume 35, number 2, page 16*. ACM.

Denormalization

In computing, **denormalization** is the process of attempting to optimize the read performance of a database by adding redundant data or by grouping data.[1][2] In some cases, denormalization is a means of addressing performance or scalability in relational database software.

A normalized design will often store different but related pieces of information in separate logical tables (called relations). If these relations are stored physically as separate disk files, completing a database query that draws information from several relations (a *join operation*) can be slow. If many relations are joined, it may be prohibitively slow. There are two strategies for dealing with this. The preferred method is to keep the logical design normalized, but allow the database management system (DBMS) to store additional redundant information on disk to optimise query response. In this case it is the DBMS software's responsibility to ensure that any redundant copies are kept consistent. This method is often implemented in SQL as indexed views (Microsoft SQL Server) or materialised views (Oracle). A view represents information in a format convenient for querying, and the index ensures that queries against the view are optimised.

The more usual approach is to denormalize the logical data design. With care this can achieve a similar improvement in query response, but at a cost—it is now the database designer's responsibility to ensure that the denormalized database does not become inconsistent. This is done by creating rules in the database called *constraints*, that specify how the redundant copies of information must be kept synchronised. It is the increase in logical complexity of the database design and the added complexity of the additional constraints that make this approach hazardous. Moreover, constraints introduce a trade-off, speeding up reads (`SELECT` in SQL) while slowing down writes (`INSERT`, `UPDATE`, and `DELETE`). This means a denormalized database under heavy write load may actually offer *worse* performance than its functionally equivalent normalized counterpart.

A denormalized data model is not the same as a data model that has not been normalized, and denormalization should only take place after a satisfactory level of normalization has taken place and that any required constraints and/or rules have been created to deal with the inherent anomalies in the design. For example, all the relations are in third normal form and any relations with join and multi-valued dependencies are handled appropriately.

Examples of denormalization techniques include:

- Materialised views, which may implement the following:
  - Storing the count of the "many" objects in a one-to-many relationship as an attribute of the "one" relation
  - Adding attributes to a relation from another relation with which it will be joined
- Star schemas, which are also known as fact-dimension models and have been extended to snowflake schemas
- Prebuilt summarisation or OLAP cubes

Denormalization techniques are often used to improve the scalability of Web applications.[3]

References

[1] G. L. Sanders and S. K. Shin. Denormalization effects on performance of RDBMS (http://www.hicss.hawaii.edu/HICSS_34/PDFs/DTDMK04.pdf). In Proceedings of the HICSS Conference, January 2001.

[2] S. K. Shin and G. L. Sanders. Denormalization strategies for data retrieval from data warehouses (http://portal.acm.org/citation.cfm?id=1217757). Decision Support Systems, 42(1):267-282, October 2006.

[3] Z. Wei, J. Dejun, G. Pierre, C.-H. Chi and M. van Steen. Service-Oriented Data Denormalization for Scalable Web Applications (http://www.globule.org/publi/SODDSWA_www2008.html). In Proceedings of the International World-Wide Web conference, April 2008.

Domain/key normal form

Domain/key normal form (**DKNF**) is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

A domain constraint specifies the permissible values for a given attribute, while a key constraint specifies the attributes that uniquely identify a row in a given table.

The domain/key normal form is achieved when every constraint on the relation is a logical consequence of the definition of keys and domains, and enforcing key and domain restraints and conditions causes all constraints to be met. Thus, it avoids all non-temporal anomalies.

The reason to use domain/key normal form is to avoid having general constraints in the database that are not clear domain or key constraints. Most databases can easily test domain and key constraints on attributes. General constraints however would normally require special database programming in the form of stored procedures that are expensive to maintain and expensive for the database to execute. Therefore general constraints are split into domain and key constraints.

It's much easier to build a database in domain/key normal form than it is to convert lesser databases which may contain numerous anomalies. However, successfully building a domain/key normal form database remains a difficult task, even for experienced database programmers. Thus, while the domain/key normal form eliminates the problems found in most databases, it tends to be the most costly normal form to achieve. However, failing to achieve the domain/key normal form may carry long-term, hidden costs due to anomalies which appear in databases adhering only to lower normal forms over time.

The third normal form, Boyce–Codd normal form, fourth normal form and fifth normal form are special cases of the domain/key normal form. All have either functional, multi-valued or join dependencies that can be converted into (super)keys. The domains on those normal forms were unconstrained so all domain constraints are satisfied. However, transforming a higher normal form into domain/key normal form is not always a dependency-preserving transformation and therefore not always possible.

Example

A violation of DKNF occurs in the following table:

Wealthy Person

Wealthy Person	Wealthy Person Type	Net Worth in Dollars
Steve	Eccentric Millionaire	124,543,621
Roderick	Evil Billionaire	6,553,228,893
Katrina	Eccentric Billionaire	8,829,462,998
Gary	Evil Millionaire	495,565,211

(Assume that the domain for Wealthy Person consists of the names of all wealthy people in a pre-defined sample of wealthy people; the domain for Wealthy Person Type consists of the values 'Eccentric Millionaire', 'Eccentric Billionaire', 'Evil Millionaire', and 'Evil Billionaire'; and the domain for Net Worth in Dollars consists of all integers greater than or equal to 1,000,000.)

There is a constraint linking Wealthy Person Type to Net Worth in Dollars, even though we cannot deduce one from the other. The constraint dictates that an Eccentric Millionaire or Evil Millionaire will have a net worth of 1,000,000 to 999,999,999 inclusive, while an Eccentric Billionaire or Evil Billionaire will have a net worth of 1,000,000,000 or higher. This constraint is neither a domain constraint nor a key constraint; therefore we cannot rely on domain

constraints and key constraints to guarantee that an inconsistent Wealthy Person Type / Net Worth in Dollars combination does not make its way into the database.

The DKNF violation could be eliminated by altering the Wealthy Person Type domain to make it consist of just two values, 'Evil' and 'Eccentric' (the wealthy person's status as a millionaire or billionaire is implicit in their Net Worth in Dollars, so no useful information is lost).

Wealthy Person

Wealthy Person	Wealthy Person Type	Net Worth in Dollars
Steve	Eccentric	124,543,621
Roderick	Evil	6,553,228,893
Katrina	Eccentric	8,829,462,998
Gary	Evil	495,565,211

Wealthiness Status

Status	Minimum	Maximum
Millionaire	1,000,000	999,999,999
Billionaire	1,000,000,000	999,999,999,999

DKNF is frequently difficult to achieve in practice.[*citation needed*]

References

- Fagin, Ronald (1981). "A Normal Form for Relational Databases That Is Based on Domains and Keys" [1]. *ACM Transactions on Database Systems* **6**: 387–415. doi:10.1145/319587.319592 [2].

[1] http://www.almaden.ibm.com/cs/people/fagin/tods81.pdf
[2] http://dx.doi.org/10.1145%2F319587.319592

External links

- Database Normalization Basics (http://databases.about.com/od/specificproducts/a/normalization.htm) by Mike Chapple (About.com)
- An Introduction to Database Normalization (http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html) by Mike Hillyer.
- Normalization (http://www.utexas.edu/its-archive/windows/database/datamodeling/rm/rm7.html) by ITS, University of Texas.
- A tutorial on the first 3 normal forms (http://phlonx.com/resources/nf3/) by Fred Coulson
- Description of the database normalization basics (http://support.microsoft.com/kb/283878) by Microsoft

Single Source of Truth

In Information Systems design and theory, as instantiated at the *Enterprise Level*, **Single Source Of Truth** (SSOT) refers to the practice of structuring information models and associated schemata such that every data element is stored exactly once (e.g., in no more than a single row of a single table). Any possible linkages to this data element (possibly in other areas of the relational schema or even in distant federated databases) are by reference only. Thus, when any such data element is updated, this update propagates to the enterprise at large, without the possibility of a duplicate value somewhere in the distant enterprise not being updated (because there would be no duplicate values that *needed* updating).[*citation needed*]

Deployment of an SSOT architecture is becoming increasingly important in enterprise settings where incorrectly linked duplicate or de-normalized data elements (a direct consequence of intentional or unintentional denormalization of any explicit data model) poses a risk for retrieval of outdated, and therefore incorrect, information. A common example would be the electronic health record, where it is imperative to accurately validate patient identity against a single referential repository, which serves as the SSOT. Duplicate representations of data within the enterprise would be implemented by the use of pointers rather than duplicate database tables, rows, or cells. This ensures that data updates to elements in the authoritative location are comprehensively distributed to all federated database constituencies in the larger overall enterprise architecture.[*citation needed*]

SSOT systems provide data that is authentic, relevant, and referable.Wikipedia:Please clarify[*citation needed*]

Implementing a Single Source of Truth

The "ideal" implementation of SSOT as described above is rarely possible in most enterprises. This is because many organisations have multiple information systems, each of which needs access to data relating to the same entities (e.g., customer). Often these systems are purchased "off-the-shelf" from vendors and cannot be modified in non-trivial ways. Each of these various systems therefore needs to store its own version of common data or entities, and therefore each system must retain its own copy of a record (hence immediately violating the SSOT approach defined above). For example, an ERP (Enterprise Resource Planning) system (such as SAP or Oracle e-Business Suite) may store a customer record; the CRM (Customer Relationship Management) system also needs a copy of the customer record (or part of it) and the warehouse despatch system might also need a copy of some or all of the customer data (e.g., shipping address). In cases where vendors do not support such modifications, it is not always possible to replace these records with pointers to the SSOT.

For organisations (with more than one information system) wishing to implement a Single Source of Truth (without modifying all but one master system to store pointers to other systems for all entities), three supporting technologies are commonly used:[*citation needed*]

- Enterprise Service Bus (ESB)
- Master Data Management (MDM)
- Data Warehouse (DW)

Enterprise Service Bus (ESB)

An Enterprise Service Bus (ESB) allows any number of systems in an organisation to receive updates of data that has changed in another system. To implement a Single Source of Truth, a single source system of correct data for any entity must be identified. Changes to this entity (creates, updates, and deletes) are then published via the ESB; other systems which need to retain a copy of that data subscribe to this update, and update their own records accordingly. For any given entity, the master source must be identified (sometimes called the Golden Record). It should be noted that any given system could publish (be the source of truth for) information on a particular entity (e.g., customer) and also subscribe to updates from another system for information on some other entity (e.g., product).[*citation needed*]

An alternative approach is point-to-point data updates, but these become exponentially more expensive to maintain as the number of systems increases, and this approach is increasingly out of favour as an IT architecture.[*citation needed*]

Master Data Management (MDM)

An MDM system can act as the source of truth for any given entity that might not necessarily have an alternative "source of truth" in another system. Typically the MDM acts as a hub for multiple systems, many of which could allow (be the source of truth for) updates to different aspects of information on a given entity. For example, the CRM system may be the "source of truth" for most aspects of the customer, and is updated by a call centre operator. However, a customer may (for example) also update their address via a customer service web site, with a different back-end database from the CRM system. The MDM application receives updates from multiple sources, acts as a broker to determine which updates are to be regarded as authoritative (the Golden Record) and then syndicates this updated data to all subscribing systems. The MDM application normally requires an ESB to syndicate its data to multiple subscribing systems.[*citation needed*]

Customer Data Integration (CDI), as a common application of Master Data Management, is sometimes abbreviated CDI-MDM.[*citation needed*]

Data Warehouse (DW)

While the primary purpose of a data warehouse is to support reporting and analysis of data that has been combined from multiple sources, the fact that such data has been combined (according to business logic embedded in the data transformation and integration processes) means that the data warehouse is often used as a *de facto* SSOT. Generally, however, the data available from the data warehouse is not used to update other systems; rather the DW becomes the "single source of truth" for reporting to multiple stakeholders. In this context, the Data Warehouse is more correctly referred to as a "Single Version of the Truth" since other versions of the truth exist in its operational data sources (no data originates in the DW; it is simply a reporting mechanism for data loaded from operational systems).[*citation needed*]

References

Single version of the truth

In computerized business management, **SVOT**, or **Single Version of the Truth**, is a technical concept describing the data warehousing ideal of having either a single centralised database, or at least a distributed synchronised database, which stores all of an organisation's data in a consistent and non-redundant form. This contrasts with the related concept of Single Source of Truth (SSOT), which refers to is a data storage principle to always source a particular piece of information from one place.[*citation needed*]

SVOT applied to message sequencing

In some systems and in the context of message processing systems (often realtime systems), this term also refers to the goal of establishing a single agreed sequence of messages within a database formed by a particular but arbitrary sequencing of records. The key concept is that data combined in a certain sequence is a "truth" which may be analyzed and processed giving particular results, and that although the sequence is arbitrary (and thus another correct but equally arbitrary sequencing would ultimately provide different results in any analysis), it is desirable to agree that the sequence enshrined in the "single version of the truth" is the version that will be considered "the truth", and that any conclusions drawn from analysis of the database are valid and unarguable, and (in a technical context) the database may be duplicated to a backup environment to ensure a persistent record is kept of the "single version of the truth".

The key point is when the database is created using an external data source (such as a sequence of trading messages from a stock exchange) an arbitrary selection is made of one possibility from two or more equally valid representations of the input data, but henceforth the decision sets "in stone" one and only one version of the truth.

Critique of SVOT as applied to message sequencing

Critics of "SVOT as applied to message sequencing" argue that this concept is not scalable. As the world moves towards systems spread over many processing nodes, the effort involved in negotiating a single agreed-upon sequence becomes prohibitive.

References

Bibliography

- Julia King (2003-12-22). "Business Intelligence: One Version of the Truth" (http://www.computerworld.com/databasetopics/data/story/0,10801,88349,00.html). *ComputerWorld*.
- Leslie Lamport (July 1978). "Time, Clocks, and the Ordering of Events in a Distributed System" (http://research.microsoft.com/en-us/um/people/lamport/pubs/time-clocks.pdf). *CACM: Operating Systems*.
- Bill Inmon (2004-09-09). "The Single Version Of The Truth" (http://www.b-eye-network.com/view/282). *Business Intelligence Network* (Powell Media LLC).
- http://www.industryweek.com/EventDetail.aspx?EventID=179
- Chisholm, Malcolm (December 2006), "There is No Single Version of the Truth" (http://www.information-management.com/issues/20061201/1069851-1.html), *Information Management Magazine*, retrieved 2010-03-15

Principle of Orthogonal Design

The **Principle of Orthogonal Design** (abbreviated POOD) was developed by database researchers David McGoveran and Christopher J. Date in the early 1990s, and first published "A New Database Design Principle" in the July 1994 issue of Database Programming and Design and reprinted several times. It is the second of the two principles of database design, which seek to prevent databases from being too complicated or redundant, the first principle being the Principle of Full Normalization (POFN).

Simply put, it says that no two relations in a relational database should be defined in such a way that they can represent the same facts. As with database normalization, POOD serves to eliminate uncontrolled storage redundancy and expressive ambiguity, especially useful for applying updates to virtual relations (e.g., view (database)). Although simple in concept, POOD is frequently misunderstood and the formal expression of POOD continues to be refined.

The principle is a restatement of the requirement that a database is a minimum cover set of the relational algebra. The relational algebra allows data duplication in the relations that are the elements of the algebra. One of the efficiency requirements of a database is that there be no data duplication. This requirement is met by the minimum cover set of the relational algebra.

Sources

- Database Debunkings: The Principle of Orthogonal Design, Part I, by D. McGoveran and C. J. Date [1]
- Database Debunkings: The Principle of Orthogonal Design, Part II, by D. McGoveran and C. J. Date [2]
- David McGoveran (personal interview) [3]

References

[1] http://www.dbdebunk.com/page/page/622331.htm
[2] http://www.dbdebunk.com/page/page/622312.htm
[3] http://www.alternativetech.com

Transaction Management

Database transaction

A **transaction** comprises a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. Transactions in a database environment have two main purposes:

1. To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.
2. To provide isolation between programs accessing a database concurrently. If this isolation is not provided, the program's outcome are possibly erroneous.

A database transaction, by definition, must be atomic, consistent, isolated and durable.[1] Database practitioners often refer to these properties of database transactions using the acronym ACID.

Transactions provide an "all-or-nothing" proposition, stating that each work-unit performed in a database must either complete in its entirety or have no effect whatsoever. Further, the system must isolate each transaction from other transactions, results must conform to existing constraints in the database, and transactions that complete successfully must get written to durable storage.

Purpose

Databases and other data stores which treat the integrity of data as paramount often include the ability to handle transactions to maintain the integrity of data. A single transaction consists of one or more independent units of work, each reading and/or writing information to a database or other data store. When this happens it is often important to ensure that all such processing leaves the database or data store in a consistent state.

Examples from double-entry accounting systems often illustrate the concept of transactions. In double-entry accounting every debit requires the recording of an associated credit. If one writes a check for $100 to buy groceries, a transactional double-entry accounting system must record the following two entries to cover the single transaction:

1. Debit $100 to Groceries Expense Account
2. Credit $100 to Checking Account

A transactional system would make both entries pass or both entries would fail. By treating the recording of multiple entries as an atomic transactional unit of work the system maintains the integrity of the data recorded. In other words, nobody ends up with a situation in which a debit is recorded but no associated credit is recorded, or vice versa.

Transactional databases

A **transactional database** is a DBMS where write transactions on the database are able to be rolled back if they are not completed properly (e.g. due to power or connectivity loss).

Most modern[2] relational database management systems fall into the category of databases that support transactions.

In a database system a transaction might consist of one or more data-manipulation statements and queries, each reading and/or writing information in the database. Users of database systems consider consistency and integrity of data as highly important. A simple transaction is usually issued to the database system in a language like SQL wrapped in a transaction, using a pattern similar to the following:

1. Begin the transaction
2. Execute a set of data manipulations and/or queries
3. If no errors occur then commit the transaction and end it
4. If errors occur then rollback the transaction and end it

If no errors occurred during the execution of the transaction then the system commits the transaction. A transaction commit operation applies all data manipulations within the scope of the transaction and persists the results to the database. If an error occurs during the transaction, or if the user specifies a rollback operation, the data manipulations within the transaction are not persisted to the database. In no case can a partial transaction be committed to the database since that would leave the database in an inconsistent state.

Internally, multi-user databases store and process transactions, often by using a transaction ID or XID.

There are multiple varying ways for transactions to be implemented other than the simple way documented above. Nested transactions, for example, are transactions which contain statements within them that start new transactions (i.e. sub-transactions). *Multi-level transactions* are a variant of nested transactions where the sub-transactions take place at different levels of a layered system architecture (e.g., with one operation at the database-engine level, one operation at the operating-system level) [2] Another type of transaction is the compensating transaction.

In SQL

Transactions are available in most SQL database implementations, though with varying levels of robustness. (MySQL, for example, does not support transactions in the MyISAM storage engine, which was its default storage engine before version 5.5.)

A transaction is typically started using the command `BEGIN` (although the SQL standard specifies `START TRANSACTION`). When the system processes a `COMMIT` statement, the transaction ends with successful completion. A `ROLLBACK` statement can also end the transaction, undoing any work performed since `BEGIN TRANSACTION`. If autocommit was disabled using `START TRANSACTION`, autocommit will also be re-enabled at the transaction's end.

One can set the isolation level for individual transactional operations as well as globally. At the READ COMMITTED level, the result of any work done after a transaction has commenced, but before it has ended, will remain invisible to other database-users until it has ended. At the lowest level (READ UNCOMMITTED), which may occasionally be used to ensure high concurrency, such changes will be visible.

Distributed transactions

Database systems implement distributed transactions as transactions against multiple applications or hosts. A distributed transaction enforces the ACID properties over multiple systems or data stores, and might include systems such as databases, file systems, messaging systems, and other applications. In a distributed transaction a coordinating service ensures that all parts of the transaction are applied to all relevant systems. As with database and other transactions, if any part of the transaction fails, the entire transaction is rolled back across all affected systems.

Transactional filesystems

The Namesys Reiser4 filesystem for Linux[3] supports transactions, and as of Microsoft Windows Vista, the Microsoft NTFS filesystem[4] supports distributed transactions across networks.

References

[1] A transaction is a group of operations that are atomic, consistent, isolated, and durable ([[ACID (http://msdn.microsoft.com/en-us/library/aa366402(VS.85).aspx)]).]

[2] Beeri, C., Bernstein, P.A., and Goodman, N. A model for concurrency in nested transactions systems. Journal of the ACM, 36(1):230-269, 1989

[3] namesys.com (http://namesys.com/v4/v4.html#committing)

[4] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/portal.asp

Further reading

- Philip A. Bernstein, Eric Newcomer (2009): *Principles of Transaction Processing*, 2nd Edition (http://www.elsevierdirect.com/product.jsp?isbn=9781558606234), Morgan Kaufmann (Elsevier), ISBN 978-1-55860-623-4
- Gerhard Weikum, Gottfried Vossen (2001), *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, ISBN 1-55860-508-8

Transaction processing

In computer science, **transaction processing** is information processing that is divided into individual, indivisible operations, called *transactions.* Each transaction must succeed or fail as a complete unit; it cannot be only partially complete.

Since most, though not necessarily all, transaction processing today is interactive the term is often treated as synonymous with *online transaction processing*.

Description

Transaction processing is designed to maintain a systems Integrity (typically a database or some modern filesystems) in a known, consistent state, by ensuring that interdependent operations on the system are either all completed successfully or all canceled successfully.

For example, consider a typical banking transaction that involves moving $700 from a customer's savings account to a customer's checking account. This transaction involves at least two separate operations in computer terms: debiting the savings account by $700, and crediting the checking account by $700. If one operation succeeds but the other does not, the books of the bank will not balance at the end of the day. There must therefore be a way to ensure that either both operations succeed or both fail, so that there is never any inconsistency in the bank's database as a whole.

Transaction processing links multiple individual operations in a single, indivisible transaction, and ensures that either all operations in a transaction are completed without error, or none of them are. If some of the operations are completed but errors occur when the others are attempted, the transaction-processing system "rolls back" *all* of the operations of the transaction (including the successful ones), thereby erasing all traces of the transaction and restoring the system to the consistent, known state that it was in before processing of the transaction began. If all operations of a transaction are completed successfully, the transaction is committed by the system, and all changes to the database are made permanent; the transaction cannot be rolled back once this is done.

Transaction processing guards against hardware and software errors that might leave a transaction partially completed. If the computer system crashes in the middle of a transaction, the transaction processing system guarantees that all operations in any uncommitted transactions are cancelled.

Generally, transactions are issued concurrently. If they overlap (i.e. need to touch the same portion of the database), this can create conflicts. For example, if the customer mentioned in the example above has $150 in his savings account and attempts to transfer $100 to a different person while at the same time moving $100 to the checking account, only one of them can succeed. However, forcing transactions to be processed sequentially is inefficient. Therefore, concurrent implementations of transaction processing is programmed to guarantee that the end result reflects a conflict-free outcome, the same as could be reached if executing the transactions sequentially in any order (a property called serializability). In our example, this means that no matter which transaction was issued first, either the transfer to a different person or the move to the checking account succeeds, while the other one fails.

Methodology

The basic principles of all transaction-processing systems are the same. However, the terminology may vary from one transaction-processing system to another, and the terms used below are not necessarily universal.

Rollback

Transaction-processing systems ensure database integrity by recording intermediate states of the database as it is modified, then using these records to restore the database to a known state if a transaction cannot be committed. For example, copies of information on the database *prior* to its modification by a transaction are set aside by the system before the transaction can make any modifications (this is sometimes called a *before image*). If any part of the transaction fails before it is committed, these copies are used to restore the database to the state it was in before the transaction began.

Rollforward

It is also possible to keep a separate journal of all modifications to a database (sometimes called *after images*). This is not required for rollback of failed transactions but it is useful for updating the database in the event of a database failure, so some transaction-processing systems provide it. If the database fails entirely, it must be restored from the most recent back-up. The back-up will not reflect transactions committed since the back-up was made. However, once the database is restored, the journal of after images can be applied to the database (*rollforward*) to bring the database up to date. Any transactions in progress at the time of the failure can then be rolled back. The result is a database in a consistent, known state that includes the results of all transactions committed up to the moment of failure.

Deadlocks

In some cases, two transactions may, in the course of their processing, attempt to access the same portion of a database at the same time, in a way that prevents them from proceeding. For example, transaction A may access portion X of the database, and transaction B may access portion Y of the database. If, at that point, transaction A then tries to access portion Y of the database while transaction B tries to access portion X, a *deadlock* occurs, and neither transaction can move forward. Transaction-processing systems are designed to detect these deadlocks when they occur. Typically both transactions will be cancelled and rolled back, and then they will be started again in a different order, automatically, so that the deadlock doesn't occur again. Or sometimes, just one of the deadlocked transactions will be cancelled, rolled back, and automatically restarted after a short delay.

Deadlocks can also occur between three or more transactions. The more transactions involved, the more difficult they are to detect, to the point that transaction processing systems find there is a practical limit to the deadlocks they can detect.

Compensating transaction

In systems where commit and rollback mechanisms are not available or undesirable, a compensating transaction is often used to undo failed transactions and restore the system to a previous state.

ACID criteria

Jim Gray defined properties of a reliable transaction system in the late 1970s under the acronym *ACID* — atomicity, consistency, isolation, and durability.

Atomicity

A transaction's changes to the state are atomic: either all happen or none happen. These changes include database changes, messages, and actions on transducers.

Consistency

Consistency: A transaction is a correct transformation of the state. The actions taken as a group do not violate any of the integrity constraints associated with the state.

Isolation

Even though transactions execute concurrently, it appears to each transaction T, that others executed either before T or after T, but not both.

Durability

Once a transaction completes successfully (commits), its changes to the state survive failures.

Benefits

Transaction processing has these benefits:

- It allows sharing of computer resources among many users
- It shifts the time of job processing to when the computing resources are less busy
- It avoids idling the computing resources without minute-by-minute human interaction and supervision
- It is used on expensive classes of computers to help amortize the cost by keeping high rates of utilization of those expensive resources

Implementations

Standard transaction-processing software, notably IBM's Information Management System, was first developed in the 1960s, and was often closely coupled to particular database management systems. Client–server computing implemented similar principles in the 1980s with mixed success. However, in more recent years, the distributed client–server model has become considerably more difficult to maintain. As the number of transactions grew in response to various online services (especially the Web), a single distributed database was not a practical solution. In addition, most online systems consist of a whole suite of programs operating together, as opposed to a strict client–server model where the single server could handle the transaction processing. Today a number of transaction processing systems are available that work at the inter-program level and which scale to large systems, including mainframes.

One well-known[*citation needed*] (and open) industry standard is the X/Open Distributed Transaction Processing (DTP) (see also JTA the Java Transaction API). However, proprietary transaction-processing environments such as IBM's CICS are still very popular,[*citation needed*] although CICS has evolved to include open industry standards as well.

The term 'Extreme Transaction Processing' (XTP) has been used to describe transaction processing systems with uncommonly challenging requirements, particularly throughput requirements (transactions per second). Such systems may be implemented via distributed or cluster style architectures.

References

External references

- Nuts and Bolts of Transaction Processing (http://www.subbu.org/articles/nuts-and-bolts-of-transaction-processing)
- Managing Transaction Processing for SQL Database Integrity (http://www.informit.com/articles/article.aspx?p=174375)

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8
- Jim Gray, Andreas Reuter, Transaction Processing — Concepts and Techniques, 1993, Morgan Kaufmann, ISBN 1-55860-190-2
- Philip A. Bernstein, Eric Newcomer, Principles of Transaction Processing, 1997, Morgan Kaufmann, ISBN 1-55860-415-4
- Ahmed K. Elmagarmid (Editor), Transaction Models for Advanced Database Applications, Morgan-Kaufmann, 1992, ISBN 1-55860-214-3

Concurrency control

In information technology and computer science, especially in the fields of computer programming, operating systems, multiprocessors, and databases, **concurrency control** ensures that correct results for concurrent operations are generated, while getting those results as quickly as possible.

Computer systems, both software and hardware, consist of modules, or components. Each component is designed to operate correctly, i.e., to obey or to meet certain consistency rules. When components that operate concurrently interact by messaging or by sharing accessed data (in memory or storage), a certain component's consistency may be violated by another component. The general area of concurrency control provides rules, methods, design methodologies, and theories to maintain the consistency of components operating concurrently while interacting, and thus the consistency and correctness of the whole system. Introducing concurrency control into a system means applying operation constraints which typically result in some performance reduction. Operation consistency and correctness should be achieved with as good as possible efficiency, without reducing performance below reasonable levels. Concurrency control can require significant additional complexity and overhead in a concurrent algorithm compared to the simpler sequential algorithm.

For example, a failure in concurrency control can result in data corruption from torn read or write operations.

Concurrency control in databases

Comments:

1. This section is applicable to all transactional systems, i.e., to all systems that use *database transactions* (*atomic transactions*; e.g., transactional objects in Systems management and in networks of smartphones which typically implement private, dedicated database systems), not only general-purpose database management systems (DBMSs).
2. DBMSs need to deal also with concurrency control issues not typical just to database transactions but rather to operating systems in general. These issues (e.g., see *Concurrency control in operating systems* below) are out of the scope of this section.

Concurrency control in Database management systems (DBMS; e.g., Bernstein et al. 1987, Weikum and Vossen 2001), other transactional objects, and related distributed applications (e.g., Grid computing and Cloud computing) ensures that *database transactions* are performed concurrently without violating the data integrity of the respective databases. Thus concurrency control is an essential element for correctness in any system where two database transactions or more, executed with time overlap, can access the same data, e.g., virtually in any general-purpose database system. Consequently a vast body of related research has been accumulated since database systems emerged in the early 1970s. A well established concurrency control theory for database systems is outlined in the references mentioned above: serializability theory, which allows to effectively design and analyze concurrency control methods and mechanisms. An alternative theory for concurrency control of atomic transactions over abstract data types is presented in (Lynch et al. 1993), and not utilized below. This theory is more refined, complex, with a wider scope, and has been less utilized in the Database literature than the classical theory above. Each theory has its pros and cons, emphasis and insight. To some extent they are complementary, and their merging may be useful.

To ensure correctness, a DBMS usually guarantees that only *serializable* transaction schedules are generated, unless *serializability* is intentionally relaxed to increase performance, but only in cases where application correctness is not harmed. For maintaining correctness in cases of failed (aborted) transactions (which can always happen for many reasons) schedules also need to have the *recoverability* (from abort) property. A DBMS also guarantees that no effect of *committed* transactions is lost, and no effect of *aborted* (rolled back) transactions remains in the related database. Overall transaction characterization is usually summarized by the ACID rules below. As databases have become distributed, or needed to cooperate in distributed environments (e.g., Federated databases in the early 1990, and Cloud computing currently), the effective distribution of concurrency control mechanisms has received special attention.

Database transaction and the ACID rules

The concept of a *database transaction* (or *atomic transaction*) has evolved in order to enable both a well understood database system behavior in a faulty environment where crashes can happen any time, and *recovery* from a crash to a well understood database state. A database transaction is a unit of work, typically encapsulating a number of operations over a database (e.g., reading a database object, writing, acquiring lock, etc.), an abstraction supported in database and also other systems. Each transaction has well defined boundaries in terms of which program/code executions are included in that transaction (determined by the transaction's programmer via special transaction commands). Every database transaction obeys the following rules (by support in the database system; i.e., a database system is designed to guarantee them for the transactions it runs):

- **Atomicity** - Either the effects of all or none of its operations remain ("all or nothing" semantics) when a transaction is completed (*committed* or *aborted* respectively). In other words, to the outside world a committed transaction appears (by its effects on the database) to be indivisible, atomic, and an aborted transaction does not leave effects on the database at all, as if never existed.
- **Consistency** - Every transaction must leave the database in a consistent (correct) state, i.e., maintain the predetermined integrity rules of the database (constraints upon and among the database's objects). A transaction

must transform a database from one consistent state to another consistent state (however, it is the responsibility of the transaction's programmer to make sure that the transaction itself is correct, i.e., performs correctly what it intends to perform (from the application's point of view) while the predefined integrity rules are enforced by the DBMS). Thus since a database can be normally changed only by transactions, all the database's states are consistent. An aborted transaction does not change the database state it has started from, as if it never existed (atomicity above).

- **Isolation** - Transactions cannot interfere with each other (as an end result of their executions). Moreover, usually (depending on concurrency control method) the effects of an incomplete transaction are not even visible to another transaction. Providing isolation is the main goal of concurrency control.
- **Durability** - Effects of successful (committed) transactions must persist through crashes (typically by recording the transaction's effects and its commit event in a non-volatile memory).

The concept of atomic transaction has been extended during the years to what has become Business transactions which actually implement types of Workflow and are not atomic. However also such enhanced transactions typically utilize atomic transactions as components.

Why is concurrency control needed?

If transactions are executed *serially*, i.e., sequentially with no overlap in time, no transaction concurrency exists. However, if concurrent transactions with interleaving operations are allowed in an uncontrolled manner, some unexpected, undesirable result may occur. Here are some typical examples:

1. The lost update problem: A second transaction writes a second value of a data-item (datum) on top of a first value written by a first concurrent transaction, and the first value is lost to other transactions running concurrently which need, by their precedence, to read the first value. The transactions that have read the wrong value end with incorrect results.
2. The dirty read problem: Transactions read a value written by a transaction that has been later aborted. This value disappears from the database upon abort, and should not have been read by any transaction ("dirty read"). The reading transactions end with incorrect results.
3. The incorrect summary problem: While one transaction takes a summary over the values of all the instances of a repeated data-item, a second transaction updates some instances of that data-item. The resulting summary does not reflect a correct result for any (usually needed for correctness) precedence order between the two transactions (if one is executed before the other), but rather some random result, depending on the timing of the updates, and whether certain update results have been included in the summary or not.

Most high-performance transactional systems need to run transactions concurrently to meet their performance requirements. Thus, without concurrency control such systems can neither provide correct results nor maintain their databases consistent.

Concurrency control mechanisms

Categories

The main categories of concurrency control mechanisms are:

- **Optimistic** - Delay the checking of whether a transaction meets the isolation and other integrity rules (e.g., serializability and recoverability) until its end, without blocking any of its (read, write) operations ("...and be optimistic about the rules being met..."), and then abort a transaction to prevent the violation, if the desired rules are to be violated upon its commit. An aborted transaction is immediately restarted and re-executed, which incurs an obvious overhead (versus executing it to the end only once). If not too many transactions are aborted, then being optimistic is usually a good strategy.

- **Pessimistic** - Block an operation of a transaction, if it may cause violation of the rules, until the possibility of violation disappears. Blocking operations is typically involved with performance reduction.
- **Semi-optimistic** - Block operations in some situations, if they may cause violation of some rules, and do not block in other situations while delaying rules checking (if needed) to transaction's end, as done with optimistic.

Different categories provide different performance, i.e., different average transaction completion rates (*throughput*), depending on transaction types mix, computing level of parallelism, and other factors. If selection and knowledge about trade-offs are available, then category and method should be chosen to provide the highest performance.

The mutual blocking between two transactions (where each one blocks the other) or more results in a deadlock, where the transactions involved are stalled and cannot reach completion. Most non-optimistic mechanisms (with blocking) are prone to deadlocks which are resolved by an intentional abort of a stalled transaction (which releases the other transactions in that deadlock), and its immediate restart and re-execution. The likelihood of a deadlock is typically low.

Blocking, deadlocks, and aborts all result in performance reduction, and hence the trade-offs between the categories.

Methods

Many methods for concurrency control exist. Most of them can be implemented within either main category above. The major methods,[1] which have each many variants, and in some cases may overlap or be combined, are:

1. Locking (e.g., **Two-phase locking** - 2PL) - Controlling access to data by locks assigned to the data. Access of a transaction to a data item (database object) locked by another transaction may be blocked (depending on lock type and access operation type) until lock release.
2. **Serialization graph checking** (also called Serializability, or Conflict, or Precedence graph checking) - Checking for cycles in the schedule's graph and breaking them by aborts.
3. **Timestamp ordering** (TO) - Assigning timestamps to transactions, and controlling or checking access to data by timestamp order.
4. **Commitment ordering** (or Commit ordering; CO) - Controlling or checking transactions' chronological order of commit events to be compatible with their respective precedence order.

Other major concurrency control types that are utilized in conjunction with the methods above include:

- **Multiversion concurrency control** (MVCC) - Increasing concurrency and performance by generating a new version of a database object each time the object is written, and allowing transactions' read operations of several last relevant versions (of each object) depending on scheduling method.
- **Index concurrency control** - Synchronizing access operations to indexes, rather than to user data. Specialized methods provide substantial performance gains.
- **Private workspace model** (**Deferred update**) - Each transaction maintains a private workspace for its accessed data, and its changed data become visible outside the transaction only upon its commit (e.g., Weikum and Vossen 2001). This model provides a different concurrency control behavior with benefits in many cases.

The most common mechanism type in database systems since their early days in the 1970s has been *Strong strict Two-phase locking* (SS2PL; also called *Rigorous scheduling* or *Rigorous 2PL*) which is a special case (variant) of both Two-phase locking (2PL) and Commitment ordering (CO). It is pessimistic. In spite of its long name (for historical reasons) the idea of the **SS2PL** mechanism is simple: "Release all locks applied by a transaction only after the transaction has ended." SS2PL (or Rigorousness) is also the name of the set of all schedules that can be generated by this mechanism, i.e., these are SS2PL (or Rigorous) schedules, have the SS2PL (or Rigorousness) property.

Major goals of concurrency control mechanisms

Concurrency control mechanisms firstly need to operate correctly, i.e., to maintain each transaction's integrity rules (as related to concurrency; application-specific integrity rule are out of the scope here) while transactions are running concurrently, and thus the integrity of the entire transactional system. Correctness needs to be achieved with as good performance as possible. In addition, increasingly a need exists to operate effectively while transactions are distributed over processes, computers, and computer networks. Other subjects that may affect concurrency control are recovery and replication.

Correctness

Serializability

For correctness, a common major goal of most concurrency control mechanisms is generating schedules with the *Serializability* property. Without serializability undesirable phenomena may occur, e.g., money may disappear from accounts, or be generated from nowhere. **Serializability** of a schedule means equivalence (in the resulting database values) to some *serial* schedule with the same transactions (i.e., in which transactions are sequential with no overlap in time, and thus completely isolated from each other: No concurrent access by any two transactions to the same data is possible). Serializability is considered the highest level of isolation among database transactions, and the major correctness criterion for concurrent transactions. In some cases compromised, relaxed forms of serializability are allowed for better performance (e.g., the popular *Snapshot isolation* mechanism) or to meet availability requirements in highly distributed systems (see *Eventual consistency*), but only if application's correctness is not violated by the relaxation (e.g., no relaxation is allowed for money transactions, since by relaxation money can disappear, or appear from nowhere).

Almost all implemented concurrency control mechanisms achieve serializability by providing *Conflict serializablity*, a broad special case of serializability (i.e., it covers, enables most serializable schedules, and does not impose significant additional delay-causing constraints) which can be implemented efficiently.

Recoverability

See *Recoverability* in *Serializability*

Comment: While in the general area of systems the term "recoverability" may refer to the ability of a system to recover from failure or from an incorrect/forbidden state, within concurrency control of database systems this term has received a specific meaning.

Concurrency control typically also ensures the *Recoverability* property of schedules for maintaining correctness in cases of aborted transactions (which can always happen for many reasons). **Recoverability** (from abort) means that no committed transaction in a schedule has read data written by an aborted transaction. Such data disappear from the database (upon the abort) and are parts of an incorrect database state. Reading such data violates the consistency rule of ACID. Unlike Serializability, Recoverability cannot be compromised, relaxed at any case, since any relaxation results in quick database integrity violation upon aborts. The major methods listed above provide serializability mechanisms. None of them in its general form automatically provides recoverability, and special considerations and mechanism enhancements are needed to support recoverability. A commonly utilized special case of recoverability is *Strictness*, which allows efficient database recovery from failure (but excludes optimistic implementations; e.g., Strict CO (SCO) cannot have an optimistic implementation, but has semi-optimistic ones).

Comment: Note that the *Recoverability* property is needed even if no database failure occurs and no database *recovery* from failure is needed. It is rather needed to correctly automatically handle transaction aborts, which may be unrelated to database failure and recovery from it.

Distribution

With the fast technological development of computing the difference between local and distributed computing over low latency networks or buses is blurring. Thus the quite effective utilization of local techniques in such distributed environments is common, e.g., in computer clusters and multi-core processors. However the local techniques have their limitations and use multi-processes (or threads) supported by multi-processors (or multi-cores) to scale. This often turns transactions into distributed ones, if they themselves need to span multi-processes. In these cases most local concurrency control techniques do not scale well.

Distributed serializability and Commitment ordering

See *Distributed serializability* in *Serializability*

As database systems have become distributed, or started to cooperate in distributed environments (e.g., Federated databases in the early 1990s, and nowadays Grid computing, Cloud computing, and networks with smartphones), some transactions have become distributed. A distributed transaction means that the transaction spans processes, and may span computers and geographical sites. This generates a need in effective distributed concurrency control mechanisms. Achieving the Serializability property of a distributed system's schedule (see *Distributed serializability* and *Global serializability* (*Modular serializability*)) effectively poses special challenges typically not met by most of the regular serializability mechanisms, originally designed to operate locally. This is especially due to a need in costly distribution of concurrency control information amid communication and computer latency. The only known general effective technique for distribution is Commitment ordering, which was disclosed publicly in 1991 (after being patented). **Commitment ordering** (Commit ordering, CO; Raz 1992) means that transactions' chronological order of commit events is kept compatible with their respective precedence order. CO does not require the distribution of concurrency control information and provides a general effective solution (reliable, high-performance, and scalable) for both distributed and global serializability, also in a heterogeneous environment with database systems (or other transactional objects) with different (any) concurrency control mechanisms. CO is indifferent to which mechanism is utilized, since it does not interfere with any transaction operation scheduling (which most mechanisms control), and only determines the order of commit events. Thus, CO enables the efficient distribution of all other mechanisms, and also the distribution of a mix of different (any) local mechanisms, for achieving distributed and global serializability. The existence of such a solution has been considered "unlikely" until 1991, and by many experts also later, due to misunderstanding of the CO solution (see Quotations in *Global serializability*). An important side-benefit of CO is automatic distributed deadlock resolution. Contrary to CO, virtually all other techniques (when not combined with CO) are prone to distributed deadlocks (also called global deadlocks) which need special handling. CO is also the name of the resulting schedule property: A schedule has the CO property if the chronological order of its transactions' commit events is compatible with the respective transactions' precedence (partial) order.

SS2PL mentioned above is a variant (special case) of CO and thus also effective to achieve distributed and global serializability. It also provides automatic distributed deadlock resolution (a fact overlooked in the research literature even after CO's publication), as well as Strictness and thus Recoverability. Possessing these desired properties together with known efficient locking based implementations explains SS2PL's popularity. SS2PL has been utilized to efficiently achieve Distributed and Global serializability since the 1980, and has become the de facto standard for it. However, SS2PL is blocking and constraining (pessimistic), and with the proliferation of distribution and utilization of systems different from traditional database systems (e.g., as in Cloud computing), less constraining types of CO (e.g., Optimistic CO) may be needed for better performance.

Comments:

1. The *Distributed conflict serializability* property in its general form is difficult to achieve efficiently, but it is achieved efficiently via its special case *Distributed CO*: Each local component (e.g., a local DBMS) needs both to provide some form of CO, and enforce a special *vote ordering strategy* for the *Two-phase commit protocol* (2PC:

utilized to commit distributed transactions). Differently from the general Distributed CO, *Distributed SS2PL* exists automatically when all local components are SS2PL based (in each component CO exists, implied, and the vote ordering strategy is now met automatically). This fact has been known and utilized since the 1980s (i.e., that SS2PL exists globally, without knowing about CO) for efficient Distributed SS2PL, which implies Distributed serializability and strictness (e.g., see Raz 1992, page 293; it is also implied in Bernstein et al. 1987, page 78). Less constrained Distributed serializability and strictness can be efficiently achieved by Distributed Strict CO (SCO), or by a mix of SS2PL based and SCO based local components.

2. About the references and Commitment ordering: (Bernstein et al. 1987) was published before the discovery of CO in 1990. The CO schedule property is called *Dynamic atomicity* in (Lynch et al. 1993, page 201). CO is described in (Weikum and Vossen 2001, pages 102, 700), but the description is partial and misses CO's essence. (Raz 1992) was the first refereed and accepted for publication article about CO algorithms (however, publications about an equivalent Dynamic atomicity property can be traced to 1988). Other CO articles followed. (Bernstein and Newcomer 2009) note CO as one of the four major concurrency control methods, and CO's ability to provide interoperability among other methods.

Distributed recoverability

Unlike Serializability, *Distributed recoverability* and *Distributed strictness* can be achieved efficiently in a straightforward way, similarly to the way Distributed CO is achieved: In each database system they have to be applied locally, and employ a vote ordering strategy for the Two-phase commit protocol (2PC; Raz 1992, page 307).

As has been mentioned above, Distributed SS2PL, including Distributed strictness (recoverability) and Distributed commitment ordering (serializability), automatically employs the needed vote ordering strategy, and is achieved (globally) when employed locally in each (local) database system (as has been known and utilized for many years; as a matter of fact locality is defined by the boundary of a 2PC participant (Raz 1992)).

Other major subjects of attention

The design of concurrency control mechanisms is often influenced by the following subjects:

Recovery

All systems are prone to failures, and handling *recovery* from failure is a must. The properties of the generated schedules, which are dictated by the concurrency control mechanism, may have an impact on the effectiveness and efficiency of recovery. For example, the Strictness property (mentioned in the section Recoverability above) is often desirable for an efficient recovery.

Replication

For high availability database objects are often *replicated*. Updates of replicas of a same database object need to be kept synchronized. This may affect the way concurrency control is done (e.g., Gray et al. 1996).

References

- Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* [2] (free PDF download), Addison Wesley Publishing Company, 1987, ISBN 0-201-10715-5
- Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* [3], Elsevier, ISBN 1-55860-508-8
- Nancy Lynch, Michael Merritt, William Weihl, Alan Fekete (1993): *Atomic Transactions in Concurrent and Distributed Systems* [4], Morgan Kauffman (Elsevier), August 1993, ISBN 978-1-55860-104-8, ISBN 1-55860-104-X
- Yoav Raz (1992): "The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Managers Using Atomic Commitment." [5] (PDF [6]),

Proceedings of the Eighteenth International Conference on Very Large Data Bases (VLDB), pp. 292-312, Vancouver, Canada, August 1992. (also DEC-TR 841, Digital Equipment Corporation, November 1990)

Footnotes

[1] Philip A. Bernstein, Eric Newcomer (2009): *Principles of Transaction Processing*, 2nd Edition (http://www.elsevierdirect.com/product.jsp?isbn=9781558606234), Morgan Kaufmann (Elsevier), June 2009, ISBN 978-1-55860-623-4 (page 145)
[2] http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx
[3] http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description
[4] http://www.elsevier.com/wps/find/bookdescription.cws_home/680521/description#description
[5] http://www.informatik.uni-trier.de/~ley/db/conf/vldb/Raz92.html
[6] http://www.vldb.org/conf/1992/P292.PDF

Concurrency control in operating systems

Multitasking operating systems, especially real-time operating systems, need to maintain the illusion that all tasks running on top of them are all running at the same time, even though only one or a few tasks really are running at any given moment due to the limitations of the hardware the operating system is running on. Such multitasking is fairly simple when all tasks are independent from each other. However, when several tasks try to use the same resource, or when tasks try to share information, it can lead to confusion and inconsistency. The task of concurrent computing is to solve that problem. Some solutions involve "locks" similar to the locks used in databases, but they risk causing problems of their own such as deadlock. Other solutions are Non-blocking algorithms and Read-copy-update.

References

- Andrew S. Tanenbaum, Albert S Woodhull (2006): *Operating Systems Design and Implementation, 3rd Edition*, Prentice Hall, ISBN 0-13-142938-8
- Silberschatz, Avi; Galvin, Peter; Gagne, Greg (2008). *Operating Systems Concepts, 8th edition*. John Wiley & Sons. ISBN 0-470-12872-0.

Transaction Control Language

A **Transaction Control Language** (**TCL**) is a computer language and a subset of SQL, used to control transactional processing in a database. A transaction is logical unit of work that comprises one or more SQL statements, usually a group of Data Manipulation Language (DML) statements.

Examples of TCL commands include:

- **COMMIT** to apply the transaction by saving the database changes.
- **ROLLBACK** to undo all changes of a transaction.
- **SAVEPOINT** to divide the transaction into smaller sections. It defines breakpoints for a transaction to allow partial rollbacks.

ACID

In computer science, **ACID** (*Atomicity, Consistency, Isolation, Durability*) is a set of properties that guarantee that database transactions are processed reliably. In the context of databases, a single logical operation on the data is called a transaction. For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction.

Jim Gray defined these properties of a reliable transaction system in the late 1970s and developed technologies to achieve them automatically.[1]

In 1983, Andreas Reuter and Theo Härder coined the acronym *ACID* to describe them.[2]

Characteristics

Atomicity

Atomicity requires that each transaction is "all or nothing": if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes. To the outside world, a committed transaction appears (by its effects on the database) to be indivisible ("atomic"), and an aborted transaction does not happen.

Consistency

The consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules, including but not limited to constraints, cascades, triggers, and any combination thereof. This does not guarantee correctness of the transaction in all ways the application programmer might have wanted (that is the responsibility of application-level code) but merely that any programming errors do not violate any defined rules.

Isolation

The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e. one after the other. Providing isolation is the main goal of concurrency control. Depending on concurrency control method, the effects of an incomplete transaction might not even be visible to another transaction.[*citation needed*]

Durability

Durability means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, for instance, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately thereafter). To defend against power loss, transactions (or their effects) must be recorded in a non-volatile memory.

Examples

The following examples further illustrate the ACID properties. In these examples, the database table has two columns, A and B. An integrity constraint requires that the value in A and the value in B must sum to 100. The following SQL code creates a table as described above:

```
CREATE TABLE acidtest (A INTEGER, B INTEGER CHECK (A + B = 100));
```

Atomicity failure

Assume that a transaction attempts to subtract 10 from A and add 10 to B. This is a valid transaction, since the data continue to satisfy the constraint after it has executed. However, assume that after removing 10 from A, the transaction is unable to modify B. If the database retained A's new value, atomicity requires that both parts of this transaction, or neither, be complete.

Consistency failure

Consistency is a very general term which demands that the data must meet all validation rules. In the previous example, the validation is a requirement that A + B = 100. Also, it may be inferred that both A and B must be integers. A valid range for A and B may also be inferred. All validation rules must be checked to ensure consistency.

Assume that a transaction attempts to subtract 10 from A without altering B. Because consistency is checked after each transaction, it is known that A + B = 100 before the transaction begins. If the transaction removes 10 from A successfully, atomicity will be achieved. However, a validation check will show that A + B = 90, which is inconsistent with the rules of the database. The entire transaction must be cancelled and the affected rows rolled back to their pre-transaction state. If there had been other constraints, triggers, or cascades, every single change operation would have been checked in the same way as above before the transaction was committed.

Isolation failure

To demonstrate isolation, we assume two transactions execute at the same time, each attempting to modify the same data. One of the two must wait until the other completes in order to maintain isolation.

Consider two transactions. T_1 transfers 10 from A to B. T_2 transfers 10 from B to A. Combined, there are four actions:

- T_1 subtracts 10 from A.
- T_1 adds 10 to B.
- T_2 subtracts 10 from B.
- T_2 adds 10 to A.

If these operations are performed in order, isolation is maintained, although T_2 must wait. Consider what happens if T_1 fails half-way through. The database eliminates T_1's effects, and T_2 sees only valid data.

By interleaving the transactions, the actual order of actions might be:

- T_1 subtracts 10 from A.
- T_2 subtracts 10 from B.
- T_2 adds 10 to A.
- T_1 adds 10 to B.

Again, consider what happens if T_1 fails halfway through. By the time T_1 fails, T_2 has already modified A; it cannot be restored to the value it had before T_1 without leaving an invalid database. This is known as a write-write failure,[*citation needed*] because two transactions attempted to write to the same data field. In a typical system, the problem would be resolved by reverting to the last known good state, canceling the failed transaction T_1, and restarting the interrupted transaction T_2 from the good state.

Durability failure

Assume that a transaction transfers 10 from A to B. It removes 10 from A. It then adds 10 to B. At this point, a "success" message is sent to the user. However, the changes are still queued in the disk buffer waiting to be committed to the disk. Power fails and the changes are lost. The user assumes (understandably) that the changes have been made.

Implementation

Processing a transaction often requires a sequence of operations that is subject to failure for a number of reasons. For instance, the system may have no room left on its disk drives, or it may have used up its allocated CPU time.

There are two popular families of techniques: write ahead logging and shadow paging. In both cases, locks must be acquired on all information that is updated, and depending on the level of isolation, possibly on all data that is read as well. In write ahead logging, atomicity is guaranteed by copying the original (unchanged) data to a log before changing the database.Wikipedia:Disputed statement That allows the database to return to a consistent state in the event of a crash.

In shadowing, updates are applied to a partial copy of the database, and the new copy is activated when the transaction commits.

Locking vs multiversioning

Many databases rely upon locking to provide ACID capabilities. Locking means that the transaction marks the data that it accesses so that the DBMS knows not to allow other transactions to modify it until the first transaction succeeds or fails. The lock must always be acquired before processing data, including data that are read but not modified. Non-trivial transactions typically require a large number of locks, resulting in substantial overhead as well as blocking other transactions. For example, if user A is running a transaction that has to read a row of data that user B wants to modify, user B must wait until user A's transaction completes. Two phase locking is often applied to guarantee full isolation.[*citation needed*]

An alternative to locking is multiversion concurrency control, in which the database provides each reading transaction the prior, unmodified version of data that is being modified by another active transaction. This allows readers to operate without acquiring locks, i.e. writing transactions do not block reading transactions, and readers do not block writers. Going back to the example, when user A's transaction requests data that user B is modifying, the database provides A with the version of that data that existed when user B started his transaction. User A gets a consistent view of the database even if other users are changing data. One implementation, namely snapshot isolation, relaxes the isolation property.

Distributed transactions

Guaranteeing ACID properties in a distributed transaction across a distributed database where no single node is responsible for all data affecting a transaction presents additional complications. Network connections might fail, or one node might successfully complete its part of the transaction and then be required to roll back its changes, because of a failure on another node. The two-phase commit protocol (not to be confused with two-phase locking) provides atomicity for distributed transactions to ensure that each participant in the transaction agrees on whether the transaction should be committed or not.[*citation needed*] Briefly, in the first phase, one node (the coordinator) interrogates the other nodes (the participants) and only when all reply that they are prepared does the coordinator, in the second phase, formalize the transaction.

References

[1] Gray, Jim, and Reuter, Andreas, *Distributed Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993. ISBN 1-55860-190-2.

[2] These four properties, atomicity, consistency, isolation, and durability (ACID), describe the major highlights of the transaction paradigm, which has influenced many aspects of development in database systems.

Atomicity (database systems)

In database systems, **atomicity** (or **atomicness**; from Greek *a-tomos*, *undividable*) is one of the ACID transaction properties. In an **atomic transaction**, a series of database operations either *all* occur, or *nothing* occurs. A guarantee of atomicity prevents updates to the database occurring only partially, which can cause greater problems than rejecting the whole series outright. In other words, atomicity means *indivisibility* and *irreducibility*.

The etymology of the phrase originates in the Classical Greek concept of a fundamental and indivisible component; see atom.

An example of atomicity is ordering an airline ticket where two actions are required: payment, and a seat reservation. The potential passenger must either:

1. both pay for and reserve a seat; OR
2. neither pay for nor reserve a seat.

The booking system does not consider it acceptable for a customer to pay for a ticket without securing the seat, nor to reserve the seat without payment succeeding.

Another example: If one wants to transfer some amount of money from one account to another, then he/she would start a procedure to do it. However, if a failure occurs, then due to atomicity, the amount will either be transferred completely or will not even start. Thus atomicity protects the user from losing money due to a failed transaction.

Orthogonality

Atomicity does not behave completely orthogonally with regard to the other ACID properties of the transactions. For example, isolation relies on atomicity to roll back changes in the event of isolation failures such as deadlock; consistency also relies on rollback in the event of a consistency-violation by an illegal transaction. Finally, atomicity itself relies on durability to ensure the atomicity of transactions even in the face of external failures.

As a result of this, failure to detect errors and roll back the enclosing transaction may cause failures of isolation and consistency.

Implementation

Typically, systems implement atomicity by providing some mechanism to indicate which transactions have started and which finished; or by keeping a copy of the data before any changes occurred (read-copy-update). Several filesystems have developed methods for avoiding the need to keep multiple copies of data, using journaling (see journaling file system). Databases usually implement this using some form of logging/journaling to track changes. The system synchronizes the logs (often the metadata) as necessary once the actual changes have successfully taken place. Afterwards, crash recovery simply ignores incomplete entries. Although implementations vary depending on factors such as concurrency issues, the principle of atomicity — i.e. complete success or complete failure — remain.

Ultimately, any application-level implementation relies on operating-system functionality. At the file-system level, POSIX-compliant systems provide system calls such as `open(2)` and `flock(2)` that allow applications to atomically open or lock a file. At the process level, POSIX Threads provide adequate synchronization primitives.

The hardware level requires atomic operations such as Test-and-set, Fetch-and-add, Compare-and-swap, or Load-Link/Store-Conditional, together with memory barriers. Portable operating systems cannot simply block interrupts to implement synchronization, since hardware that lacks actual concurrent execution such as hyper-threading or multi-processing is now extremely rare.

In NoSQL data stores with eventual consistency, the atomicity is also weaker specified than in relational database systems, and exists only in *row*s (i.e. column families).

References

Isolation (database systems)

In database systems, **isolation** is a property that defines how/when the changes made by one operation become visible to other concurrent operations. Isolation is one of the ACID (Atomicity, Consistency, Isolation, Durability) properties.

Concurrency control

Concurrency control comprises the underlying mechanisms in a DBMS which handles isolation and guarantees related correctness. It is heavily utilized by the database and storage engines (see above) both to guarantee the correct execution of concurrent transactions, and (different mechanisms) the correctness of other DBMS processes. The transaction-related mechanisms typically constrain the database data access operations' timing (transaction schedules) to certain orders characterized as the serializability and recoverability schedule properties. Constraining database access operation execution typically means reduced performance (rates of execution), and thus concurrency control mechanisms are typically designed to provide the best performance possible under the constraints. Often, when possible without harming correctness, the serializability property is compromised for better performance. However, recoverability cannot be compromised, since such typically results in a quick database integrity violation.

Two-phase locking is the most common transaction concurrency control method in DBMSs, used to provide both serializability and recoverability for correctness. In order to access a database object a transaction first needs to acquire a lock for this object. Depending on the access operation type (e.g., reading or writing an object) and on the lock type, acquiring the lock may be blocked and postponed, if another transaction is holding a lock for that object.

Isolation levels

Of the four ACID properties in a DBMS (Database Management System), the isolation property is the one most often relaxed. When attempting to maintain the highest level of isolation, a DBMS usually acquires locks on data or implements multiversion concurrency control, which may result in a loss of concurrency. This requires adding logic for the application to function correctly.

Most DBMSs offer a number of *transaction isolation levels*, which control the degree of locking that occurs when selecting data. For many database applications, the majority of database transactions can be constructed to avoid requiring high isolation levels (e.g. SERIALIZABLE level), thus reducing the locking overhead for the system. The programmer must carefully analyze database access code to ensure that any relaxation of isolation does not cause software bugs that are difficult to find. Conversely, if higher isolation levels are used, the possibility of deadlock is increased, which also requires careful analysis and programming techniques to avoid.

The isolation levels defined by the ANSI/ISO SQL standard are listed as follows.

Serializable

This is the *highest* isolation level.

With a lock-based concurrency control DBMS implementation, serializability requires read and write locks (acquired on selected data) to be released at the end of the transaction. Also *range-locks* must be acquired when a SELECT query uses a ranged *WHERE* clause, especially to avoid the *phantom reads* phenomenon (see below).

When using non-lock based concurrency control, no locks are acquired; however, if the system detects a *write collision* among several concurrent transactions, only one of them is allowed to commit. See *snapshot isolation* for more details on this topic.

Repeatable reads

In this isolation level, a lock-based concurrency control DBMS implementation keeps read and write locks (acquired on selected data) until the end of the transaction. However, *range-locks* are not managed, so the *phantom reads* phenomenon can occur (see below).

Read committed

In this isolation level, a lock-based concurrency control DBMS implementation keeps write locks (acquired on selected data) until the end of the transaction, but read locks are released as soon as the SELECT operation is performed (so the *non-repeatable reads* phenomenon can occur in this isolation level, as discussed below). As in the previous level, *range-locks* are not managed.

Putting it in simpler words, read committed is an isolation level that guarantees that any data read is committed at the moment it is read. It simply restricts the reader from seeing any intermediate, uncommitted, 'dirty' read. It makes no promise whatsoever that if the transaction re-issues the read, it will find the same data; data is free to change after it is read.

Read uncommitted

This is the *lowest* isolation level. In this level, ***dirty reads*** are allowed (see below), so one transaction may see *not-yet-committed* changes made by other transactions.

Since each isolation level is stronger than those below, in that no higher isolation level allows an action forbidden by a lower one, the standard permits a DBMS to run a transaction at an isolation level stronger than that requested (e.g., a "Read committed" transaction may actually be performed at a "Repeatable read" isolation level).

Default isolation level

The *default isolation level* of different DBMS's varies quite widely. Most databases that feature transactions allow the user to set any isolation level. Some DBMS's also require additional syntax when performing a SELECT statement to acquire locks (e.g. *SELECT ... FOR UPDATE* to acquire exclusive write locks on accessed rows).

However, the definitions above have been criticised as being ambiguous, and as not accurately reflecting the isolation provided by many databases:

> This paper shows a number of weaknesses in the anomaly approach to defining isolation levels. The three ANSI phenomena are ambiguous. Even their broadest interpretations do not exclude anomalous behavior. This leads to some counter-intuitive results. In particular, lock-based isolation levels have different characteristics than their ANSI equivalents. This is disconcerting because commercial database systems typically use locking. Additionally, the ANSI phenomena do not distinguish among several isolation levels popular in commercial systems.

There are also other criticisms concerning ANSI SQL's isolation definition, in that it encourages implementors to do "bad things":

> ... it relies in subtle ways on an assumption that a locking schema is used for concurrency control, as opposed to an optimistic or multi-version concurrency scheme. This implies that the proposed semantics are *ill-defined*.

Read phenomena

The ANSI/ISO standard SQL 92 refers to three different *read phenomena* when Transaction 1 reads data that Transaction 2 might have changed.

In the following examples, two transactions take place. In the first, Query 1 is performed. Then, in the second transaction, Query 2 is performed and committed. Finally, in the first transaction, Query 1 is performed again.

The queries use the following data table:

users

id	name	age
1	Joe	20
2	Jill	25

Dirty reads (Uncommitted Dependency)

A dirty read occurs when a transaction is allowed to read data from a row that has been modified by another running transaction and not yet committed.

Dirty reads work similarly to non-repeatable reads; however, the second transaction would not need to be committed for the first query to return a different result. The only thing that may be prevented in the READ UNCOMMITTED isolation level is updates appearing out of order in the results; that is, earlier updates will always appear in a result set before later updates.

In our example, Transaction 2 changes a row, but does not commit the changes. Transaction 1 then reads the uncommitted data. Now if Transaction 2 rolls back its changes (already read by Transaction 1) or updates different changes to the database, then the view of the data may be wrong in the records of Transaction 1.

Transaction 1

```
<font size="9.40">
/* Query 1 */
SELECT age FROM users WHERE id = 1;
/* will read 20 */
</font>
```

Transaction 2

```
<font size="9.40">
/* Query 2 */
UPDATE users SET age = 21 WHERE id = 1;
/* No commit here */
</font>
```

```
<font size="9.40">
/* Query 1 */
SELECT age FROM users WHERE id = 1;
/* will read 21 */
</font>
```

```
<font size="9.40">
ROLLBACK; /* lock-based DIRTY READ */
</font>
```

But in this case no row exists that has an id of 1 and an age of 21.

Non-repeatable reads

A *non-repeatable read* occurs, when during the course of a transaction, a row is retrieved twice and the values within the row differ between reads.

Non-repeatable reads phenomenon may occur in a lock-based concurrency control method when read locks are not acquired when performing a SELECT, or when the acquired locks on affected rows are released as soon as the SELECT operation is performed. Under the multiversion concurrency control method, *non-repeatable reads* may occur when the requirement that a transaction affected by a commit conflict must roll back is relaxed.

Transaction 1	Transaction 2

`<font size="9.40">`
`/* Query 1 */`
`SELECT * FROM users WHERE id = 1;`
`</font>`	

	`<font size="9.40">`
	`/* Query 2 */`
	`UPDATE users SET age = 21 WHERE id = 1;`
	`COMMIT; /* in multiversion concurrency control, or lock-based READ COMMITTED */`
	`</font>`

`<font size="9.40">`
`/* Query 1 */`
`SELECT * FROM users WHERE id = 1;`
`COMMIT; /* lock-based REPEATABLE READ */`
`</font>`	

In this example, Transaction 2 commits successfully, which means that its changes to the row with id 1 should become visible. However, Transaction 1 has already seen a different value for *age* in that row. At the SERIALIZABLE and REPEATABLE READ isolation levels, the DBMS must return the old value for the second SELECT. At READ COMMITTED and READ UNCOMMITTED, the DBMS may return the updated value; this is a non-repeatable read.

There are two basic strategies used to prevent non-repeatable reads. The first is to delay the execution of Transaction 2 until Transaction 1 has committed or rolled back. This method is used when locking is used, and produces the serial schedule **T1, T2**. A serial schedule exhibits *repeatable reads* behaviour.

In the other strategy, as used in *multiversion concurrency control*, Transaction 2 is permitted to commit first, which provides for better concurrency. However, Transaction 1, which commenced prior to Transaction 2, must continue to operate on a past version of the database — a snapshot of the moment it was started. When Transaction 1 eventually tries to commit, the DBMS checks if the result of committing Transaction 1 would be equivalent to the schedule **T1, T2**. If it is, then Transaction 1 can proceed. If it cannot be seen to be equivalent, however, Transaction 1 must roll back with a serialization failure.

Using a lock-based concurrency control method, at the REPEATABLE READ isolation mode, the row with ID = 1 would be locked, thus blocking Query 2 until the first transaction was committed or rolled back. In READ COMMITTED mode, the second time Query 1 was executed, the age would have changed.

Under multiversion concurrency control, at the SERIALIZABLE isolation level, both SELECT queries see a snapshot of the database taken at the start of Transaction 1. Therefore, they return the same data. However, if Transaction 1 then attempted to UPDATE that row as well, a serialization failure would occur and Transaction 1 would be forced to roll back.

At the READ COMMITTED isolation level, each query sees a snapshot of the database taken at the start of each query. Therefore, they each see different data for the updated row. No serialization failure is possible in this mode (because no promise of serializability is made), and Transaction 1 will not have to be retried.

Phantom reads

A *phantom read* occurs when, in the course of a transaction, two identical queries are executed, and the collection of rows returned by the second query is different from the first.

This can occur when *range locks* are not acquired on performing a *SELECT ... WHERE* operation. The *phantom reads* anomaly is a special case of *Non-repeatable reads* when Transaction 1 repeats a ranged *SELECT ... WHERE* query and, between both operations, Transaction 2 creates (i.e. INSERT) new rows (in the target table) which fulfill that *WHERE* clause.

Transaction 1	Transaction 2

`<font size="9.50">`
`/* Query 1 */`
`SELECT * FROM users`
`WHERE age BETWEEN 10 AND 30;`
`</font>`	

	`<font size="9.50">`
	`/* Query 2 */`
	`INSERT INTO users VALUES ( 3, 'Bob', 27 );`
	`COMMIT;`
	`</font>`

`<font size="9.50">`
`/* Query 1 */`
`SELECT * FROM users`
`WHERE age BETWEEN 10 AND 30;`
`COMMIT;`
`</font>`	

Note that Transaction 1 executed the same query twice. If the highest level of isolation were maintained, the same set of rows should be returned both times, and indeed that is what is mandated to occur in a database operating at the SQL SERIALIZABLE isolation level. However, at the lesser isolation levels, a different set of rows may be returned the second time.

In the SERIALIZABLE isolation mode, Query 1 would result in all records with age in the range 10 to 30 being locked, thus Query 2 would block until the first transaction was committed. In REPEATABLE READ mode, the range would not be locked, allowing the record to be inserted and the second execution of Query 1 to include the new row in its results.

Isolation Levels, Read Phenomena and Locks

Isolation Levels vs Read Phenomena

Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	may occur	may occur	may occur
Read Committed	-	may occur	may occur
Repeatable Read	-	-	may occur
Serializable	-	-	-

Anomaly Serializable is not the same as Serializable. That is, it is necessary, but not sufficient that a Serializable schedule should be free of all three phenomena types. See [1] below.

"may occur" means that the isolation level suffers that phenomenon, while "-" means that it does not suffer it.

Isolation Levels vs Lock Duration

[*citation needed*]

In lock-based concurrency control, isolation level determines the duration that locks are held.
"C" - Denotes that locks are held until the transaction commits.
"S" - Denotes that the locks are held only during the currently executing statement. Note that if locks are released after a statement, the underlying data could be changed by another transaction before the current transaction commits, thus creating a violation.

Isolation level	Write Operation	Read Operation	Range Operation (...where...)
Read Uncommitted	S	S	S
Read Committed	C	S	S
Repeatable Read	C	C	S
Serializable	C	C	C

References

External links

- Oracle® Database Concepts (http://docs.oracle.com/cd/B12037_01/server.101/b10743/toc.htm), chapter 13 Data Concurrency and Consistency, Preventable Phenomena and Transaction Isolation Levels (http://docs.oracle.com/cd/B12037_01/server.101/b10743/consist.htm#sthref1919)
- Oracle® Database SQL Reference (http://docs.oracle.com/cd/B19306_01/server.102/b14200/toc.htm), chapter 19 SQL Statements: SAVEPOINT to UPDATE (http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10.htm#i2068385), SET TRANSACTION (http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10005.htm#i2067247)
- in JDBC: Connection constant fields (http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html#field_summary), Connection.getTransactionIsolation() (http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html#getTransactionIsolation()), Connection.setTransactionIsolation(int) (http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html#setTransactionIsolation(int))
- in Spring Framework: @Transactional (http://static.springsource.org/spring/docs/current/javadoc-api/org/springframework/transaction/annotation/Transactional.html), Isolation (http://static.springsource.org/spring/docs/current/javadoc-api/org/springframework/transaction/annotation/Isolation.html)

Durability (database systems)

In database systems, **durability** is the ACID property which guarantees that transactions that have committed will survive permanently. For example, if a flight booking reports that a seat has successfully been booked, then the seat will remain booked even if the system crashes.

Durability can be achieved by flushing the transaction's log records to non-volatile storage before acknowledging commitment.

In distributed transactions, all participating servers must coordinate before commit can be acknowledged. This is usually done by a two-phase commit protocol.

Many DBMSs implement durability by writing transactions into a transaction log that can be reprocessed to recreate the system state right before any later failure. A transaction is deemed committed only after it is entered in the log.

Atomic commit

An **atomic commit** is an operation in which a set of distinct changes is applied as a single operation. If the changes are applied then the atomic commit is said to have succeeded. If there is a failure before the atomic commit can be completed then all of the changes completed in the atomic commit are reversed. This ensures that the system is always left in a consistent state. The other key property of isolation comes from their nature as atomic operations. Isolation ensures that only one atomic commit is processed at a time. The most common uses of atomic commits are in database systems and revision control systems.

The problem with atomic commits is that they require coordination between multiple systems. As computer networks are unreliable services this means no algorithm can coordinate with all systems as proven in the Two Generals Problem. As databases become more and more distributed this coordination will increase the difficulty of making truly atomic commits.

Necessity for Atomic Commits

Atomic commits are essential for multi-step updates to data. This can be clearly shown in a simple example of a money transfer between two checking accounts.

This example is complicated by a transaction to check the balance of account Y during a transaction for transferring 100 dollars from account X to Y. To start, first 100 dollars is removed from account X. Second, 100 dollars is added to account Y. If the entire operation is not completed as one atomic commit, then several problems could occur. If the system fails in the middle of the operation, after removing the money from X and before adding into Y, then 100 dollars has just disappeared. Another issue is if the balance of Y is checked before the 100 dollars is added. The wrong balance for Y will be reported.

With atomic commits neither of these cases can happen, in the first case of the system failure, the atomic commit would be rolled back and the money returned to X. In the second case, the request of the balance of Y cannot occur until the atomic commit is fully completed.

Database System

Atomic commits in database systems fulfil two of the key properties of ACID, atomicity and consistency. Consistency is only achieved if each change in the atomic commit is consistent.

As shown in the example atomic commits are critical to multistep operations in databases. Due to modern hardware design the physical disk on which the database resides true atomic commits cannot exist. The smallest area that can be written to on disk is known as a sector. A single database entry may span several different sectors. Only one sector can be written at a time. This writing limit is why true atomic commits are not possible. After the database entries in memory have been modified they are queued up to be written to disk. This means the same problems identified in the example have reoccurred. Any algorithmic solution to this problem will still encounter the Two Generals' Problem. The two-phase commit protocol and three-phase commit protocol attempt to solve this and some of the other problems associated with atomic commits.

The two-phase commit protocol requires a coordinator to maintain all the information needed to recover the original state of the database if something goes wrong. As the name indicates there are two phases, voting and commit.

During the voting phase each node writes the changes in the atomic commit to its own disk. The nodes then report their status to the coordinator. If any node does not report to the coordinator or their status message is lost the coordinator assumes the node's write failed. Once all of the nodes have reported to the coordinator the second phase begins.

During the commit phase the coordinator sends a commit message to each of the nodes to record in their individual logs. Until this message is added to a node's log, any changes made will be recorded as incomplete. If any of the nodes reported a failure the coordinator will instead send a rollback message. This will remove any changes the nodes have written to disk.

The three-phase commit protocol seeks to remove the main problem with the two phase commit protocol which occurs if a coordinator and another node fail at the same time during the commit phase neither can tell what action should occur. To solve this problem a third phase is added to the protocol. The prepare to commit phase occurs after the voting phase and before the commit phase.

In the voting phase, similar to the two-phase commit, the coordinator requests that each node is ready to commit. If any node fails the coordinator will timeout while waiting for the failed node. If this happens the coordinator sends an abort message to every node. The same action will be undertaken if any of the nodes return a failure message.

Upon receiving success messages from each node in the voting phase the prepare to commit phase begins. During this phase the coordinator sends a prepare message to each node. Each node must acknowledge the prepare message and reply. If any reply is missed or any node return that they are not prepared then the coordinator sends an abort message. Any node that does not receive a prepare message before the timeout expires aborts the commit.

After all nodes have replied to the prepare message then the commit phase begins. In this phase the coordinator sends a commit message to each node. When each node receives this message it performs the actual commit. If the commit message does not reach a node due to the message being lost or the coordinator fails they will perform the commit if the timeout expires. If the coordinator fails upon recovery it will send a commit message to each node.

Revision Control

The other area where atomic commits are employed is revision control systems. This allows multiple modified files to be uploaded and merged into the source. Most revision control systems support atomic commits (CVS and VSS are the major exceptions).

Like database systems commits may fail due to a problem in applying the changes on disk. Unlike a database system which overwrites any existing data with the data from the changeset, revision control systems merge the modification in the changeset into the existing data. If the system cannot complete the merge then the commit will be rejected. If a merge cannot be resolved by the revision control software it is up to the user to merge the changes. For revision control systems that support atomic commits, this failure in merging would result in a failed commit.

Atomic commits are crucial for maintaining a consistent state in the repository. Without atomic commits some changes a developer has made may be applied but other changes may not. If these changes have any kind of coupling this will result in errors. Atomic commits prevent this by not applying partial changes which would create these errors. Note that if the changes already contain errors, atomic commits offer no fix.

Atomic Commit Convention

When using a revision control systems a common convention is to use small commits. These are sometimes referred to as atomic commits as they (ideally) only affect a single aspect of the system. These atomic commits allow for greater understandability, less effort to roll back changes, easier bug identification.

The greater understandability comes from the small size and focused nature of the commit. It is much easier to understand what is changed and reasoning behind the changes if you are only looking for one kind of change. This becomes especially important when making format changes to the source code. If format and functional changes are combined it becomes very difficult to identify useful changes. Imagine if the spacing in a file is changed from using tabs to three spaces every tab in the file will show as having been changed. This becomes critical if some functional changes are also made as a reviewer may simply not see the functional changes.

If only atomic commits are made then commits that introduce errors become much simpler to identify. You are not required to look though every commit to see if it was the cause of the error, only the commits dealing with that functionality need to be examined. If the error is to be rolled back, atomic commits again make the job much simpler. Instead of having to revert to the offending revision and remove the changes manually before integrating any later changes; the developer can simply revert any changes in the identified commit. This also reduces the risk of a developer accidentally removing unrelated changes that happened to be in the same commit.

Atomic commits also allow bug fixes to be easily reviewed if only a single bug fixes committed at a time. Instead of having to check multiple potentially unrelated files the reviewer must only check files and changes that directly impact the bug being fixed. This also means that bug fixes can be easily packaged for testing as only the changes that fix the bug are in the commit.

References

Schedule (computer science)

In the fields of databases and transaction processing (transaction management), a **schedule** (or **history**) of a system is an abstract model to describe execution of transactions running in the system. Often it is a *list* of operations (actions) ordered by time, performed by a set of transactions that are executed together in the system. If order in time between certain operations is not determined by the system, then a *partial order* is used. Examples of such operations are requesting a read operation, reading, writing, aborting, committing, requesting lock, locking, etc. Not all transaction operation types should be included in a schedule, and typically only selected operation types (e.g., data access operations) are included, as needed to reason about and describe certain phenomena. Schedules and schedule properties are fundamental concepts in database concurrency control theory.

Formal description

The following is an example of a schedule:

$$D = \begin{bmatrix} T1 & T2 & T3 \\ R(X) & & \\ W(X) & & \\ Com. & & \\ & R(Y) & \\ & W(Y) & \\ & Com. & \\ & & R(Z) \\ & & W(Z) \\ & & Com. \end{bmatrix}$$

In this example, the horizontal axis represents the different transactions in the schedule D. The vertical axis represents time order of operations. Schedule D consists of three transactions T1, T2, T3. The schedule describes the actions of the transactions as seen by the DBMS. First T1 Reads and Writes to object X, and then Commits. Then T2 Reads and Writes to object Y and Commits, and finally T3 Reads and Writes to object Z and Commits. This is an example of a *serial* schedule, i.e., sequential with no overlap in time, because the actions of in all three transactions are sequential, and the transactions are not interleaved in time.

Representing the schedule D above by a table (rather than a list) is just for the convenience of identifying each transaction's operations in a glance. This notation is used throughout the article below. A more common way in the technical literature for representing such schedule is by a list:

D = R1(X) W1(X) Com1 R2(Y) W2(Y) Com2 R3(Z) W3(Z) Com3

Usually, for the purpose of reasoning about concurrency control in databases, an operation is modeled as *atomic*, occurring at a point in time, without duration. When this is not satisfactory start and end time-points and possibly other point events are specified (rarely). Real executed operations always have some duration and specified respective times of occurrence of events within them (e.g., "exact" times of beginning and completion), but for concurrency control reasoning usually only the precedence in time of the whole operations (without looking into the quite complex details of each operation) matters, i.e., which operation is before, or after another operation. Furthermore, in many cases the before/after relationships between two specific operations do not matter and should not be specified, while being specified for other pairs of operations.

In general operations of transactions in a schedule can interleave (i.e., transactions can be executed concurrently), while time orders between operations in each transaction remain unchanged as implied by the transaction's program. Since not always time orders between all operations of all transactions matter and need to be specified, a schedule is, in general, a *partial order* between operations rather than a *total order* (where order for each pair is determined, as in

a list of operations). Also in the general case each transaction may consist of several processes, and itself be properly represented by a partial order of operations, rather than a total order. Thus in general a schedule is a partial order of operations, containing (embedding) the partial orders of all its transactions.

Time-order between two operations can be represented by an *ordered pair* of these operations (e.g., the existence of a pair (OP1,OP2) means that OP1 is always before OP2), and a schedule in the general case is a set of such ordered pairs. Such a set, a schedule, is a partial order which can be represented by an *acyclic directed graph* (or *directed acyclic graph*, DAG) with operations as nodes and time-order as a directed edge (no cycles are allowed since a cycle means that a first (any) operation on a cycle can be both before and after (any) another second operation on the cycle, which contradicts our perception of Time). In many cases a graphical representation of such graph is used to demonstrate a schedule.

Comment: Since a list of operations (and the table notation used in this article) always represents a total order between operations, schedules that are not a total order cannot be represented by a list (but always can be represented by a DAG).

Types of schedule

Serial

The transactions are executed non-interleaved (see example above) i.e., a serial schedule is one in which no transaction starts until a running transaction has ended.

Serializable

A schedule that is equivalent (in its outcome) to a serial schedule has the serializability property.

In schedule E, the order in which the actions of the transactions are executed is not the same as in D, but in the end, E gives the same result as D.

$$E = \begin{bmatrix} T1 & T2 & T3 \\ R(X) & & \\ & R(Y) & \\ & & R(Z) \\ W(X) & & \\ & W(Y) & \\ & & W(Z) \\ Com. & Com. & Com. \end{bmatrix}$$

Conflicting actions

Two actions are said to be in conflict (conflicting pair) if:

1. The actions belong to different transactions.
2. At least one of the actions is a write operation.
3. The actions access the same object (read or write).

The following set of actions is conflicting:

- R1(X), W2(X), W3(X) (3 conflicting pairs)

While the following sets of actions are not:

- R1(X), R2(X), R3(X)
- R1(X), W2(Y), R3(X)

Conflict equivalence

The schedules S1 and S2 are said to be conflict-equivalent if following two conditions are satisfied:

1. Both schedules S1 and S2 involve the same set of transactions (including ordering of actions within each transaction).
2. The set of conflicting pairs in S1 is the same as in S2.

Conflict-serializable

A schedule is said to be conflict-serializable when the schedule is conflict-equivalent to one or more serial schedules.

Another definition for conflict-serializability is that a schedule is conflict-serializable if and only if its precedence graph/serializability graph, when only committed transactions are considered, is acyclic (if the graph is defined to include also uncommitted transactions, then cycles involving uncommitted transactions may occur without conflict serializability violation).

$$G = \begin{bmatrix} T1 & T2 \\ R(A) & \\ & R(A) \\ W(B) & \\ Com. & \\ & W(A) \\ & Com. \end{bmatrix}$$

Which is conflict-equivalent to the serial schedule <T1,T2>, but not <T2,T1>.

Commitment-ordered

A schedule is said to be commitment-ordered (commit-ordered), or commitment-order-serializable, if it obeys the Commitment ordering (CO; also commit-ordering or commit-order-serializability) schedule property. This means that the order in time of transactions' commitment events is compatible with the precedence (partial) order of the respective transactions, as induced by their schedule's acyclic precedence graph (serializability graph, conflict graph). This implies that it is also conflict-serializable. The CO property is especially effective for achieving Global serializability in distributed systems.

Comment: Commitment ordering, which was discovered in 1990, is obviously not mentioned in (Bernstein et al. 1987). Its correct definition appears in (Weikum and Vossen 2001), however the description there of its related techniques and theory is partial, inaccurate, and misleading.Wikipedia:Avoid weasel words For an extensive coverage of commitment ordering and its sources see *Commitment ordering* and *The History of Commitment Ordering*.

View equivalence

Two schedules S1 and S2 are said to be view-equivalent when the following conditions are satisfied:

1. If the transaction T_i in S1 reads an initial value for object X, so does the transaction T_i in S2.
2. If the transaction T_i in S1 reads the value written by transaction T_j in S1 for object X, so does the transaction T_i in S2.
3. If the transaction T_i in S1 is the final transaction to write the value for an object X, so is the transaction T_i in S2.

View-serializable

A schedule is said to be view-serializable if it is view-equivalent to some serial schedule. Note that by definition, all conflict-serializable schedules are view-serializable.

$$G = \begin{bmatrix} T1 & T2 \\ R(A) & \\ & R(A) \\ W(B) & \end{bmatrix}$$

Notice that the above example (which is the same as the example in the discussion of conflict-serializable) is both view-serializable and conflict-serializable at the same time.) There are however view-serializable schedules that are not conflict-serializable: those schedules with a transaction performing a blind write:

$$H = \begin{bmatrix} T1 & T2 & T3 \\ R(A) & & \\ & W(A) & \\ & Com. & \\ W(A) & & \\ Com. & & \\ & & W(A) \\ & & Com. \\ & & \end{bmatrix}$$

The above example is not conflict-serializable, but it is view-serializable since it has a view-equivalent serial schedule <T1, T2, T3>.

Since determining whether a schedule is view-serializable is NP-complete, view-serializability has little practical interest.

Recoverable

Transactions commit only after all transactions whose changes they read, commit.

$$F = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Com. & \\ & Com. \\ & \end{bmatrix} \quad F2 = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Abort & \\ & Abort \\ & \end{bmatrix}$$

These schedules are recoverable. F is recoverable because T1 commits before T2, that makes the value read by T2 correct. Then T2 can commit itself. In F2, if T1 aborted, T2 has to abort because the value of A it read is incorrect. In both cases, the database is left in a consistent state.

Unrecoverable

If a transaction T1 aborts, and a transaction T2 commits, but T2 relied on T1, we have an unrecoverable schedule.

$$G = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ & Com. \\ Abort & \\ & \end{bmatrix}$$

In this example, G is unrecoverable, because T2 read the value of A written by T1, and committed. T1 later aborted, therefore the value read by T2 is wrong, but since T2 committed, this schedule is unrecoverable.

Avoids cascading aborts (rollbacks)

Also named cascadeless. A single transaction abort leads to a series of transaction rollback. Strategy to prevent cascading aborts is to disallow a transaction from reading uncommitted changes from another transaction in the same schedule.

The following examples are the same as the one from the discussion on recoverable:

$$F = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Com. & \\ & Com. \\ & \end{bmatrix} \quad F2 = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Abort & \\ & Abort \\ & \end{bmatrix}$$

In this example, although F2 is recoverable, it does not avoid cascading aborts. It can be seen that if T1 aborts, T2 will have to be aborted too in order to maintain the correctness of the schedule as T2 has already read the uncommitted value written by T1.

The following is a recoverable schedule which avoids cascading abort. Note, however, that the update of A by T1 is always lost (since T1 is aborted).

$$F3 = \begin{bmatrix} T1 & T2 \\ & R(A) \\ R(A) & \\ W(A) & \\ & W(A) \\ Abort & \\ & Commit \\ & \end{bmatrix}$$

Cascading aborts avoidance is sufficient but not necessary for a schedule to be recoverable.

Strict

A schedule is strict - has the strictness property - if for any two transactions T1, T2, if a write operation of T1 precedes a *conflicting* operation of T2 (either read or write), then the commit event of T1 also precedes that conflicting operation of T2.

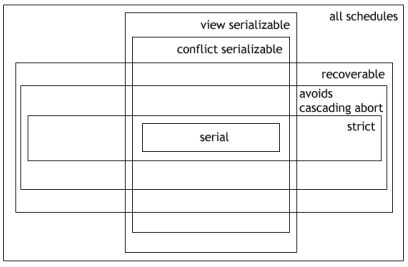
Any strict schedule is cascadeless, but not the converse. Strictness allows efficient recovery of databases from failure.

Hierarchical relationship between serializability classes

The following expressions illustrate the hierarachical (containment) relationships between serializability and recoverability classes:

- Serial ⊂ commitment-ordered ⊂ conflict-serializable ⊂ view-serializable ⊂ all schedules
- Serial ⊂ strict ⊂ avoids cascading aborts ⊂ recoverable ⊂ all schedules

The Venn diagram (below) illustrates the above clauses graphically.



Venn diagram for serializability and recoverability classes

Practical implementations

In practice, most general purpose database systems employ conflict-serializable and recoverable (primarily strict) schedules.

References

- Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman: *Concurrency Control and Recovery in Database Systems* [2], Addison Wesley Publishing Company, 1987, ISBN 0-201-10715-5
- Gerhard Weikum, Gottfried Vossen: *Transactional Information Systems* [3], Elsevier, 2001, ISBN 1-55860-508-8

Serializability

In concurrency control of databases,[1][2] transaction processing (transaction management), and various transactional applications (e.g., transactional memory[3] and software transactional memory), both centralized and distributed, a transaction schedule is **serializable** if its outcome (e.g., the resulting database state) is equal to the outcome of its transactions executed serially, i.e., sequentially without overlapping in time. Transactions are normally executed concurrently (they overlap), since this is the most efficient way. Serializability is the major correctness criterion for concurrent transactions' executions. It is considered the highest level of isolation between transactions, and plays an essential role in concurrency control. As such it is supported in all general purpose database systems. *Strong strict two-phase locking* (SS2PL) is a popular serializability mechanism utilized in most of the database systems (in various variants) since their early days in the 1970s.

Serializability theory provides the formal framework to reason about and analyze serializability and its techniques. Though it is mathematical in nature, its fundamentals are informally (without Mathematics notation) introduced below.

Database transaction

For this discussion a *database transaction* is a specific intended run (with specific parameters, e.g., with transaction identification, at least) of a computer program (or programs) that accesses a database (or databases). Such a program is written with the assumption that it is running in *isolation* from other executing programs, i.e., when running, its accessed data (after the access) are not changed by other running programs. Without this assumption the transaction's results are unpredictable and can be wrong. The same transaction can be executed in different situations, e.g., in different times and locations, in parallel with different programs. A *live* transaction (i.e., exists in a computing environment with already allocated computing resources; to distinguish from a *transaction request*, waiting to get execution resources) can be in one of three states, or phases:

1. *Running* - Its program(s) is (are) executing.
2. *Ready* - Its program's execution has ended, and it is waiting to be *Ended (Completed)*.
3. *Ended* (or *Completed*) - It is either *Committed* or *Aborted (Rolled-back)*, depending whether the execution is considered a success or not, respectively . When committed, all its *recoverable* (i.e., with states that can be controlled for this purpose), *durable* resources (typically *database data*) are put in their *final* states, states after running. When aborted, all its recoverable resources are put back in their *initial* states, as before running.

A failure in transaction's computing environment before ending typically results in its abort. However, a transaction may be aborted also for other reasons as well (e.g., see below).

Upon being ended (completed), transaction's allocated computing resources are released and the transaction disappears from the computing environment. However, the effects of a committed transaction remain in the database, while the effects of an aborted (rolled-back) transaction disappear from the database. The concept of *atomic transaction* ("all or nothing" semantics) was designed to exactly achieve this behavior, in order to control correctness in complex faulty systems.

Correctness

Correctness - serializability

Serializability is a property of a transaction schedule (history). It relates to the *isolation* property of a database transaction.

> **Serializability** of a schedule means equivalence (in the outcome, the database state, data values) to a *serial schedule* (i.e., sequential with no transaction overlap in time) with the same transactions. It is the major criterion for the correctness of concurrent transactions' schedule, and thus supported in all general purpose database systems.
>
> **The rationale behind serializability** is the following:
>
> If each transaction is correct by itself, i.e., meets certain integrity conditions, then a schedule that comprises any *serial* execution of these transactions is correct (its transactions still meet their conditions): "Serial" means that transactions do not overlap in time and cannot interfere with each other, i.e, complete *isolation* between each other exists. Any order of the transactions is legitimate, if no dependencies among them exist, which is assumed (see comment below). As a result, a schedule that comprises any execution (not necessarily serial) that is equivalent (in its outcome) to any serial execution of these transactions, is correct.

Schedules that are not serializable are likely to generate erroneous outcomes. Well known examples are with transactions that debit and credit accounts with money: If the related schedules are not serializable, then the total sum of money may not be preserved. Money could disappear, or be generated from nowhere. This and violations of possibly needed other invariant preservations are caused by one transaction writing, and "stepping on" and erasing what has been written by another transaction before it has become permanent in the database. It does not happen if serializability is maintained.

If any specific order between some transactions is requested by an application, then it is enforced independently of the underlying serializability mechanisms. These mechanisms are typically indifferent to any specific order, and generate some unpredictable partial order that is typically compatible with multiple serial orders of these transactions. This partial order results from the scheduling orders of concurrent transactions' data access operations, which depend on many factors.

Correctness - recoverability

A major characteristic of a database transaction is *atomicity*, which means that it either *commits*, i.e., all its operations' results take effect in the database, or *aborts* (rolled-back), all its operations' results do not have any effect on the database ("all or nothing" semantics of a transaction). In all real systems transactions can abort for many reasons, and serializability by itself is not sufficient for correctness. Schedules also need to possess the *recoverability* (from abort) property. **Recoverability** means that committed transactions have not read data written by aborted transactions (whose effects do not exist in the resulting database states). While serializability is currently compromised on purpose in many applications for better performance (only in cases when application's correctness is not harmed), compromising recoverability would quickly violate the database's integrity, as well as that of transactions' results external to the database. A schedule with the recoverability property (a *recoverable* schedule) "recovers" from aborts by itself, i.e., aborts do not harm the integrity of its committed transactions and resulting database. This is false without recoverability, where the likely integrity violations (resulting incorrect database data) need special, typically manual, corrective actions in the database.

Implementing recoverability in its general form may result in *cascading aborts*: Aborting one transaction may result in a need to abort a second transaction, and then a third, and so on. This results in a waste of already partially executed transactions, and may result also in a performance penalty. **Avoiding cascading aborts** (ACA, or Cascadelessness) is a special case of recoverability that exactly prevents such phenomenon. Often in practice a

special case of ACA is utilized: **Strictness**. Strictness allows an efficient database recovery from failure.

Note that the *recoverability* property is needed even if no database failure occurs and no database *recovery* from failure is needed. It is rather needed to correctly automatically handle aborts, which may be unrelated to database failure and recovery from failure.

Relaxing serializability

In many applications, unlike with finances, absolute correctness is not needed. For example, when retrieving a list of products according to specification, in most cases it does not matter much if a product, whose data was updated a short time ago, does not appear in the list, even if it meets the specification. It will typically appear in such a list when tried again a short time later. Commercial databases provide concurrency control with a whole range of isolation levels which are in fact (controlled) serializability violations in order to achieve higher performance. Higher performance means better transaction execution rate and shorter average transaction response time (transaction duration). *Snapshot isolation* is an example of a popular, widely utilized efficient relaxed serializability method with many characteristics of full serializability, but still short of some, and unfit in many situations.

Another common reason nowadays for distributed serializability relaxation (see below) is the requirement of availability of internet products and services. This requirement is typically answered by large-scale data replication. The straightforward solution for synchronizing replicas' updates of a same database object is including all these updates in a single atomic distributed transaction. However, with many replicas such a transaction is very large, and may span several computers and networks that some of them are likely to be unavailable. Thus such a transaction is likely to end with abort and miss its purpose. Consequently Optimistic replication (Lazy replication) is often utilized (e.g., in many products and services by Google, Amazon, Yahoo, and alike), while serializability is relaxed and compromised for eventual consistency. Again in this case, relaxation is done only for applications that are not expected to be harmed by this technique.

Classes of schedules defined by *relaxed serializability* properties either contain the serializability class, or are incomparable with it.

View and conflict serializability

Mechanisms that enforce serializability need to execute in real time, or almost in real time, while transactions are running at high rates. In order to meet this requirement special cases of serializability, sufficient conditions for serializability which can be enforced effectively, are utilized.

Two major types of serializability exist: *view-serializability*, and *conflict-serializability*. View-serializability matches the general definition of serializability given above. Conflict-serializability is a broad special case, i.e., any schedule that is conflict-serializable is also view-serializable, but not necessarily the opposite. Conflict-serializability is widely utilized because it is easier to determine and covers a substantial portion of the view-serializable schedules. Determining view-serializability of a schedule is an NP-complete problem (a class of problems with only difficult-to-compute, excessively time-consuming known solutions).

> **View-serializability** of a schedule is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that respective transactions in the two schedules read and write the same data values ("view" the same data values).
>
> **Conflict-serializability** is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that both schedules have the same sets of respective chronologically ordered pairs of conflicting operations (same precedence relations of respective conflicting operations).

Operations upon data are *read* or *write* (a write: either *insert* or *modify* or *delete*). Two operations are *conflicting*, if they are of different transactions, upon the same datum (data item), and at least one of them is *write*. Each such pair of conflicting operations has a *conflict type*: It is either a *read-write*, or *write-read*, or a *write-write* conflict. The

transaction of the second operation in the pair is said to be *in conflict* with the transaction of the first operation. A more general definition of conflicting operations (also for complex operations, which may consist each of several "simple" read/write operations) requires that they are noncommutative (changing their order also changes their combined result). Each such operation needs to be atomic by itself (by proper system support) in order to be considered an operation for a commutativity check. For example, read-read operations are commutative (unlike read-write and the other possibilities) and thus read-read is not a conflict. Another more complex example: the operations *increment* and *decrement* of a *counter* are both *write* operations (both modify the counter), but do not need to be considered conflicting (write-write conflict type) since they are commutative (thus increment-decrement is not a conflict; e.g., already has been supported in the old IBM's IMS "fast path"). Only precedence (time order) in pairs of conflicting (non-commutative) operations is important when checking equivalence to a serial schedule, since different schedules consisting of the same transactions can be transformed from one to another by changing orders between different transactions' operations (different transactions' interleaving), and since changing orders of commutative operations (non-conflicting) does not change an overall operation sequence result, i.e., a schedule outcome (the outcome is preserved through order change between non-conflicting operations, but typically not when conflicting operations change order). This means that if a schedule can be transformed to any serial schedule without changing orders of conflicting operations (but changing orders of non-conflicting, while preserving operation order inside each transaction), then the outcome of both schedules is the same, and the schedule is conflict-serializable by definition.

Conflicts are the reason for blocking transactions and delays (non-materialized conflicts), or for aborting transactions due to serializability violations prevention. Both possibilities reduce performance. Thus reducing the number of conflicts, e.g., by commutativity (when possible), is a way to increase performance.

A transaction can issue/request a conflicting operation and be *in conflict* with another transaction while its conflicting operation is delayed and not executed (e.g., blocked by a lock). Only executed (*materialized*) conflicting operations are relevant to *conflict serializability* (see more below).

Enforcing conflict serializability

Testing conflict serializability

Schedule compliance with conflict serializability can be tested with the precedence graph (*serializability graph*, *serialization graph*, *conflict graph*) for committed transactions of the schedule. It is the directed graph representing precedence of transactions in the schedule, as reflected by precedence of conflicting operations in the transactions.

> In the **precedence graph** transactions are nodes and precedence relations are directed edges. There exists an edge from a first transaction to a second transaction, if the second transaction is *in conflict* with the first (see Conflict serializability above), and the conflict is **materialized** (i.e., if the requested conflicting operation is actually executed: in many cases a requested/issued conflicting operation by a transaction is delayed and even never executed, typically by a lock on the operation's object, held by another transaction, or when writing to a transaction's temporary private workspace and materializing, copying to the database itself, upon commit; as long as a requested/issued conflicting operation is not executed upon the database itself, the conflict is **non-materialized**; non-materialized conflicts are not represented by an edge in the precedence graph).
>
> **Comment:** In many text books only *committed transactions* are included in the precedence graph. Here all transactions are included for convenience in later discussions.

The following observation is a **key characterization of conflict serializability**:

> A schedule is *conflict-serializable* if and only if its precedence graph of *committed transactions* (when only *committed* transactions are considered) is *acyclic*. This means that a cycle consisting of committed transactions only is generated in the (general) precedence graph, if and only if conflict-serializability is violated.

Cycles of committed transactions can be prevented by aborting an *undecided* (neither committed, nor aborted) transaction on each cycle in the precedence graph of all the transactions, which can otherwise turn into a cycle of committed transactions (and a committed transaction cannot be aborted). One transaction aborted per cycle is both required and sufficient number to break and eliminate the cycle (more aborts are possible, and can happen in some mechanisms, but unnecessary for serializability). The probability of cycle generation is typically low, but nevertheless, such a situation is carefully handled, typically with a considerable overhead, since correctness is involved. Transactions aborted due to serializability violation prevention are *restarted* and executed again immediately.

Serializability enforcing mechanisms typically do not maintain a precedence graph as a data structure, but rather prevent or break cycles implicitly (e.g., SS2PL below).

Common mechanism - SS2PL

Strong strict two phase locking (SS2PL) is a common mechanism utilized in database systems since their early days in the 1970s (the "SS" in the name SS2PL is newer though) to enforce both conflict serializability and *strictness* (a special case of recoverability which allows effective database recovery from failure) of a schedule. In this mechanism each datum is locked by a transaction before accessing it (any read or write operation): The item is marked by, associated with a *lock* of a certain type, depending on operation (and the specific implementation; various models with different lock types exist; in some models locks may change type during the transaction's life). As a result access by another transaction may be blocked, typically upon a conflict (the lock delays or completely prevents the conflict from being materialized and be reflected in the precedence graph by blocking the conflicting operation), depending on lock type and the other transaction's access operation type. Employing an SS2PL mechanism means that all locks on data on behalf of a transaction are released only after the transaction has ended (either committed or aborted).

SS2PL is the name of the resulting schedule property as well, which is also called *rigorousness*. SS2PL is a special case (proper subset) of Two-phase locking (2PL)

Mutual blocking between transactions results in a *deadlock*, where execution of these transactions is stalled, and no completion can be reached. Thus deadlocks need to be resolved to complete these transactions' execution and release related computing resources. A deadlock is a reflection of a potential cycle in the precedence graph, that would occur without the blocking when conflicts are materialized. A deadlock is resolved by aborting a transaction involved with such potential cycle, and breaking the cycle. It is often detected using a *wait-for graph* (a graph of conflicts blocked by locks from being materialized; it can be also defined as the graph of non-materialized conflicts; conflicts not materialized are not reflected in the precedence graph and do not affect serializability), which indicates which transaction is "waiting for" lock release by which transaction, and a cycle means a deadlock. Aborting one transaction per cycle is sufficient to break the cycle. Transactions aborted due to deadlock resolution are *restarted* and executed again immediately.

Other enforcing techniques

Other known mechanisms include:

- Precedence graph (or Serializability graph, Conflict graph) cycle elimination
- Two-phase locking (2PL)
- Timestamp ordering (TO)
- Serializable snapshot isolation[4] (SerializableSI)

The above (conflict) serializability techniques in their general form do not provide recoverability. Special enhancements are needed for adding recoverability.

Optimistic versus pessimistic techniques

Concurrency control techniques are of three major types:

1. *Pessimistic*: In Pessimistic concurrency control a transaction blocks data access operations of other transactions upon conflicts, and conflicts are *non-materialized* until blocking is removed. This is done to ensure that operations that may violate serializability (and in practice also recoverability) do not occur.
2. *Optimistic*: In Optimistic concurrency control data access operations of other transactions are not blocked upon conflicts, and conflicts are immediately *materialized*. When the transaction reaches the *ready* state, i.e., its *running* state has been completed, possible serializability (and in practice also recoverability) violation by the transaction's operations (relatively to other running transactions) is checked: If violation has occurred, the transaction is typically *aborted* (sometimes aborting *another* transaction to handle serializability violation is preferred). Otherwise it is *committed*.
3. *Semi-optimistic*: Mechanisms that mix blocking in certain situations with not blocking in other situations and employ both materialized and non-materialized conflicts

The main differences between the technique types is the conflict types that are generated by them. A pessimistic method blocks a transaction operation upon conflict and generates a non-materialized conflict, while an optimistic method does not block and generates a materialized conflict. A semi-optimistic method generates both conflict types. Both conflict types are generated by the chronological orders in which transaction operations are invoked, independently of the type of conflict. A cycle of committed transactions (with materialized conflicts) in the *precedence graph* (conflict graph) represents a serializability violation, and should be avoided for maintaining serializability. A cycle of (non-materialized) conflicts in the *wait-for graph* represents a deadlock situation, which should be resolved by breaking the cycle. Both cycle types result from conflicts, and should be broken. At any technique type conflicts should be detected and considered, with similar overhead for both materialized and non-materialized conflicts (typically by using mechanisms like locking, while either blocking for locks, or not blocking but recording conflict for materialized conflicts). In a blocking method typically a context switching occurs upon conflict, with (additional) incurred overhead. Otherwise blocked transactions' related computing resources remain idle, unutilized, which may be a worse alternative. When conflicts do not occur frequently, optimistic methods typically have an advantage. With different transactions loads (mixes of transaction types) one technique type (i.e., either optimistic or pessimistic) may provide better performance than the other.

Unless schedule classes are *inherently blocking* (i.e., they cannot be implemented without data-access operations blocking; e.g., 2PL, SS2PL and SCO above; see chart), they can be implemented also using optimistic techniques (e.g., Serializability, Recoverability).

Serializable multi-version concurrency control

See also Multiversion concurrency control (partial coverage)

and Serializable_Snapshot_Isolation in Snapshot isolation

Multi-version concurrency control (MVCC) is a common way today to increase concurrency and performance by generating a new version of a database object each time the object is written, and allowing transactions' read operations of several last relevant versions (of each object), depending on scheduling method. MVCC can be combined with all the serializability techniques listed above (except SerializableSI which is originally MVCC based). It is utilized in most general-purpose DBMS products.

MVCC is especially popular nowadays through the *relaxed serializability* (see above) method *Snapshot isolation* (SI) which provides better performance than most known serializability mechanisms (at the cost of possible serializability violation in certain cases). SerializableSI, which is an efficient enhancement of SI to make it serializable, is intended to provide an efficient serializable solution. SerializableSI has been analyzed[5] via a general theory of MVCC

Distributed serializability

Overview

Distributed serializability is the serializability of a schedule of a transactional distributed system (e.g., a distributed database system). Such system is characterized by *distributed transactions* (also called *global transactions*), i.e., transactions that span computer processes (a process abstraction in a general sense, depending on computing environment; e.g., operating system's thread) and possibly network nodes. A distributed transaction comprises more than one *local sub-transactions* that each has states as described above for a database transaction. A local sub-transaction comprises a single process, or more processes that typically fail together (e.g., in a single processor core). Distributed transactions imply a need in Atomic commit protocol to reach consensus among its local sub-transactions on whether to commit or abort. Such protocols can vary from a simple (one-phase) hand-shake among processes that fail together, to more sophisticated protocols, like Two-phase commit, to handle more complicated cases of failure (e.g., process, node, communication, etc. failure). Distributed serializability is a major goal of distributed concurrency control for correctness. With the proliferation of the Internet, Cloud computing, Grid computing, and small, portable, powerful computing devices (e.g., smartphones) the need for effective distributed serializability techniques to ensure correctness in and among distributed applications seems to increase.

Distributed serializability is achieved by implementing distributed versions of the known centralized techniques. Typically all such distributed versions require utilizing conflict information (either of materialized or non-materialized conflicts, or equivalently, transaction precedence or blocking information; conflict serializability is usually utilized) that is not generated locally, but rather in different processes, and remote locations. Thus information distribution is needed (e.g., precedence relations, lock information, timestamps, or tickets). When the distributed system is of a relatively small scale, and message delays across the system are small, the centralized concurrency control methods can be used unchanged, while certain processes or nodes in the system manage the related algorithms. However, in a large-scale system (e.g., *Grid* and *Cloud*), due to the distribution of such information, substantial performance penalty is typically incurred, even when distributed versions of the methods (Vs. centralized) are used, primarily due to computer and communication latency. Also, when such information is distributed, related techniques typically do not scale well. A well-known example with scalability problems is a distributed lock manager, which distributes lock (non-materialized conflict) information across the distributed system to implement locking techniques.

Notes

[1] Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* (http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx) (free PDF download), Addison Wesley Publishing Company, ISBN 0-201-10715-5

[2] Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description), Elsevier, ISBN 1-55860-508-8

[3] Maurice Herlihy and J. Eliot B. Moss. *Transactional memory: architectural support for lock-free data structures.* Proceedings of the 20th annual international symposium on Computer architecture (ISCA '93). Volume 21, Issue 2, May 1993.

[4] Michael J. Cahill, Uwe Röhm, Alan D. Fekete (2008): "Serializable isolation for snapshot databases" (http://portal.acm.org/citation.cfm?id=1376690), *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 729-738, Vancouver, Canada, June 2008, ISBN 978-1-60558-102-6 (SIGMOD 2008 best paper award)

[5] Alan Fekete (2009), "Snapshot Isolation and Serializable Execution" (http://www.it.usyd.edu.au/~fekete/teaching/serializableSI-Fekete.pdf), Presentation, Page 4, 2009, The university of Sydney (Australia). Retrieved 16 September 2009

References

- Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* (http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx), Addison Wesley Publishing Company, ISBN 0-201-10715-5
- Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description), Elsevier, ISBN 1-55860-508-8

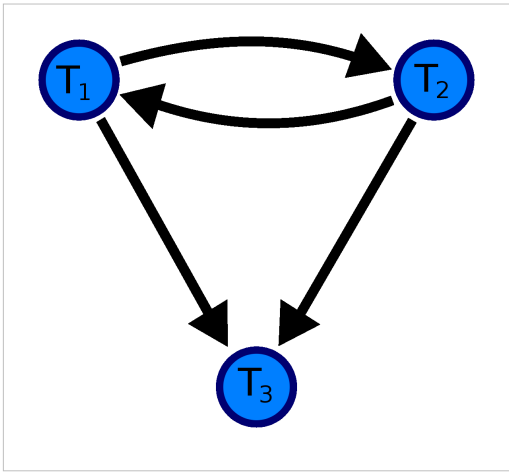
Precedence graph

A **precedence graph**, also named **conflict graph** and **serializability graph**, is used in the context of concurrency control in databases.

The precedence graph for a schedule S contains:

- A node for each committed transaction in S
- An arc from T_i to T_j if an action of T_i precedes and conflicts with one of T_j's actions.

Precedence graph example



$$D = \begin{bmatrix} T1 & T2 & T3 \\ R(A) & & \\ & W(A) & \\ & Com. & \\ W(A) & & \\ Com. & & \\ & & W(A) \\ & & Com. \end{bmatrix}$$

or

$$D = R1(A)\ W2(A)\ Com.2\ W1(A)\ Com.1\ W3(A)\ Com.3$$

A precedence graph of the schedule D, with 3 transactions. As there is a cycle (of length 2; with two edges) through the committed transactions T1 and T2, this schedule (history) is *not* Conflict serializable.

Testing Serializability with Precedence Graph

The drawing sequence for the precedence graph:-

1. For each transaction T_i participating in schedule S, create a node labelled T_i in the precedence graph. So the precedence graph contains T_1, T_2, T_3
2. For each case in S where T_i executes a write_item(X) then T_j executes a read_item(X), create an edge (T_i --> T_j) in the precedence graph. This occurs nowhere in the above example, as there is no read after write.
3. For each case in S where T_i executes a read_item(X) then T_j executes a write_item(X), create an edge (T_i --> T_j) in the precedence graph. This will bring to front a directed graph from T_1 to T_2.
4. For each case in S where T_i executes a write_item(X) then T_j executes a write_item(X), create an edge (T_i --> T_j) in the precedence graph. It creates a directed graph from T_2 to T_1, T_1 to T_3, and T_2 to T_3.
5. The schedule S is serializable if the precedence graph has no cycles. As T_1 and T_2 constitute a bicycle, then we cannot declare S as serializable or not and serializability has to be checked using other methods.

External links

- The Fundamentals of Database Systems, 5th Edition [1] the use of precedence graphs is discussed in chapter 17, as they relate to tests for conflict serializability.
- Abraham Silberschatz, Henry Korth, and S. Sudarshan. 2005. Database Systems Concepts (5 ed.), PP. 628–630. McGraw-Hill, Inc., New York, NY, USA.

References

[1] http://portal.acm.org/citation.cfm?id=1202608&dl=GUIDE&coll=GUIDE&CFID=9802819&CFTOKEN=82728908

Serializability theory

In concurrency control of databases,[1][2] transaction processing (transaction management), and various transactional applications (e.g., transactional memory[3] and software transactional memory), both centralized and distributed, a transaction schedule is **serializable** if its outcome (e.g., the resulting database state) is equal to the outcome of its transactions executed serially, i.e., sequentially without overlapping in time. Transactions are normally executed concurrently (they overlap), since this is the most efficient way. Serializability is the major correctness criterion for concurrent transactions' executions. It is considered the highest level of isolation between transactions, and plays an essential role in concurrency control. As such it is supported in all general purpose database systems. *Strong strict two-phase locking* (SS2PL) is a popular serializability mechanism utilized in most of the database systems (in various variants) since their early days in the 1970s.

Serializability theory provides the formal framework to reason about and analyze serializability and its techniques. Though it is mathematical in nature, its fundamentals are informally (without Mathematics notation) introduced below.

Database transaction

For this discussion a *database transaction* is a specific intended run (with specific parameters, e.g., with transaction identification, at least) of a computer program (or programs) that accesses a database (or databases). Such a program is written with the assumption that it is running in *isolation* from other executing programs, i.e., when running, its accessed data (after the access) are not changed by other running programs. Without this assumption the transaction's results are unpredictable and can be wrong. The same transaction can be executed in different situations, e.g., in different times and locations, in parallel with different programs. A *live* transaction (i.e., exists in a computing

environment with already allocated computing resources; to distinguish from a *transaction request*, waiting to get execution resources) can be in one of three states, or phases:

1. *Running* - Its program(s) is (are) executing.
2. *Ready* - Its program's execution has ended, and it is waiting to be *Ended (Completed)*.
3. *Ended* (or *Completed*) - It is either *Committed* or *Aborted (Rolled-back)*, depending whether the execution is considered a success or not, respectively . When committed, all its *recoverable* (i.e., with states that can be controlled for this purpose), *durable* resources (typically *database data*) are put in their *final* states, states after running. When aborted, all its recoverable resources are put back in their *initial* states, as before running.

A failure in transaction's computing environment before ending typically results in its abort. However, a transaction may be aborted also for other reasons as well (e.g., see below).

Upon being ended (completed), transaction's allocated computing resources are released and the transaction disappears from the computing environment. However, the effects of a committed transaction remain in the database, while the effects of an aborted (rolled-back) transaction disappear from the database. The concept of *atomic transaction* ("all or nothing" semantics) was designed to exactly achieve this behavior, in order to control correctness in complex faulty systems.

Correctness

Correctness - serializability

Serializability is a property of a transaction schedule (history). It relates to the *isolation* property of a database transaction.

> **Serializability** of a schedule means equivalence (in the outcome, the database state, data values) to a *serial schedule* (i.e., sequential with no transaction overlap in time) with the same transactions. It is the major criterion for the correctness of concurrent transactions' schedule, and thus supported in all general purpose database systems.
>
> **The rationale behind serializability** is the following:
>
> If each transaction is correct by itself, i.e., meets certain integrity conditions, then a schedule that comprises any *serial* execution of these transactions is correct (its transactions still meet their conditions): "Serial" means that transactions do not overlap in time and cannot interfere with each other, i.e, complete *isolation* between each other exists. Any order of the transactions is legitimate, if no dependencies among them exist, which is assumed (see comment below). As a result, a schedule that comprises any execution (not necessarily serial) that is equivalent (in its outcome) to any serial execution of these transactions, is correct.

Schedules that are not serializable are likely to generate erroneous outcomes. Well known examples are with transactions that debit and credit accounts with money: If the related schedules are not serializable, then the total sum of money may not be preserved. Money could disappear, or be generated from nowhere. This and violations of possibly needed other invariant preservations are caused by one transaction writing, and "stepping on" and erasing what has been written by another transaction before it has become permanent in the database. It does not happen if serializability is maintained.

If any specific order between some transactions is requested by an application, then it is enforced independently of the underlying serializability mechanisms. These mechanisms are typically indifferent to any specific order, and generate some unpredictable partial order that is typically compatible with multiple serial orders of these transactions. This partial order results from the scheduling orders of concurrent transactions' data access operations, which depend on many factors.

Correctness - recoverability

A major characteristic of a database transaction is *atomicity*, which means that it either *commits*, i.e., all its operations' results take effect in the database, or *aborts* (rolled-back), all its operations' results do not have any effect on the database ("all or nothing" semantics of a transaction). In all real systems transactions can abort for many reasons, and serializability by itself is not sufficient for correctness. Schedules also need to possess the *recoverability* (from abort) property. **Recoverability** means that committed transactions have not read data written by aborted transactions (whose effects do not exist in the resulting database states). While serializability is currently compromised on purpose in many applications for better performance (only in cases when application's correctness is not harmed), compromising recoverability would quickly violate the database's integrity, as well as that of transactions' results external to the database. A schedule with the recoverability property (a *recoverable* schedule) "recovers" from aborts by itself, i.e., aborts do not harm the integrity of its committed transactions and resulting database. This is false without recoverability, where the likely integrity violations (resulting incorrect database data) need special, typically manual, corrective actions in the database.

Implementing recoverability in its general form may result in *cascading aborts*: Aborting one transaction may result in a need to abort a second transaction, and then a third, and so on. This results in a waste of already partially executed transactions, and may result also in a performance penalty. **Avoiding cascading aborts** (ACA, or Cascadelessness) is a special case of recoverability that exactly prevents such phenomenon. Often in practice a special case of ACA is utilized: **Strictness**. Strictness allows an efficient database recovery from failure.

Note that the *recoverability* property is needed even if no database failure occurs and no database *recovery* from failure is needed. It is rather needed to correctly automatically handle aborts, which may be unrelated to database failure and recovery from failure.

Relaxing serializability

In many applications, unlike with finances, absolute correctness is not needed. For example, when retrieving a list of products according to specification, in most cases it does not matter much if a product, whose data was updated a short time ago, does not appear in the list, even if it meets the specification. It will typically appear in such a list when tried again a short time later. Commercial databases provide concurrency control with a whole range of isolation levels which are in fact (controlled) serializability violations in order to achieve higher performance. Higher performance means better transaction execution rate and shorter average transaction response time (transaction duration). *Snapshot isolation* is an example of a popular, widely utilized efficient relaxed serializability method with many characteristics of full serializability, but still short of some, and unfit in many situations.

Another common reason nowadays for distributed serializability relaxation (see below) is the requirement of availability of internet products and services. This requirement is typically answered by large-scale data replication. The straightforward solution for synchronizing replicas' updates of a same database object is including all these updates in a single atomic distributed transaction. However, with many replicas such a transaction is very large, and may span several computers and networks that some of them are likely to be unavailable. Thus such a transaction is likely to end with abort and miss its purpose. Consequently Optimistic replication (Lazy replication) is often utilized (e.g., in many products and services by Google, Amazon, Yahoo, and alike), while serializability is relaxed and compromised for eventual consistency. Again in this case, relaxation is done only for applications that are not expected to be harmed by this technique.

Classes of schedules defined by *relaxed serializability* properties either contain the serializability class, or are incomparable with it.

View and conflict serializability

Mechanisms that enforce serializability need to execute in real time, or almost in real time, while transactions are running at high rates. In order to meet this requirement special cases of serializability, sufficient conditions for serializability which can be enforced effectively, are utilized.

Two major types of serializability exist: *view-serializability*, and *conflict-serializability*. View-serializability matches the general definition of serializability given above. Conflict-serializability is a broad special case, i.e., any schedule that is conflict-serializable is also view-serializable, but not necessarily the opposite. Conflict-serializability is widely utilized because it is easier to determine and covers a substantial portion of the view-serializable schedules. Determining view-serializability of a schedule is an NP-complete problem (a class of problems with only difficult-to-compute, excessively time-consuming known solutions).

> **View-serializability** of a schedule is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that respective transactions in the two schedules read and write the same data values ("view" the same data values).
>
> **Conflict-serializability** is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that both schedules have the same sets of respective chronologically ordered pairs of conflicting operations (same precedence relations of respective conflicting operations).

Operations upon data are *read* or *write* (a write: either *insert* or *modify* or *delete*). Two operations are *conflicting*, if they are of different transactions, upon the same datum (data item), and at least one of them is *write*. Each such pair of conflicting operations has a *conflict type*: It is either a *read-write*, or *write-read*, or a *write-write* conflict. The transaction of the second operation in the pair is said to be *in conflict* with the transaction of the first operation. A more general definition of conflicting operations (also for complex operations, which may consist each of several "simple" read/write operations) requires that they are noncommutative (changing their order also changes their combined result). Each such operation needs to be atomic by itself (by proper system support) in order to be considered an operation for a commutativity check. For example, read-read operations are commutative (unlike read-write and the other possibilities) and thus read-read is not a conflict. Another more complex example: the operations *increment* and *decrement* of a *counter* are both *write* operations (both modify the counter), but do not need to be considered conflicting (write-write conflict type) since they are commutative (thus increment-decrement is not a conflict; e.g., already has been supported in the old IBM's IMS "fast path"). Only precedence (time order) in pairs of conflicting (non-commutative) operations is important when checking equivalence to a serial schedule, since different schedules consisting of the same transactions can be transformed from one to another by changing orders between different transactions' operations (different transactions' interleaving), and since changing orders of commutative operations (non-conflicting) does not change an overall operation sequence result, i.e., a schedule outcome (the outcome is preserved through order change between non-conflicting operations, but typically not when conflicting operations change order). This means that if a schedule can be transformed to any serial schedule without changing orders of conflicting operations (but changing orders of non-conflicting, while preserving operation order inside each transaction), then the outcome of both schedules is the same, and the schedule is conflict-serializable by definition.

Conflicts are the reason for blocking transactions and delays (non-materialized conflicts), or for aborting transactions due to serializability violations prevention. Both possibilities reduce performance. Thus reducing the number of conflicts, e.g., by commutativity (when possible), is a way to increase performance.

A transaction can issue/request a conflicting operation and be *in conflict* with another transaction while its conflicting operation is delayed and not executed (e.g., blocked by a lock). Only executed (*materialized*) conflicting operations are relevant to *conflict serializability* (see more below).

Enforcing conflict serializability

Testing conflict serializability

Schedule compliance with conflict serializability can be tested with the precedence graph (*serializability graph*, *serialization graph*, *conflict graph*) for committed transactions of the schedule. It is the directed graph representing precedence of transactions in the schedule, as reflected by precedence of conflicting operations in the transactions.

> In the **precedence graph** transactions are nodes and precedence relations are directed edges. There exists an edge from a first transaction to a second transaction, if the second transaction is *in conflict* with the first (see Conflict serializability above), and the conflict is **materialized** (i.e., if the requested conflicting operation is actually executed: in many cases a requested/issued conflicting operation by a transaction is delayed and even never executed, typically by a lock on the operation's object, held by another transaction, or when writing to a transaction's temporary private workspace and materializing, copying to the database itself, upon commit; as long as a requested/issued conflicting operation is not executed upon the database itself, the conflict is **non-materialized**; non-materialized conflicts are not represented by an edge in the precedence graph).
>
> **Comment:** In many text books only *committed transactions* are included in the precedence graph. Here all transactions are included for convenience in later discussions.

The following observation is a **key characterization of conflict serializability**:

> A schedule is *conflict-serializable* if and only if its precedence graph of *committed transactions* (when only *committed* transactions are considered) is *acyclic*. This means that a cycle consisting of committed transactions only is generated in the (general) precedence graph, if and only if conflict-serializability is violated.

Cycles of committed transactions can be prevented by aborting an *undecided* (neither committed, nor aborted) transaction on each cycle in the precedence graph of all the transactions, which can otherwise turn into a cycle of committed transactions (and a committed transaction cannot be aborted). One transaction aborted per cycle is both required and sufficient number to break and eliminate the cycle (more aborts are possible, and can happen in some mechanisms, but unnecessary for serializability). The probability of cycle generation is typically low, but nevertheless, such a situation is carefully handled, typically with a considerable overhead, since correctness is involved. Transactions aborted due to serializability violation prevention are *restarted* and executed again immediately.

Serializability enforcing mechanisms typically do not maintain a precedence graph as a data structure, but rather prevent or break cycles implicitly (e.g., SS2PL below).

Common mechanism - SS2PL

Strong strict two phase locking (SS2PL) is a common mechanism utilized in database systems since their early days in the 1970s (the "SS" in the name SS2PL is newer though) to enforce both conflict serializability and *strictness* (a special case of recoverability which allows effective database recovery from failure) of a schedule. In this mechanism each datum is locked by a transaction before accessing it (any read or write operation): The item is marked by, associated with a *lock* of a certain type, depending on operation (and the specific implementation; various models with different lock types exist; in some models locks may change type during the transaction's life). As a result access by another transaction may be blocked, typically upon a conflict (the lock delays or completely prevents the conflict from being materialized and be reflected in the precedence graph by blocking the conflicting operation), depending on lock type and the other transaction's access operation type. Employing an SS2PL mechanism means that all locks on data on behalf of a transaction are released only after the transaction has ended (either committed or aborted).

SS2PL is the name of the resulting schedule property as well, which is also called *rigorousness*. SS2PL is a special case (proper subset) of Two-phase locking (2PL)

Mutual blocking between transactions results in a *deadlock*, where execution of these transactions is stalled, and no completion can be reached. Thus deadlocks need to be resolved to complete these transactions' execution and release related computing resources. A deadlock is a reflection of a potential cycle in the precedence graph, that would occur without the blocking when conflicts are materialized. A deadlock is resolved by aborting a transaction involved with such potential cycle, and breaking the cycle. It is often detected using a *wait-for graph* (a graph of conflicts blocked by locks from being materialized; it can be also defined as the graph of non-materialized conflicts; conflicts not materialized are not reflected in the precedence graph and do not affect serializability), which indicates which transaction is "waiting for" lock release by which transaction, and a cycle means a deadlock. Aborting one transaction per cycle is sufficient to break the cycle. Transactions aborted due to deadlock resolution are *restarted* and executed again immediately.

Other enforcing techniques

Other known mechanisms include:

- Precedence graph (or Serializability graph, Conflict graph) cycle elimination
- Two-phase locking (2PL)
- Timestamp ordering (TO)
- Serializable snapshot isolation[4] (SerializableSI)

The above (conflict) serializability techniques in their general form do not provide recoverability. Special enhancements are needed for adding recoverability.

Optimistic versus pessimistic techniques

Concurrency control techniques are of three major types:

1. *Pessimistic*: In Pessimistic concurrency control a transaction blocks data access operations of other transactions upon conflicts, and conflicts are *non-materialized* until blocking is removed. This is done to ensure that operations that may violate serializability (and in practice also recoverability) do not occur.
2. *Optimistic*: In Optimistic concurrency control data access operations of other transactions are not blocked upon conflicts, and conflicts are immediately *materialized*. When the transaction reaches the *ready* state, i.e., its *running* state has been completed, possible serializability (and in practice also recoverability) violation by the transaction's operations (relatively to other running transactions) is checked: If violation has occurred, the transaction is typically *aborted* (sometimes aborting *another* transaction to handle serializability violation is preferred). Otherwise it is *committed*.
3. *Semi-optimistic*: Mechanisms that mix blocking in certain situations with not blocking in other situations and employ both materialized and non-materialized conflicts

The main differences between the technique types is the conflict types that are generated by them. A pessimistic method blocks a transaction operation upon conflict and generates a non-materialized conflict, while an optimistic method does not block and generates a materialized conflict. A semi-optimistic method generates both conflict types. Both conflict types are generated by the chronological orders in which transaction operations are invoked, independently of the type of conflict. A cycle of committed transactions (with materialized conflicts) in the *precedence graph* (conflict graph) represents a serializability violation, and should be avoided for maintaining serializability. A cycle of (non-materialized) conflicts in the *wait-for graph* represents a deadlock situation, which should be resolved by breaking the cycle. Both cycle types result from conflicts, and should be broken. At any technique type conflicts should be detected and considered, with similar overhead for both materialized and non-materialized conflicts (typically by using mechanisms like locking, while either blocking for locks, or not blocking but recording conflict for materialized conflicts). In a blocking method typically a context switching occurs upon conflict, with (additional) incurred overhead. Otherwise blocked transactions' related computing resources remain idle, unutilized, which may be a worse alternative. When conflicts do not occur frequently, optimistic

methods typically have an advantage. With different transactions loads (mixes of transaction types) one technique type (i.e., either optimistic or pessimistic) may provide better performance than the other.

Unless schedule classes are *inherently blocking* (i.e., they cannot be implemented without data-access operations blocking; e.g., 2PL, SS2PL and SCO above; see chart), they can be implemented also using optimistic techniques (e.g., Serializability, Recoverability).

Serializable multi-version concurrency control

See also Multiversion concurrency control (partial coverage)

and Serializable_Snapshot_Isolation in Snapshot isolation

Multi-version concurrency control (MVCC) is a common way today to increase concurrency and performance by generating a new version of a database object each time the object is written, and allowing transactions' read operations of several last relevant versions (of each object), depending on scheduling method. MVCC can be combined with all the serializability techniques listed above (except SerializableSI which is originally MVCC based). It is utilized in most general-purpose DBMS products.

MVCC is especially popular nowadays through the *relaxed serializability* (see above) method *Snapshot isolation* (SI) which provides better performance than most known serializability mechanisms (at the cost of possible serializability violation in certain cases). SerializableSI, which is an efficient enhancement of SI to make it serializable, is intended to provide an efficient serializable solution. SerializableSI has been analyzed[5] via a general theory of MVCC

Distributed serializability

Overview

Distributed serializability is the serializability of a schedule of a transactional distributed system (e.g., a distributed database system). Such system is characterized by *distributed transactions* (also called *global transactions*), i.e., transactions that span computer processes (a process abstraction in a general sense, depending on computing environment; e.g., operating system's thread) and possibly network nodes. A distributed transaction comprises more than one *local sub-transactions* that each has states as described above for a database transaction. A local sub-transaction comprises a single process, or more processes that typically fail together (e.g., in a single processor core). Distributed transactions imply a need in Atomic commit protocol to reach consensus among its local sub-transactions on whether to commit or abort. Such protocols can vary from a simple (one-phase) hand-shake among processes that fail together, to more sophisticated protocols, like Two-phase commit, to handle more complicated cases of failure (e.g., process, node, communication, etc. failure). Distributed serializability is a major goal of distributed concurrency control for correctness. With the proliferation of the Internet, Cloud computing, Grid computing, and small, portable, powerful computing devices (e.g., smartphones) the need for effective distributed serializability techniques to ensure correctness in and among distributed applications seems to increase.

Distributed serializability is achieved by implementing distributed versions of the known centralized techniques. Typically all such distributed versions require utilizing conflict information (either of materialized or non-materialized conflicts, or equivalently, transaction precedence or blocking information; conflict serializability is usually utilized) that is not generated locally, but rather in different processes, and remote locations. Thus information distribution is needed (e.g., precedence relations, lock information, timestamps, or tickets). When the distributed system is of a relatively small scale, and message delays across the system are small, the centralized concurrency control methods can be used unchanged, while certain processes or nodes in the system manage the related algorithms. However, in a large-scale system (e.g., *Grid* and *Cloud*), due to the distribution of such information, substantial performance penalty is typically incurred, even when distributed versions of the methods (Vs. centralized) are used, primarily due to computer and communication latency. Also, when such information is

distributed, related techniques typically do not scale well. A well-known example with scalability problems is a distributed lock manager, which distributes lock (non-materialized conflict) information across the distributed system to implement locking techniques.

Notes

[1] Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* (http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx) (free PDF download), Addison Wesley Publishing Company, ISBN 0-201-10715-5

[2] Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description), Elsevier, ISBN 1-55860-508-8

[3] Maurice Herlihy and J. Eliot B. Moss. *Transactional memory: architectural support for lock-free data structures.* Proceedings of the 20th annual international symposium on Computer architecture (ISCA '93). Volume 21, Issue 2, May 1993.

[4] Michael J. Cahill, Uwe Röhm, Alan D. Fekete (2008): "Serializable isolation for snapshot databases" (http://portal.acm.org/citation.cfm?id=1376690), *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 729-738, Vancouver, Canada, June 2008, ISBN 978-1-60558-102-6 (SIGMOD 2008 best paper award)

[5] Alan Fekete (2009), "Snapshot Isolation and Serializable Execution" (http://www.it.usyd.edu.au/~fekete/teaching/serializableSI-Fekete.pdf), Presentation, Page 4, 2009, The university of Sydney (Australia). Retrieved 16 September 2009

References

- Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* (http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx), Addison Wesley Publishing Company, ISBN 0-201-10715-5
- Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description), Elsevier, ISBN 1-55860-508-8

Read–write conflict

In computer science, in the field of databases, **Read-Write Conflict**, also known as **unrepeatable reads**, is a computational anomaly associated with interleaved execution of transactions.

Given a schedule S

$$S = \begin{bmatrix} T1 & T2 \\ R(A) & \\ & R(A) \\ & W(A) \\ & Com. \\ R(A) & \\ W(A) & \\ Com. & \end{bmatrix}$$

In this example, T1 has read the original value of A, and is waiting for T2 to finish. T2 also reads the original value of A, overwrites A, and commits.

However, when T1 reads to A, it discovers two different versions of A, and T1 would be forced to abort, because T1 would not know what to do. This is an unrepeatable read. This could never occur in a serial schedule. Strict two-phase locking (Strict 2PL) prevents this conflict.

Real world example

Alice and Bob are using Ticketmaster website to book tickets for a specific show. Only one ticket is left for the specific show. Alice signs on to Ticketmaster first and finds one left, and finds it expensive. Alice takes time to decide. Bob signs on and finds one ticket left, orders it instantly. Bob purchases and logs off. Alice decides to buy a ticket, to find there are no tickets. This is a typical Read-Write Conflict situation.

Write–read conflict

In computer science, in the field of databases, **Write-Read Conflict**, also known as **reading uncommitted data**, is a computational anomaly associated with interleaved execution of transactions.

Given a schedule S

$$S = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ & R(B) \\ & W(B) \\ & Com. \\ R(B) & \\ W(B) & \\ Com. & \end{bmatrix}$$

T2 could read a database object A, modified by T1 which hasn't committed. This is a ***dirty read***.

T1 may write some value into A which makes the database inconsistent. It is possible that interleaved execution can expose this inconsistency and lead to inconsistent final database state, violating ACID rules.

Strict 2PL overcomes this inconsistency by locking T2 out from performing a Read/Write on A. Note however that Strict 2PL can have a number of drawbacks, such as the possibility of deadlocks.

References

Write–write conflict

In computer science, in the field of databases, **Write-Write Conflict**, also known as **overwriting uncommitted data** is a computational anomaly associated with interleaved execution of transactions.

Given a schedule S

$$S = \begin{bmatrix} T1 & T2 \\ W(A) & \\ & W(B) \\ W(B) & \\ Com. & \\ & W(A) \\ & Com. \end{bmatrix}$$

note that there is no read in this schedule. The writes are called ***blind writes***.

We have a ***lost update***. Any attempts to make this schedule serial would give off two different results (either T1's version of A and B is shown, or T2's version of A and B is shown), and would not be the same as the above schedule. This schedule would not be serializable.

Strict 2PL, overcomes this inconsistency by locking T1 out from B. Unfortunately, deadlocks are something Strict 2PL does not overcome all the time.

References

Lock (database)

A **lock**, as a **read lock** or **write lock**, is used when multiple users need to access a database concurrently. This prevents data from being corrupted or invalidated when multiple users try to read while others write to the database. Any single user can only modify those database records (that is, items in the database) to which they have applied a lock that gives them exclusive access to the record until the lock is released. Locking not only provides exclusivity to writes but also prevents (or controls) reading of unfinished modifications (AKA uncommitted data).

A read lock can be used to prevent other users from reading a record (or page) which is being updated, so that others will not act upon soon-to-be-outdated information.

Mechanisms for locking

There are two mechanisms for locking data in a database: *pessimistic locking*, and *optimistic locking*. In pessimistic locking a record or page is locked immediately when the lock is requested, while in an optimistic lock the record or page is only locked when the changes made to that record are updated. The latter situation is only appropriate when there is less chance of someone needing to access the record while it is locked; otherwise it cannot be certain that the update will succeed because the attempt to update the record will fail if another user updates the record first. With pessimistic locking it is guaranteed that the record will be updated.

The degree of locking can be controlled by isolation level. Change of a lock is called lock conversion and the lock may be upgraded (lock upgrade) or downgraded (lock downgrade).

Transactional isolation is usually implemented by locking whatever is accessed in a transaction. There are two different approaches to transactional locking: Pessimistic locking and optimistic locking. The disadvantage of pessimistic locking is that a resource is locked from the time it is first accessed in a transaction until the transaction

is finished, making it inaccessible to other transactions during that time. If most transactions simply look at the resource and never change it, an exclusive lock may be overkill as it may cause lock contention, and optimistic locking may be a better approach. With pessimistic locking, locks are applied in a fail-safe way. In the banking application example, an account is locked as soon as it is accessed in a transaction. Attempts to use the account in other transactions while it is locked will either result in the other process being delayed until the account lock is released, or that the process transaction will be rolled back. The lock exists until the transaction has either been committed or rolled back. With optimistic locking, a resource is not actually locked when it is first is accessed by a transaction. Instead, the state of the resource at the time when it would have been locked with the pessimistic locking approach is saved. Other transactions are able to concurrently access to the resource and the possibility of conflicting changes is possible. At commit time, when the resource is about to be updated in persistent storage, the state of the resource is read from storage again and compared to the state that was saved when the resource was first accessed in the transaction. If the two states differ, a conflicting update was made, and the transaction will be rolled back. In the banking application example, the balance of an account is saved when the account is first accessed in a transaction. If the transaction changes the account balance, the balance is read from the store again just before the balance is about to be updated. If the balance has changed since the transaction began, the transaction will fail itself, otherwise the new balance is written to persistent storage. A lock is used when multiple users need to access a database concurrently. This prevents data from being corrupted or invalidated when multiple users try to write to the database. Any single user can only modify those database records (that is, items in the database) to which they have applied a lock that gives them exclusive access to the record until the lock is released. Locking not only provides exclusivity to writes but also prevents (or controls) reading of unfinished modifications (AKA uncommitted data).

References

Record locking

Record locking is the technique of preventing simultaneous access to data in a database, to prevent inconsistent results.

The classic example is demonstrated by two bank clerks attempting to update the same bank account for two different transactions. Clerks 1 and 2 both retrieve (i.e., copy) the account's record. Clerk 1 applies and saves a transaction. Clerk 2 applies a different transaction to his saved copy, and saves the result, based on the original record and his changes, overwriting the transaction entered by clerk 1. The record no longer reflects the first transaction, as if it had never taken place.

A simple way to prevent this is to lock the file whenever a record is being modified by any user, so that no other user can save data. This prevents records from being overwritten incorrectly, but allows only one record to be processed at a time, locking out other users who need to edit records at the same time.

To allow several users to edit a database table at the same time and also prevent inconsistencies created by unrestricted access, a single record can be *locked* when retrieved for editing or updating. Anyone attempting to retrieve the same record for editing is denied write access because of the lock (although, depending on the implementation, they may be able to view the record without editing it). Once the record is saved or edits are canceled, the lock is released. Records can never be saved so as to overwrite other changes, preserving data integrity.

In database management theory, locking is used to implement *isolation* among multiple database users. This is the "I" in the acronym ACID.

A thorough and authoritative description of locking was written by Jim Gray.

Granularity of locks

If the bank clerks (to follow the illustration above) are serving two customers, but their accounts are contained in one ledger, then the entire ledger, or one or more database tables, would need to be made available for editing to the clerks in order for each to complete a transaction, one at a time (file locking). While safe, this method can cause unnecessary waiting.

If the clerks can remove one page from the ledger, containing the account of the current customer (plus several other accounts), then multiple customers can be serviced concurrently, provided that each customer's account is found on a different page than the others. If two customers have accounts on the same page, then only one may be serviced at a time. This is analogous to a *page level lock* in a database.

A higher degree of granularity is achieved if each individual account may be taken by a clerk. This would allow any customer to be serviced without waiting for another customer who is accessing a different account. This is analogous to a *record level lock* and is normally the highest degree of locking granularity in a database management system.

In a SQL database, a record is typically called a "row."

The introduction of granular (subset) locks creates the possibility for a situation called deadlock. Deadlock is possible when *incremental locking* (locking one entity, then locking one or more additional entities) is used. To illustrate, if two bank customers asked two clerks to obtain their account information so they could transfer some money into other accounts, the two accounts would essentially be locked. Then, if the customers told their clerks that the money was to be transferred into each other's accounts, the clerks would search for the other accounts but find them to be "in use" and wait for them to be returned. Unknowingly, the two clerks are waiting for each other, and neither of them can complete their transaction until the other gives up and returns the account. Various techniques are used to avoid such problems.

Use of locks

Record locks need to be managed between the entities requesting the records such that no entity is given too much service via successive **grants**, and no other entity is effectively locked out. The entities that request a lock can be either individual applications (programs) or an entire processor.

The application or system should be designed such that any lock is held for the shortest time possible. Data reading, without editing facilities, does not require a lock, and reading locked records is usually permissible.

Two main types of locks can be requested:

Exclusive locks

Exclusive locks are, as the name implies, exclusively held by a single entity, usually for the purpose of writing to the record. If the locking schema was represented by a list, the **holder list** would contain only one entry. Since this type of lock effectively blocks any other entity that requires the lock from processing, care must be used to:

- ensure the lock is held for the shortest time possible;
- not hold the lock across system or function calls where the entity is no longer running on the processor - this can lead to deadlock;
- ensure that if the entity is unexpectedly exited for any reason, the lock is freed.

Non-holders of the lock (aka **waiters**) can be held in a list that is serviced in a round robin fashion, or in a FIFO queue. This would ensure that any possible waiter would get equal chance to obtain the lock and not be locked out. To further speed up the process, if an entity has gone to sleep waiting for a lock, performance is improved if the entity is notified of the grant, instead of discovering it on some sort of system timeout driven wakeup.

Shared locks

Shared locks differ from exclusive locks in that the **holder list** can contain multiple entries. Shared locks allow all holders to read the contents of the record knowing that the record cannot be changed until after the lock has been released by all holders. Exclusive locks cannot be obtained when a record is already locked (exclusively or shared) by another entity.

If lock requests for the same entity are queued, then once a shared lock is granted, any queued shared locks may also be granted. If an exclusive lock is found next on the queue, it must wait until all shared locks have been released. As with exclusive locks, these shared locks should be held for the least time possible.

References

Multiple granularity locking

In computer science, **multiple granularity locking** (MGL) is a locking method used in database management systems (DBMS) and relational databases.

In MGL, locks are set on objects that contain other objects. MGL exploits the hierarchical nature of the *contains* relationship. For example, a database may have files, which contain pages, which further contain records. This can be thought of as a tree of objects, where each node contains its children. A lock on such as a shared or exclusive lock locks the targeted node as well as all of its descendants.

Multiple granularity locking is usually used with non-strict two-phase locking to guarantee serializability.

Lock Modes

In addition to shared (S) locks and exclusive (X) locks from other locking schemes, like strict two-phase locking, MGL also uses *intention shared* and *intention exclusive* locks. IS locks conflict with X locks, while IX locks conflict with S and X locks. The null lock (NL) is compatible with everything.

To lock a node in S (or X), MGL has the transaction lock on all of its ancestors with IS (or IX), so if a transaction locks a node in S (or X), no other transaction can access its ancestors in X (or S and X). This protocol is shown in the following table:

To Get	Must Have on all Ancestors
IS or S	IS or IX
IX, SIX or X	IX or SIX

Determining what level of granularity to use for locking is done by locking the finest level possible (at the lowest leaf level), and then escalating these locks to higher levels in the file hierarchy to cover more records or file elements as needed. This process is known as Lock Escalation. MGL locking modes are compatible with each other as defined in the following matrix.

Mode	NL	IS	IX	S	SIX	X
NL	Yes	Yes	Yes	Yes	Yes	Yes
IS	Yes	Yes	Yes	Yes	Yes	No
IX	Yes	Yes	Yes	No	No	No
S	Yes	Yes	No	Yes	No	No
SIX	Yes	Yes	No	No	No	No
X	Yes	No	No	No	No	No

Following the locking protocol and the compatibility matrix, if one transaction holds a node in S mode, no other transactions can have locked any ancestor in X mode.

References

Granularity of Locks and Degrees of Consistency, J. Gray, R. Lorie, G.F. Putzolu, and I.L. Traiger, Modeling in Data Base Management Systems, G.M. Nijssen ed., North Holland Pub., 1976, pp. 364-394.

Two-phase locking

In databases and transaction processing, **two-phase locking** (**2PL**) is a concurrency control method that guarantees serializability.[1][2] It is also the name of the resulting set of database transaction schedules (histories). The protocol utilizes locks, applied by a transaction to data, which may block (interpreted as signals to stop) other transactions from accessing the same data during the transaction's life.

By the 2PL protocol locks are applied and removed in two phases:

1. Expanding phase: locks are acquired and no locks are released.
2. Shrinking phase: locks are released and no locks are acquired.

Two types of locks are utilized by the basic protocol: *Shared* and *Exclusive* locks. Refinements of the basic protocol may utilize more lock types. Using locks that block processes, 2PL may be subject to deadlocks that result from the mutual blocking of two or more transactions.

2PL is a superset of **strong strict two-phase locking** (**SS2PL**),[3] also called **rigorousness**,[4] which has been widely utilized for concurrency control in general-purpose database systems since the 1970s. SS2PL implementations have many variants. SS2PL was called *strict 2PL* but this name usage is not recommended now. Now **strict 2PL** (**S2PL**) is the intersection of strictness and 2PL, which is different from SS2PL. SS2PL is also a special case of commitment ordering, and inherits many of CO's useful properties. SS2PL actually comprises only one phase: phase-2 does not exist, and all locks are released only after transaction end. Thus this useful 2PL type is not two-phased at all.

Neither 2PL nor S2PL in their general forms are known to be used in practice. Thus 2PL by itself does not seem to have much practical importance, and whenever 2PL or S2PL utilization has been mentioned in the literature, the intention has been SS2PL. What has made SS2PL so popular (probably the most utilized serializability mechanism) is the effective and efficient locking-based combination of two ingredients (the first does not exist in both general 2PL and S2PL; the second does not exist in general 2PL):

1. Commitment ordering, which provides both serializability, and effective distributed serializability and global serializability, and
2. Strictness, which provides cascadelessness (ACA, cascade-less recoverability) and (independently) allows efficient database recovery from failure.

Additionally SS2PL is easier, with less overhead to implement than both 2PL and S2PL, provides exactly the same locking, but sometimes releases locks later. However, practically (though not simplistically theoretically) such later lock release occurs only slightly later, and this apparent disadvantage is insignificant and disappears next to the advantages of SS2PL.

Thus, the importance of the general *Two-phase locking* (2PL) is historic only, while *Strong strict two-phase locking* (SS2PL) is practically the important mechanism and resulting schedule property.

Data-access locks

A lock is a system object associated with a shared resource such as a data item of an elementary type, a row in a database, or a page of memory. In a database, a lock on a database object (a data-access lock) may need to be acquired by a transaction before accessing the object. Correct use of locks prevents undesired, incorrect or inconsistent operations on shared resources by other concurrent transactions. When a database object with an existing lock acquired by one transaction needs to be accessed by another transaction, the existing lock for the object and the type of the intended access are checked by the system. If the existing lock type does not allow this specific attempted concurrent access type, the transaction attempting access is blocked (according to a predefined agreement/scheme). In practice a lock on an object does not directly block a transaction's operation upon the object, but rather blocks that transaction from acquiring another lock on the same object, needed to be held/owned by the transaction before performing this operation. Thus, with a locking mechanism, needed operation blocking is controlled by a proper lock blocking scheme, which indicates which lock type blocks which lock type.

Two major types of locks are utilized:

- **Write-lock** (**exclusive lock**) is associated with a database object by a transaction (Terminology: "the transaction locks the object," or "acquires lock for it") before *writing* (inserting/modifying/deleting) this object.
- **Read-lock** (**shared lock**) is associated with a database object by a transaction before *reading* (retrieving the state of) this object.

The common interactions between these lock types are defined by blocking behavior as follows:

- An existing *write-lock* on a database object blocks an intended *write* upon the same object (already requested/issued) by another transaction by blocking a respective *write-lock* from being acquired by the other transaction. The second write-lock will be acquired and the requested write of the object will take place (materialize) after the existing write-lock is released.
- A *write-lock* blocks an intended (already requested/issued) *read* by another transaction by blocking the respective *read-lock* .
- A *read-lock* blocks an intended *write* by another transaction by blocking the respective *write-lock* .
- A *read-lock* does not block an intended *read* by another transaction. The respective *read-lock* for the intended read is acquired (shared with the previous read) immediately after the intended read is requested, and then the intended read itself takes place.

Several variations and refinements of these major lock types exist, with respective variations of blocking behavior. If a first lock blocks another lock, the two locks are called *incompatible*; otherwise the locks are *compatible*. Often lock types blocking interactions are presented in the technical literature by a *Lock compatibility table*. The following is an example with the common, major lock types:

Lock compatibility table

Lock type	read-lock	write-lock
read-lock		**X**
write-lock	**X**	**X**

X indicates incompatibility, i.e, a case when a lock of the first type (in left column) on an object blocks a lock of the second type (in top row) from being acquired on the same object (by another transaction). An object typically has a queue of waiting requested (by transactions) operations with respective locks. The first blocked lock for operation in the queue is acquired as soon as the existing blocking lock is removed from the object, and then its respective operation is executed. If a lock for operation in the queue is not blocked by any existing lock (existence of multiple compatible locks on a same object is possible concurrently) it is acquired immediately.

Comment: In some publications the table entries are simply marked "compatible" or "incompatible", or respectively "yes" or "no".

Two-phase locking and its special cases

Two-phase locking

According to the **two-phase locking** protocol a transaction handles its locks in two distinct, consecutive phases during the transaction's execution:

1. **Expanding phase** (aka Growing phase): locks are acquired and no locks are released (the number of locks can only increase).
2. **Shrinking phase**: locks are released and no locks are acquired.

The serializability property is guaranteed for a schedule with transactions that obey the protocol. The 2PL *schedule class* is defined as the class of all the schedules comprising transactions with data access orders that could be generated by the 2PL protocol (or in other words, all the schedules that the 2PL protocol can generate).

Typically, without explicit knowledge in a transaction on end of phase-1, it is safely determined only when a transaction has entered its *ready* state in all its processes (processing has ended, and it is ready to be committed; no additional data access and locking are needed and can happen). In this case phase-2 can end immediately (no additional processing is needed), and actually no phase-2 is needed. Also, if several processes (two or more) are involved, then a synchronization point (similar to *atomic commitment*) among them is needed to determine end of phase-1 for all of them (i.e., in the entire distributed transaction), to start releasing locks in phase-2 (otherwise it is very likely that both 2PL and Serializability are quickly violated). Such synchronization point is usually too costly (involving a distributed protocol similar to atomic commitment), and end of phase-1 is usually postponed to be merged with transaction end (atomic commitment protocol for a multi-process transaction), and again phase-2 is not needed. This turns 2PL to SS2PL (see below). All known implementations of 2PL in products are SS2PL based, and whenever 2PL (or Strict 2PL, S2PL) practical utilization has been mentioned in the professional literature, the intention has been SS2PL.

Strict two-phase locking

The **strict two-phase locking** (S2PL) class of schedules is the intersection of the 2PL class with the class of schedules possessing the *Strictness* property.

To comply with the S2PL protocol a transaction needs to comply with 2PL, and release its *write (exclusive)* locks only after it has ended, i.e., being either *committed* or *aborted*. On the other hand, *read (shared)* locks are released regularly during phase 2. Implementing general S2PL requires explicit support of phase-1 end, separate from transaction end, and no such widely utilized product implementation is known.

S2PL is a special case of 2PL, i.e., the S2PL class is a proper subclass of 2PL

Strong strict two-phase locking

or **Rigorousness**, or **Rigorous scheduling**, or **Rigorous two-phase locking**

To comply with **strong strict two-phase locking** (SS2PL) the locking protocol releases both *write (exclusive)* and *read (shared)* locks applied by a transaction only after the transaction has ended, i.e., only after both completing executing (being *ready*) and becoming either *committed* or *aborted*. This protocol also complies with the S2PL rules. A transaction obeying SS2PL can be viewed as having phase-1 that lasts the transaction's entire execution duration, and no phase-2 (or a degenerate phase-2). Thus, only one phase is actually left, and "two-phase" in the name seems to be still utilized due to the historical development of the concept from 2PL, and 2PL being a super-class. The SS2PL property of a schedule is also called **Rigorousness**. It is also the name of the class of schedules having this property, and an SS2PL schedule is also called a "rigorous schedule". The term "Rigorousness" is free of the unnecessary legacy of "two-phase," as well as being independent of any (locking) mechanism (in principle other blocking mechanisms can be utilized). The property's respective locking mechanism is sometimes referred to as **Rigorous 2PL**.

SS2PL is a special case of S2PL, i.e., the SS2PL class of schedules is a proper subclass of S2PL (every SS2PL schedule is also an S2PL schedule, but S2PL schedules exist that are not SS2PL).

SS2PL has been the concurrency control protocol of choice for most database systems and utilized since their early days in the 1970s. It is proven to be an effective mechanism in many situations, and provides besides Serializability also Strictness (a special case of cascadeless Recoverability), which is instrumental for efficient database recovery, and also Commitment ordering (CO) for participating in distributed environments where a CO based distributed serializability and global serializability solutions are employed. Being a subset of CO, an efficient implementation of *distributed SS2PL* exists without a distributed lock manager (DLM), while distributed deadlocks (see below) are resolved automatically. The fact that SS2PL employed in multi database systems ensures global serializability has been known for years before the discovery of CO, but only with CO came the understanding of the role of an atomic commitment protocol in maintaining global serializability, as well as the observation of automatic distributed deadlock resolution (see a detailed example of Distributed SS2PL). As a matter of fact, SS2PL inheriting properties of Recoverability and CO is more significant than being a subset of 2PL, which by itself in its general form, besides comprising a simple serializability mechanism (however serializability is also implied by CO), in not known to provide SS2PL with any other significant qualities. 2PL in its general form, as well as when combined with Strictness, i.e., Strict 2PL (S2PL), are not known to be utilized in practice. The popular SS2PL does not require to mark "end of phase-1" as 2PL and S2PL do, and thus is simpler to implement. Also unlike the general 2PL, SS2PL provides, as mentioned above, the useful Strictness and Commitment ordering properties.

Many variants of SS2PL exist that utilize various lock types with various semantics in different situations, including cases of lock-type change during a transaction. Notable are variants that use Multiple granularity locking.

Comments:

1. SS2PL Vs. S2PL: Both provide Serializability and Strictness. Since S2PL is a super class of SS2PL it may, in principle, provide more concurrency. However no concurrency advantage is typically practically noticed (exactly

same locking exists for both, with practically not much earlier lock release for S2PL), and the overhead of dealing with an end-of-phase-1 mechanism in S2PL, separate from transaction-end, is not justified. Also, while SS2PL provides Commitment ordering, S2PL does not. This explains the preference of SS2PL over S2PL.

2. Especially before 1990, but also after, in many articles and books, e.g., (Bernstein et al. 1987, p. 59), the term "Strict 2PL" (S2PL) has been frequently defined by the locking protocol "Release all locks only after transaction end," which is the protocol of SS2PL. Thus, "Strict 2PL" could not be there the name of the intersection of Strictness and 2PL, which is larger than the class generated by the SS2PL protocol. This has caused confusion. With an explicit definition of S2PL as the intersection of Strictness and 2PL, a new name for SS2PL, and an explicit distinction between the classes S2PL and SS2PL, the articles (Breitbart et al. 1991) and (Raz 1992) have intended to clear the confusion: The first using the name "Rigorousness," and the second "SS2PL."
3. A more general property than SS2PL exists (a schedule super-class), **Strict commitment ordering** (Strict CO, or SCO), which as well provides both serializability, strictness, and CO, and has similar locking overhead. Unlike SS2PL, SCO does not block upon a read-write conflict (a read-lock does not block acquiring a write-lock; both SCO and SS2PL have the same behavior for write-read and write-write conflicts) at the cost of a possible delayed commit, and upon such conflict type SCO has shorter average transaction completion time and better performance than SS2PL.[5] While SS2PL obeys the *lock compatibility table* above, SCO has the following table:

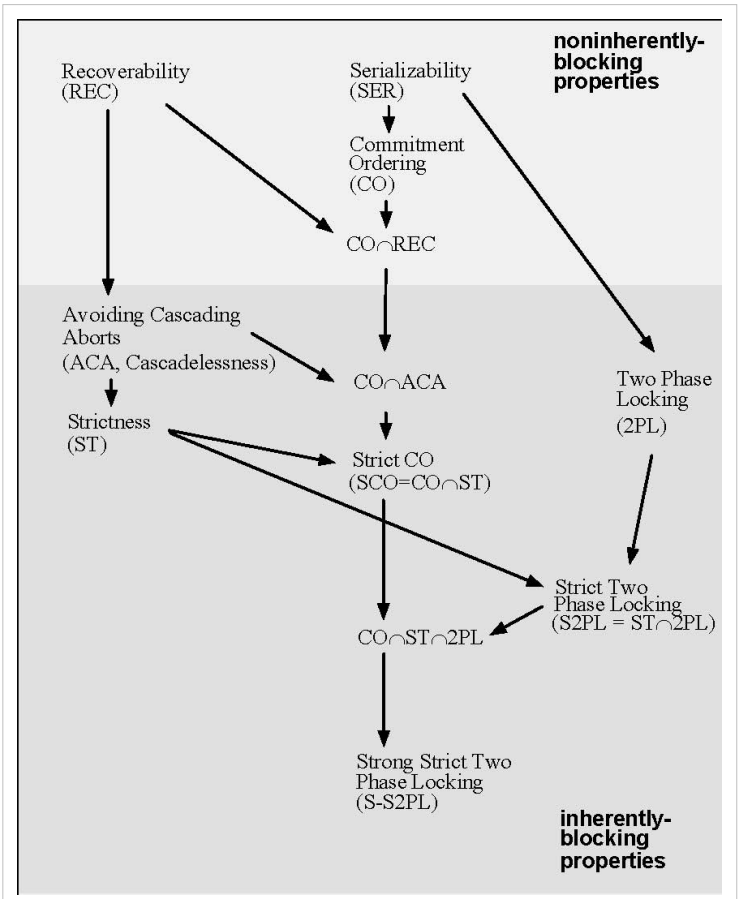
Lock compatibility for SCO

Lock type	read-lock	write-lock
read-lock		
write-lock	**X**	**X**

Note that though SCO releases all locks at transaction end and complies with the 2PL locking rules, SCO is not a subset of 2PL because of its different lock compatibility table. SCO allows materialized read-write conflicts between two transactions in their phases 1, which 2PL does not allow in phase-1 (see about materialized conflicts in Serializability). On the other hand 2PL allows other materialized conflict types in phase-2 that SCO does not allow at all. Together this implies that the schedule classes 2PL and SCO are incomparable (i.e., no class contains the other class).

Summary - Relationships among classes

Between any two schedule classes (define by their schedules' respective properties) that have common schedules, either one *contains* the other (*strictly contains* if they are not equal), or they are *incomparable*. The containment relationships among the 2PL classes and other major schedule classes are summarized in the following diagram. 2PL and its subclasses are *inherently blocking*, which means that no optimistic implementations for them exist (and whenever "Optimistic 2PL" is mentioned it refers to a different mechanism with a class that includes also schedules not in the 2PL class).



Schedule classes containment: An arrow from class A to class B indicates that class A strictly contains B; a lack of a directed path between classes means that the classes are incomparable.A property is inherently blocking, if it can be enforced only by blocking transaction's data access operations until certain events occur in other transactions. (#Raz1992Raz 1992)

Deadlocks in 2PL

Locks block data-access operations. Mutual blocking between transactions results in a *deadlock*, where execution of these transactions is stalled, and no completion can be reached. Thus deadlocks need to be resolved to complete these transactions' executions and release related computing resources. A deadlock is a reflection of a potential cycle in the *precedence graph*, that would occur without the blocking. A deadlock is resolved by aborting a transaction involved with such potential cycle, and breaking the cycle. It is often detected using a *wait-for graph* (a graph of conflicts blocked by locks from being materialized; conflicts not materialized in the database due to blocked operations are not reflected in the precedence graph and do not affect *serializability*), which indicates which transaction is "waiting for" lock release by which transaction, and a cycle means a deadlock. Aborting one transaction per cycle is sufficient to break the cycle. Transactions aborted due to deadlock resolution are executed again immediately.

In a distributed environment an atomic commitment protocol, typically the Two-phase commit (2PC) protocol, is utilized for atomicity. When recoverable data (data under transaction control) are partitioned among 2PC participants (i.e., each data object is controlled by a single 2PC participant), then distributed (global) deadlocks, deadlocks involving two or more participants in 2PC, are resolved automatically as follows:

When SS2PL is effectively utilized in a distributed environment, then global deadlocks due to locking generate voting-deadlocks in 2PC, and are resolved automatically by 2PC (see Commitment ordering (CO), in Exact characterization of voting-deadlocks by global cycles; No reference except the CO articles is known to notice this). For the general case of 2PL, global deadlocks are similarly resolved automatically by the synchronization point protocol of phase-1 end in a distributed transaction (synchronization point is achieved by "voting" (notifying local phase-1 end), and being propagated to the participants in a distributed transaction the same way as a decision point in atomic commitment; in analogy to decision point in CO, a conflicting operation in 2PL cannot happen before

phase-1 end synchronization point, with the same resulting voting-deadlock in the case of a global data-access deadlock; the voting-deadlock (which is also a locking based global deadlock) is automatically resolved by the protocol aborting some transaction involved, with a missing vote, typically using a timeout).

Comment:

When data are partitioned among the *atomic commitment protocol* (e.g., 2PC) participants, automatic *global deadlock* resolution has been overlooked in the database research literature, though deadlocks in such systems has been a quite intensive research area:

- For CO and its special case SS2PL, the automatic resolution by the *atomic commitment protocol* has been noticed only in the CO articles. However, it has been noticed in practice that in many cases global deadlocks are very infrequently detected by the dedicated resolution mechanisms, less than could be expected ("Why do we see so few global deadlocks?"). The reason is probably the deadlocks that are automatically resolved and thus not handled and uncounted by the mechanisms;
- For 2PL in general, the automatic resolution by the (mandatory) *end-of-phase-one synchronization point protocol* (which has same voting mechanism as atomic commitment protocol, and same missing vote handling upon voting deadlock, resulting in global deadlock resolution) has not been mentioned until today (2009). Practically only the special case SS2PL is utilized, where no end-of-phase-one synchronization is needed in addition to atomic commit protocol.

In a distributed environment where recoverable data are not partitioned among atomic commitment protocol participants, no such automatic resolution exists, and distributed deadlocks need to be resolved by dedicated techniques.

References

[1] Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* (http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx), Addison Wesley Publishing Company, ISBN 0-201-10715-5

[2] Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description), Elsevier, ISBN 1-55860-508-8

[3] Yoav Raz (1992): "The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Managers Using Atomic Commitment" (http://www.informatik.uni-trier.de/~ley/db/conf/vldb/Raz92.html) (PDF (http://www.vldb.org/conf/1992/P292.PDF)), *Proceedings of the Eighteenth International Conference on Very Large Data Bases* (VLDB), pp. 292-312, Vancouver, Canada, August 1992, ISBN 1-55860-151-1 (also DEC-TR 841, Digital Equipment Corporation, November 1990)

[4] Yuri Breitbart, Dimitrios Georgakopoulos, Marek Rusinkiewicz, Abraham Silberschatz (1991): "On Rigorous Transaction Scheduling" (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=92915), *IEEE Transactions on Software Engineering* (TSE), September 1991, Volume 17, Issue 9, pp. 954-960, ISSN: 0098-5589

[5] Yoav Raz (1991): "Locking Based Strict Commitment Ordering, or How to improve Concurrency in Locking Based Resource Managers", DEC-TR 844, December 1991.

Readers–writer lock

In computer science, a **readers-writer** or **shared-exclusive** lock (also known as the **multiple readers / single-writer lock** or the **multi-reader lock**,[1] or by typographical variants such as **readers/writers lock**) is a **synchronization primitive** that solves one of the readers-writers problems. A readers-writer lock is like a mutex, in that it controls access to a shared resource, allowing concurrent access to multiple threads for reading but restricting access to a single thread for writes (or other changes) to the resource. A common use might be to control access to a data structure in memory that can't be updated atomically and isn't valid (and shouldn't be read by another thread) until the update is complete.

One potential problem with a conventional RW lock is that it can lead to write-starvation if contention is high enough, meaning that as long as at least one reading thread holds the lock, no writer thread will be able to acquire it. Since multiple reader threads may hold the lock at once, this means that a writer thread may continue waiting for the lock while new reader threads are able to acquire the lock, even to the point where the writer may still be waiting after all of the readers which were holding the lock when it first attempted to acquire it have released the lock. To avoid writer starvation, a variant on a readers-writer lock can be constructed which prevents any *new* readers from acquiring the lock if there is a writer queued and waiting for the lock, so that the writer will acquire the lock as soon as the readers which were already holding the lock are finished with it. The downside is that it's less performant because each operation, taking or releasing the lock for either read or write, is more complex, internally requiring taking and releasing two mutexes instead of one. This variation is sometimes known as a "write-preferring" or "write-biased" readers-writer lock.[2][3]

A solution to this problem by Brandenburg and Anderson introduces the phase fair reader-writer lock The releasing of the lock alternates between readers and writers. When no writer is requesting a lock, readers are allowed through. After a writer shows up, new readers are queued until existing readers and then the new writer finish with the lock, at which time the backlogged queued readers are released en masse. A C language phase fair implementation that requires 8 bytes per lock:

Readers–writer locks are usually constructed on top of mutexes and condition variables, or on top of semaphores.

The read-copy-update (RCU) algorithm is one solution to the readers-writers problem. RCU is wait-free for readers. The Linux-Kernel implements a special solution for few writers called seqlock.

A **read/write lock pattern** or simply **RWL** is a software design pattern that allows concurrent read access to an object but requires exclusive access for write operations. In this pattern, multiple readers can read the data in parallel but an exclusive lock is needed while writing the data. When a writer is writing the data, readers will be blocked until the writer is finished writing.

Implementations

- The POSIX standard `pthread_rwlock_t` and associated operations.
- The C language Win32 multiple-reader/single-writer lock used in Hamilton C shell. The Hamilton lock presumes contention is low enough that writers are unlikely to be starved, prompting Jordan Zimmerman to suggest a modified version to avoid starvation.
- The ReadWriteLock interface and the ReentrantReadWriteLock locks in Java version 5 or above.
- A simple Windows API implementation by Glenn Slayden.
- The Microsoft `System.Threading.ReaderWriterLockSlim` lock for C# and other .NET languages.
- The `boost::shared_mutex` read/write lock in the Boost C++ Libraries.
- A pseudo-code implementation in the Readers-writers problem article.

References

[1] "Practical lock-freedom" (http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-579.pdf) by Keir Fraser 2004

[2] "ReaderWriterLock Alternative" (http://www.codeplex.com/ReaderWriterLockAlt) an open source C# implementation of a write-biased readers-writer lock

[3] Java readers-writer lock implementation offers a "fair" mode

Blind write

In computing, a **blind write** occurs when a transaction writes a value without reading it. Any view serializable schedule that is not conflict serializable must contain a blind write.

Conservative two-phase locking

In computer science, **conservative two-phase locking (C2PL)** is a locking method used in DBMS and relational databases.

Conservative 2PL *prevents* deadlocks.

The difference between 2PL and C2PL is that C2PL's transactions obtain all the locks they need before the transactions begin. This is to ensure that a transaction that already holds some locks will not block waiting for other locks.

In heavy lock contention, C2PL reduces the time locks are held on average, relative to 2PL and Strict 2PL, because transactions that hold locks are never blocked.

In light lock contention, C2PL holds more locks than is necessary, because it is hard to tell what locks will be needed in the future, thus leads to higher overhead.

Also, a transaction will not even obtain any locks if it cannot obtain all the locks it needs in its initial request. Furthermore, each transaction needs to declare its read and write set (data items to be read/written during transaction), which is not always possible. Because of these limitations, C2PL is not used very frequently.

Strong strict two-phase locking

In databases and transaction processing, **two-phase locking** (**2PL**) is a concurrency control method that guarantees serializability.[1][2] It is also the name of the resulting set of database transaction schedules (histories). The protocol utilizes locks, applied by a transaction to data, which may block (interpreted as signals to stop) other transactions from accessing the same data during the transaction's life.

By the 2PL protocol locks are applied and removed in two phases:

1. Expanding phase: locks are acquired and no locks are released.
2. Shrinking phase: locks are released and no locks are acquired.

Two types of locks are utilized by the basic protocol: *Shared* and *Exclusive* locks. Refinements of the basic protocol may utilize more lock types. Using locks that block processes, 2PL may be subject to deadlocks that result from the mutual blocking of two or more transactions.

2PL is a superset of **strong strict two-phase locking** (**SS2PL**),[3] also called **rigorousness**,[4] which has been widely utilized for concurrency control in general-purpose database systems since the 1970s. SS2PL implementations have many variants. SS2PL was called *strict 2PL* but this name usage is not recommended now. Now **strict 2PL** (**S2PL**) is the intersection of strictness and 2PL, which is different from SS2PL. SS2PL is also a special case of commitment ordering, and inherits many of CO's useful properties. SS2PL actually comprises only one phase: phase-2 does not exist, and all locks are released only after transaction end. Thus this useful 2PL type is not two-phased at all.

Neither 2PL nor S2PL in their general forms are known to be used in practice. Thus 2PL by itself does not seem to have much practical importance, and whenever 2PL or S2PL utilization has been mentioned in the literature, the intention has been SS2PL. What has made SS2PL so popular (probably the most utilized serializability mechanism) is the effective and efficient locking-based combination of two ingredients (the first does not exist in both general 2PL and S2PL; the second does not exist in general 2PL):

1. Commitment ordering, which provides both serializability, and effective distributed serializability and global serializability, and
2. Strictness, which provides cascadelessness (ACA, cascade-less recoverability) and (independently) allows efficient database recovery from failure.

Additionally SS2PL is easier, with less overhead to implement than both 2PL and S2PL, provides exactly the same locking, but sometimes releases locks later. However, practically (though not simplistically theoretically) such later lock release occurs only slightly later, and this apparent disadvantage is insignificant and disappears next to the advantages of SS2PL.

Thus, the importance of the general *Two-phase locking* (2PL) is historic only, while *Strong strict two-phase locking* (SS2PL) is practically the important mechanism and resulting schedule property.

Data-access locks

A lock is a system object associated with a shared resource such as a data item of an elementary type, a row in a database, or a page of memory. In a database, a lock on a database object (a data-access lock) may need to be acquired by a transaction before accessing the object. Correct use of locks prevents undesired, incorrect or inconsistent operations on shared resources by other concurrent transactions. When a database object with an existing lock acquired by one transaction needs to be accessed by another transaction, the existing lock for the object and the type of the intended access are checked by the system. If the existing lock type does not allow this specific attempted concurrent access type, the transaction attempting access is blocked (according to a predefined agreement/scheme). In practice a lock on an object does not directly block a transaction's operation upon the object, but rather blocks that transaction from acquiring another lock on the same object, needed to be held/owned by the transaction before performing this operation. Thus, with a locking mechanism, needed operation blocking is

controlled by a proper lock blocking scheme, which indicates which lock type blocks which lock type.

Two major types of locks are utilized:

- **Write-lock** (**exclusive lock**) is associated with a database object by a transaction (Terminology: "the transaction locks the object," or "acquires lock for it") before *writing* (inserting/modifying/deleting) this object.
- **Read-lock** (**shared lock**) is associated with a database object by a transaction before *reading* (retrieving the state of) this object.

The common interactions between these lock types are defined by blocking behavior as follows:

- An existing *write-lock* on a database object blocks an intended *write* upon the same object (already requested/issued) by another transaction by blocking a respective *write-lock* from being acquired by the other transaction. The second write-lock will be acquired and the requested write of the object will take place (materialize) after the existing write-lock is released.
- A *write-lock* blocks an intended (already requested/issued) *read* by another transaction by blocking the respective *read-lock* .
- A *read-lock* blocks an intended *write* by another transaction by blocking the respective *write-lock* .
- A *read-lock* does not block an intended *read* by another transaction. The respective *read-lock* for the intended read is acquired (shared with the previous read) immediately after the intended read is requested, and then the intended read itself takes place.

Several variations and refinements of these major lock types exist, with respective variations of blocking behavior. If a first lock blocks another lock, the two locks are called *incompatible*; otherwise the locks are *compatible*. Often lock types blocking interactions are presented in the technical literature by a *Lock compatibility table*. The following is an example with the common, major lock types:

Lock compatibility table

Lock type	read-lock	write-lock
read-lock		**X**
write-lock	**X**	**X**

X indicates incompatibility, i.e, a case when a lock of the first type (in left column) on an object blocks a lock of the second type (in top row) from being acquired on the same object (by another transaction). An object typically has a queue of waiting requested (by transactions) operations with respective locks. The first blocked lock for operation in the queue is acquired as soon as the existing blocking lock is removed from the object, and then its respective operation is executed. If a lock for operation in the queue is not blocked by any existing lock (existence of multiple compatible locks on a same object is possible concurrently) it is acquired immediately.

Comment: In some publications the table entries are simply marked "compatible" or "incompatible", or respectively "yes" or "no".

Two-phase locking and its special cases

Two-phase locking

According to the **two-phase locking** protocol a transaction handles its locks in two distinct, consecutive phases during the transaction's execution:

1. **Expanding phase** (aka Growing phase): locks are acquired and no locks are released (the number of locks can only increase).
2. **Shrinking phase**: locks are released and no locks are acquired.

The serializability property is guaranteed for a schedule with transactions that obey the protocol. The 2PL *schedule class* is defined as the class of all the schedules comprising transactions with data access orders that could be generated by the 2PL protocol (or in other words, all the schedules that the 2PL protocol can generate).

Typically, without explicit knowledge in a transaction on end of phase-1, it is safely determined only when a transaction has entered its *ready* state in all its processes (processing has ended, and it is ready to be committed; no additional data access and locking are needed and can happen). In this case phase-2 can end immediately (no additional processing is needed), and actually no phase-2 is needed. Also, if several processes (two or more) are involved, then a synchronization point (similar to *atomic commitment*) among them is needed to determine end of phase-1 for all of them (i.e., in the entire distributed transaction), to start releasing locks in phase-2 (otherwise it is very likely that both 2PL and Serializability are quickly violated). Such synchronization point is usually too costly (involving a distributed protocol similar to atomic commitment), and end of phase-1 is usually postponed to be merged with transaction end (atomic commitment protocol for a multi-process transaction), and again phase-2 is not needed. This turns 2PL to SS2PL (see below). All known implementations of 2PL in products are SS2PL based, and whenever 2PL (or Strict 2PL, S2PL) practical utilization has been mentioned in the professional literature, the intention has been SS2PL.

Strict two-phase locking

The **strict two-phase locking** (S2PL) class of schedules is the intersection of the 2PL class with the class of schedules possessing the *Strictness* property.

To comply with the S2PL protocol a transaction needs to comply with 2PL, and release its *write (exclusive)* locks only after it has ended, i.e., being either *committed* or *aborted*. On the other hand, *read (shared)* locks are released regularly during phase 2. Implementing general S2PL requires explicit support of phase-1 end, separate from transaction end, and no such widely utilized product implementation is known.

S2PL is a special case of 2PL, i.e., the S2PL class is a proper subclass of 2PL

Strong strict two-phase locking

or **Rigorousness**, or **Rigorous scheduling**, or **Rigorous two-phase locking**

To comply with **strong strict two-phase locking** (SS2PL) the locking protocol releases both *write (exclusive)* and *read (shared)* locks applied by a transaction only after the transaction has ended, i.e., only after both completing executing (being *ready*) and becoming either *committed* or *aborted*. This protocol also complies with the S2PL rules. A transaction obeying SS2PL can be viewed as having phase-1 that lasts the transaction's entire execution duration, and no phase-2 (or a degenerate phase-2). Thus, only one phase is actually left, and "two-phase" in the name seems to be still utilized due to the historical development of the concept from 2PL, and 2PL being a super-class. The SS2PL property of a schedule is also called **Rigorousness**. It is also the name of the class of schedules having this property, and an SS2PL schedule is also called a "rigorous schedule". The term "Rigorousness" is free of the unnecessary legacy of "two-phase," as well as being independent of any (locking) mechanism (in principle other blocking mechanisms can be utilized). The property's respective locking mechanism is sometimes referred to as

Rigorous 2PL.

SS2PL is a special case of S2PL, i.e., the SS2PL class of schedules is a proper subclass of S2PL (every SS2PL schedule is also an S2PL schedule, but S2PL schedules exist that are not SS2PL).

SS2PL has been the concurrency control protocol of choice for most database systems and utilized since their early days in the 1970s. It is proven to be an effective mechanism in many situations, and provides besides Serializability also Strictness (a special case of cascadeless Recoverability), which is instrumental for efficient database recovery, and also Commitment ordering (CO) for participating in distributed environments where a CO based distributed serializability and global serializability solutions are employed. Being a subset of CO, an efficient implementation of *distributed SS2PL* exists without a distributed lock manager (DLM), while distributed deadlocks (see below) are resolved automatically. The fact that SS2PL employed in multi database systems ensures global serializability has been known for years before the discovery of CO, but only with CO came the understanding of the role of an atomic commitment protocol in maintaining global serializability, as well as the observation of automatic distributed deadlock resolution (see a detailed example of Distributed SS2PL). As a matter of fact, SS2PL inheriting properties of Recoverability and CO is more significant than being a subset of 2PL, which by itself in its general form, besides comprising a simple serializability mechanism (however serializability is also implied by CO), in not known to provide SS2PL with any other significant qualities. 2PL in its general form, as well as when combined with Strictness, i.e., Strict 2PL (S2PL), are not known to be utilized in practice. The popular SS2PL does not require to mark "end of phase-1" as 2PL and S2PL do, and thus is simpler to implement. Also unlike the general 2PL, SS2PL provides, as mentioned above, the useful Strictness and Commitment ordering properties.

Many variants of SS2PL exist that utilize various lock types with various semantics in different situations, including cases of lock-type change during a transaction. Notable are variants that use Multiple granularity locking.

Comments:

1. SS2PL Vs. S2PL: Both provide Serializability and Strictness. Since S2PL is a super class of SS2PL it may, in principle, provide more concurrency. However no concurrency advantage is typically practically noticed (exactly same locking exists for both, with practically not much earlier lock release for S2PL), and the overhead of dealing with an end-of-phase-1 mechanism in S2PL, separate from transaction-end, is not justified. Also, while SS2PL provides Commitment ordering, S2PL does not. This explains the preference of SS2PL over S2PL.
2. Especially before 1990, but also after, in many articles and books, e.g., (Bernstein et al. 1987, p. 59), the term "Strict 2PL" (S2PL) has been frequently defined by the locking protocol "Release all locks only after transaction end," which is the protocol of SS2PL. Thus, "Strict 2PL" could not be there the name of the intersection of Strictness and 2PL, which is larger than the class generated by the SS2PL protocol. This has caused confusion. With an explicit definition of S2PL as the intersection of Strictness and 2PL, a new name for SS2PL, and an explicit distinction between the classes S2PL and SS2PL, the articles (Breitbart et al. 1991) and (Raz 1992) have intended to clear the confusion: The first using the name "Rigorousness," and the second "SS2PL."
3. A more general property than SS2PL exists (a schedule super-class), **Strict commitment ordering** (Strict CO, or SCO), which as well provides both serializability, strictness, and CO, and has similar locking overhead. Unlike SS2PL, SCO does not block upon a read-write conflict (a read-lock does not block acquiring a write-lock; both SCO and SS2PL have the same behavior for write-read and write-write conflicts) at the cost of a possible delayed commit, and upon such conflict type SCO has shorter average transaction completion time and better performance than SS2PL.[5] While SS2PL obeys the *lock compatibility table* above, SCO has the following table:

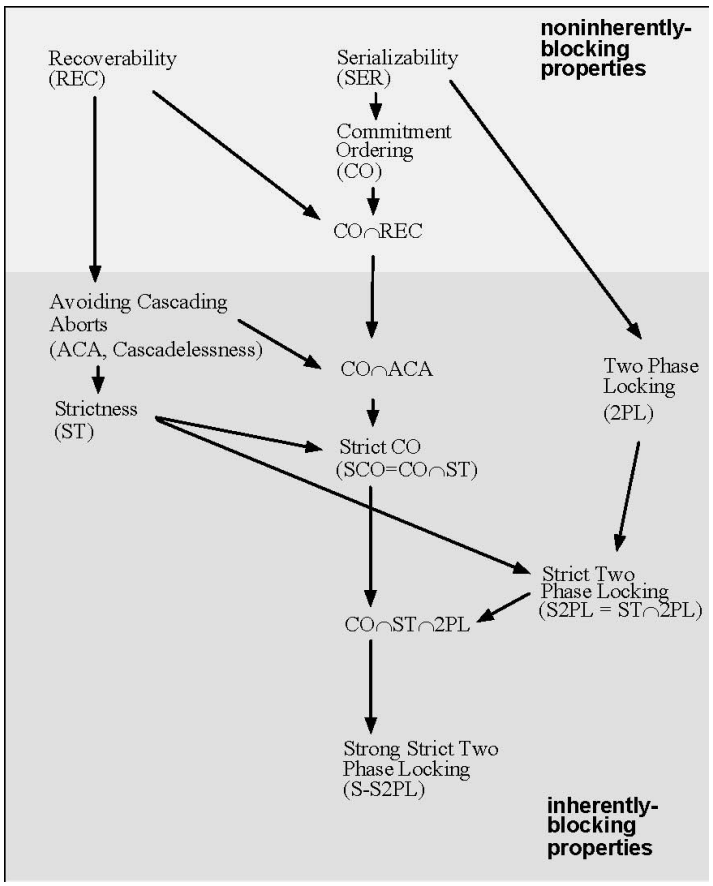
Lock compatibility for SCO

Lock type	read-lock	write-lock
read-lock		
write-lock	X	X

Note that though SCO releases all locks at transaction end and complies with the 2PL locking rules, SCO is not a subset of 2PL because of its different lock compatibility table. SCO allows materialized read-write conflicts between two transactions in their phases 1, which 2PL does not allow in phase-1 (see about materialized conflicts in Serializability). On the other hand 2PL allows other materialized conflict types in phase-2 that SCO does not allow at all. Together this implies that the schedule classes 2PL and SCO are incomparable (i.e., no class contains the other class).

Summary - Relationships among classes

Between any two schedule classes (define by their schedules' respective properties) that have common schedules, either one *contains* the other (*strictly contains* if they are not equal), or they are *incomparable*. The containment relationships among the 2PL classes and other major schedule classes are summarized in the following diagram. 2PL and its subclasses are *inherently blocking*, which means that no optimistic implementations for them exist (and whenever "Optimistic 2PL" is mentioned it refers to a different mechanism with a class that includes also schedules not in the 2PL class).



Schedule classes containment: An arrow from class A to class B indicates that class A strictly contains B; a lack of a directed path between classes means that the classes are incomparable.A property is inherently blocking, if it can be enforced only by blocking transaction's data access operations until certain events occur in other transactions. (#Raz1992Raz 1992)

Deadlocks in 2PL

Locks block data-access operations. Mutual blocking between transactions results in a *deadlock*, where execution of these transactions is stalled, and no completion can be reached. Thus deadlocks need to be resolved to complete these transactions' executions and release related computing resources. A deadlock is a reflection of a potential cycle in the *precedence graph*, that would occur without the blocking. A deadlock is resolved by aborting a transaction involved with such potential cycle, and breaking the cycle. It is often detected using a *wait-for graph* (a graph of conflicts blocked by locks from being materialized; conflicts not materialized in the database due to blocked operations are not reflected in the precedence graph and do not affect

serializability), which indicates which transaction is "waiting for" lock release by which transaction, and a cycle means a deadlock. Aborting one transaction per cycle is sufficient to break the cycle. Transactions aborted due to deadlock resolution are executed again immediately.

In a distributed environment an atomic commitment protocol, typically the Two-phase commit (2PC) protocol, is utilized for atomicity. When recoverable data (data under transaction control) are partitioned among 2PC participants (i.e., each data object is controlled by a single 2PC participant), then distributed (global) deadlocks, deadlocks involving two or more participants in 2PC, are resolved automatically as follows:

When SS2PL is effectively utilized in a distributed environment, then global deadlocks due to locking generate voting-deadlocks in 2PC, and are resolved automatically by 2PC (see Commitment ordering (CO), in Exact characterization of voting-deadlocks by global cycles; No reference except the CO articles is known to notice this). For the general case of 2PL, global deadlocks are similarly resolved automatically by the synchronization point protocol of phase-1 end in a distributed transaction (synchronization point is achieved by "voting" (notifying local phase-1 end), and being propagated to the participants in a distributed transaction the same way as a decision point in atomic commitment; in analogy to decision point in CO, a conflicting operation in 2PL cannot happen before phase-1 end synchronization point, with the same resulting voting-deadlock in the case of a global data-access deadlock; the voting-deadlock (which is also a locking based global deadlock) is automatically resolved by the protocol aborting some transaction involved, with a missing vote, typically using a timeout).

Comment:

When data are partitioned among the *atomic commitment protocol* (e.g., 2PC) participants, automatic *global deadlock* resolution has been overlooked in the database research literature, though deadlocks in such systems has been a quite intensive research area:

- For CO and its special case SS2PL, the automatic resolution by the *atomic commitment protocol* has been noticed only in the CO articles. However, it has been noticed in practice that in many cases global deadlocks are very infrequently detected by the dedicated resolution mechanisms, less than could be expected ("Why do we see so few global deadlocks?"). The reason is probably the deadlocks that are automatically resolved and thus not handled and uncounted by the mechanisms;
- For 2PL in general, the automatic resolution by the (mandatory) *end-of-phase-one synchronization point protocol* (which has same voting mechanism as atomic commitment protocol, and same missing vote handling upon voting deadlock, resulting in global deadlock resolution) has not been mentioned until today (2009). Practically only the special case SS2PL is utilized, where no end-of-phase-one synchronization is needed in addition to atomic commit protocol.

In a distributed environment where recoverable data are not partitioned among atomic commitment protocol participants, no such automatic resolution exists, and distributed deadlocks need to be resolved by dedicated techniques.

References

[1] Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* (http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx), Addison Wesley Publishing Company, ISBN 0-201-10715-5

[2] Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description), Elsevier, ISBN 1-55860-508-8

[3] Yoav Raz (1992): "The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Managers Using Atomic Commitment" (http://www.informatik.uni-trier.de/~ley/db/conf/vldb/Raz92.html) (PDF (http://www.vldb.org/conf/1992/P292.PDF)), *Proceedings of the Eighteenth International Conference on Very Large Data Bases* (VLDB), pp. 292-312, Vancouver, Canada, August 1992, ISBN 1-55860-151-1 (also DEC-TR 841, Digital Equipment Corporation, November 1990)

[4] Yuri Breitbart, Dimitrios Georgakopoulos, Marek Rusinkiewicz, Abraham Silberschatz (1991): "On Rigorous Transaction Scheduling" (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=92915), *IEEE Transactions on Software Engineering* (TSE), September 1991, Volume 17, Issue 9, pp. 954-960, ISSN: 0098-5589

[5] Yoav Raz (1991): "Locking Based Strict Commitment Ordering, or How to improve Concurrency in Locking Based Resource Managers", DEC-TR 844, December 1991.

Index locking

In databases an *index* is a data structure, part of the database, used by a database system to effectively navigate access to *user data*. Index data are system data distinct from user data, and consist primarily of pointers. Changes in a database (by insert, delete, or modify operations), may require indexes to be updated to maintain accurate user data accesses.[1] **Index locking** is a technique used to maintain index integrity. A portion of an index is locked during a database transaction when this portion is being accessed by the transaction as a result of attempt to access related user data. Additionally, special database system transactions (not user-invoked transactions) may be invoked to maintain and modify an index, as part of a system's self-maintenance activities. When a portion of an index is locked by a transaction, other transactions may be blocked from accessing this index portion (blocked from modifying, and even from reading it, depending on lock type and needed operation).

Specialized concurrency control techniques exist for accessing indexes. These techniques depend on the index type, and take advantage of its structure. They are typically much more effective than applying to indexes common concurrency control methods applied to user data. Notable and widely researched are specialized techniques for B-trees (B-Tree concurrency control[2]) which are regularly used as database indexes.

Index locks are used to coordinate threads accessing indexes concurrently, and typically shorter-lived than the common transaction locks on user data. In professional literature, they are often called *latches*.

References

[1] Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description) Chapter 9, Elsevier, ISBN 1-55860-508-8

[2] Goetz Graefe (2010): "A survey of B-tree locking techniques" (http://portal.acm.org/citation.cfm?id=1806908) *ACM Transactions on Database Systems* (TODS), Volume 35 Issue 3, July 2010 (also HPL-2010-9 (http://www.hpl.hp.com/techreports/2010/HPL-2010-9.pdf), HP Laboratories).

Snapshot isolation

In databases, and transaction processing (transaction management), **snapshot isolation** is a guarantee that all reads made in a transaction will see a consistent snapshot of the database (in practice it reads the last committed values that existed at the time it started), and the transaction itself will successfully commit only if no updates it has made conflict with any concurrent updates made since that snapshot.

Snapshot isolation has been adopted by several major database management systems, such as SQL Anywhere, InterBase, Firebird, Oracle, PostgreSQL and Microsoft SQL Server (2005 and later). The main reason for its adoption is that it allows better performance than serializability, yet still avoids most of the concurrency anomalies that serializability avoids (but not always all). In practice snapshot isolation is implemented within multiversion concurrency control (MVCC), where generational values of each data item (versions) are maintained: MVCC is a common way to increase concurrency and performance by generating a new version of a database object each time the object is written, and allowing transactions' read operations of several last relevant versions (of each object). Snapshot isolation has also been used to critique the ANSI SQL-92 standard's definition of isolation levels, as it exhibits none of the "anomalies" that the SQL standard prohibited, yet is not serializable (the anomaly-free isolation level defined by ANSI).

Snapshot isolation is called "serializable" mode in Oracle[1][2][3] and PostgreSQL versions prior to 9.1,[4][5] which may cause confusion with the "real serializability" mode. There are arguments both for and against this decision; what is clear is that users must be aware of the distinction to avoid possible undesired anomalous behavior in their database system logic.

Definition

A transaction executing under snapshot isolation appears to operate on a personal *snapshot* of the database, taken at the start of the transaction. When the transaction concludes, it will successfully commit only if the values updated by the transaction have not been changed externally since the snapshot was taken. Such a write-write conflict will cause the transaction to abort.

In a *write skew* anomaly, two transactions (T1 and T2) concurrently read an overlapping data set (e.g. values V1 and V2), concurrently make disjoint updates (e.g. T1 updates V1, T2 updates V2), and finally concurrently commit, neither having seen the update performed by the other. Were the system serializable, such an anomaly would be impossible, as either T1 or T2 would have to occur "first", and be visible to the other. In contrast, snapshot isolation permits write skew anomalies.

As a concrete example, imagine V1 and V2 are two balances held by a single person, Phil. The bank will allow either V1 or V2 to run a deficit, provided the total held in both is never negative (i.e. $V1 + V2 \geq 0$). Both balances are currently $100. Phil initiates two transactions concurrently, T1 withdrawing $200 from V1, and T2 withdrawing $200 from V2.

If the database guaranteed serializable transactions, the simplest way of coding T1 is to deduct $200 from V1, and then verify that $V1 + V2 \geq 0$ still holds, aborting if not. T2 similarly deducts $200 from V2 and then verifies $V1 + V2 \geq 0$. Since the transactions must serialize, either T1 happens first, leaving V1 = -$100, V2 = $100, and preventing T2 from succeeding (since V1 + (V2 - $200) is now -$200), or T2 happens first and similarly prevents T1 from committing.

Under snapshot isolation, however, T1 and T2 operate on private snapshots of the database: each deducts $200 from an account, and then verifies that the new total is zero, using the other account value that held when the snapshot was taken. Since neither *update* conflicts, both commit successfully, leaving V1 = V2 = -$100, and V1 + V2 = -$200.

If built on multiversion concurrency control, snapshot isolation allows transactions to proceed without worrying about concurrent operations, and more importantly without needing to re-verify all read operations when the

transaction finally commits. The only information that must be stored during the transaction is a list of updates made, which can be scanned for conflicts fairly easily before being committed.

Serializable Snapshot Isolation

Potential inconsistency problems arising from write skew anomalies can be fixed by adding (otherwise unnecessary) updates to the transactions in order to enforce the serializability property.

- **Materialize the conflict**: Add a special conflict table, which both transactions update in order to create a direct write-write conflict.
- **Promotion**: Have one transaction "update" a read-only location (replacing a value with the same value) in order to create a direct write-write conflict (or use an equivalent promotion, e.g. Oracle's SELECT FOR UPDATE).

In the example above, we can materialize the conflict by adding a new table which makes the hidden constraint explicit, mapping each person to their *total balance*. Phil would start off with a total balance of $200, and each transaction would attempt to subtract $200 from this, creating a write-write conflict that would prevent the two from succeeding concurrently. This approach violates the normal form.[*citation needed*]

Alternatively, we can promote one of the transaction's reads to a write. For instance, T2 could set V1 = V1, creating an artificial write-write conflict with T1 and, again, preventing the two from succeeding concurrently. This solution may not always be possible.

In general, therefore, snapshot isolation puts some of the problem of maintaining non-trivial constraints onto the user, who may not appreciate either the potential pitfalls or the possible solutions. The upside to this transfer is better performance.

With additional communication between transactions, the anomalies that snapshot isolation normally allows can be blocked by aborting one of the transactions involved, turning a snapshot isolation implementation into a full serializability guarantee.[6] This implementation of serializability is well-suited to multiversion concurrency control databases, and has been adopted in PostgreSQL 9.1, where it is referred to as "Serializable Snapshot Isolation", abbreviated to SSI. When used consistently, this eliminates the need for the above workarounds. The downside over snapshot isolation is an increase in aborted transactions. This can perform better or worse than snapshot isolation with the above workarounds, depending on workload.

History

Snapshot isolation arose from work on multiversion concurrency control databases, where multiple versions of the database are maintained concurrently to allow readers to execute without colliding with writers. Such a system allows a natural definition and implementation of such an isolation level. InterBase later owned by Borland provided SI as far back as 1984[*citation needed*].

Unfortunately, the ANSI SQL-92 standard was written with a lock-based database in mind, and hence is rather vague when applied to MVCC systems. Berenson *et al.* wrote a paper in 1995 critiquing the SQL standard, and cited snapshot isolation as an example of an isolation level that did not exhibit the standard anomalies described in the ANSI SQL-92 standard, yet still had anomalous behaviour when compared with serializable transactions.

References

[1] Oracle Database Concepts 10g Release 1 (10.1) Chapter 13 : Data Concurrency and Consistency — Oracle Isolation Levels (http://docs.oracle.com/cd/B12037_01/server.101/b10743/consist.htm#i17856)
[2] Ask Tom : On Transaction Isolation Levels (http://asktom.oracle.com/pls/asktom/f?p=100:11:::::P11_QUESTION_ID:7636765105002)
[3] Ask Tom : "Serializable Transaction" (http://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:3233191441609)
[4] PostgreSQL 9.0 Documentation: 13.2.2.1. Serializable Isolation versus True Serializability (http://www.postgresql.org/docs/9.0/static/transaction-iso.html#MVCC-SERIALIZABILITY)
[5] PostgreSQL 9.1 press release (http://www.postgresql.org/about/news.1349)
[6] Michael J. Cahill, Uwe Röhm, Alan D. Fekete (2008) "Serializable isolation for snapshot databases" (http://portal.acm.org/citation.cfm?id=1376690), *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 729–738, ISBN 978-1-60558-102-6 (SIGMOD 2008 best paper award)

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8
- Khuzaima Daudjee, Kenneth Salem, *Lazy Database Replication with Snapshot Isolation*, VLDB 2006: pages 715-726

Non-lock concurrency control

In Computer Science, in the field of databases, **non-lock concurrency control** is a concurrency control method used in relational databases without using locking.

There are several non-lock concurrency control methods, which involve the use of timestamps on transaction to determine transaction priority:

- Optimistic concurrency control
- Timestamp-based concurrency control
- Multiversion concurrency control

Commitment ordering

Commitment ordering (**CO**) is a class of interoperable *serializability* techniques in concurrency control of databases, transaction processing, and related applications. It allows optimistic (non-blocking) implementations. With the proliferation of multi-core processors, CO has been also increasingly utilized in concurrent programming, transactional memory, and especially in software transactional memory (STM) for achieving serializability optimistically. CO is also the name of the resulting transaction schedule (history) property, which was originally defined in 1988 with the name *dynamic atomicity*.[1] In a CO compliant schedule the chronological order of commitment events of transactions is compatible with the precedence order of the respective transactions. CO is a broad special case of *conflict serializability*, and effective means (reliable, high-performance, distributed, and scalable) to achieve global serializability (modular serializability) across any collection of database systems that possibly use different concurrency control mechanisms (CO also makes each system serializability compliant, if not already).

Each not-CO-compliant database system is augmented with a CO component (the commitment order coordinator—COCO) which orders the commitment events for CO compliance, with neither data-access nor any other transaction operation interference. As such CO provides a low overhead, general solution for global serializability (and distributed serializability), instrumental for global concurrency control (and distributed concurrency control) of multi database systems and other transactional objects, possibly highly distributed (e.g., within cloud computing, grid computing, and networks of smartphones). An atomic commitment protocol (ACP; of any type) is a fundamental part of the solution, utilized to break global cycles in the conflict (precedence, serializability) graph. CO is the most general property (a necessary condition) that guarantees global serializability, if the database systems involved do not share concurrency control information beyond atomic commitment protocol (unmodified) messages, and have no knowledge whether transactions are global or local (the database systems are *autonomous*). Thus CO (with its variants) is the only general technique that does not require the typically costly distribution of local concurrency control information (e.g., local precedence relations, locks, timestamps, or tickets). It generalizes the popular *strong strict two-phase locking* (SS2PL) property, which in conjunction with the *two-phase commit protocol* (2PC) is the de facto standard to achieve global serializability across (SS2PL based) database systems. As a result CO compliant database systems (with any, different concurrency control types) can transparently join such SS2PL based solutions for global serializability.

In addition, locking based *global deadlocks* are resolved automatically in a CO based multi-database environment, an important side-benefit (including the special case of a completely SS2PL based environment; a previously unnoticed fact for SS2PL).

Furthermore, **strict commitment ordering** (SCO; Raz 1991c), the intersection of *Strictness* and CO, provides better performance (shorter average transaction completion time and resulting better transaction throughput) than SS2PL whenever read-write conflicts are present (identical blocking behavior for write-read and write-write conflicts; comparable locking overhead). The advantage of SCO is especially significant during lock contention. Strictness allows both SS2PL and SCO to use the same effective *database recovery* mechanisms.

Two major generalizing variants of CO exist, **extended CO** (ECO; Raz 1993a) and **multi-version CO** (MVCO; Raz 1993b). They as well provide global serializability without local concurrency control information distribution, can be combined with any relevant concurrency control, and allow optimistic (non-blocking) implementations. Both use additional information for relaxing CO constraints and achieving better concurrency and performance. **Vote ordering** (VO or Generalized CO (GCO); Raz 2009) is a container schedule set (property) and technique for CO and all its variants. Local VO is a necessary condition for guaranteeing global serializability, if the atomic commitment protocol (ACP) participants do not share concurrency control information (have the *generalized autonomy* property). CO and its variants inter-operate transparently, guaranteeing global serializability and automatic global deadlock resolution also together in a mixed, heterogeneous environment with different variants.

Overview

The *Commitment ordering* (CO; Raz 1990, 1992, 1994, 2009) schedule property has been referred to also as *Dynamic atomicity* (since 1988), *commit ordering*, *commit order serializability*, and *strong recoverability* (since 1991). The latter is a misleading name since CO is incomparable with *recoverability*, and the term "strong" implies a special case. This means that a schedule with a strong recoverability property does not necessarily have the CO property, and vice versa.

In 2009 CO has been characterized as a major concurrency control method, together with the previously known (since the 1980s) three major methods: *Locking*, *Time-stamp ordering*, and *Serialization graph testing*, and as an enabler for the interoperability of systems using different concurrency control mechanisms.[2]

In a federated database system or any other more loosely defined multidatabase system, which are typically distributed in a communication network, transactions span multiple and possibly Distributed databases. Enforcing global serializability in such system is problematic. Even if every local schedule of a single database is serializable, still, the global schedule of a whole system is not necessarily serializable. The massive communication exchanges of conflict information needed between databases to reach conflict serializability would lead to unacceptable performance, primarily due to computer and communication latency. The problem of achieving global serializability effectively had been characterized as open until the public disclosure of CO in 1991 by its inventor Yoav Raz (Raz 1991a; see also Global serializability).

Enforcing CO is an effective way to enforce conflict serializability globally in a distributed system, since enforcing CO locally in each database (or other transactional object) also enforces it globally. Each database may use any, possibly different, type of concurrency control mechanism. With a local mechanism that already provides conflict serializability, enforcing CO locally does not cause any additional aborts, since enforcing CO locally does not affect the data access scheduling strategy of the mechanism (this scheduling determines the serializability related aborts; such a mechanism typically does not consider the commitment events or their order). The CO solution requires no communication overhead, since it uses (unmodified) *atomic commitment* protocol messages only, already needed by each distributed transaction to reach atomicity. An atomic commitment protocol plays a central role in the distributed CO algorithm, which enforces CO globally, by breaking global cycles (cycles that span two or more databases) in the global conflict graph. CO, its special cases, and its generalizations are interoperable, and achieve global serializability while transparently being utilized together in a single heterogeneous distributed environment comprising objects with possibly different concurrency control mechanisms. As such, *Commitment ordering*, including its special cases, and together with its generalizations (see CO variants below), provides a general, high performance, fully distributed solution (no central processing component or central data structure are needed) for guaranteeing global serializability in heterogeneous environments of multidatabase systems and other multiple transactional objects (objects with states accessed and modified only by transactions; e.g., in the framework of transactional processes, and within Cloud computing and Grid computing). The CO solution scales up with network size and the number of databases without any negative impact on performance (assuming the statistics of a single distributed transaction, e.g., the average number of databases involved with a single transaction, are unchanged).

With the proliferation of Multi-core processors, Optimistic CO (OCO) has been also increasingly utilized to achieve serializability in software transactional memory, and numerous STM articles and patents utilizing "commit order" have already been published (e.g., Zhang et al. 2006).

The commitment ordering solution for global serializability

General characterization of CO

Commitment ordering (CO) is a special case of conflict serializability. CO can be enforced with *non-blocking* mechanisms (each transaction can complete its task without having its data-access blocked, which allows optimistic concurrency control; however, commitment could be blocked). In a CO schedule the commitment events' (partial) precedence order of the transactions corresponds to the precedence (partial) order of the respective transactions in the (directed) conflict graph (precedence graph, serializability graph), as induced by their conflicting access operations (usually read and write (insert/modify/delete) operations; CO also applies to higher level operations, where they are conflicting if noncommutative, as well as to conflicts between operations upon multi-version data).

- **Definition: commitment ordering**

  Let T_1, T_2 be two *committed* transactions in a schedule, such that T_2 is *in a conflict* with T_1 (T_1 *precedes* T_2). The schedule has the **Commitment ordering** (CO) property, if for every two such transactions T_1 commits before T_2 commits.

The commitment decision events are generated by either a local commitment mechanism, or an atomic commitment protocol, if different processes need to reach consensus on whether to commit or abort. The protocol may be distributed or centralized. Transactions may be committed concurrently, if the commit partial order allows (if they do not have conflicting operations). If different conflicting operations induce different partial orders of same transactions, then the conflict graph has cycles, and the schedule will violate serializability when all the transactions on a cycle are committed. In this case no partial order for commitment events can be found. Thus, cycles in the conflict graph need to be broken by aborting transactions. However, any conflict serializable schedule can be made CO without aborting any transaction, by properly delaying commit events to comply with the transactions' precedence partial order.

CO enforcement by itself is not sufficient as a concurrency control mechanism, since CO lacks the recoverability property, which should be supported as well.

The distributed CO algorithm

A fully distributed *Global commitment ordering* enforcement algorithm exists, that uses local CO of each participating database, and needs only (unmodified) Atomic commitment protocol messages with no further communication. The distributed algorithm is the combination of local (to each database) CO algorithm processes, and an atomic commitment protocol (which can be fully distributed). Atomic commitment protocol is essential to enforce atomicity of each distributed transaction (to decide whether to commit or abort it; this procedure is always carried out for distributed transactions, independently of concurrency control and CO). A common example of an atomic commitment protocol is the *two-phase commit protocol*, which is resilient to many types of system failure. In a reliable environment, or when processes usually fail together (e.g., in the same integrated circuit), a simpler protocol for atomic commitment may be used (e.g., a simple handshake of distributed transaction's participating processes with some arbitrary but known special participant, the transaction's coordinator, i.e., a type of *one-phase commit* protocol). An atomic commitment protocol reaches consensus among participants on whether to *commit* or *abort* a distributed (global) transaction that spans these participants. An essential stage in each such protocol is the **YES vote** (either explicit, or implicit) by each participant, which means an obligation of the voting participant to obey the decision of the protocol, either commit or abort. Otherwise a participant can unilaterally abort the transaction by an explicit NO vote. The protocol commits the transaction only if YES votes have been received from *all* participants, and thus typically a missing YES vote of a participant is considered a NO vote by this participant. Otherwise the protocol aborts the transaction. The various atomic commit protocols only differ in their abilities to handle different computing environment failure situations, and the amounts of work and other computing resources needed in different situations.

The entire CO solution for global serializability is based on the fact that in case of a missing vote for a distributed transaction, the atomic commitment protocol eventually aborts this transaction.

Enforcing global CO

In each database system a local CO algorithm determines the needed commitment order for that database. By the characterization of CO above, this order depends on the local precedence order of transactions, which results from the local data access scheduling mechanisms. Accordingly YES votes in the atomic commitment protocol are scheduled for each (unaborted) distributed transaction (in what follows "a vote" means a YES vote). If a precedence relation (conflict) exists between two transactions, then the second will not be voted on before the first is completed (either committed or aborted), to prevent possible commit order violation by the atomic commitment protocol. Such can happen since the commit order by the protocol is not necessarily the same as the voting order. If no precedence relation exists, both can be voted on concurrently. This *vote ordering strategy* ensures that also the atomic commitment protocol maintains commitment order, and it is a *necessary condition* for guaranteeing Global CO (and the local CO of a database; without it both Global CO and Local CO (a property meaning that each database is CO compliant) may be violated).

However, since database systems schedule their transactions independently, it is possible that the transactions' precedence orders in two databases or more are not compatible (no global partial order exists that can embed the respective local partial orders together). With CO precedence orders are also the commitment orders. When participating databases in a same distributed transaction do not have compatible local precedence orders for that transaction (without "knowing" it; typically no coordination between database systems exists on conflicts, since the needed communication is massive and unacceptably degrades performance) it means that the transaction resides on a global cycle (involving two or more databases) in the global conflict graph. In this case the atomic commitment protocol will fail to collect all the votes needed to commit that transaction: By the *vote ordering strategy* above at least one database will delay its vote for that transaction indefinitely, to comply with its own commitment (precedence) order, since it will be waiting to the completion of another, preceding transaction on that global cycle, delayed indefinitely by another database with a different order. This means a ***voting-deadlock*** situation involving the databases on that cycle. As a result the protocol will eventually abort some deadlocked transaction on this global cycle, since each such transaction is missing at least one participant's vote. Selection of the specific transaction on the cycle to be aborted depends on the atomic commitment protocol's abort policies (a timeout mechanism is common, but it may result in more than one needed abort per cycle; both preventing unnecessary aborts and abort time shortening can be achieved by a dedicated abort mechanism for CO). Such abort will break the global cycle involving that distributed transaction. Both deadlocked transactions and possibly other in conflict with the deadlocked (and thus blocked) will be free to be voted on. It is worthwhile noting that each database involved with the voting-deadlock continues to vote regularly on transactions that are not in conflict with its deadlocked transaction, typically almost all the outstanding transactions. Thus, in case of incompatible local (partial) commitment orders, no action is needed since the atomic commitment protocol resolves it automatically by aborting a transaction that is a cause of incompatibility. This means that the above *vote ordering strategy* is also a *sufficient condition* for guaranteeing Global CO.

The following is concluded:

- **The Vote ordering strategy for Global CO Enforcing Theorem**

  Let T_1, T_2 be undecided (neither committed nor aborted) transactions in a database system that enforces CO for local transactions, such that T_2 is *global* and *in conflict* with T_1 (T_1 *precedes* T_2). Then, having T_1 ended (either committed or aborted) before T_2 is voted on to be committed (the *vote ordering strategy*), in each such database system in a multidatabase environment, is a necessary and sufficient condition for guaranteeing Global CO (the condition guarantees Global CO, which may be violated without it).
  Comments:

1. The *vote ordering strategy* that enforces global CO is referred to as CD^3C in (Raz 1992).
2. The Local CO property of a global schedule means that each database is CO compliant. From the necessity discussion part above it directly follows that the theorem is true also when replacing "Global CO" with "Local CO" when global transactions are present. Together it means that Global CO is guaranteed if and only if Local CO is guaranteed (which is untrue for Global conflict serializability and Local conflict serializability: Global implies Local, but not the opposite).

Global CO implies Global serializability.

The **Global CO algorithm** comprises enforcing (local) CO in each participating database system by ordering commits of local transactions (see Enforcing CO locally below) and enforcing the *vote ordering strategy* in the theorem above (for global transactions).

Exact characterization of voting-deadlocks by global cycles

The above global cycle elimination process by a **voting deadlock** can be explained in detail by the following observation:

First it is assumed, for simplicity, that every transaction reaches the ready-to-commit state and is voted on by at least one database (this implies that no blocking by locks occurs). Define a *"wait for vote to commit" graph* as a directed graph with transactions as nodes, and a directed edge from any first transaction to a second transaction if the first transaction blocks the vote to commit of the second transaction (opposite to conventional edge direction in a wait-for graph). Such blocking happens only if the second transaction is in a conflict with the first transaction (see above). Thus this "wait for vote to commit" graph is identical to the global conflict graph. A cycle in the "wait for vote to commit" graph means a deadlock in voting. Hence there is a deadlock in voting if and only if there is a cycle in the conflict graph. Local cycles (confined to a single database) are eliminated by the local serializability mechanisms. Consequently only global cycles are left, which are then eliminated by the atomic commitment protocol when it aborts deadlocked transactions with missing (blocked) respective votes.

Secondly, also local commits are dealt with: Note that when enforcing CO also waiting for a regular local commit of a local transaction can block local commits and votes of other transactions upon conflicts, and the situation for global transactions does not change also without the simplifying assumption above: The final result is the same also with local commitment for local transactions, without voting in atomic commitment for them.

Finally, blocking by a lock (which has been excluded so far) needs to be considered: A lock blocks a conflicting operation and prevents a conflict from being materialized. If the lock is released only after transaction end, it may block indirectly either a vote or a local commit of another transaction (which now cannot get to ready state), with the same effect as of a direct blocking of a vote or a local commit. In this case a cycle is generated in the conflict graph only if such a blocking by a lock is also represented by an edge. With such added edges representing events of blocking-by-a-lock, the conflict graph is becoming an *augmented conflict graph*.

- **Definition: augmented conflict graph**

  An **augmented conflict graph** is a conflict graph with added edges: In addition to the original edges a directed edge exists from transaction T_1 to transaction T_2 if two conditions are met:

1. T_2 is blocked by a data-access lock applied by T_1 (the blocking prevents the conflict of T_2 with T_1 from being materialized and have an edge in the regular conflict graph), and
2. This blocking will not stop before T_1 ends (commits or aborts; true for any locking-based CO)

   The graph can also be defined as the union of the (regular) *conflict graph* with the (reversed edge, regular) *wait-for graph*

   Comments:

1. Here, unlike the regular conflict graph, which has edges only for materialized conflicts, all conflicts, both materialized and non-materialized, are represented by edges.

2. Note that all the new edges are all the (reversed to the conventional) edges of the *wait-for graph*. The *wait-for graph* can be defined also as the graph of non-materialized conflicts. By the common conventions edge direction in a *conflict graph* defines time order between conflicting operations which is opposite to the time order defined by an edge in a *wait-for graph*.
3. Note that such global graph contains (has embedded) all the (reversed edge) regular local *wait-for* graphs, and also may include locking based global cycles (which cannot exist in the local graphs). For example, if all the databases on a global cycle are SS2PL based, then all the related vote blocking situations are caused by locks (this is the classical, and probably the only global deadlock situation dealt with in the database research literature). This is a global deadlock case where each related database creates a portion of the cycle, but the complete cycle does not reside in any local wait-for graph.

In the presence of CO the *augmented conflict graph* is in fact a (reversed edge) *local-commit and voting wait-for graph*: An edge exists from a first transaction, either local or global, to a second, if the second is waiting for the first to end in order to be either voted on (if global), or locally committed (if local). All *global cycles* (across two or more databases) in this graph generate voting-deadlocks. The graph's global cycles provide complete characterization for voting deadlocks and may include any combination of materialized and non-materialized conflicts. Only cycles of (only) materialized conflicts are also cycles of the regular conflict graph and affect serializability. One or more (lock related) non-materialized conflicts on a cycle prevent it from being a cycle in the regular conflict graph, and make it a locking related deadlock. All the global cycles (voting-deadlocks) need to be broken (resolved) to both maintain global serializability and resolve global deadlocks involving data access locking, and indeed they are all broken by the atomic commitment protocol due to missing votes upon a voting deadlock.

Comment: This observation also explains the correctness of *Extended CO (ECO)* below: Global transactions' voting order must follow the conflict graph order with vote blocking when order relation (graph path) exists between two global transactions. Local transactions are not voted on, and their (local) commits are not blocked upon conflicts. This results in same voting-deadlock situations and resulting global cycle elimination process for ECO.

The *voting-deadlock* situation can be summarized as follows:

- **The CO Voting-Deadlock Theorem**

  Let a multidatabase environment comprise CO compliant (which eliminates *local cycles*) database systems that enforce, each, *Global CO* (using the condition in the theorem above). Then a *voting-deadlock* occurs if and only if a *global cycle* (spans two or more databases) exists in the *Global augmented conflict graph* (also blocking by a data-access lock is represented by an edge). If the cycle does not break by any abort, then all the *global transactions* on it are involved with the respective voting-deadlock, and eventually each has its vote blocked (either directly, or indirectly by a data-access lock); if a local transaction resides on the cycle, eventually it has its (local) commit blocked.

  Comment: A rare situation of a voting deadlock (by missing blocked votes) can happen, with no voting for any transaction on the related cycle by any of the database systems involved with these transactions. This can occur when local sub-transactions are multi-threaded. The highest probability instance of such rare event involves two transactions on two simultaneous opposite cycles. Such global cycles (deadlocks) overlap with local cycles which are resolved locally, and thus typically resolved by local mechanisms without involving atomic commitment. Formally it is also a global cycle, but practically it is local (portions of local cycles generate a global one; to see this, split each global transaction (node) to local sub-transactions (its portions confined each to a single database); a directed edge exists between transactions if an edge exists between any respective local sub-transactions; a cycle is local if all its edges originate from a cycle among sub-transactions of the same database, and global if not; global and local can overlap: a same cycle among transactions can result from several different cycles among sub-transactions, and be both local and global).

Also the following locking based special case is concluded:

- **The CO Locking-based Global-Deadlock Theorem**

In a CO compliant multidatabase system a locking-based global-deadlock, involving at least one data-access lock (non-materialized conflict), and two or more database systems, is a reflection of a global cycle in the *Global augmented conflict graph*, which results in a voting-deadlock. Such cycle is not a cycle in the (regular) *Global conflict graph* (which reflects only materialized conflicts, and thus such cycle does not affect *serializability*).

Comments:

1. Any blocking (edge) in the cycle that is not by a data-access lock is a direct blocking of either voting or local commit. All voting-deadlocks are resolved (almost all by *Atomic commitment*; see comment above), including this locking-based type.
2. Locking-based global-deadlocks can be generated also in a completely SS2PL-based distributed environment (special case of CO based), where all the vote blocking (and voting-deadlocks) are caused by data-access locks. Many research articles have dealt for years with resolving such global deadlocks, but none (except the CO articles) is known (as of 2009) to notice that *atomic commitment* automatically resolves them. Such automatic resolutions are regularly occurring unnoticed in all existing SS2PL based multidatabase systems, often bypassing dedicated resolution mechanisms.

Voting-deadlocks are the key for the operation of distributed CO.

Global cycle elimination (here voting-deadlock resolution by *atomic commitment*) and resulting aborted transactions' re-executions are time consuming, regardless of concurrency control used. If databases schedule transactions independently, global cycles are unavoidable (in a complete analogy to cycles/deadlocks generated in local SS2PL; with distribution, any transaction or operation scheduling coordination results in autonomy violation, and typically also in substantial performance penalty). However, in many cases their likelihood can be made very low by implementing database and transaction design guidelines that reduce the number of conflicts involving a global transaction. This, primarily by properly handling hot spots (database objects with frequent access), and avoiding conflicts by using commutativity when possible (e.g., when extensively using counters, as in finances, and especially multi-transaction *accumulation counters*, which are typically hot spots).

Atomic commitment protocols are intended and designed to achieve atomicity without considering database concurrency control. They abort upon detecting or heuristically finding (e.g., by timeout; sometimes mistakenly, unnecessarily) missing votes, and typically unaware of global cycles. These protocols can be specially enhanced for CO (including CO's variants below) both to prevent unnecessary aborts, and to accelerate aborts used for breaking global cycles in the global augmented conflict graph (for better performance by earlier release upon transaction-end of computing resources and typically locked data). For example, existing locking based global deadlock detection methods, other than timeout, can be generalized to consider also local commit and vote direct blocking, besides data access blocking. A possible compromise in such mechanisms is effectively detecting and breaking the most frequent and relatively simple to handle length-2 global cycles, and using timeout for undetected, much less frequent, longer cycles.

Enforcing CO locally

Commitment ordering can be enforced locally (in a single database) by a dedicated CO algorithm, or by any algorithm/protocol that provides any special case of CO. An important such protocol, being utilized extensively in database systems, which generates a CO schedule, is the *strong strict two phase locking* protocol (SS2PL: "release transaction's locks only after the transaction has been either committed or aborted"; see below). SS2PL is a proper subset of the intersection of 2PL and strictness.

A generic local CO algorithm

A **generic local CO algorithm** (Raz 1992; Algorithm 4.1) is an algorithm independent of implementation details, that enforces exactly the CO property. It does not block data access (nonblocking), and consists of aborting a certain set of transactions (only if needed) upon committing a transaction. It aborts a (uniquely determined at any given time) minimal set of other undecided (neither committed, nor aborted) transactions that run locally and can cause serializability violation in the future (can later generate cycles of committed transactions in the conflict graph; this is the ABORT set of a committed transaction T; after committing T no transaction in ABORT at commit time can be committed, and all of them are doomed to be aborted). This set consists of all undecided transactions with directed edges in the conflict graph to the committed transaction. The size of this set cannot increase when that transaction is waiting to be committed (in ready state: processing has ended), and typically decreases in time as its transactions are being decided. Thus, unless real-time constraints exist to complete that transaction, it is preferred to wait with committing that transaction and let this set decrease in size. If another serializability mechanism exists locally (which eliminates cycles in the local conflict graph), or if no cycle involving that transaction exists, the set will be empty eventually, and no abort of set member is needed. Otherwise the set will stabilize with transactions on local cycles, and aborting set members will have to occur to break the cycles. Since in the case of CO conflicts generate blocking on commit, local cycles in the *augments conflict graph* (see above) indicate local commit-deadlocks, and deadlock resolution techniques as in SS2PL can be used (e.g., like *timeout* and *wait-for graph*). A local cycle in the *augmented conflict graph* with at least one non-materialized conflict reflects a locking-based deadlock. The local algorithm above, applied to the local augmented conflict graph rather than the regular local conflict graph, comprises the **generic enhanced local CO algorithm**, a single local cycle elimination mechanism, for both guaranteeing local serializability and handling locking based local deadlocks. Practically an additional concurrency control mechanism is always utilized, even solely to enforce recoverability. The generic CO algorithm does not affect local data access scheduling strategy, when it runs alongside of any other local concurrency control mechanism. It affects only the commit order, and for this reason it does not need to abort more transactions than those needed to be aborted for serializability violation prevention by any combined local concurrency control mechanism. The net effect of CO may be, at most, a delay of commit events (or voting in a distributed environment), to comply with the needed commit order (but not more delay than its special cases, for example, SS2PL, and on the average significantly less).

The following theorem is concluded:

- **The Generic Local CO Algorithm Theorem**

  When running alone or alongside any concurrency control mechanism in a database system then

1. The *Generic local CO algorithm* guarantees (local) CO (a CO compliant schedule).
2. The *Generic enhanced local CO algorithm* guarantees both (local) CO and (local) locking based deadlock resolution.

   and (when not using *timeout*, and no *real-time* transaction completion constraints are applied) neither algorithm aborts more transactions than the minimum needed (which is determined by the transactions' operations scheduling, out of the scope of the algorithms).

Example: Concurrent programming and Transactional memory

See also *Concurrent programming and Transactional memory*

With the proliferation of Multi-core processors, variants of the Generic local CO algorithm have been also increasingly utilized in Concurrent programming, Transactional memory, and especially in Software transactional memory for achieving serializability optimistically by "commit order" (e.g., Ramadan et al. 2009,[3] Zhang et al. 2006,[] von Parun et al. 2007[4]). Numerous related articles and patents utilizing CO have already been published.

Implementation considerations: The Commitment Order Coordinator (COCO)

A database system in a multidatabase environment is assumed. From a software architecture point of view a CO component that implements the generic CO algorithm locally, the *Commitment Order Coordinator* (COCO), can be designed in a straightforward way as a mediator between a (single) database system and an atomic commitment protocol component (Raz 1991b). However, the COCO is typically an integral part of the database system. The COCO's functions are to vote to commit on ready global transactions (processing has ended) according to the local commitment order, to vote to abort on transactions for which the database system has initiated an abort (the database system can initiate abort for any transaction, for many reasons), and to pass the atomic commitment decision to the database system. For local transactions (when can be identified) no voting is needed. For determining the commitment order the COCO maintains an updated representation of the local conflict graph (or local augmented conflict graph for capturing also locking deadlocks) of the undecided (neither committed nor aborted) transactions as a data structure (e.g., utilizing mechanisms similar to locking for capturing conflicts, but with no data-access blocking). The COCO component has an interface with its database system to receive "conflict," "ready" (processing has ended; readiness to vote on a global transaction or commit a local one), and "abort" notifications from the database system. It also interfaces with the atomic commitment protocol to vote and to receive the atomic commitment protocol's decision on each global transaction. The decisions are delivered from the COCO to the database system through their interface, as well as local transactions' commit notifications, at a proper commit order. The COCO, including its interfaces, can be enhanced, if it implements another variant of CO (see below), or plays a role in the database's concurrency control mechanism beyond voting in atomic commitment.

The COCO also guarantees CO locally in a single, isolated database system with no interface with an atomic commitment protocol.

CO is a necessary condition for global serializability across autonomous database systems

If the databases that participate in distributed transactions (i.e., transactions that span more than a single database) do not use any shared concurrency control information and use unmodified atomic commitment protocol messages (for reaching atomicity), then maintaining (local) *commitment ordering* or one of its generalizing variants (see below) is a necessary condition for guaranteeing global serializability (a proof technique can be found in (Raz 1992), and a different proof method for this in (Raz 1993a)); it is also a sufficient condition. This is a mathematical fact derived from the definitions of *serializability* and a *transaction*. It means that if not complying with CO, then global serializability cannot be guaranteed under this condition (the condition of no local concurrency control information sharing between databases beyond atomic commit protocol messages). Atomic commitment is a minimal requirement for a distributed transaction since it is always needed, which is implied by the definition of transaction.

(Raz 1992) defines *database autonomy* and *independence* as complying with this requirement without using any additional local knowledge:

- **Definition:** (concurrency control based) **autonomous database system**

  A database system is **Autonomous**, if it does not share with any other entity any concurrency control information beyond unmodified atomic commitment protocol messages. In addition it does not use for concurrency control any additional local information beyond conflicts (the last sentence does not appear explicitly but rather implied by further discussion in Raz 1992).

Using this definition the following is concluded:

- **The CO and Global serializability Theorem**

1. CO compliance of every *autonomous* database system (or transactional object) in a multidatabase environment is a *necessary condition* for guaranteeing Global serializability (without CO Global serializability may be violated).
2. CO compliance of every database system is a *sufficient condition* for guaranteeing Global serializability.

However, the definition of autonomy above implies, for example, that transactions are scheduled in a way that local transactions (confined to a single database) cannot be identified as such by an autonomous database system. This is realistic for some transactional objects, but too restrictive and less realistic for general purpose database systems. If autonomy is augmented with the ability to identify local transactions, then compliance with a more general property, *Extended commitment ordering* (ECO, see below), makes ECO the necessary condition.

Only in (Raz 2009) the notion of *Generalized autonomy* captures the intended notion of autonomy:

- **Definition: generalized autonomy**

  A database system has the *Generalized autonomy* property, if it does not share with any other database system any local concurrency information beyond (unmodified) atomic commit protocol messages (however any local information can be utilized).

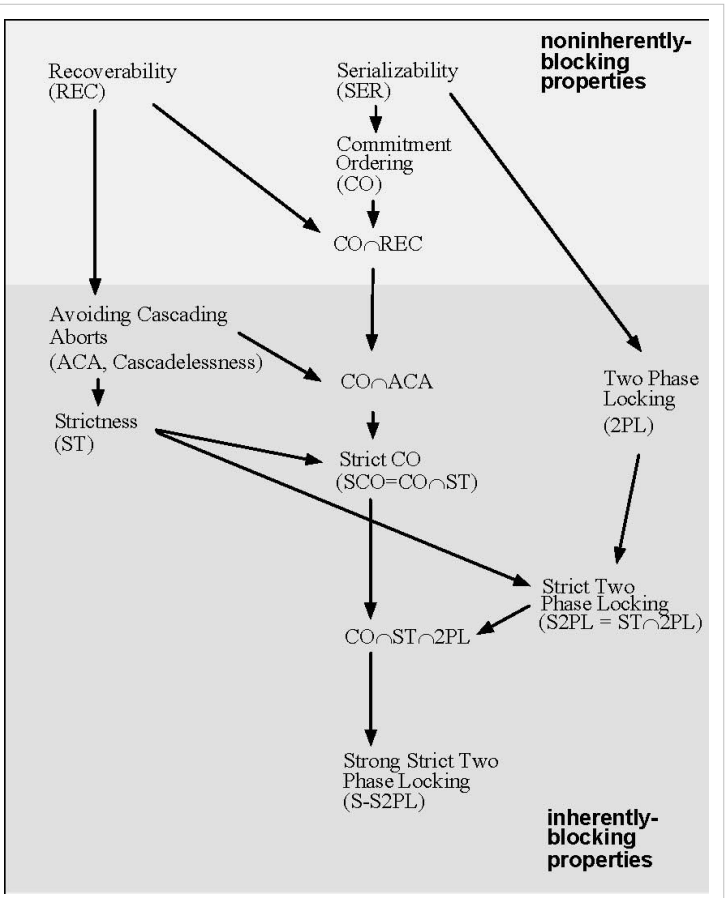
This definition is probably the broadest such definition possible in the context of database concurrency control, and it makes CO together with any of its (useful: No concurrency control information distribution) generalizing variants (Vote ordering (VO); see CO variants below) the necessary condition for Global serializability (i.e., the union of CO and its generalizing variants is the necessary set VO, which may include also new unknown useful generalizing variants).

Summary

The *Commitment ordering* (CO) solution (technique) for global serializability can be summarized as follows:

If each *database* (or any other *transactional object*) in a multidatabase environment complies with CO, i.e., arranges its local transactions' commitments and its votes on (global, distributed) transactions to the *atomic commitment* protocol according to the local (to the database) partial order induced by the local conflict graph (serializability graph) for the respective transactions, then *Global CO* and *Global serializability* are guaranteed. A database's CO compliance can be achieved effectively with any local conflict serializability based concurrency control mechanism, with neither affecting any transaction's execution process or scheduling, nor aborting it. Also the database's autonomy is not violated. The only low overhead incurred is detecting conflicts (e.g., as with locking, but with no data-access blocking; if not already detected for other purposes), and ordering votes and local transactions' commits according to the conflicts.

In case of incompatible partial orders of two or more databases (no global partial order can embed the respective local partial orders together), a global cycle (spans two databases or more) in the global conflict graph is generated. This, together with CO, results in a cycle of blocked votes, and a *voting-deadlock* occurs for the databases on that cycle (however, allowed concurrent voting in each database, typically for almost all the outstanding votes, continue to execute). In this case the atomic commitment protocol fails to collect all the votes needed for the blocked transactions on that global cycle, and consequently the protocol aborts some transaction with a missing vote. This breaks the global cycle, the voting-deadlock is resolved, and the related blocked votes are free to be executed. Breaking the global cycle in the global conflict graph ensures that both global CO and global serializability are maintained. Thus, in case of incompatible local (partial) commitment orders no action is needed since the atomic commitment protocol resolves it automatically by aborting a transaction that is a cause for the incompatibility. Furthermore, also global deadlocks due to locking (global cycles in the *augmented conflict graph* with at least one data access blocking) result in voting deadlocks and are resolved automatically by the same mechanism.



Schedule classes containment: An arrow from class A to class B indicates that class A strictly contains B; a lack of a directed path between classes means that the classes are incomparable.A property is inherently blocking, if it can be enforced only by blocking transaction's data access operations until certain events occur in other transactions. (#Raz1992Raz 1992)

Local CO is a necessary condition for guaranteeing *Global serializability,* if the databases involved do not share any concurrency control information beyond (unmodified) atomic commitment protocol messages, i.e., if the databases are *autonomous* in the context of concurrency control. This means that every global serializability solution for autonomous databases must comply with CO. Otherwise global serializability may be violated (and thus, is likely to be violated very quickly in a high-performance environment).

The CO solution scales up with network size and the number of databases without performance penalty when it utilizes common distributed atomic commitment architecture.

Distributed serializability and CO

Distributed CO

A distinguishing characteristic of the CO solution to distributed serializability from other techniques is the fact that it requires no conflict information distributed (e.g., local precedence relations, locks, timestamps, tickets), which makes it uniquely effective. It utilizes (unmodified) atomic commitment protocol messages (which are already used) instead.

A common way to achieve distributed serializability in a (distributed) system is by a distributed lock manager (DLM). DLMs, which communicate lock (non-materialized conflict) information in a distributed environment, typically suffer from computer and communication latency, which reduces the performance of the system. CO allows to achieve distributed serializability under very general conditions, without a distributed lock manager, exhibiting the benefits already explored above for multidatabase environments; in particular: reliability, high performance, scalability, possibility of using *optimistic concurrency control* when desired, no conflict information related communications over the network (which have incurred overhead and delays), and automatic distributed deadlock resolution.

All *distributed transactional systems* rely on some atomic commitment protocol to coordinate atomicity (whether to commit or abort) among processes in a distributed transaction. Also, typically *recoverable data* (i.e., data under transactions' control, e.g., database data; not to be confused with the *recoverability* property of a schedule) are directly accessed by a single *transactional data manager* component (also referred to as a *resource manager*) that handles local sub-transactions (the distributed transaction's portion in a single location, e.g., network node), even if these data are accessed indirectly by other entities in the distributed system during a transaction (i.e., indirect access requires a direct access through a local sub-transaction). Thus recoverable data in a distributed transactional system are typically partitioned among transactional data managers. In such system these transactional data managers typically comprise the participants in the system's atomic commitment protocol. If each participant complies with CO (e.g., by using SS2PL, or COCOs, or a combination; see above), then the entire distributed system provides CO (by the theorems above; each participant can be considered a separate transactional object), and thus (distributed) serializability. Furthermore: When CO is utilized together with an atomic commitment protocol also *distributed deadlocks* (i.e., deadlocks that span two or more data managers) caused by data-access locking are resolved automatically. Thus the following corollary is concluded:

- **The CO Based Distributed Serializability Theorem**

  Let a *distributed transactional system* (e.g., a distributed database system) comprise *transactional data managers* (also called *resource managers*) that manage all the system's *recoverable data*. The data managers meet three conditions:

1. **Data partition:** Recoverable data are partitioned among the data managers, i.e., each recoverable datum (data item) is controlled by a single data manager (e.g., as common in a Shared nothing architecture; even copies of a same datum under different data managers are physically distinct, *replicated*).
2. **Participants in atomic commitment protocol:** These data managers are the participants in the system's atomic commitment protocol for coordinating distributed transactions' atomicity.
3. **CO compliance:** Each such data manager is CO compliant (or some CO variant compliant; see below).

   Then

1. The entire distributed system guarantees (distributed CO and) *serializability*, and
2. Data-access based *distributed deadlocks* (deadlocks involving two or more data managers with at least one non-materialized conflict) are resolved automatically.

   Furthermore: The data managers being CO compliant is a *necessary condition* for (distributed) serializability in a system meeting conditions 1, 2 above, when the data managers are *autonomous*, i.e., do not share

concurrency control information beyond unmodified messages of atomic commitment protocol.

This theorem also means that when SS2PL (or any other CO variant) is used locally in each transactional data manager, and each data manager has exclusive control of its data, no distributed lock manager (which is often utilized to enforce distributed SS2PL) is needed for distributed SS2PL and serializability. It is relevant to a wide range of distributed transactional applications, which can be easily designed to meet the theorem's conditions.

Distributed optimistic CO (DOCO)

For implementing Distributed Optimistic CO (DOCO) the generic local CO algorithm is utilized in all the atomic commitment protocol participants in the system with no data access blocking and thus with no local deadlocks. The previous theorem has the following corollary:

- **The Distributed optimistic CO (DOCO) Theorem**

  If DOCO is utilized, then:

  1. No local deadlocks occur, and
  2. Global (voting) deadlocks are resolved automatically (and all are serializability related (with non-blocking conflicts) rather than locking related (with blocking and possibly also non-blocking conflicts)).

  Thus, no deadlock handling is needed.

Examples

Distributed SS2PL

A distributed database system that utilizes SS2PL resides on two remote nodes, A and B. The database system has two *transactional data managers* (*resource managers*), one on each node, and the database data are partitioned between the two data managers in a way that each has an exclusive control of its own (local to the node) portion of data: Each handles its own data and locks without any knowledge on the other manager's. For each distributed transaction such data managers need to execute the available atomic commitment protocol.

Two distributed transactions, $T_1$and T_2, are running concurrently, and both access data x and y. x is under the exclusive control of the data manager on A (B's manager cannot access x), and y under that on B.

T_1reads x on A and writes y on B, i.e., $T_1 = R_{1A}(x)\ W_{1B}(y)$when using notation common for concurrency control.

T_2reads y on B and writes x on A, i.e., $T_2 = R_{2B}(y)\ W_{2A}(x)$

The respective *local sub-transactions* on A and B (the portions of $T_1$and T_2on each of the nodes) are the following:

Local sub-transactions

Transaction \ Node	A	B
T_1	$T_{1A} = R_{1A}(x)$	$T_{1B} = W_{1B}(y)$
T_2	$T_{2A} = W_{2A}(x)$	$T_{2B} = R_{2B}(y)$

The database system's schedule at a certain point in time is the following:

$R_{1A}(x)\ R_{2B}(y)$

(also $R_{2B}(y)\ R_{1A}(x)$is possible)

T_1holds a read-lock on x and T_2holds read-locks on y. Thus $W_{1B}(y)$and $W_{2A}(x)$are blocked by the lock compatibility rules of SS2PL and cannot be executed. This is a distributed deadlock situation, which is also a voting-deadlock (see below) with a distributed (global) cycle of length 2 (number of edges, conflicts; 2 is the most

frequent length). The local sub-transactions are in the following states:

T_{1A} is *ready* (execution has ended) and *voted* (in atomic commitment)

T_{1B} is *running* and blocked (a non-materialized conflict situation; no vote on it can occur)

T_{2B} is *ready* and *voted*

T_{2A} is *running* and blocked (a non-materialized conflict; no vote).

Since the atomic commitment protocol cannot receive votes for blocked sub-transactions (a voting-deadlock), it will eventually abort some transaction with a missing vote(s) by timeout, either T_1, or T_2, (or both, if the timeouts fall very close). This will resolve the global deadlock. The remaining transaction will complete running, be voted on, and committed. An aborted transaction is immediately *restarted* and re-executed.

Comments:

1. The data partition (x on A; y on B) is important since without it, for example, x can be accessed directly from B. If a transaction T_3 is running on B concurrently with T_1 and T_2 and directly writes x, then, without a distributed lock manager the read-lock for x held by T_1 on A is not visible on B and cannot block the write of T_3 (or signal a materialized conflict for a non-blocking CO variant; see below). Thus serializability can be violated.
2. Due to data partition, x cannot be accessed directly from B. However, functionality is not limited, and a transaction running on B still can issue a write or read request of x (not common). This request is communicated to the transaction's local sub-transaction on A (which is generated, if does not exist already) which issues this request to the local data manager on A.

Variations

In the scenario above both conflicts are *non-materialized*, and the global voting-deadlock is reflected as a cycle in the global *wait-for graph* (but not in the global *conflict graph*; see Exact characterization of voting-deadlocks by global cycles above). However the database system can utilize any CO variant with exactly the same conflicts and voting-deadlock situation, and same resolution. Conflicts can be either *materialized* or *non-materialized*, depending on CO variant used. For example, if SCO (below) is used by the distributed database system instead of SS2PL, then the two conflicts in the example are *materialized*, all local sub-transactions are in *ready* states, and vote blocking occurs in the two transactions, one on each node, because of the CO voting rule applied independently on both A and B: due to conflicts $T_{2A} = W_{2A}(x)$ is not voted on before $T_{1A} = R_{1A}(x)$ ends, and $T_{1B} = W_{1B}(y)$ is not voted on before $T_{2B} = R_{2B}(y)$ ends, which is a voting-deadlock. Now the *conflict graph* has the global cycle (all conflicts are materialized), and again it is resolved by the atomic commitment protocol, and distributed serializability is maintained. Unlikely for a distributed database system, but possible in principle (and occurs in a multi-database), A can employ SS2PL while B employs SCO. In this case the global cycle is neither in the wait-for graph nor in the serializability graph, but still in the *augmented conflict graph* (the union of the two). The various combinations are summarized in the following table:

Voting-deadlock situations

Case	Node A	Node B	Possible schedule	Materialized conflicts on cycle	Non-materialized conflicts	$T_{1A} = R_{1A}(x)$	$T_{1B} = W_{1B}(y)$	$T_{2A} = W_{2A}(x)$	$T_{2B} = R_{2B}(y)$
1	SS2PL	SS2PL	$R_{1A}(x)\ R_{2B}(y)$	0	2	Ready Voted	Running (Blocked)	Running (Blocked)	Ready Voted
2	SS2PL	SCO	$R_{1A}(x)\ R_{2B}(y)\ W_{1B}(y)$	1	1	Ready Voted	Ready Vote blocked	Running (Blocked)	Ready Voted
3	SCO	SS2PL	$R_{1A}(x)\ R_{2B}(y)\ W_{2A}(x)$	1	1	Ready Voted	Running (Blocked)	Ready Vote blocked	Ready Voted

4	SCO	SCO	$R_{1A}(x)\ R_{2B}(y)\ W_{1B}(y)\ W_{2A}(x)$	2	0	Ready Voted	Ready Vote blocked	Ready Vote blocked	Ready Voted

Comments:

1. Conflicts and thus cycles in the *augmented conflict graph* are determined by the transactions and their initial scheduling only, independently of the concurrency control utilized. With any variant of CO, any *global cycle* (i.e., spans two databases or more) causes a *voting deadlock*. Different CO variants may differ on whether a certain conflict is *materialized* or *non-materialized*.
2. Some limited operation order changes in the schedules above are possible, constrained by the orders inside the transactions, but such changes do not change the rest of the table.
3. As noted above, only case 4 describes a cycle in the (regular) conflict graph which affects serializability. Cases 1-3 describe cycles of locking based global deadlocks (at least one lock blocking exists). All cycle types are equally resolved by the atomic commitment protocol. Case 1 is the common Distributed SS2PL, utilized since the 1980s. However, no research article, except the CO articles, is known to notice this automatic locking global deadlock resolution as of 2009. Such global deadlocks typically have been dealt with by dedicated mechanisms.
4. Case 4 above is also an example for a typical voting-deadlock when Distributed optimistic CO (DOCO) is used (i.e., Case 4 is unchanged when Optimistic CO (OCO; see below) replaces SCO on both A and B): No data-access blocking occurs, and only materialized conflicts exist.

Hypothetical Multi Single-Threaded Core (MuSiC) environment

Comment: While the examples above describe real, recommended utilization of CO, this example is hypothetical, for demonstration only.

Certain experimental distributed memory-resident databases advocate multi single-threaded core (MuSiC) transactional environments. "Single-threaded" refers to transaction threads only, and to *serial* execution of transactions. The purpose is possible orders of magnitude gain in performance (e.g., H-Store[5] and VoltDB) relatively to conventional transaction execution in multiple threads on a same core. In what described below MuSiC is independent of the way the cores are distributed. They may reside in one integrated circuit (chip), or in many chips, possibly distributed geographically in many computers. In such an environment, if recoverable (transactional) data are partitioned among threads (cores), and it is implemented in the conventional way for distributed CO, as described in previous sections, then DOCO and Strictness exist automatically. However, downsides exist with this straightforward implementation of such environment, and its practicality as a general-purpose solution is questionable. On the other hand tremendous performance gain can be achieved in applications that can bypass these downsides in most situations.

Comment: The MuSiC straightforward implementation described here (which uses, for example, as usual in distributed CO, voting (and transaction thread) blocking in atomic commitment protocol when needed) is for demonstration only, and has **no connection** to the implementation in H-Store or any other project.

In a MuSiC environment local schedules are *serial*. Thus both local Optimistic CO (OCO; see below) and the *Global CO enforcement vote ordering strategy* condition for the atomic commitment protocol are met automatically. This results in both distributed CO compliance (and thus distributed serializability) and automatic global (voting) deadlock resolution.

Furthermore, also local *Strictness* follows automatically in a serial schedule. By Theorem 5.2 in (Raz 1992; page 307), when the CO vote ordering strategy is applied, also Global Strictness is guaranteed. Note that *serial* locally is the only mode that allows strictness and "optimistic" (no data access blocking) together.

The following is concluded:

- **The MuSiC Theorem**

  In MuSiC environments, if recoverable (transactional) data are partitioned among cores (threads), then both

1. *OCO* (and implied *Serializability*; i.e., DOCO and Distributed serializability)
2. *Strictness* (allowing effective recovery; 1 and 2 implying Strict CO—see SCO below) and
3. (voting) *deadlock resolution*

automatically exist globally with unbounded scalability in number of cores used.

Comment: However, two major downsides, which need special handling, may exist:

1. Local sub-transactions of a global transaction are blocked until commit, which makes the respective cores idle. This reduces core utilization substantially, even if scheduling of the local sub-transactions attempts to execute all of them in time proximity, almost together. It can be overcome by detaching execution from commit (with some atomic commitment protocol) for global transactions, at the cost of possible cascading aborts.
2. increasing the number of cores for a given amount of recoverable data (database size) decreases the average amount of (partitioned) data per core. This may make some cores idle, while others very busy, depending on data utilization distribution. Also a local (to a core) transaction may become global (multi-core) to reach its needed data, with additional incurred overhead. Thus, as the number of cores increases, the amount and type of data assigned to each core should be balanced according to data usage, so a core is neither overwhelmed to become a bottleneck, nor becoming idle too frequently and underutilized in a busy system. Another consideration is putting in a same core partition all the data that are usually accessed by a same transaction (if possible), to maximize the number of local transactions (and minimize the number of global, distributed transactions). This may be achieved by occasional data re-partition among cores based on load balancing (data access balancing) and patterns of data usage by transactions. Another way to considerably mitigate this downside is by proper physical data replication among some core partitions in a way that read-only global transactions are possibly (depending on usage patterns) completely avoided, and replication changes are synchronized by a dedicated commit mechanism.

CO variants: Interesting special cases and generalizations

Special case schedule property classes (e.g., SS2PL and SCO below) are strictly contained in the CO class. The generalizing classes (ECO and MVCO) strictly contain the CO class (i.e., include also schedules that are not CO compliant). The generalizing variants also guarantee global serializability without distributing local concurrency control information (each database has the *generalized autonomy* property: it uses only local information), while relaxing CO constraints and utilizing additional (local) information for better concurrency and performance: ECO uses knowledge about transactions being local (i.e., confined to a single database), and MVCO uses availability of data versions values. Like CO, both generalizing variants are *non-blocking*, do not interfere with any transaction's operation scheduling, and can be seamlessly combined with any relevant concurrency control mechanism.

The term **CO variant** refers in general to CO, ECO, MVCO, or a combination of each of them with any relevant concurrency control mechanism or property (including Multi-version based ECO, MVECO). No other interesting generalizing variants (which guarantee global serializability with no local concurrency control information distribution) are known, but may be discovered.

Strong strict two phase locking (SS2PL)

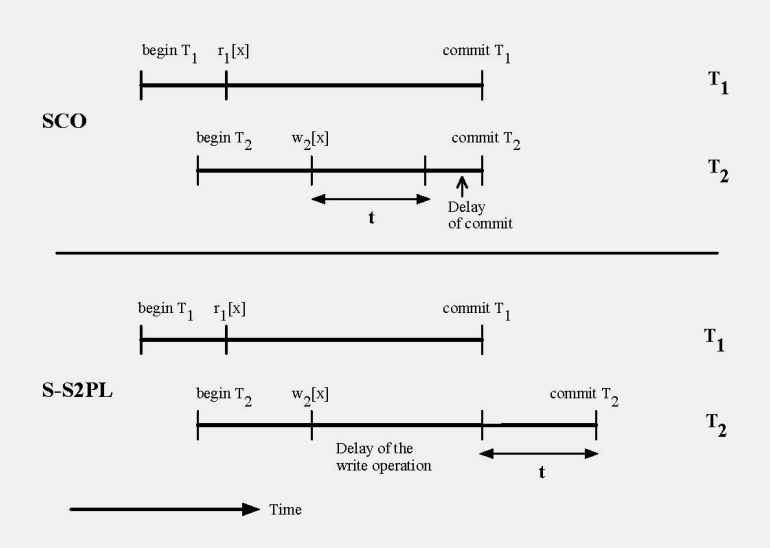
Strong Strict Two Phase Locking (SS2PL; also referred to as *Rigorousness* or *Rigorous scheduling*) means that both read and write locks of a transaction are released only after the transaction has ended (either committed or aborted). The set of SS2PL schedules is a proper subset of the set of CO schedules. This property is widely utilized in database systems, and since it implies CO, databases that use it and participate in global transactions generate together a serializable global schedule (when using any atomic commitment protocol, which is needed for atomicity in a multi-database environment). No database modification or addition is needed in this case to participate in a CO distributed solution: The set of undecided transactions to be aborted before committing in the local generic CO algorithm above is empty because of the locks, and hence such an algorithm is unnecessary in this case. A transaction can be voted on by a database system immediately after entering a "ready" state, i.e., completing running

its task locally. Its locks are released by the database system only after it is decided by the atomic commitment protocol, and thus the condition in the *Global CO enforcing theorem* above is kept automatically. Interestingly, if a local timeout mechanism is used by a database system to resolve (local) SS2PL deadlocks, then aborting blocked transactions breaks not only potential local cycles in the global conflict graph (real cycles in the augmented conflict graph), but also database system's potential global cycles as a side effect, if the atomic commitment protocol's abort mechanism is relatively slow. Such independent aborts by several entities typically may result in unnecessary aborts for more than one transaction per global cycle. The situation is different for a local *wait-for graph* based mechanisms: Such cannot identify global cycles, and the atomic commitment protocol will break the global cycle, if the resulting voting deadlock is not resolved earlier in another database.

Local SS2PL together with atomic commitment implying global serializability can also be deduced directly: All transactions, including distributed, obey the 2PL (SS2PL) rules. The atomic commitment protocol mechanism is not needed here for consensus on commit, but rather for the end of phase-two synchronization point. Probably for this reason, without considering the atomic commitment voting mechanism, automatic global deadlock resolution has not been noticed before CO.

Strict CO (SCO)

Strict Commitment Ordering (SCO; (Raz 1991c)) is the intersection of strictness (a special case of recoverability) and CO, and provides an upper bound for a schedule's concurrency when both properties exist. It can be implemented using blocking mechanisms (locking) similar to those used for the popular SS2PL with similar overheads.



Read-write conflict: SCO Vs. SS2PL. Duration of transaction T2 is longer with SS2PL than with SCO.SS2PL delays write operation w2[x] of T2 until T1 commits, due to a lock on x by T1 following read operation r1[x]. If t time units are needed for transaction T2 after starting write operation w2[x] in order to reach ready state, than T2 commits t time units after T1 commits. However, SCO does not block w2[x], and T2 can commit immediately after T1 commits. (#Raz1991cRaz 1991c)

Unlike SS2PL, SCO does not block on a read-write conflict but possibly blocks on commit instead. SCO and SS2PL have identical blocking behavior for the other two conflict types: write-read, and write-write. As a result SCO has shorter average blocking periods, and more concurrency (e.g., performance simulations of a single database for the most significant variant of *locks with ordered sharing,* which is identical to SCO, clearly show this, with approximately 100% gain for some transaction loads; also for identical transaction loads SCO can reach higher transaction rates than SS2PL before *lock thrashing* occurs). More concurrency means that with given computing resources more transactions are completed in time unit (higher transaction rate, throughput), and the average duration of a transaction is shorter (faster completion; see chart). The advantage of SCO is especially significant during lock contention.

- **The SCO Vs. SS2PL Performance Theorem**

  SCO provides shorter average transaction completion time than SS2PL, if read-write conflicts exist. SCO and SS2PL are identical otherwise (have identical blocking behavior with write-read and write-write conflicts).

SCO is as practical as SS2PL since as SS2PL it provides besides serializability also strictness, which is widely utilized as a basis for efficient recovery of databases from failure. An SS2PL mechanism can be converted to an SCO one for better performance in a straightforward way without changing recovery methods. A description of a SCO implementation can be found in (Perrizo and Tatarinov 1998). See also *Semi-optimistic database scheduler*.

SS2PL is a proper subset of SCO (which is another explanation why SCO is less constraining and provides more concurrency than SS2PL).

Optimistic CO (OCO)

For implementing **Optimistic commitment ordering** (OCO) the generic local CO algorithm is utilized without data access blocking, and thus without local deadlocks. OCO without transaction or operation scheduling constraints covers the entire CO class, and is not a special case of the CO class, but rather a useful CO variant and mechanism characterization.

Extended CO (ECO)

General characterization of ECO

Extended Commitment Ordering (ECO; (Raz 1993a)) generalizes CO. When local transactions (transactions confined to a single database) can be distinguished from global (distributed) transactions (transactions that span two databases or more), commitment order is applied to global transactions only. Thus, for a local (to a database) schedule to have the ECO property, the chronological (partial) order of commit events of global transactions only (unimportant for local transactions) is consistent with their order on the respective local conflict graph.

- **Definition: extended commitment ordering**

  Let T_1, T_2 be two committed *global* transactions in a schedule, such that a *directed path* of unaborted transactions exists in the *conflict graph* (precedence graph) from T_1 to T_2 (T_1 precedes T_2, possibly transitively, indirectly). The schedule has the **Extended commitment ordering** (ECO) property, if for every two such transactions T_1 commits before T_2 commits.

A distributed algorithm to guarantee global ECO exists. As for CO, the algorithm needs only (unmodified) atomic commitment protocol messages. In order to guarantee global serializability, each database needs to guarantee also the conflict serializability of its own transactions by any (local) concurrency control mechanism.

- **The ECO and Global Serializability Theorem**

1. (Local, which implies global) ECO together with local conflict serializability, is a sufficient condition to guarantee global conflict serializability.
2. When no concurrency control information beyond atomic commitment messages is shared outside a database (autonomy), and local transactions can be identified, it is also a necessary condition.

   See a necessity proof in (Raz 1993a).

This condition (ECO with local serializability) is weaker than CO, and allows more concurrency at the cost of a little more complicated local algorithm (however, no practical overhead difference with CO exists).

When all the transactions are assumed to be global (e.g., if no information is available about transactions being local), ECO reduces to CO.

The ECO algorithm

Before a global transaction is committed, a generic local (to a database) ECO algorithm aborts a minimal set of undecided transactions (neither committed, nor aborted; either local transactions, or global that run locally), that can cause later a cycle in the conflict graph. This set of aborted transactions (not unique, contrary to CO) can be optimized, if each transaction is assigned with a weight (that can be determined by transaction's importance and by the computing resources already invested in the running transaction; optimization can be carried out, for example, by a reduction from the *Max flow in networks* problem (Raz 1993a)). Like for CO such a set is time dependent, and becomes empty eventually. Practically, almost in all needed implementations a transaction should be committed only when the set is empty (and no set optimization is applicable). The local (to the database) concurrency control mechanism (separate from the ECO algorithm) ensures that local cycles are eliminated (unlike with CO, which implies serializability by itself; however, practically also for CO a local concurrency mechanism is utilized, at least to ensure Recoverability). Local transactions can be always committed concurrently (even if a precedence relation exists, unlike CO). When the overall transactions' local partial order (which is determined by the local conflict graph, now only with possible temporary local cycles, since cycles are eliminated by a local serializability mechanism) allows, also global transactions can be voted on to be committed concurrently (when all their transitively (indirect) preceding (via conflict) *global* transactions are committed, while transitively preceding local transactions can be at any state. This in analogy to the distributed CO algorithm's stronger concurrent voting condition, where all the transitively preceding transactions need to be committed).

The condition for guaranteeing *Global ECO* can be summarized similarly to CO:

- **The Global ECO Enforcing Vote ordering strategy Theorem**

  Let T_1, T_2 be undecided (neither committed nor aborted) *global transactions* in a database system that ensures serializability locally, such that a *directed path* of unaborted transactions exists in the *local conflict graph* (that of the database itself) from T_1 to T_2. Then, having T_1 ended (either committed or aborted) before T_2 is voted on to be committed, in every such database system in a multidatabase environment, is a necessary and sufficient condition for guaranteeing Global ECO (the condition guarantees Global ECO, which may be violated without it).

Global ECO (all global cycles in the global conflict graph are eliminated by atomic commitment) together with Local serializability (i.e., each database system maintains serializability locally; all local cycles are eliminated) imply Global serializability (all cycles are eliminated). This means that if each database system in a multidatabase environment provides local serializability (by *any* mechanism) and enforces the *vote ordering strategy* in the theorem above (a generalization of CO's vote ordering strategy), then *Global serializability* is guaranteed (no local CO is needed anymore).

Similarly to CO as well, the ECO *voting-deadlock* situation can be summarized as follows:

- **The ECO Voting-Deadlock Theorem**

  Let a multidatabase environment comprise database systems that enforce, each, both *Global ECO* (using the condition in the theorem above) and *local conflict serializability* (which eliminates local cycles in the global conflict graph). Then, a *voting-deadlock* occurs if and only if a *global cycle* (spans two or more databases) exists in the *Global augmented conflict graph* (also blocking by a data-access lock is represented by an edge). If the cycle does not break by any abort, then all the *global transactions* on it are involved with the respective voting-deadlock, and eventually each has its vote blocked (either directly, or indirectly by a data-access lock). If a local transaction resides on the cycle, it may be in any unaborted state (running, ready, or committed; unlike CO no local commit blocking is needed).

As with CO this means that also global deadlocks due to data-access locking (with at least one lock blocking) are voting deadlocks, and are automatically resolved by atomic commitment.

Multi-version CO (MVCO)

Multi-version Commitment Ordering (MVCO; (Raz 1993b)) is a generalization of CO for databases with multi-version resources. With such resources *read-only transactions* do not block or being blocked for better performance. Utilizing such resources is a common way nowadays to increase concurrency and performance by generating a new version of a database object each time the object is written, and allowing transactions' read operations of several last relevant versions (of each object). MVCO implies *One-copy-serializability* (1SER or 1SR) which is the generalization of serializability for multi-version resources. Like CO, MVCO is non-blocking, and can be combined with any relevant multi-version concurrency control mechanism without interfering with it. In the introduced underlying theory for MVCO conflicts are generalized for different versions of a same resource (differently from earlier multi-version theories). For different versions conflict chronological order is replaced by version order, and possibly reversed, while keeping the usual definitions for conflicting operations. Results for the regular and augmented conflict graphs remain unchanged, and similarly to CO a distributed MVCO enforcing algorithm exists, now for a mixed environment with both single-version and multi-version resources (now single-version is a special case of multi-version). As for CO, the MVCO algorithm needs only (unmodified) atomic commitment protocol messages with no additional communication overhead. Locking-based global deadlocks translate to voting deadlocks and are resolved automatically. In analogy to CO the following holds:

- **The MVCO and Global one-copy-serializability Theorem**

1. MVCO compliance of every *autonomous* database system (or transactional object) in a mixed multidatabase environment of single-version and multi-version databases is a *necessary condition* for guaranteeing Global one-copy-serializability (1SER).
2. MVCO compliance of every database system is a *sufficient condition* for guaranteeing Global 1SER.
3. Locking-based global deadlocks are resolved automatically.

Comment: Now a CO compliant single-version database system is automatically also MVCO compliant.

MVCO can be further generalized to employ the generalization of ECO (MVECO).

Example: CO based snapshot isolation (COSI)

CO based snapshot isolation (COSI) is the intersection of *Snapshot isolation* (SI) with MVCO. SI is a multiversion concurrency control method widely utilized due to good performance and similarity to serializability (1SER) in several aspects. The theory in (Raz 1993b) for MVCO described above is utilized later in (Fekete et al. 2005) and other articles on SI, e.g., (Cahill et al. 2008);[6] see also Making snapshot isolation serializable and the references there), for analyzing conflicts in SI in order to make it serializable. The method presented in (Cahill et al. 2008), *Serializable snapshot isolation* (SerializableSI), a low overhead modification of SI, provides good performance results versus SI, with only small penalty for enforcing serializability. A different method, by combining SI with MVCO (COSI), makes SI serializable as well, with a relatively low overhead, similarly to combining the generic CO algorithm with single-version mechanisms. Furthermore, the resulting combination, COSI, being MVCO compliant, allows COSI compliant database systems to inter-operate and transparently participate in a CO solution for distributed/global serializability (see below). Besides overheads also protocols' behaviors need to be compared quantitatively. On one hand, all serializable SI schedules can be made MVCO by COSI (by possible commit delays when needed) without aborting transactions. On the other hand, SerializableSI is known to unnecessarily abort and restart certain percentages of transactions also in serializable SI schedules.

CO and its variants are transparently interoperable for global serializability

With CO and its variants (e.g., SS2PL, SCO, OCO, ECO, and MVCO above) global serializability is achieved via *atomic commitment* protocol based distributed algorithms. For CO and all its variants atomic commitment protocol is the instrument to eliminate global cycles (cycles that span two or more databases) in the *global augmented* (and thus also regular) *conflict graph* (implicitly; no global data structure implementation is needed). In cases of either incompatible local commitment orders in two or more databases (when no global partial order can embed the respective local partial orders together), or a data-access locking related voting deadlock, both implying a global cycle in the global augmented conflict graph and missing votes, the atomic commitment protocol breaks such cycle by aborting an undecided transaction on it (see The distributed CO algorithm above). Differences between the various variants exist at the local level only (within the participating database systems). Each local CO instance of any variant has the same role, to determine the position of every global transaction (a transaction that spans two or more databases) within the local commitment order, i.e., to determine when it is the transaction's turn to be voted on locally in the atomic commitment protocol. Thus, all the CO variants exhibit the same behavior in regard to atomic commitment. This means that they are all interoperable via atomic commitment (using the same software interfaces, typically provided as services, some already standardized for atomic commitment, primarily for the two phase commit protocol, e.g., X/Open XA) and transparently can be utilized together in any distributed environment (while each CO variant instance is possibly associated with any relevant local concurrency control mechanism type).

In summary, any single global transaction can participate simultaneously in databases that may employ each any, possibly different, CO variant (while concurrently running processes in each such database, and running concurrently with local and other global transactions in each such database). The atomic commitment protocol is indifferent to CO, and does not distinguish between the various CO variants. Any *global cycle* generated in the augmented global conflict graph may span databases of different CO variants, and generate (if not broken by any local abort) a voting deadlock that is resolved by atomic commitment exactly the same way as in a single CO variant environment. *local cycles* (now possibly with mixed materialized and non-materialized conflicts, both serializability and data-access-locking deadlock related, e.g., SCO) are resolved locally (each by its respective variant instance's own local mechanisms).

Vote ordering (VO or Generalized CO (GCO); Raz 2009), the union of CO and all its above variants, is a useful concept and global serializability technique. To comply with VO, local serializability (in it most general form, commutativity based, and including multi-versioning) and the *vote order strategy* (voting by local precedence order) are needed.

Combining results for CO and its variants, the following is concluded:

- **The CO Variants Interoperability Theorem**

1. In a multi-database environment, where each database system (transactional object) is compliant with some CO variant property (VO compliant), any global transaction can participate simultaneously in databases of possibly different CO variants, and Global serializability is guaranteed (*sufficient condition* for Global serializability; and Global one-copy-serializability (1SER), for a case when a multi-version database exists).
2. If only local (to a database system) concurrency control information is utilized by every database system (each has the *generalized autonomy* property, a generalization of *autonomy*), then compliance of each with some (any) CO variant property (VO compliance) is a *necessary condition* for guaranteeing Global serializability (and Global 1SER; otherwise they may be violated).
3. Furthermore, in such environment data-access-locking related global deadlocks are resolved automatically (each such deadlock is generated by a global cycle in the *augmented conflict graph* (i.e., a *voting deadlock*; see above), involving at least one data-access lock (non-materialized conflict) and two database systems; thus, not a cycle in the regular conflict graph and does not affect serializability).

References

- Raz, Yoav (August 1992), "The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Managers Using Atomic Commitment" [6], *Proceedings of the Eighteenth International Conference on Very Large Data Bases* (Vancouver, Canada): 292–312 (also DEC-TR 841, Digital Equipment Corporation, November 1990)
- Raz, Yoav (September 1994), "Serializability by Commitment Ordering", *Information Processing Letters* **51** (5): 257–264, doi:10.1016/0020-0190(94)90005-1 [7]
- Raz, Yoav (June 2009), *Theory of Commitment Ordering: Summary* [8], retrieved November 11, 2011
- Raz, Yoav (November 1990), *On the Significance of Commitment Ordering* [9], Digital Equipment Corporation
- Yoav Raz (1991a): US patents 5,504,899 (ECO) [10] 5,504,900 (CO) [11] 5,701,480 (MVCO) [12]
- Yoav Raz (1991b): "The Commitment Order Coordinator (COCO) of a Resource Manager, or Architecture for Distributed Commitment Ordering Based Concurrency Control", DEC-TR 843, Digital Equipment Corporation, December 1991.
- Yoav Raz (1991c): "Locking Based Strict Commitment Ordering, or How to improve Concurrency in Locking Based Resource Managers", DEC-TR 844, December 1991.
- Yoav Raz (1993a): "Extended Commitment Ordering or Guaranteeing Global Serializability by Applying Commitment Order Selectivity to Global Transactions." [13] *Proceedings of the Twelfth ACM Symposium on Principles of Database Systems* (PODS), Washington, DC, pp. 83-96, May 1993. (also DEC-TR 842, November 1991)
- Yoav Raz (1993b): "Commitment Ordering Based Distributed Concurrency Control for Bridging Single and Multi Version Resources." [14] *Proceedings of the Third IEEE International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems* (RIDE-IMS), Vienna, Austria, pp. 189-198, April 1993. (also DEC-TR 853, July 1992)

Footnotes

[1] Alan Fekete, Nancy Lynch, Michael Merritt, William Weihl (1988): *Commutativity-based locking for nested transactions* (PDF) (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA200980&Location=U2&doc=GetTRDoc.pdf) MIT, LCS lab, Technical report MIT/LCS/TM-370, August 1988.

[2] Philip A. Bernstein, Eric Newcomer (2009): *Principles of Transaction Processing*, 2nd Edition (http://www.elsevierdirect.com/product.jsp?isbn=9781558606234), Morgan Kaufmann (Elsevier), June 2009, ISBN 978-1-55860-623-4 (pages 145, 360)

[3] Hany E. Ramadan, Indrajit Roy, Maurice Herlihy, Emmett Witchel (2009): "Committing conflicting transactions in an STM" (http://portal.acm.org/citation.cfm?id=1504201) (PDF (http://www.cs.utexas.edu/~indrajit/pubs/ppopp121-ramadan.pdf)) *Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming* (PPoPP '09), ISBN 978-1-60558-397-6

[4] Christoph von Praun, Luis Ceze, Calin Cascaval (2007) "Implicit Parallelism with Ordered Transactions" (http://portal.acm.org/citation.cfm?id=1229443) (PDF (http://www.cs.washington.edu/homes/luisceze/publications/ipot_ppopp07.pdf)), *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming* (PPoPP '07), ACM New York ©2007, ISBN 978-1-59593-602-8 doi 10.1145/1229428.1229443

[5] Robert Kallman, Hideaki Kimura, Jonathan Natkins, Andrew Pavlo, Alex Rasin, Stanley Zdonik, Evan Jones, Yang Zhang, Samuel Madden, Michael Stonebraker, John Hugg, Daniel Abadi (2008): "H-Store: A High-Performance, Distributed Main Memory Transaction Processing System" (http://portal.acm.org/citation.cfm?id=1454211), *Proceedings of the 2008 VLDB*, pages 1496 - 1499, Auckland, New-Zealand, August 2008.

[6] Michael J. Cahill, Uwe Röhm, Alan D. Fekete (2008): "Serializable isolation for snapshot databases" (http://portal.acm.org/citation.cfm?id=1376690), *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 729-738, Vancouver, Canada, June 2008, ISBN 978-1-60558-102-6 (SIGMOD 2008 best paper award

[7] http://dx.doi.org/10.1016%2F0020-0190%2894%2990005-1

[8] http://sites.google.com/site/yoavraz2/home/theory-of-commitment-ordering

[9] http://yoavraz.googlepages.com/DEC-CO-MEMO-90-11-16.pdf

[10] http://patft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=3&f=G&l=50&co1=AND&d=PTXT&s1=%22commitment+ordering%22.TI.&OS=TTL/

[11] http://patft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=2&f=G&l=50&co1=AND&d=PTXT&s1=%22commitment+ordering%22.TI.&OS=TTL/
[12] http://patft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=1&f=G&l=50&co1=AND&d=PTXT&s1=%22commitment+ordering%22.TI.&OS=TTL/
[13] http://portal.acm.org/citation.cfm?id=153858
[14] http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=281924

External links

- Yoav Raz's Commitment ordering page (http://sites.google.com/site/yoavraz2/the_principle_of_co)

Long-running transaction

Long-running transactions are computer database transactions that avoid locks on non-local resources, use compensation to handle failures, potentially aggregate smaller ACID transactions (also referred to as atomic transactions), and typically use a coordinator to complete or abort the transaction. In contrast to rollback in ACID transactions, compensation restores the original state, or an equivalent, and is business-specific. For example, the compensating action for making a hotel reservation is canceling that reservation, possibly with a penalty.

A number of protocols have been specified for long-running transactions using Web services within business processes. OASIS Business Transaction Processing [1] and WS-CAF [2] are examples. These protocols use a coordinator to mediate the successful completion or use of compensation in a long-running transaction.

References

[1] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction
[2] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf

Timestamp-based concurrency control

In computer science, a **timestamp-based concurrency control** algorithm is a non-lock concurrency control method. It is used in some databases to safely handle transactions, using timestamps.

Operation

Assumptions

- Every timestamp value is unique and accurately represents an instant in time.
- No two timestamps can be the same.
- A higher-valued timestamp occurs later in time than a lower-valued timestamp.

Generating a Timestamp

A number of different ways have been used to generate timestamp

- Use the value of the system's clock at the start of a transaction as the timestamp.
- Use a thread-safe shared counter that is incremental at the start of a transaction as the timestamp.
- A combination of the above two methods.

Formal

Each transaction (T_i) is an ordered list of actions (A_{ix}). Before the transaction performs its first action (A_{i1}), it is marked with the current timestamp, or any other strictly totally ordered sequence: $TS(T_i) = NOW()$. Every transaction is also given an initially empty set of transactions upon which it depends, $DEP(T_i) = []$, and an initially empty set of old objects which it updated, $OLD(T_i) = []$.
Each object (O_j) in the database is given two timestamp fields which are not used other than for concurrency control: $RTS(O_j)$ is the time at which the value of object was last used by a transaction, $WTS(O_j)$ is the time at which the value of the object was last updated by a transaction.
For all T_i:

For each action A_{ix}:

If A_{ix} wishes to use the value of O_j:

If $WTS(O_j) > TS(T_i)$ then **abort** (a more recent thread has overwritten the value),

Otherwise update the set of dependencies $DEP(T_i).\text{add}(WTS(O_j))$ and set $RTS(O_j) = \max(RTS(O_j), TS(T_i))$;

If A_{ix} wishes to update the value of O_j:

If $RTS(O_j) > TS(T_i)$ then **abort** (a more recent thread is already relying on the old value),

If $WTS(O_j) > TS(T_i)$ then **skip** (the Thomas Write Rule),

Otherwise store the previous values, $OLD(T_i).\text{add}(O_j, WTS(O_j))$, set $WTS(O_j) = TS(T_i)$, and update the value of O_j.

While there is a transaction in $DEP(T_i)$ that has not ended: **wait**

If there is a transaction in $DEP(T_i)$ that aborted then **abort**

Otherwise: **commit**.

To **abort**:

For each $(\text{old}O_j, \text{old}WTS(O_j))$ in $OLD(T_i)$

If $WTS(O_j)$ equals $TS(T_i)$ then restore $O_j = \text{old}O_j$ and $WTS(O_j) = \text{old}WTS(O_j)$

Informal

Whenever a transaction starts, it is given a timestamp. This is so we can tell which order that the transactions are supposed to be applied in. So given two transactions that affect the same object, the transaction that has the earlier timestamp is meant to be applied before the other one. However, if the wrong transaction is actually presented first, it is aborted and must be restarted.

Every object in the database has a **read timestamp**, which is updated whenever the object's data is read, and a **write timestamp**, which is updated whenever the object's data is changed.

If a transaction wants to read an object,

- but the transaction started *before* the object's **write timestamp** it means that something changed the object's data after the transaction started. In this case, the transaction is canceled and must be restarted.
- and the transaction started *after* the object's **write timestamp**, it means that it is *safe* to read the object. In this case, if the transaction timestamp is after the object's **read timestamp**, the read timestamp is set to the transaction timestamp.

If a transaction wants to write to an object,

- but the transaction started *before* the object's **read timestamp** it means that something has had a look at the object, and we assume it took a copy of the object's data. So we can't write to the object as that would make any copied data invalid, so the transaction is aborted and must be restarted.
- and the transaction started *before* the object's **write timestamp** it means that something has changed the object since we started our transaction. In this case we use the Thomas Write Rule and simply skip our write operation and continue as normal; the transaction does not have to be aborted or restarted
- otherwise, the transaction writes to the object, and the object's **write timestamp** is set to the transaction's timestamp.

Recoverability

For an explanation of the terms **recoverable** (RC), **avoids cascading aborts** (ACA) and **strict** (ST) see Schedule (computer science).

Note that timestamp ordering in its basic form does not produce recoverable histories. Consider for example the following history with transactions T_1 and T_2:

$$W_1(x)\ R_2(x)\ W_2(y)\ C_2\ R_1(z)\ C_1$$

This could be produced by a TO scheduler, but is not recoverable, as T_2 commits even though having read from an uncomitted transaction. To make sure the it produces recoverable histories, a scheduler can keep a list of other transactions each transaction has *read from*, and not let a transaction commit before this list consisted of only committed transactions. To avoid cascading aborts, the scheduler could tag data written by uncommitted transactions as *dirty*, and never let a read operation commence on such a data item before it was untagged. To get a strict history, the scheduler should not allow any operations on dirty items.

Implementation Issues

Timestamp Resolution

This is the minimum time elapsed between two adjacent timestamps. If the resolution of the timestamp is too large (coarse), the possibility of two or more timestamps being equal is increased and thus enabling some transactions to commit out of correct order. For example, assuming that we have a system that can create one hundred unique timestamps per second, and given two events that occur 2 milliseconds apart, they will probably be given the same timestamp even though they actually occurred at different times.

Timestamp Locking

Even though this technique is a non-locking one, in as much as the Object is not locked from concurrent access for the duration of a transaction, the act of recording each timestamp against the Object requires an extremely short duration lock on the Object or its proxy.

Pseudoconversational transaction

In transaction processing, a **pseudoconversational transaction** is a type of transaction that emulates a true conversation in an interactive session. To the end user, it appears as though the program has simply "paused" to request further input, whereas in reality, most resources are released while the input is waiting to be received.

Transparent termination and restart

The controlling program has deliberately saved most of its state during the delay, terminated, and then, on being restarted through new input, restores its previous state. A single control variable is usually retained to hold the current state in terms of the stage of input reached (and therefore what must be recovered at any stage in order to resume processing). The state, including the control variable, is usually preserved in a 'temporary storage record' that maps the variables needing restoration as an aggregate set, usually contained in a single structure (other variables will be re-initialized on restart).

Conserving resources

This method of programming frees up pooled resources (such as memory) for an indeterminate time. This delay is the end-user 'thinking time' (or response time) and depends on human factors including speed of typing. For systems supporting many thousands of users on a single processor, it allows the transparent 'look and feel' of a true conversational session without tying up limited resources.

References

External links

- Pseudoconversational and conversational design (http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=/com.ibm.cics.ts31.doc/dfhp3/dfhp35g.htm) by IBM

Thomas write rule

In computer science, particularly the field of databases, the **Thomas Write rule** is a rule in timestamp-based concurrency control. It can be summarized as *ignore outdated writes*.

It states that, if a more recent transaction has already written the value of an object, then a less recent transaction does not need perform its own write since it will eventually be overwritten by the more recent one.

The Thomas Write rule is applied in situations where a predefined **logical** order is assigned to transactions when they start. For example a transactions might be assigned a monotonically increasing timestamp when it is created. The rule prevents changes in the order in which the transactions are executed from creating different outputs: The outputs will always be consistent with the predefined logical order.

For example consider a database with 3 variables (A, B, C), and two atomic operations C := A (T1), and C := B (T2). Each transaction involves a read (A or B), and a write (C). The only conflict between these transactions is the write on C. The following is one possible schedule for the operations of these transactions:

$$\begin{bmatrix} T_1 & T_2 \\ & Read(A) \\ Read(B) & \\ & Write(C) \\ Write(C) & \\ Commit & \\ & Commit \end{bmatrix} \Longleftrightarrow \begin{bmatrix} T_1 & T_2 \\ & Read(A) \\ Read(B) & \\ & Write(C) \\ & \\ Commit & \\ & Commit \end{bmatrix}$$

If (when the transactions are created) T1 is assigned a timestamp that precedes T2 (i.e., according to the logical order, T1 comes first), then only T2's write should be visible. If, however, T1's write is executed after T2's write, then we need a way to detect this and discard the write.

One practical approach to this is to label each value with a write timestamp (WTS) that indicates the timestamp of the last transaction to modify the value. Enforcing the Thomas Write rule only requires checking to see if the write timestamp of the object is greater than the time stamp of the transaction performing a write. If so, the write is discarded

In the example above, if we call TS(T) the timestamp of transaction T, and WTS(O) the write timestamp of object O, then T2's write sets WTS(C) to TS(T2). When T1 tries to write C, it sees that TS(T1) < WTS(C), and discards the write. If a third transaction T3 (with TS(T3) > TS(T2)) were to then write to C, it would get TS(T3) > WTS(C), and the write would be allowed.

References

Robert H. Thomas (1979). "A majority consensus approach to concurrency control for multiple copy databases". *ACM Transactions on Database Systems* **4** (2): 180–209. doi:10.1145/320071.320076 [1].

References

[1] http://dx.doi.org/10.1145%2F320071.320076

Global concurrency control

Global concurrency control typically pertains to the concurrency control of a system comprising several components, each with its own concurrency control. The overall concurrency control of the whole system, the *Global concurrency control*, is determined by the concurrency control of its components, modules. In this case also the term **Modular concurrency control** is used.

In many cases a system may be distributed over a communication network. In this case we deal with distributed concurrency control of the system, and the two terms sometimes overlap. However, distributed concurrency control typically relates to a case where the distributed system's components do not have each concurrency control of its own, but rather are involved with a concurrency control mechanism that spans several components in order to operate. For example, as typical in a distributed database.

In *database systems* and *transaction processing* (*transaction management*) global concurrency control relates to the concurrency control of a *multidatabase system* (for example, a Federated database; other examples are Grid computing and Cloud computing environments). It deals with the properties of the *global schedule*, which is the unified schedule of the multidatabase system, comprising all the individual schedules of the database systems and possibly other transactional objects in the system. A major goal for global concurrency control is *Global serializability* (or *Modular serializability*). The problem of achieving global serializability in a heterogeneous environment had been open for many years, until an effective solution based on Commitment ordering (CO) has been proposed (see Global serializability). Global concurrency control deals also with relaxed forms of global serializability which compromise global serializability (and in many applications also correctness, and thus are avoided there). While local (to a database system) relaxed serializability methods compromise serializability for performance gain (utilized when the application allows), it is unclear that the various proposed relaxed global serializability methods provide any performance gain over CO, which guarantees global serializability.

Global serializability

In concurrency control of *databases*, *transaction processing* (*transaction management*), and other transactional distributed applications, **Global serializability** (or **Modular serializability**) is a property of a *global schedule* of transactions. A global schedule is the unified schedule of all the individual database (and other transactional object) schedules in a multidatabase environment (e.g., federated database). Complying with global serializability means that the global schedule is *serializable*, has the *serializability* property, while each component database (module) has a serializable schedule as well. In other words, a collection of serializable components provides overall system serializability, which is usually incorrect. A need in correctness across databases in multidatabase systems makes global serializability a major goal for *global concurrency control* (or *modular concurrency control*). With the proliferation of the Internet, Cloud computing, Grid computing, and small, portable, powerful computing devices (e.g., smartphones), as well as increase in systems management sophistication, the need for atomic distributed transactions and thus effective global serializability techniques, to ensure correctness in and among distributed transactional applications, seems to increase.

In a federated database system or any other more loosely defined multidatabase system, which are typically distributed in a communication network, transactions span multiple (and possibly distributed) databases. Enforcing global serializability in such system, where different databases may use different types of concurrency control, is problematic. Even if every local schedule of a single database is serializable, the global schedule of a whole system is not necessarily serializable. The massive communication exchanges of conflict information needed between databases to reach conflict serializability globally would lead to unacceptable performance, primarily due to computer and communication latency. Achieving global serializability effectively over different types of concurrency control has been open for several years. *Commitment ordering* (or Commit ordering; CO), a serializability technique publicly introduced in 1991 by Yoav Raz from Digital Equipment Corporation (DEC), provides an effective general solution for global (conflict) serializability across any collection of database systems and other transactional objects, with possibly different concurrency control mechanisms. CO does not need the distribution of conflict information, but rather utilizes the already needed (unmodified) atomic commitment protocol messages without any further communication between databases. It also allows optimistic (non-blocking) implementations. CO generalizes *Strong strict two phase locking* (SS2PL), which in conjunction with the *Two-phase commit* (2PC) protocol is the de facto standard for achieving global serializability across (SS2PL based) database systems. As a result CO compliant database systems (with any, different concurrency control types) can transparently join existing SS2PL based solutions for global serializability. The same applies also to all other multiple (transactional) object systems that use atomic transactions and need global serializability for correctness (see examples above; nowadays such need is not smaller than with database systems, the origin of atomic transactions).

The most significant aspects of CO that make it a uniquely effective general solution for global serializability are the following:

1. Seamless, low overhead integration with any concurrency control mechanism, with neither changing any transaction's operation scheduling or blocking it, nor adding any new operation.
2. Heterogeneity: Global serializability is achieved across multiple transactional objects (e.g., database management systems) with different (any) concurrency control mechanisms, without interfering with the mechanisms' operations.
3. Modularity: Transactional objects can be added and removed transparently.
4. Autonomy of transactional objects: No need of conflict or equivalent information distribution (e.g., local precedence relations, locks, timestamps, or tickets; no object needs other object's information).
5. Scalability: With "normal" global transactions, computer network size and number of transactional objects can increase unboundedly with no impact on performance, and

6. Automatic global deadlock resolution.

All these aspects, except the first two, are also possessed by the popular SS2PL, which is a (constrained, blocking) special case of CO and inherits many of CO's qualities.

The global serializability problem

Problem statement

The difficulties described above translate into the following problem:

> Find an efficient (high-performance and fault tolerant) method to enforce *Global serializability* (global conflict serializability) in a heterogeneous distributed environment of multiple autonomous database systems. The database systems may employ different concurrency control methods. No limitation should be imposed on the operations of either local transactions (confined to a single database system) or global transactions (span two or more database systems).

Quotations

Lack of an appropriate solution for the global serializability problem has driven researchers to look for alternatives to serializability as a correctness criterion in a multidatabase environment (e.g., see *Relaxing global serializability* below), and the problem has been characterized as difficult and *open*. The following two quotations demonstrate the mindset about it by the end of the year 1991, with similar quotations in numerous other articles:

- "Without knowledge about local as well as global transactions, it is highly unlikely that efficient global concurrency control can be provided... Additional complications occur when different component DBMSs [Database Management Systems] and the FDBMSs [Federated Database Management Systems] support different concurrency mechanisms... It is unlikely that a theoretically elegant solution that provides conflict serializability without sacrificing performance (i.e., concurrency and/or response time) and availability exists."[1]

Commitment ordering, publicly introduced in May 1991 (see below), provides an efficient elegant general solution, from both practical and theoretical points of view, to the global serializability problem across database systems with possibly different concurrency control mechanisms. It provides conflict serializability with no negative effect on availability, and with no worse performance than the de facto standard for global serializability, CO's special case Strong strict two-phase locking (SS2PL). It requires knowledge about neither local nor global transactions.

- "Transaction management in a heterogeneous, distributed database system is a difficult issue. The main problem is that each of the local database management systems may be using a different type of concurrency control scheme. Integrating this is a challenging problem, made worse if we wish to preserve the local autonomy of each of the local databases, and allow local and global transactions to execute in parallel. One simple solution is to restrict global transactions to retrieve-only access. However, the issue of reliable transaction management in the general case, where global and local transactions are allowed to both read and write data, is still open."[2]

The commitment ordering solution comprises effective integration of autonomous database management systems with possibly different concurrency control mechanisms. This while local and global transactions execute in parallel without restricting any read or write operation in either local or global transactions, and without compromising the systems' autonomy.

Even in later years, after the public introduction of the Commitment ordering general solution in 1991, the problem still has been considered by many unsolvable:

- "We present a transaction model for multidatabase systems with autonomous component systems, coined heterogeneous 3-level transactions. It has become evident that in such a system the requirements of guaranteeing full ACID properties and full local autonomy can not be reconciled..."[3]

The quotation above is from a 1997 article proposing a relaxed global serializability solution (see *Relaxing global serializability* below), and referencing Commitment ordering (CO) articles. The CO solution supports effectively both full ACID properties and full local autonomy, as well as meeting the other requirements posed above in the *Problem statement* section, and apparently has been misunderstood.

Similar thinking we see also in the following quotation from a 1998 article:

- "The concept of serializability has been the traditionally accepted correctness criterion in database systems. However in multidatabase systems (MDBSs), ensuring global serializability is a difficult task. The difficulty arises due to the heterogeneity of the concurrency control protocols used by the participating local database management systems (DBMSs), and the desire to preserve the autonomy of the local DBMSs. In general, solutions to the global serializability problem result in executions with a low degree of concurrency. The alternative, relaxed serializability, may result in data inconsistency."[4]

Also the above quoted article proposes a relaxed global serializability solution, while referencing the CO work. The CO solution for global serializability both bridges between different concurrency control protocols with no substantial concurrency reduction (and typically minor, if at all), and maintains the autonomy of local DBMSs. Evidently also here CO has been misunderstood. This misunderstanding continues to 2010 in a textbook by some of the same authors, where the same relaxed global serializability technique, *Two level serializability*, is emphasized and described in detail, and CO is not mentioned at all.[5]

On the other hand, the following quotation on CO appears in a 2009 book:[6]

- "Not all concurrency control algorithms use locks... Three other techniques are timestamp ordering, serialization graph testing, and commit ordering. **Timestamp ordering** assigns each transaction a timestamp and ensures that conflicting operations execute in timestamp order. **Serialization graph testing** tracks conflicts and ensures that the serialization graph is acyclic. **Commit ordering** ensures that conflicting operations are consistent with the relative order in which their transactions commit, which can enable interoperability of systems using different concurrency control mechanisms."

Comments:

1. Beyond the common locking based algorithm SS2PL, which is a CO variant itself, also additional variants of CO that use locks exist, (see below). However, generic, or "pure" CO does not use locks.
2. Since CO mechanisms order the commit events according to conflicts that already have occurred, it is better to describe CO as "**Commit ordering** ensures that the relative order in which transactions commit is consistent with the order of their respective conflicting operations."

The characteristics and properties of the CO solution are discussed below.

Proposed solutions

Several solutions, some partial, have been proposed for the global serializability problem. Among them:

- *Global conflict graph* (serializability graph, precedence graph) *checking*
- *Distributed Two phase locking* (Distributed 2PL)
- *Distributed Timestamp ordering*
- *Tickets* (local logical timestamps which define local total orders, and are propagated to determine global partial order of transactions)
- *Commitment ordering*

Technology perspective

The problem of global serializability has been a quite intensively researched subject in the late 1980s and early 1990s. *Commitment ordering* (CO) has provided an effective general solution to the problem, insight into it, and understanding about possible generalizations of *strong strict two phase locking* (SS2PL), which practically and almost exclusively has been utilized (in conjunction with the *Two-phase commit protocol* (2PC)) since the 1980s to achieve global serializability across databases. An important side-benefit of CO is the automatic *global deadlock* resolution that it provides (this is applicable also to distributed SS2PL; though global deadlocks have been an important research subject for SS2PL, automatic resolution has been overlooked, except in the CO articles, until today (2009)). At that time quite many commercial database system types existed, many non-relational, and databases were relatively very small. Multi database systems were considered a key for database scalability by database systems interoperability, and global serializability was urgently needed. Since then the tremendous progress in computing power, storage, and communication networks, resulted in orders of magnitude increases in both centralized databases' sizes, transaction rates, and remote access to database capabilities, as well as blurring the boundaries between centralized computing and distributed one over fast, low-latency local networks (e.g., Infiniband). These, together with progress in database vendors' distributed solutions (primarily the popular SS2PL with 2PC based, a de facto standard that allows interoperability among different vendors' (SS2PL-based) databases; both SS2PL and 2PC technologies have gained substantial expertise and efficiency), workflow management systems, and database replication technology, in most cases have provided satisfactory and sometimes better information technology solutions without multi database atomic distributed transactions over databases with different concurrency control (bypassing the problem above). As a result, the sense of urgency that existed with the problem at that period, and in general with high-performance distributed atomic transactions over databases with different concurrency control types, has reduced. However, the need in concurrent distributed atomic transactions as a fundamental element of reliability exists in distributed systems also beyond database systems, and so the need in global serializability as a fundamental correctness criterion for such transactional systems (see also Distributed serializability in Serializability). With the proliferation of the Internet, Cloud computing, Grid computing, small, portable, powerful computing devices (e.g., smartphones), and sophisticated systems management the need for effective global serializability techniques to ensure correctness in and among distributed transactional applications seems to increase, and thus also the need in Commitment ordering (including the popular for databases special case SS2PL; SS2PL, though, does not meet the requirements of many other transactional objects).

The commitment ordering solution

Commitment ordering[3][] (or Commit ordering; CO) is the only high-performance, fault tolerant, conflict serializability providing solution that has been proposed as a fully distributed (no central computing component or data-structure are needed), general mechanism that can be combined seamlessly with any local (to a database) concurrency control mechanism (see technical summary). Since the CO property of a schedule is a necessary condition for global serializability of *autonomous databases* (in the context of concurrency control), it provides the only general solution for autonomous databases (i.e., if autonomous databases do not comply with CO, then global serializability may be violated). Seemingly by sheer luck, the CO solution possesses many attractive properties:

1. does not interfere with any transaction's operation, particularly neither block, restrict nor delay any data-access operation (read or write) for either local or global transactions (and thus does not cause any extra aborts); thus allows seamless integration with any concurrency control mechanism.
2. allows optimistic implementations (*non-blocking*, i.e., non data access blocking).
3. allows heterogeneity: Global serializability is achieved across multiple transactional objects with different (any) concurrency control mechanisms, without interfering with the mechanisms' operations.
4. allows modularity: Transactional objects can be added and removed transparently.
5. allows full ACID transaction support.

6. maintains each database's autonomy, and does not need any concurrency control information distribution (e.g., local precedence relations, locks, timestamps, or tickets).
7. does not need any knowledge about the transactions.
8. requires no communication overhead since it only uses already needed, unmodified *atomic commitment* protocol messages (any such protocol; using fault tolerant atomic commitment protocols and database systems makes the CO solution fault tolerant).
9. automatically resolves global deadlocks due to locking.
10. scales up effectively with computer network size and number of databases, almost without any negative impact on performance, since each global transaction is typically confined to certain relatively small numbers of databases and network nodes.
11. requires no additional, artificial transaction access operations (e.g., "take timestamp" or "take ticket"), which typically result in additional, artificial conflicts that reduce concurrency.
12. requires low overhead.

The only overhead incurred by the CO solution is locally detecting conflicts (which is already done by any known serializability mechanism, both pessimistic and optimistic) and locally ordering in each database system both the (local) commits of local transactions and the voting for atomic commitment of global transactions. Such overhead is low. The net effect of CO may be some delays of commit events (but never more delay than SS2PL, and on the average less). This makes CO instrumental for global concurrency control of multidatabase systems (e.g., federated database systems). The underlying *Theory of Commitment ordering*,[] part of Serializability theory, is both sound and elegant (and even "mathematically beautiful"; referring to structure and dynamics of conflicts, graph cycles, and deadlocks), with interesting implications for transactional distributed applications.

All the qualities of CO in the list above, except the first three, are also possessed by SS2PL, which is a special case of CO, but blocking and constraining. This partially explains the popularity of SS2PL as a solution (practically, the only solution, for many years) for achieving global serializability. However, property 9 above, automatic resolution of global deadlocks, has not been noticed for SS2PL in the database research literature until today (2009; except in the CO publications). This, since the phenomenon of voting-deadlocks in such environments and their automatic resolution by the atomic commitment protocol has been overlooked.

Most existing database systems, including all major commercial database systems, are *strong strict two phase locking (SS2PL)* based and already CO compliant. Thus they can participate in a CO based solution for global serializability in multidatabase environments without any modification (except for the popular *multiversioning*, where additional CO aspects should be considered). Achieving global serializability across SS2PL based databases using atomic commitment (primarily using *two phase commit, 2PC*) has been employed for many years (i.e., using the same CO solution for a specific special case; however, no reference is known prior to CO, that notices this special case's automatic global deadlock resulotion by the atomic commitment protocol's augmented-conflict-graph global cycle elimination process). Virtually all existing distributed transaction processing environments and supporting products rely on SS2PL and provide 2PC. As a matter of fact SS2PL together with 2PC have become a de facto standard. This solution is a homogeneous concurrency control one, suboptimal (when both Serializability and Strictness are needed; see Strict commitment ordering; SCO) but still quite effective in most cases, sometimes at the cost of increased computing power needed relatively to the optimum. (However for better performance relaxed serializability is used whenever applications allow). It allows inter-operation among SS2PL-compliant different database system types, i.e., allows heterogeneity in aspects other than concurrency control. SS2PL is a very constraining schedule property, and "takes over" when combined with any other property. For example, when combined with any optimistic property, the result is not optimistic anymore, but rather characteristically SS2PL. On the other hand, CO does not change data-access scheduling patterns at all, and *any* combined property's characteristics remain unchanged. Since also CO uses atomic commitment (e.g., 2PC) for achieving global serializability, as SS2PL does, any CO compliant database system or transactional object can transparently join existing SS2PL based environments, use 2PC, and maintain global serializability without any environment change.

This makes CO a straightforward, natural generalization of SS2PL for any conflict serializability based database system, for all practical purposes.

Commitment ordering has been quite widely known inside the *transaction processing* and *databases* communities at *Digital Equipment Corporation* (DEC) since 1990. It has been under *company confidentiality* due to patenting[] [] processes. CO was disclosed outside of DEC by lectures and technical reports' distribution to database researches in May 1991, immediately after its first patent filing. It has been misunderstood by many database researchers years after its introduction, which is evident by the quotes above from articles in 1997-1998 referencing Commitment ordering articles. On the other hand CO has been utilized extensively as a solution for global serializability in works on Transactional processes, [7] [8] and more recently in the related **Re:GRIDiT**, [9] [10] which is an approach for transaction management in the converging Grid computing and Cloud computing. See more in *The History of Commitment Ordering*.

Relaxing global serializability

Some techniques have been developed for **relaxed global serializability** (i.e., they do not guarantee global serializability; see also *Relaxing serializability*). Among them (with several publications each):

- *Quasi serializability*[11]
- *Two-level serializability*

While local (to a database system) relaxed serializability methods compromise *serializability* for performance gain (and are utilized only when the application can tolerate possible resulting inaccuracies, or its integrity is unharmed), it is unclear that various proposed *relaxed global serializability* methods which compromise *global serializability*, provide any performance gain over *commitment ordering* which guarantees global serializability. Typically, the declared intention of such methods has not been performance gain over effective global serializability methods (which apparently have been unknown to the inventors), but rather correctness criteria alternatives due to lack of a known effective global serializability method. Oddly, some of them were introduced years after CO had been introduced, and some even quote CO without realizing that it provides an effective global serializability solution, and thus without providing any performance comparison with CO to justify them as alternatives to global serializability for some applications (e.g., *Two-level serializability*). *Two-level serializability* is even presented as a major global concurrency control method in a 2010 edition of a text-book on databases (authored by two of the original authors of Two-level serializability, where one of them, Avi Silberschatz, is also an author of the original *Strong recoverability* articles). This book neither mentions CO nor references it, and strangely, apparently does not consider CO a valid *Global serializability* solution.

Another common reason nowadays for Global serializability relaxation is the requirement of availability of internet products and services. This requirement is typically answered by large scale data replication. The straightforward solution for synchronizing replicas' updates of a same database object is including all these updates in a single atomic distributed transaction. However, with many replicas such a transaction is very large, and may span several computers and networks that some of them are likely to be unavailable. Thus such a transaction is likely to end with abort and miss its purpose. Consequently Optimistic replication (Lazy replication) is often utilized (e.g., in many products and services by Google, Amazon, Yahoo, and alike), while Global serializability is relaxed and compromised for Eventual consistency. In this case relaxation is done only for applications that are not expected to be harmed by it.

Classes of schedules defined by *relaxed global serializability* properties either contain the global serializability class, or are incomparable with it. What differentiates techniques for *relaxed global conflict serializability* (RGCSR) properties from those of *relaxed conflict serializability* (RCSR) properties that are not RGCSR is typically the different way *global cycles* (span two or more databases) in the *global conflict graph* are handled. No distinction between global and local cycles exists for RCSR properties that are not RGCSR. RCSR contains RGCSR. Typically RGCSR techniques eliminate local cycles, i.e., provide *local serializability* (which can be achieved effectively by

regular, known concurrency control methods), however, obviously they do not eliminate all global cycles (which would achieve global serializability).

References

[1] Amit Sheth, James Larson (1990): "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases" (http://www.informatik.uni-trier.de/~ley/db/journals/csur/ShethL90.html), *ACM Computing Surveys*, Vol. 22, No 3, pp. 183-236, September 1990 (quotation from page 227)

[2] Abraham Silberschatz, Michael Stonebraker, and Jeffrey Ullman (1991): "Database Systems: Achievements and Opportunities" (http://www.informatik.uni-trier.de/~ley/db/journals/cacm/SilberschatzSU91.html), *Communications of the ACM*, Vol. 34, No. 10, pp. 110-120, October 1991 (quotation from page 120)

[3] Peter Muth (1997): "Application Specific Transaction Management in Multidatabase Systems" (http://portal.acm.org/citation.cfm?id=264226), *Distributed and Parallel Databases*, Volume 5, Issue 4, pp. 357 - 403, October 1997, ISSN: 0926-8782 (quotation from the article's Abstract)

[4] Sharad Mehrotra, Rajeev Rastogi, Henry Korth, Abraham Silberschatz (1998): "Ensuring Consistency in Multidatabases by Preserving Two-Level Serializability" (http://portal.acm.org/citation.cfm?id=277629), *ACM Transactions on Database Systems* (TODS), Vol. 23, No. 2, pp. 199-230, June 1998 (quotation from the article's Abstract)

[5] Avi Silberschatz, Henry F Korth, S. Sudarshan (2010): *Database System Concepts* (http://highered.mcgraw-hill.com/sites/0073523321/), 6th Edition, McGraw-Hill, ISBN 0-07-295886-3

[6] Philip A. Bernstein, Eric Newcomer (2009): *Principles of Transaction Processing*, 2nd Edition (http://www.elsevierdirect.com/product.jsp?isbn=9781558606234), Morgan Kaufmann (Elsevier), June 2009, ISBN 978-1-55860-623-4 (quotation from page 145)

[7] Heiko Schuldt, Hans-Jörg Schek, and Gustavo Alonso (1999): "Transactional Coordination Agents for Composite Systems" (http://portal.acm.org/citation.cfm?id=853907), In *Proceedings of the 3rd International Database Engineering and Applications Symposium* (IDEAS'99), IEEE Computer Society Press, Montrteal, Canada, pp. 321–331.

[8] Klaus Haller, Heiko Schuldt, Can Türker (2005): "Decentralized coordination of transactional processes in peer-to-peer environments", (http://portal.acm.org/citation.cfm?doid=1099554.1099563) *Proceedings of the 2005 ACM CIKM, International Conference on Information and Knowledge Management*, pp. 28-35, Bremen, Germany, October 31 - November 5, 2005, ISBN 1-59593-140-6

[9] Laura Cristiana Voicu, Heiko Schuldt, Fuat Akal, Yuri Breitbart, Hans Jörg Schek (2009): "Re:GRIDiT – Coordinating Distributed Update Transactions on Replicated Data in the Grid" (http://dbis.cs.unibas.ch/publications/2009/grid2009/dbis_publication_view), *10th IEEE/ACM International Conference on Grid Computing (Grid 2009)*, Banff, Canada, 2009/10.

[10] Laura Cristiana Voicu and Heiko Schuldt (2009): "How Replicated Data Management in the Cloud can benefit from a Data Grid Protocol — the Re:GRIDiT Approach" (http://dbis.cs.unibas.ch/publications/2009/clouddb09/dbis_publication_view), *Proceedings of the 1st International Workshop on Cloud Data Management (CloudDB 2009)*, Hong Kong, China, 2009/11.

[11] Weimin Du and Ahmed K. Elmagarmid (1989): "Quasi Serializability: a Correctness Criterion for Global Concurrency Control in InterBase" (http://www.informatik.uni-trier.de/~ley/db/conf/vldb/DuE89.html), *Proceedings of the Fifteenth International Conference on Very Large Data Bases* (VLDB), August 22–25, 1989, Amsterdam, The Netherlands, pp. 347-355, Morgan Kaufmann, ISBN 1-55860-101-5

Modular concurrency control

Global concurrency control typically pertains to the concurrency control of a system comprising several components, each with its own concurrency control. The overall concurrency control of the whole system, the *Global concurrency control*, is determined by the concurrency control of its components, modules. In this case also the term **Modular concurrency control** is used.

In many cases a system may be distributed over a communication network. In this case we deal with distributed concurrency control of the system, and the two terms sometimes overlap. However, distributed concurrency control typically relates to a case where the distributed system's components do not have each concurrency control of its own, but rather are involved with a concurrency control mechanism that spans several components in order to operate. For example, as typical in a distributed database.

In *database systems* and *transaction processing* (*transaction management*) global concurrency control relates to the concurrency control of a *multidatabase system* (for example, a Federated database; other examples are Grid computing and Cloud computing environments). It deals with the properties of the *global schedule*, which is the unified schedule of the multidatabase system, comprising all the individual schedules of the database systems and possibly other transactional objects in the system. A major goal for global concurrency control is *Global serializability* (or *Modular serializability*). The problem of achieving global serializability in a heterogeneous environment had been open for many years, until an effective solution based on Commitment ordering (CO) has been proposed (see Global serializability). Global concurrency control deals also with relaxed forms of global serializability which compromise global serializability (and in many applications also correctness, and thus are avoided there). While local (to a database system) relaxed serializability methods compromise serializability for performance gain (utilized when the application allows), it is unclear that the various proposed relaxed global serializability methods provide any performance gain over CO, which guarantees global serializability.

Multiversion concurrency control

Multiversion concurrency control (**MCC** or **MVCC**), is a concurrency control method commonly used by database management systems to provide concurrent access to the database and in programming languages to implement transactional memory.[1]

If someone is reading from a database at the same time as someone else is writing to it, it is possible that the reader will see a half-written or inconsistent piece of data. There are several ways of solving this problem, known as concurrency control methods. The simplest way is to make all readers wait until the writer is done, which is known as a lock. This can be very slow, so MVCC takes a different approach: each user connected to the database sees a *snapshot* of the database at a particular instant in time. Any changes made by a writer will not be seen by other users of the database until the changes have been completed (or, in database terms: until the transaction has been committed.)

When an MVCC database needs to update an item of data, it will not overwrite the old data with new data, but instead mark the old data as obsolete and add the newer *version* elsewhere. Thus there are multiple versions stored, but only one is the latest. This allows readers to access the data that was there when they began reading, even if it was modified or deleted part way through by someone else. It also allows the database to avoid the overhead of filling in holes in memory or disk structures but requires (generally) the system to periodically sweep through and delete the old, obsolete data objects. For a document-oriented database it also allows the system to optimize documents by writing entire documents onto contiguous sections of disk—when updated, the entire document can be re-written rather than bits and pieces cut out or maintained in a linked, non-contiguous database structure.

MVCC provides *point in time* consistent views. Read transactions under MVCC typically use a timestamp or transaction ID to determine what state of the DB to read, and read these *versions* of the data. This avoids managing locks for read transactions because writes can be isolated by virtue of the old versions being maintained, rather than through a process of locks or mutexes. Writes affect a future *version* but at the transaction ID that the read is working at, everything is guaranteed to be consistent because the writes are occurring at a later transaction ID.

Implementation

MVCC uses timestamps or increasing transaction IDs to achieve transactional consistency. MVCC ensures a transaction never has to wait for a database object by maintaining several versions of an object. Each version would have a write timestamp and it would let a transaction (T_i) read the most recent version of an object which precedes the transaction timestamp ($TS(T_i)$).

If a transaction (T_i) wants to write to an object, and if there is another transaction (T_k), the timestamp of T_i must precede the timestamp of T_k (i.e., $TS(T_i) < TS(T_k)$) for the object write operation to succeed. Which is to say a write cannot complete if there are outstanding transactions with an earlier timestamp.

Every object would also have a read timestamp, and if a transaction T_i wanted to write to object P, and the timestamp of that transaction is earlier than the object's read timestamp ($TS(T_i) < RTS(P)$), the transaction T_i is aborted and restarted. Otherwise, T_i creates a new version of P and sets the read/write timestamps of P to the timestamp of the transaction $TS(T_i)$.

The obvious drawback to this system is the cost of storing multiple versions of objects in the database. On the other hand reads are never blocked, which can be important for workloads mostly involving reading values from the database. MVCC is particularly adept at implementing true snapshot isolation, something which other methods of concurrency control frequently do either incompletely or with high performance costs.

Anomalies

MVCC fails to achieve true snapshot isolation contrary to what the original paper was assuming. Under some circumstances some anomalies can arise called *skew write* and *read-read anomaly*. Those anomalies can be fixed using Serializable Snapshot Isolation and Precisely Serializable Snapshot Isolation at the expense of increasing the number of aborted transactions.

Examples

Concurrent read-write

At Time = 1, the state of a database could be:

Time	Object 1	Object 2
1	"Hello" by T1	
0	"Foo" by T0	"Bar" by T0

T0 wrote Object 1="Foo" and Object 2="Bar". After that T1 wrote Object 1="Hello" leaving Object 2 at its original value. The new value of Object 1 will supersede the value at 0 for all transaction that starts after T1 commits at which point version 0 of Object 1 can be garbage collected.

If a long running transaction T2 starts a read operation of Object 2 and Object 1 after T1 committed and there is a concurrent update transaction T3 which deletes Object 2 and adds Object 3="Foo-Bar", the database state will look like at time 2:

Time	Object 1	Object 2	Object 3
2		(deleted) by T3	"Foo-Bar" by T3
1	"Hello" by T1		
0	"Foo" by T0	"Bar" by T0	

There is a new version as of time 2 of Object 2 which is marked as deleted and a new Object 3. Since T2 and T3 run concurrently T2 sees another the version of the database before 2 i.e. before T3 committed writes, as such T2 reads Object 2="Bar" and Object 1="Hello". This is how MVCC allows snapshot isolation reads in almost every cases without any locks.

History

Multiversion concurrency control is described in some detail in the 1981 paper "Concurrency Control in Distributed Database Systems" by Philip Bernstein and Nathan Goodman, then employed by the Computer Corporation of America. Bernstein and Goodman's paper cites a 1978 dissertation by David P. Reed which quite clearly describes MVCC and claims it as an original work.

The first shipping, commercial database software product featuring MVCC was Digital's VAX Rdb/ELN. The second was InterBase, which is still an active, commercial product.

Databases with MVCC

- Altibase
- ArangoDB[2]
- Berkeley DB[3]
- Bigdata[4]
- Cloudant
- Clustrix[5]
- CouchDB
- IBM DB2 – since IBM DB2 9.7 LUW ("Cobra") under CS isolation level - in *currently committed* mode[6]
- IBM Cognos TM1 – in versions 9.5.2 and up.[7]
- Drizzle
- eXtremeDB
- Firebird[8]
- FLAIM
- GE Smallworld Version Managed Data Store
- H2 Database Engine – experimental since version 1.0.57 (2007-08-25)[9]
- Hawtdb [10]
- HBase (Apache HBase [11])
- HSQLDB – starting with version 2.0
- HypergraphDB [12] is a Typed Hypergraph Database. Hypergraphs are an extension of Graph, ObjectOriented, and list based data structures.
- InfiniDB[13]
- Ingres[14]
- InterBase – all versions
- MariaDB (MySQL fork) when used with XtraDB [15] (developed by Percona Inc.)(XtraDB is a storage engine that is an InnoDB fork and that is included in MariaDB sources and binaries)[16] or PBXT [17] (developed by PrimeBase Technologies).[18][19]
- MarkLogic Server - a bit of this is described in[20]
- MemSQL
- MDB [21]
- Meronymy SPARQL Database Server
- Microsoft SQL Server – when using READ_COMMITTED_SNAPSHOT, starting with SQL Server 2005[22]
- MySQL when used with InnoDB,[23][24] Falcon,[25] or Archive storage engines.
- NuoDB [26] - Elastic Cloud Database [27]
- Netezza
- ObjectStore
- Oracle database – all versions since Oracle 4[28]
- OrientDB[29]
- PostgreSQL[30]
- Rdb/ELN
- RDM Embedded[31]
- REAL Server
- RethinkDB[32]
- SAP HANA
- ScimoreDB
- sones GraphDB
- Sybase SQL Anywhere

- Sybase IQ
- ThinkSQL
- Tibero – all versions since Tibero 3
- Zope Object Database[33]

Other software with MVCC

- JBoss Cache – v 3.0[34]
- Ehcache – v 1.6.0-beta4[35][36]
- Clojure – language software transactional memory
- pojo-mvcc – a lightweight MVCC implementation written in Java[37]
- JVSTM [38] – Software Transactional memory that implements the concept of Versioned Boxes [39] proposed by João Cachopo and António Rito Silva, members of the Software Engineering Group - INESC-ID [40]

Version control systems

Any version control system that has the internal notion of a version (e.g. Subversion, Git, probably almost any current VCS with the notable exception of CVS) will provide explicit MVCC (you only ever access data but by its version identifier).

Among the VCSes that don't provide MVCC at the repository level, most still work with the notion of a *working copy*, which is a file tree checked out from the repository, edited without using the VCS itself and checked in after edition. This working copy provides MVCC while it is checked out.

References

[1] refs (http://clojure.org/refs). Clojure. Retrieved on 2013-09-18.
[2] ArangoDB Manual Pages: AppendOnly/MVCC (http://www.arangodb.org/manuals/1.0.0/DbaManualBasics.html#DbaManualBasicsMvcc)
[3] Berkeley DB Reference Guide: Degrees of Isolation (http://download.oracle.com/docs/cd/E17076_02/html/programmer_reference/transapp_read.html)
[4] Bigdata Blog (http://www.bigdata.com/blog)
[5] A new approach: Clustrix Sierra database engine (http://www.clustrix.com/Default.aspx?app=LeadgenDownload&shortpath=docs/Clustrix_A_New_Approach.pdf)
[6] DB2 Version 9.7 LUW Information Center, Currently committed semantics improve concurrency (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.perf.doc/doc/c0053760.html)
[7] TM1 9.5.2 Information Center, Parallel Interaction (http://publib.boulder.ibm.com/infocenter/ctm1/v9r5m0/topic/com.ibm.swg.im.cognos.tm1_nfg.9.5.2.doc/tm1_nfg_id90tm1_952_nfg_CubeVersioning_N38274.html#tm1_952_nfg_CubeVersioning_N38274)
[8] White paper by Roman Rokytsky Firebird and Multi Version Concurrency Control (http://www.firebirdsql.org/en/multi-version-concurrency-control/)
[9] Multi-Version Concurrency Control in the H2 Database Engine (http://www.h2database.com/html/advanced.html#mvcc)
[10] http://hawtdb.fusesource.org
[11] http://hbase.apache.org/
[12] http://hypergraphdb.org/index
[13] InfiniDB – the high performance, column oriented analytic database (http://infinidb.org/). Infinidb.org. Retrieved on 2013-09-18.
[14] MVCC - Ingres Community Wiki (http://community.ingres.com/wiki/MVCC). Community.ingres.com. Retrieved on 2013-09-18.
[15] https://kb.askmonty.org/en/about-xtradb/
[16] About XtraDB, About XtraDB (https://kb.askmonty.org/en/about-xtradb/)
[17] https://kb.askmonty.org/en/about-pbxt/
[18] MariaDB/Storage Engines, PBXT (http://en.wikibooks.org/wiki/MariaDB/Storage_Engines)
[19] About PBXT, About PBXT (https://kb.askmonty.org/en/about-pbxt/)
[20] Inside MarkLogic Server (http://community.marklogic.com/pubs/architecture/Inside MarkLogic Server.pdf)
[21] http://gitorious.org/mdb
[22] Snapshot Isolation in SQL Server (http://msdn.microsoft.com/en-us/library/tcbchxcb.aspx)
[23] MySQL 5.1 Reference Manual, Section 14.2.12: Implementation of Multi-Versioning (http://dev.mysql.com/doc/refman/5.1/en/innodb-multi-versioning.html)

[24] MySQL 5.1 Reference Manual, Table 14.1. Storage Engine Features (http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html)
[25] or Maria MySQL 5.1 Reference Manual, Section 14.6.1: Falcon Features (Archive) (http://mysql.netvisao.pt/doc/refman/5.1/en/se-falcon-features.html)
[26] http://nuodb.com
[27] http://www.nuodb.com/explore/sql-cloud-database-editions
[28] Oracle Database Concepts: Chapter 13 Data Concurrency and Consistency Multiversion Concurency Control (http://docs.oracle.com/cd/B19306_01/server.102/b14220/consist.htm#i17881)
[29] OrientDb Documentation (http://code.google.com/p/orient/wiki/Transactions)
[30] PostgreSQL Current Documentation, Chapter 13: Concurrency Control (http://postgresql.org/docs/current/static/mvcc.html)
[31] RDM Embedded 10.1 Reference Manual, d_trrobegin (http://docs.raima.com/rdme/10_1/Content/RM/d_trrobegin.htm)
[32] RethinkDB advanced FAQ (http://www.rethinkdb.com/docs/advanced-faq/)
[33] Proposal for MVCC in ZODB (http://wiki.zope.org/ZODB/MultiVersionConcurrencyControl)
[34] MVCC has landed (http://jbosscache.blogspot.com/2008/07/mvcc-has-landed.html)
[35] ehcache site (http://ehcache.sourceforge.net/)
[36] MVCC optimistic locking is not implemented yet (http://jira.terracotta.org/jira/browse/EHC-375)
[37] pojo-mvcc project home (http://code.google.com/p/pojo-mvcc/)
[38] http://www.esw.inesc-id.pt/~jcachopo/jvstm/
[39] http://urresearch.rochester.edu/handle/1802/2101
[40] http://www.esw.inesc-id.pt/wikiesw

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8

Optimistic concurrency control

Optimistic concurrency control (**OCC**) is a concurrency control method applied to transactional systems such as relational database management systems and software transactional memory. OCC assumes that multiple transactions can frequently complete without interfering with each other. While running, transactions use data resources without acquiring locks on those resources. Before committing, each transaction verifies that no other transaction has modified the data it has read. If the check reveals conflicting modifications, the committing transaction rolls back and can be restarted. Optimistic concurrency control was first proposed by H.T. Kung.

OCC is generally used in environments with low data contention. When conflicts are rare, transactions can complete without the expense of managing locks and without having transactions wait for other transactions' locks to clear, leading to higher throughput than other concurrency control methods. However, if contention for data resources is frequent, the cost of repeatedly restarting transactions hurts performance significantly; it is commonly thought that other concurrency control methods have better performance under these conditions. However, locking-based ("pessimistic") methods also can deliver poor performance because locking can drastically limit effective concurrency even when deadlocks are avoided.

OCC phases

More specifically, OCC transactions involve these phases:

- **Begin**: Record a timestamp marking the transaction's beginning.
- **Modify**: Read database values, and tentatively write changes.
- **Validate**: Check whether other transactions have modified data that this transaction has used (read or written). This includes transactions that completed after this transaction's start time, and optionally, transactions that are still active at validation time.
- **Commit/Rollback**: If there is no conflict, make all changes take effect. If there is a conflict, resolve it, typically by aborting the transaction, although other resolution schemes are possible. Care must be taken to avoid a

TOCTTOU bug, particularly if this phase and the previous one are not performed as a single atomic operation.

Web usage

The stateless nature of HTTP makes locking infeasible for web user interfaces. It's common for a user to start editing a record, then leave without following a "cancel" or "logout" link. If locking is used, other users who attempt to edit the same record must wait until the first user's lock times out.

HTTP does provide a form of built-in OCC: The GET method returns an ETag for a resource and subsequent PUTs use the ETag value in the If-Match headers; while the first PUT will succeed, the second will not, as the value in If-Match is based on the first version of the resource.

Some database management systems offer OCC natively - without requiring special application code. For others, the application can implement an OCC layer outside of the database, and avoid waiting or silently overwriting records. In such cases, the form includes a hidden field with the record's original content, a timestamp, a sequence number, or an opaque token. On submit, this is compared against the database. If it differs, the conflict resolution algorithm is invoked.

Examples

- MediaWiki's edit pages use OCC.[1]
- Bugzilla uses OCC; edit conflicts are called "mid-air collisions".
- The Ruby on Rails framework has an API for OCC.
- The Grails framework uses OCC in its default conventions.
- Microsoft's Entity Framework (including Code-First) has built-in support for OCC based on a binary timestamp value.[2]
- Mimer SQL is a DBMS that only implements optimistic concurrency control.
- Google App Engine data store uses OCC.
- The ElasticSearch search engine supports OCC via the version attribute.
- The MonetDB column-oriented database management system's transaction management scheme is based on OCC.
- Most implementations of software transactional memory use optimistic locking.

References

[1] Help:Edit conflict

- Most revision control systems support the "merge" model for concurrency, which is OCC.

External links

- Kung, H. T.; John T. Robinson (June 1981). "On optimistic methods for concurrency control". *ACM Transactions on Database Systems* **6** (2): 213–226. doi: 10.1145/319566.319567 (http://dx.doi.org/10.1145/319566.319567).
- Enterprise JavaBeans, 3.0, By Bill Burke, Richard Monson-Haefel, Chapter 16. Transactions, Section 16.3.5. Optimistic Locking, Publisher: O'Reilly, Pub Date: May 16, 2006,Print ISBN 0-596-00978-X,
- Hollmann, Andreas (May 2009). "Multi-Isolation: Virtues and Limitations" (http://www.andrej-hollmann.de/images/stories/informatik/multi-isolation-part-1.pdf) (PDF). *Multi-Isolation (what is between pessimistic and optimistic locking)*. 01069 Gutzkovstr. 30/F301.2, Dresden: Happy-Guys Software GbR. p. 8. Retrieved 2013-05-16.

Autocommit

In the context of data management, **autocommit** is a mode of operation of a database connection. Each individual database interaction (i.e., each SQL statement) submitted through the database connection in autocommit mode will be executed in its own transaction that is implicitly committed. A SQL statement executed in autocommit mode cannot be rolled back.

Autocommit mode, in theory, incurs per-statement transaction overhead, having often undesirable performance or resource utilization impact. Nonetheless, in systems such as Microsoft SQL Server, as well as connection technologies such as ODBC and Microsoft OLE DB, autocommit mode is the default for all statements that change data, in order to ensure that individual statements will conform to the ACID (atomicity-consistency-isolation-durability) properties of transactions.[1]

The alternative to autocommit mode (non-autocommit) means that the SQL client application itself is responsible for issuing transaction initiation (*start transaction*) and termination (*commit* or *rollback*) commands. Non-autocommit mode enables grouping of multiple data manipulation SQL commands into a single atomic transaction.

References

[1] Autocommit transactions. http://technet.microsoft.com/en-us/library/aa213069(v=sql.80).aspx

Transaction log

In the field of databases in computer science, a **transaction log** (also **transaction journal**, **database log**, **binary log** or **audit trail**) is a history of actions executed by a database management system to guarantee ACID properties over crashes or hardware failures. Physically, a log is a file of updates done to the database, stored in stable storage.

If, after a start, the database is found in an inconsistent state or not been shut down properly, the database management system reviews the database logs for uncommitted transactions and rolls back the changes made by these transactions. Additionally, all transactions that are already committed but whose changes were not yet materialized in the database are re-applied. Both are done to ensure atomicity and durability of transactions.

This term is not to be confused with other, human-readable logs that a database management system usually provides.

Anatomy of a general database log

A database log record is made up of:

- *Log Sequence Number*: A unique id for a log record. With LSNs, logs can be recovered in constant time. Most logs' LSNs are assigned in monotonically increasing order, which is useful in recovery algorithms, like ARIES.
- *Prev LSN*: A link to the last log record. This implies database logs are constructed in linked list form.
- *Transaction ID number*: A reference to the database transaction generating the log record.
- *Type*: Describes the type of database log record.
- Information about the actual changes that triggered the log record to be written.

Types of database log records

All log records include the general log attributes above, and also other attributes depending on their type (which is recorded in the *Type* attribute, as above).

- **Update Log Record** notes an update (change) to the database. It includes this extra information:
  - *PageID*: A reference to the Page ID of the modified page.
  - *Length and Offset*: Length in bytes and offset of the page are usually included.
  - *Before and After Images*: Includes the value of the bytes of page before and after the page change. Some databases may have logs which include one or both images.
- **Compensation Log Record** notes the rollback of a particular change to the database. Each correspond with exactly one other Update Log Record (although the corresponding update log record is not typically stored in the Compensation Log Record). It includes this extra information:
  - *undoNextLSN*: This field contains the LSN of the next log record that is to be undone for transaction that wrote the last Update Log.
- **Commit Record** notes a decision to commit a transaction.
- **Abort Record** notes a decision to abort and hence roll back a transaction.
- **Checkpoint Record** notes that a checkpoint has been made. These are used to speed up recovery. They record information that eliminates the need to read a long way into the log's past. This varies according to checkpoint algorithm. If all dirty pages are flushed while creating the checkpoint (as in PostgreSQL), it might contain:
  - *redoLSN*: This is a reference to the first log record that corresponds to a dirty page. i.e. the first update that wasn't flushed at checkpoint time. This is where redo must begin on recovery.
  - *undoLSN*: This is a reference to the oldest log record of the oldest in-progress transaction. This is the oldest log record needed to undo all in-progress transactions.
- **Completion Record** notes that all work has been done for this particular transaction. (It has been fully committed or aborted)

Tables

These tables are maintained in memory, and can be efficiently reconstructed (if not exactly, to an equivalent state) from the log and the database:

- **Transaction Table**: The table contains one entry for each active transaction. This includes Transaction ID and lastLSN, where lastLSN describes the LSN of the most recent log record for the transaction.
- **Dirty Page Table**: The table contains one entry for each dirty page that hasn't been written to disk. The entry contains recLSN, where recLSN is the LSN of the first log record that caused the page to be dirty.
- **Transaction Log**: A DBMS uses a transaction log to keep track of all transactions that updates the database. The information stored in this log is used by DBMS for a recovery requirement triggered by 'Roll Back' statement.

Savepoint

A **savepoint** is a way of implementing subtransactions (also known as nested transactions) within a relational database management system by indicating a point within a transaction that can be "rolled back to" without affecting any work done in the transaction before the savepoint was created. Multiple savepoints can exist within a single transaction. Savepoints are useful for implementing complex error recovery in database applications — if an error occurs in the midst of a multiple-statement transaction, the application may be able to recover from the error (by rolling back to a savepoint) without needing to abort the entire transaction.

A savepoint can be declared by issuing a `SAVEPOINT` *`name`* statement. All changes made after a savepoint has been declared can be undone by issuing a `ROLLBACK TO SAVEPOINT` *`name`* command. Issuing `RELEASE SAVEPOINT` *`name`* will cause the named savepoint to be discarded, but will not otherwise affect anything. Issuing the commands `ROLLBACK` or `COMMIT` will also discard any savepoints created since the start of the main transaction.[1]

Savepoints are supported in some form or other in database systems like PostgreSQL, Oracle, Microsoft SQL Server, MySQL, DB2, SQLite (since 3.6.8), Firebird and Informix (since version 11.50xC3). Savepoints are also defined in the SQL standard.

[1] http://docs.oracle.com/cd/B19306_01/appdev.102/b14261/savepoint_statement.htm

No-force

A **no-force** policy is used in transaction control in database theory. The term no-force refers to the disk pages related to the actual database object being modified.

With a no-force policy, when a transaction commits, the changes made to the actual objects are not required to be written to disk in-place (*forced*).

A record of the changes must still be preserved at commit time to ensure that the transaction is durable. This record is typically written to a sequential transaction log, with the actual changes to the database objects being changes which can be written at a later time.

For frequently changed objects, a no-force policy allows updates to be merged and so reduce the number of write operations to the actual database object. A no-force policy also reduces the seek time required for a commit by having mostly sequential write operations to the transaction log, rather than requiring the disk to seek to many distinct database objects during a commit.

Non-blocking algorithm

In computer science, a **non-blocking algorithm** ensures that threads competing for a shared resource do not have their execution indefinitely postponed by mutual exclusion. A non-blocking algorithm is **lock-free** if there is guaranteed system-wide progress; **wait-free** if there is also guaranteed per-thread progress.

Literature up to the turn of the 21st century used "non-blocking" synonymously with lock-free. However, since 2003, the term has been weakened to only prevent progress-blocking interactions with a preemptive scheduler. In modern usage, therefore, an algorithm is *non-blocking* if the suspension of one or more threads will not stop the potential progress of the remaining threads. They are designed to avoid requiring a critical section. Often, these algorithms allow multiple processes to make progress on a problem without ever blocking each other. For some operations, these algorithms provide an alternative to locking mechanisms.

Motivation

The traditional approach to multi-threaded programming is to use locks to synchronize access to shared resources. Synchronization primitives such as mutexes, semaphores, and critical sections are all mechanisms by which a programmer can ensure that certain sections of code do not execute concurrently, if doing so would corrupt shared memory structures. If one thread attempts to acquire a lock that is already held by another thread, the thread will block until the lock is free.

Blocking a thread is undesirable for many reasons. An obvious reason is that while the thread is blocked, it cannot accomplish anything. If the blocked thread was performing a high-priority or real-time task, it would be highly undesirable to halt its progress. Other problems are less obvious. Certain interactions between locks can lead to error conditions such as deadlock, livelock, and priority inversion. Using locks also involves a trade-off between coarse-grained locking, which can significantly reduce opportunities for parallelism, and fine-grained locking, which requires more careful design, increases locking overhead and is more prone to bugs.

Non-blocking algorithms are also safe for use in interrupt handlers: even though the preempted thread cannot be resumed, progress is still possible without it. In contrast, global data structures protected by mutual exclusion cannot safely be accessed in a handler, as the preempted thread may be the one holding the lock.

Implementation

With few exceptions, non-blocking algorithms use atomic read-modify-write primitives that the hardware must provide, the most notable of which is compare and swap (CAS). Critical sections are almost always implemented using standard interfaces over these primitives. Until recently, all non-blocking algorithms had to be written "natively" with the underlying primitives to achieve acceptable performance. However, the emerging field of software transactional memory promises standard abstractions for writing efficient non-blocking code.

Much research has also been done in providing basic data structures such as stacks, queues, sets, and hash tables. These allow programs to easily exchange data between threads asynchronously.

Additionally, some data structures are weak enough to be implemented without special atomic primitives. These exceptions include:

- single-reader single-writer ring buffer FIFO
- Read-copy-update with a single writer and any number of readers. (The readers are wait-free; the writer is usually lock-free, until it needs to reclaim memory).
- Read-copy-update with multiple writers and any number of readers. (The readers are wait-free; multiple writers generally serialize with a lock and are not obstruction-free).

Wait-freedom

Wait-freedom is the strongest non-blocking guarantee of progress, combining guaranteed system-wide throughput with starvation-freedom. An algorithm is wait-free if every operation has a bound on the number of steps the algorithm will take before the operation completes. This property is critical for real-time systems and is always nice to have as long as the performance cost is not too high.

It was shown in the 1980s that all algorithms can be implemented wait-free, and many transformations from serial code, called *universal constructions*, have been demonstrated. However, the resulting performance does not in general match even naïve blocking designs. Several papers have since improved the performance of universal constructions, but still, their performance is far below blocking designs.

Several papers have investigated the difficulty of creating wait-free algorithms. For example, it has been shown that the widely available atomic *conditional* primitives, CAS and LL/SC, cannot provide starvation-free implementations of many common data structures without memory costs growing linearly in the number of threads. But in practice these lower bounds do not present a real barrier as spending a word per thread in the shared memory is not considered too costly for practical systems.

Until 2011, wait-free algorithms were rare, both in research and in practice. However, in 2011 Kogan and Petrank presented a wait-free queue building on the CAS primitive, generally available on common hardware. Their construction expands the lock-free queue of Michael and Scott, which is an efficient queue often used in practice. A follow-up paper by Kogan and Petrank provided a methodology for making wait-free algorithms fast and used this methodology to make the wait-free queue practically as fast as its lock-free counterpart.

Lock-freedom

Lock-freedom allows individual threads to starve but guarantees system-wide throughput. An algorithm is lock-free if it satisfies that when the program threads are run sufficiently long at least one of the threads makes progress (for some sensible definition of progress). All wait-free algorithms are lock-free.

In general, a lock-free algorithm can run in four phases: completing one's own operation, assisting an obstructing operation, aborting an obstructing operation, and waiting. Completing one's own operation is complicated by the possibility of concurrent assistance and abortion, but is invariably the fastest path to completion.

The decision about when to assist, abort or wait when an obstruction is met is the responsibility of a *contention manager*. This may be very simple (assist higher priority operations, abort lower priority ones), or may be more optimized to achieve better throughput, or lower the latency of prioritized operations.

Correct concurrent assistance is typically the most complex part of a lock-free algorithm, and often very costly to execute: not only does the assisting thread slow down, but thanks to the mechanics of shared memory, the thread being assisted will be slowed, too, if it is still running.

Obstruction-freedom

Obstruction-freedom is possibly the weakest natural non-blocking progress guarantee. An algorithm is obstruction-free if at any point, a single thread executed in isolation (i.e., with all obstructing threads suspended) for a bounded number of steps will complete its operation. All lock-free algorithms are obstruction-free.

Obstruction-freedom demands only that any partially completed operation can be aborted and the changes made rolled back. Dropping concurrent assistance can often result in much simpler algorithms that are easier to validate. Preventing the system from continually live-locking is the task of a contention manager.

Obstruction-freedom is also called optimistic concurrency control.

Some obstruction-free algorithms use a pair of "consistency markers" in the data structure. Processes reading the data structure first read one consistency marker, then read the relevant data into an internal buffer, then read the other

marker, and then compare the markers. The data is consistent if the two markers are identical. Markers may be non-identical when the read is interrupted by another process updating the data structure. In such a case, the process discards the data in the internal buffer and tries again.

References

Data recovery

Data Recovery is the process of salvaging data from damaged, failed, corrupted, or inaccessible secondary storage media when it cannot be accessed normally. Often the data are being salvaged from storage media such as internal or external hard disk drives, solid-state drives (SSD), USB flash drive, storage tapes, CDs, DVDs, RAID, and other electronics. Recovery may be required due to physical damage to the storage device or logical damage to the file system that prevents it from being mounted by the host operating system (OS).

The most common "Data Recovery" scenario involves an operating system failure (typically on a single-disk, single-partition, single-OS system), in which case the goal is simply to copy all wanted files to another disk. This can be easily accomplished using a Live CD, many of which provide a means to mount the system drive and backup disks or removable media, and to move the files from the system disk to the backup media with a file manager or optical disc authoring software. Such cases can often be mitigated by disk partitioning and consistently storing valuable data files (or copies of them) on a different partition from the replaceable OS system files.

Another scenario involves a disk-level failure, such as a compromised file system or disk partition, or a hard disk failure. In any of these cases, the data cannot be easily read. Depending on the situation, solutions involve repairing the file system, partition table or master boot record, or hard disk recovery techniques ranging from software-based recovery of corrupted data, hardware-software based recovery of damaged service areas (also known as the hard drive's "firmware"), to hardware replacement on a physically damaged disk. If hard disk recovery is necessary, the disk itself has typically failed permanently, and the focus is rather on a one-time recovery, salvaging whatever data can be read.

In a third scenario, files have been "deleted" from a storage medium. Typically, the contents of deleted files are not removed immediately from the drive; instead, references to them in the directory structure are removed, and the space they occupy is made available for later overwriting. In the meantime, the original file contents remain, often in a number of disconnected fragments, and may be recoverable.

The term "data recovery" is also used in the context of forensic applications or espionage, where data which has been encrypted or hidden, rather than damaged, is recovered.

Physical damage

A wide variety of failures can cause physical damage to storage media. CD-ROMs can have their metallic substrate or dye layer scratched off; hard disks can suffer any of several mechanical failures, such as head crashes and failed motors; tapes can simply break. Physical damage always causes at least some data loss, and in many cases the logical structures of the file system are damaged as well. Any logical damage must be dealt with before files can be salvaged from the failed media.

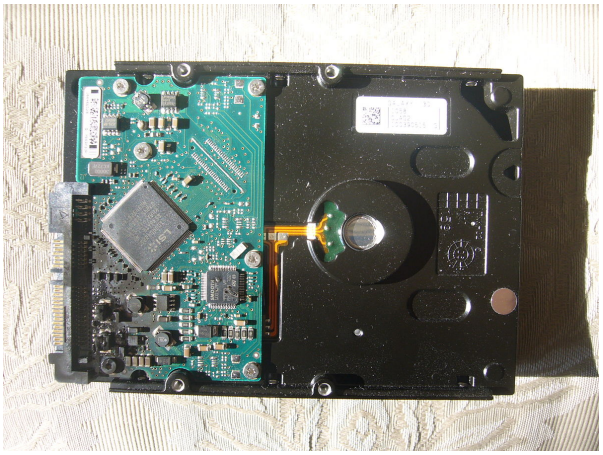
Most physical damage cannot be repaired by end users. For example, opening a hard disk drive in a normal environment can allow airborne dust to settle on the platter and become caught between the platter and the read/write head, causing new head crashes that further damage the platter and thus compromise the recovery process. Furthermore, end users generally do not have the hardware or technical expertise required to make these repairs. Consequently, data recovery companies are often employed to salvage important data with the more reputable ones

using class 100, dust- & static-free cleanrooms.

Recovery techniques

Recovering data from physically damaged hardware can involve multiple techniques. Some damage can be repaired by replacing parts in the hard disk. This alone may make the disk usable, but there may still be logical damage. A specialized disk-imaging procedure is used to recover every readable bit from the surface. Once this image is acquired and saved on a reliable medium, the image can be safely analyzed for logical damage and will possibly allow much of the original file system to be reconstructed.

Hardware repair



Media that has suffered a catastrophic electronic failure requires data recovery in order to salvage its contents.

A common misconception is that a damaged printed circuit board (PCB) may be replaced during recovery procedures by an identical PCB from a healthy drive. While this may work in rare circumstances on hard drives manufactured before 2003, it will not work on newer hard drives. Each hard drive has what is called a System Area. This portion of the drive, which is not accessible to the end user, contains adaptive data that helps the drive operate within normal parameters. One function of the System Area is to log defective sectors within the drive; essentially telling the hard drive where it can and cannot write data. The sector lists are also stored on various chips attached to the PCB, and they are unique to each hard drive. If the data on the PCB does not match what is stored on the platter, then the drive will not calibrate properly.[1] In most cases the hard drive heads will click, because they are unable to find the data matching what is stored on the PCB.

Logical damage

The term "logical damage" refers to situations in which the error is not a problem in the hardware and requires software-level solutions.

Corrupt partitions and filesystems, media errors

In some cases, data on a hard drive can be unreadable due to damage to the partition table or filesystem, or to (intermittent) media errors. In the majority of these cases, at least a portion of the original data can be recovered by repairing the damaged partition table or filesystem using specialized data recovery software such as Testdisk; software like dd rescue can image media despite intermittent errors, and image raw data when there is partition table or filesystem damage. This type of data recovery can be performed by people without expertise in drive hardware, as it requires no special physical equipment or access to platters. Sometimes data can be recovered using relatively simple methods and tools; more serious cases can require expert intervention, particularly if parts of files are irrecoverable. Data carving is the recovery of parts of damaged files using knowledge of their structure.

Result of a failed data recovery from a hard disk drive.

Overwritten data

When data has been physically overwritten on a hard disk drive it is generally assumed that the previous data is no longer possible to recover. In 1996, Peter Gutmann, a computer scientist, presented a paper that suggested overwritten data could be recovered through the use of magnetic force microscope.[2] In 2001, he presented another paper on a similar topic.[3] Substantial criticism has followed, primarily dealing with the lack of any concrete examples of significant amounts of overwritten data being recovered. Although Gutmann's theory may be correct, there is no practical evidence that overwritten data can be recovered, while research has shown to support that overwritten data cannot be recovered.Wikipedia:Citing sources To guard against this type of data recovery, Gutmann and Colin Plumb designed a method of irreversibly scrubbing data, known as the Gutmann method and used by several disk-scrubbing software packages.

Solid-state drives (SSD) overwrite data differently from hard disk drives (HDD) which makes at least some of their data easier to recover. Most SSDs use flash memory to store data in pages and blocks, referenced by logical block addresses (LBA) which are managed by the flash translation layer (FTL). When the FTL modifies a sector it writes the new data to another location and updates the map so the new data appears at the target LBA. This leaves the pre-modification data in place, with possibly many generations, and recoverable by data recovery software.

Remote data recovery

It is not always necessary for experts to have physical access to the damaged drive; where data can be recovered by software techniques, they can often be used remotely, with an expert using a computer at another location linked by an Internet or other connection to equipment at the fault site.

Remote recovery requires a stable connection of adequate bandwidth. However, it is not applicable where access to the hardware is required, as for cases of physical damage.

Four phases

It is important to understand the four phases of data recovery. Each phase stands for different level and range of data recovery capabilities, each phase requires different hdd repair tools and data recovery tools to work with and each phase must be treated properly to make sure the maximum data is finally to be recovered.

- Phase 1: Repair the hard drive
- Phase 2: Image the drive to a new drive.
- Phase 3: Logical recovery of files, partition, MBR, and MFT.
- Phase 4: Repair the damaged files that were retrieved.

References

[1] Swapping PCB's on Data Recovery Report (http://datarecoveryreport.com/#Swapping_PCB_Logic_Board)

[2] *Secure Deletion of Data from Magnetic and Solid-State Memory* (http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html), Peter Gutmann, Department of Computer Science, University of Auckland

[3] *Data Remanence in Semiconductor Devices* (http://www.cypherpunks.to/~peter/usenix01.pdf), Peter Gutmann, IBM T.J. Watson Research Center

Further reading

- Tanenbaum, A. & Woodhull, A. S. (1997). *Operating Systems: Design And Implementation,* 2nd ed. New York: Prentice Hall.

www.aurora.se (http:/ / www. aurora. se) www.dataraddning.aurora.se (http:/ / www. dataraddning. aurora. se) www.harddiskkrasch.aurora.se (http://www. harddiskkrasch. aurora. se) www.radda-data.aurora.se (http://www. radda-data. aurora. se) www.radda-harddisk.aurora.se (http:/ / www. radda-harddisk. aurora. se) www.harddiskraddning.aurora.se (http:/ / www. harddiskraddning. aurora. se) www.reparera-trasig-harddisk.aurora.se (http://www.reparera-trasig-harddisk.aurora.se)

- Data recovery (http://www.dmoz.org/Computers/Hardware/Storage/Data_Recovery/) on the Open Directory Project

Point-in-time recovery

Point-in-time recovery (PITR) in the context of computers is a system whereby a set of data or a particular setting can be restored or recovered from a time in the past. An example of this is Windows XP's feature of being able to restore operating system settings from a past date (before data corruption occurred, for example), or PostgreSQL's feature of being able to view a database table and its data as it was at a particular date in the past. Also, Time Machine for Mac OS X is an example of Point-in-time recovery.

A database with the PITR feature can be restored or recovered to the state that it had at any time since PITR logging was started for that database.

The PostgreSQL database implements Continuous Archiving and Point-In-Time Recovery [1]. In PostgreSQL, Write-ahead logging (WAL) must be enabled for a particular database, in order for PITR to be used on that database; any time after WAL is enabled for a database, that database may be restored to any later time.

External links

- PostgreSQL Continuous Archiving and Point-In-Time Recovery (PITR) blog/article [2]

References

[1] http://www.postgresql.org/docs/8.2/static/continuous-archiving.html
[2] http://blog.ganneff.de/blog/2008/02/15/postgresql-continuous-archivin.html

Redo log

In the Oracle RDBMS environment, **redo logs** comprise files in a proprietary format which log a history of all changes made to the database. Each redo log file consists of redo records. A redo record, also called a redo entry, holds a group of Change vectors, each of which describes or represents a change made to a single block in the database.

For example, if a user `UPDATE`s a salary-value in a table containing employee-related data, the DBMS generates a redo record containing change-vectors that describe changes to the data segment block for the table. And if the user then `COMMIT`s the update, Oracle generates another redo record and assigns the change a "system change number" (SCN).

Whenever something changes in a datafile, Oracle records the change in the redo log. The name *redo log* indicates its purpose: When the database crashes, the RDBMS can redo (re-process) all changes on datafiles which will take the database data back to the state it was when the last redo record was written. DBAs use the views `V$LOG`, `V$LOGFILE`, `V$LOG_HISTORY` and `V$THREAD` to find information about the redo log of the database. Each redo log file belongs to exactly one group (of which at least two must exist). Exactly one of these groups is the CURRENT group (can be queried using the column status of v$log). Oracle uses that current group to write the redo log entries. When the group is full, a **log switch** occurs, making another group the current one. Each log switch causes checkpoint, however, the converse is not true: a checkpoint does not cause a redo log switch. One can also manually cause a redo-log switch using the `ALTER SYSTEM SWITCH LOGFILE` command.

Usage

Before a user receives a "Commit complete" message, the system must first successfully write the new or changed data to a redo log file..

All changes included in the transaction are first written into the log buffer. Using memory in this way for the initial capture aims to reduce disk IO. Of course, when a transaction commits, the redo log buffer must be flushed to disk, because otherwise the recovery for that commit could not be guaranteed. It is LGWR (Log Writer) that does that flushing.

If a database crashes, the recovery process has to apply all transactions, both uncommitted as well as committed, to the data-files on disk, using the information in the redo log files. Oracle must re-do all redo-log transactions that have both a begin and a commit entry, and it must undo all transactions that have a begin entry but no commit entry. (Re-doing a transaction in this context simply means applying the information in the redo log files to the database; the system does not re-run the transaction itself.) The system thus re-creates committed transactions by applying the “after image” records in the redo log files to the database, and undoes incomplete transactions by using the “before image” records in the undo tablespace.

Change data capture can read the redo logs.

Implications

Given the verbosity of the logging, Oracle Corporation provides methods for archiving redo logs (archive-logs), and this in turn can feed into data backup solutions and standby databases.

The existence of a detailed series of individually logged transactions and actions provides the basis of several data-management enhancements such as Oracle Flashback, log-mining and point-in-time recovery.

For database tuning purposes, efficiently coping with redo logs requires plentiful and fast-access disk.

References

- Managing the Redo Log [1]

References

[1] http://download.oracle.com/docs/cd/B19306_01/server.102/b14231/onlineredo.htm

Extreme transaction processing

In Computer Science, **Extreme Transaction Processing** (XTP) is an exceptionally demanding form of transaction processing. Transactions of most high-end (more than 10,000 concurrent accesses or 500 transaction per second) or ultra-high-end (more than 100,000 concurrent accesses or 5,000 transaction per second) requirements or more would require this form of processing.[1]

Definition

Gartner defines XTP as an application style aimed at supporting the design, development, deployment, management and maintenance of distributed TP applications characterized by exceptionally demanding performance, scalability, availability, security, manageability and dependability requirements.[2][3][4]

XTP applications focuses on delivering high and consistent performance in a linearly scalable and highly available system.[5]

Major Solutions

Major solutions promising XTP are :-

1. Extreme Scale
2. Oracle Coherence
3. Red Hat's JBoss Data Grid [6], based on the open source Infinispan [7] project
4. GigaSpaces [8]
5. GemFire [9]

References

[1] Extreme Transaction Processing: High-Impact Emerging Technology - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors - Kevin Roebuck (http://books.google.com/books/about/Extreme_Transaction_Processing.html?id=m-rOjwEACAAJ&redir_esc=y)

[2] IBM Extreme Transaction Processing (XTP) Patterns (http://www.ibm.com/developerworks/websphere/library/techarticles/0906_vuong/0906_vuong.html)

[3] SOA Magazine article on XTP (http://soa.sys-con.com/node/506149)

[4] Extreme Transaction Processing Patterns: Write-behind Caching (http://www.infoq.com/articles/write-behind-caching)

[5] DEVOXX 2011 - Going extreme on Health Care (http://www.parleys.com/#st=5&id=2794&sl=31)

[6] http://www.redhat.com/products/jbossenterprisemiddleware/data-grid

[7] http://www.infinispan.org

[8] Gigaspace official Website - XAP Product Overview (http://wiki.gigaspaces.com/wiki/display/XAP9/Product+Architecture)

[9] GemFire Product Overview (http://www.vmware.com/products/application-platform/vfabric-gemfire/overview.html)

In-database processing

In-database processing, sometimes referred to as in-database analytics, refers to the integration of data analytics into data warehousing functionality. Today, many large databases, such as those used for credit card fraud detection and investment bank risk management, use this technology because it provides significant performance improvements over traditional methods.

History

Traditional approaches to data analysis require data to be moved out of the database into a separate analytics environment for processing, and then back to the database. (SPSS from IBM are examples of tools that still do this today). Doing the analysis in the database, where the data resides, eliminates the costs, time and security issues associated with the old approach by doing the processing in the data warehouse itself.

Though in-database capabilities were first commercially offered in the mid-1990s, as object-related database systems from vendors including IBM, Illustra/Informix (now IBM) and Oracle, the technology did not begin to catch on until the mid-2000s.

At that point, the need for in-database processing had become more pressing as the amount of data available to collect and analyze continues to grow exponentially (due largely to the rise of the Internet), from megabytes to gigabytes, terabytes and petabytes. This “big data” is one of the primary reasons it has become important to collect, process and analyze data efficiently and accurately.

Also, the speed of business has accelerated to the point where a performance gain of nanoseconds can make a difference in some industries. Additionally, as more people and industries use data to answer important questions, the questions they ask become more complex, demanding more sophisticated tools and more precise results.

All of these factors in combination have created the need for in-database processing. The introduction of the column-oriented database, specifically designed for analytics, data warehousing and reporting, has helped make the technology possible.

Types

There are three main types of in-database processing: translating a model into SQL code, loading C or C++ libraries into the database process space as a built-in user-defined function (UDF), and out-of-process libraries typically written in C, C++ or JAVA and registering them in the database as a built-in UDFs in a SQL statement.

Translating Models into SQL Code

In this type of in-database processing, a predictive model is converted from its source language into SQL that can run in the database usually in a stored procedure. Many analytic model-building tools have the ability to export their models in either in SQL or PMML (Predictive Modeling Markup Language). Once the SQL is loaded into a stored procedure, values can be passed in through parameters and the model is executed natively in the database. Tools that can use this approach include SAS, R and KXEN.

Loading C or C++ Libraries into the database process space

With C or C++ UDF libraries that run in process, the functions are typically registered as built-in functions within the database server and called like any other built-in function in a SQL statement. Running in process allows the function to have full access to the database server’s memory, parallelism and processing management capabilities. Because of this, the functions must be well-behaved so as not to negatively impact the database or the engine. This type of UDF gives the highest performance out of any method for OLAP, mathematical, statistical, univariate

distributions and data mining algorithms.

Out-of-Process

Out-of-Process UDFs are typically written in C, C++ or JAVA. By running out of process, they do not run the same risk to the database or the engine as they run in their own process space with their own resources. Here, they wouldn't be expected to have the same performance as an in-process UDF. They are still typically registered in the database engine and called through standard SQL, usually in a stored procedure. Out-of-process UDFs are a safe way to extend the capabilities of a database server and are an ideal way to add custom data mining libraries.

Uses

In-database processing makes data analysis more accessible and relevant for high-throughput, real-time applications including fraud detection, credit scoring, risk management, transaction processing, pricing and margin analysis, usage-based micro-segmenting, behavioral ad targeting and recommendation engines, such as those used by customer service organizations to determine next-best actions.

Vendors

In-database processing is performed and promoted as a feature by many of the major data warehousing vendors, including Teradata (and acquired Aster Data Systems), IBM Netezza, EMC Greenplum, Sybase, ParAccel, SAS, and EXASOL. Fuzzy Logix offers libraries of in-database models used for mathematical, statistical, data mining, simulation and classification modelling as well as financial models for equity, fixed income, interest rate and portfolio optimization.

Related Technologies

In-database processing is one of several technologies focused on improving data warehousing performance. Others include parallel computing, shared everything architectures, shared nothing architectures and massive parallel processing. It is an important step towards improving predictive analytics capabilities.[1]

External links

- EXASOL EXAPowerlytics [2]

References

[1] (http://timmanns.blogspot.com/2009/01/isnt-in-database-processing-old-news.html) "Isn't In-database processing old news yet?," "Blog by Tim Manns (Data Mining Blog)," January 8, 2009

[2] http://www.exasol.com/en/exasolution/exapowerlytics.html

Locks with ordered sharing

In databases and transaction processing the term **Locks with ordered sharing** comprises several variants of the *Two phase locking* (2PL) concurrency control protocol generated by changing the blocking semantics of locks upon conflicts. One variant is identical to Strict commitment ordering (SCO).

References

- D. Agrawal, A. El Abbadi, A. E. Lang: *The Performance of Protocols Based on Locks with Ordered Sharing* [1], IEEE Transactions on Knowledge and Data Engineering, Volume 6, Issue 5, October 1994, PP. 805 - 818, ISSN:1041-4347

References

[1] http://portal.acm.org/citation.cfm?id=627615

Nested transaction

A **nested transaction** is a database transaction that is started by an instruction within the scope of an already started transaction.

Nested transactions are implemented differently in different databases. However, they have in common that the changes are not made visible to any unrelated transactions until the outermost transaction has committed. This means that a commit in an inner transaction does not necessary persist updates to the database.

In some databases, changes made by the nested transaction are not seen by the 'host' transaction until the nested transaction is committed. According to some,Wikipedia:Avoid weasel words this follows from the isolation property of transactions.

The capability to handle nested transactions properly is a prerequisite for true component based application architectures. In a component-based encapsulated architecture, nested transactions can occur without the programmer knowing it. A component function may or may not contain a database transaction (this is the encapsulated secret of the component. See Information hiding). If a call to such a component function is made inside a BEGIN - COMMIT bracket, nested transactions occur. Since popular databases like MySQL do not allow nesting BEGIN - COMMIT brackets, a framework or a transaction monitor is needed to handle this. When we speak about nested transactions, it should be made clear that this feature is DBMS dependent and is not available for all databases.

Theory for nested transactions is similar to the theory for flat transactions, and was introduced in the following paper:

- Resende, R.F.; El Abbadi, A. (1994-05-25). "On the serializability theorem for nested transactions". *Information Processing Letters* **50** (4): 177–183. doi:10.1016/0020-0190(94)00033-6 [1].

The banking industry usually processes financial transactions using **Open Nested Transactions**, which is a looser variant of the nested transaction model that provides higher performance while accepting the accompanying trade-offs of inconsistency. Open Nested Transactions are discussed in the following paper:

- Weikum, Gerhard; Hans-J. Schek (1992). "Concepts and Applications of Multilevel Transactions and Open Nested Transactions" [2]. *Database Transaction Models for Advanced Applications* (Morgan Kaufmann): 515–553. ISBN 1-55860-214-3. Retrieved 2007-11-13.

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8

References

[1] http://dx.doi.org/10.1016%2F0020-0190%2894%2900033-6

[2] http://citeseer.ist.psu.edu/119221.html

Transaction processing system

Transaction processing is a style of computing that divides work into individual, indivisible operations, called transactions. A **transaction processing system (TPS)** or **transaction server** is a software system, or software/hardware combination, that supports transaction processing.

History

The first transaction processing systems was American Airlines SABRE system[*citation needed*], which became operational in 1960. Designed to process up to 83,000 transactions a day, the system ran on two IBM 7090 computers. SABRE was migrated to IBM System/360 computers in 1972, and became an IBM product first as *Airline control Program (ACP)* and later as *Transaction Processing Facility (TPF)*. In addition to airlines TPF is used by large banks, credit card companies, and hotel chains.

The Hewlett-Packard NonStop system (formerly Tandem NonStop) was a hardware and software system designed for *Online Transaction Processing (OLTP)* introduced in 1976. The systems were designed for transaction processing and provided an extreme level of availability and data integrity.

List of transaction processing systems

- IBM Transaction Processing Facility (TPF) - 1960. Unlike most other transaction processing systems TPF is a dedicated operating system for transaction processing on IBM System z mainframes. Originally Airline Control Program (ACP).
- IBM Information Management System (IMS) - 1966. A joint hierarchical database and information management system with extensive transaction processing capabilities. Runs on OS/360 and successors.
- IBM Customer Information Control System (CICS) - 1969. A transaction manager designed for rapid, high-volume online processing, CICS originally used standard system datasets, but now has a connection to IBM's DB/2 relational database system. Runs on OS/360 and successors and DOS/360 and successors, IBM AIX, VM, and OS/2. Non-mainframe versions are called *TXSeries*.
- Tuxedo - 1980s. Transactions for Unix, Extended for Distributed Operations developed by AT&T Corporation, now owned by Oracle Corporation. Tuxedo is a cross-platform TPS.
- UNIVAC Transaction Interface Package (TIP) - 1970s. A transaction processing monitor for UNIVAC 1100/2200 series computers.
- Burroughs Corporation supported transaction processing capabilities in its MCP operating systems. As of 2012 UNISYS ClearPath Enterprise Servers include Transaction Server, "an extremely flexible, high-performance message and application control system."
- Digital Equipment Corporation (DEC) Application Control and Management System (ACMS) - 1985. "Provides an environment for creating and controlling online transaction processing (OLTP) applications on the VMS operating system." Runs on VAX/VMS systems.

- Digital Equipment Corporation (DEC) Message Control System (MCS-10) for PDP-10 TOPS-10 systems.
- Honeywell Multics Transaction Processing. Feature (TP) - 1979.
- Transaction Management eXecutive (TMX) was NCR Corporation's proprietary transaction processing system running on NCR Tower 5000-series systems. This system was used mainly by financial institutions in the 1980s and 1990s.
- Hewlett-Packard NonStop system - 1976. NonStop is an integrated hardware and software system specifically designed for transaction processing. Originally from Tandem Computers.
- Transarc Encina - 1991. Transarc was purchased by IBM in 1994. Encina was discontinued as a product and folded into IBM's *TXSeries*. Encina support was discontinued in 2006.

Processing types

Transaction processing is distinct from other computer processing models — batch processing, time-sharing, and real-time processing.

Batch processing

Batch processing is execution of a series of programs (*jobs*) on a computer without manual intervention. Several transactions, called a *batch* are collected and processed at the same time. The results of each transaction are not immediately available when the transaction is being entered;[1] there is a time delay.

Real-time processing

"Real time systems attempt to guarantee an appropriate response to a stimulus or request quickly enough to affect the conditions that caused the stimulus." Each transaction in real-time processing is unique; it is not part of a group of transactions.

Time-sharing

Time sharing is the sharing of a computer system among multiple users, usually giving each user the illusion that they have exclusive control of the system. The users may be working on the same project or different projects, but there are usually few restrictions on the type of work each user is doing.

Transaction processing

Transaction processing systems also attempt to provide predictable response times to requests, although this is not as critical as for real-time systems. Rather than allowing the user to run arbitrary programs as time-sharing, transaction processing allows only predefined, structured transactions. Each transaction is usually short duration and the processing activity for each transaction is programmed in advance.

Transaction processing system features

The following features are considered important in evaluating transaction processing systems.

Performance

Fast performance with a rapid response time is critical. Transaction processing systems are usually measured by the number of transactions they can process in a given period of time.

Continuous availability

The system must be available during the time period when the users are entering transactions. Many organizations rely heavily on their TPS; a breakdown will disrupt operations or even stop the business.

Data integrity

The system must be able to handle hardware or software problems without corrupting data. Multiple users must be protected from attempting to change the same piece of data at the same time, for example two operators cannot sell the same seat on an airplane.

Ease of use

Often users of transaction processing systems are casual users. The system should be simple for them to understand, protect them from data-entry errors as much as possible, and allow them to easily correct their errors.

Modular growth

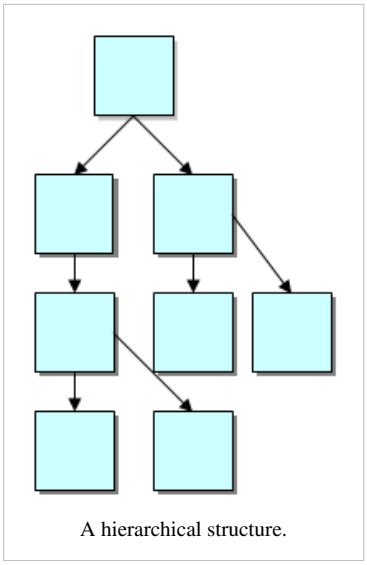
The system should be capable of growth at incremental costs, rather than requiring a complete replacement. It should be possible to add, replace, or update hardware and software components without shutting down the system.

Databases and files

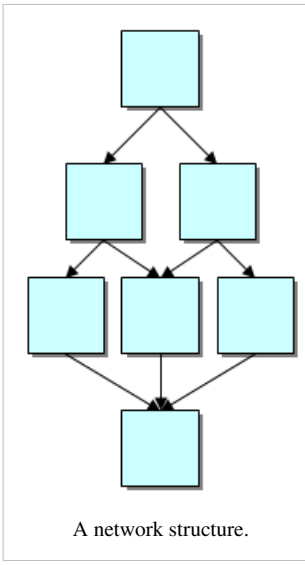
A database is an organized collection of data. Databases offer fast retrieval times for non-structured requests as in a typical transaction processing application.

Databases may be constructed using hierarchical, network, or relational structures.

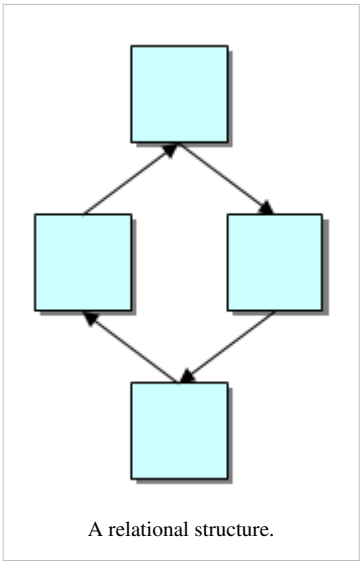
- Hierarchical structure: organizes data in a series of levels. Its top to bottom like structure consists of nodes and branches; each child node has branches and is only linked to one higher level parent node.
- Network structure: network structures also organizes data using nodes and branches. But, unlike hierarchical, each child node can be linked to multiple, higher parent nodes.
- Relational structure: a relational database organizes its data in a series of related tables. This gives flexibility as relationships between the tables are built.



A hierarchical structure.



A network structure.



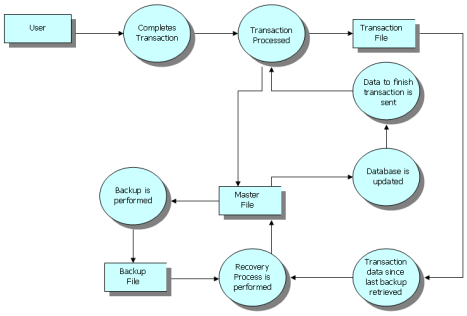
A relational structure.

The following features are desirable in a database system used in transaction processing systems:

- **Good data placement**: The database should be designed to access patterns of data from many simultaneous users.
- **Short transactions**: Short transactions enables quick processing. This avoids concurrency and paces the systems.
- **Real-time backup**: Backup should be scheduled between low times of activity to prevent lag of the server.
- **High normalization**: This lowers redundant information to increase the speed and improve concurrency, this also improves backups.
- **Archiving of historical data**: Uncommonly used data are moved into other databases or backed up tables. This keeps tables small and also improves backup times.
- **Good hardware configuration**: Hardware must be able to handle many users and provide quick response times.

Backup procedures

Since business organizations have become very dependent on TPSs, a breakdown in their TPS may stop the business' regular routines and thus stopping its operation for a certain amount of time. In order to prevent data loss and minimize disruptions when a TPS breaks down a well-designed backup and recovery procedure is put into use. The recovery process can rebuild the system when it goes down.



A Dataflow Diagram of backup and recovery procedures.

Recovery process

A TPS may fail for many reasons. These reasons could include a system failure, human errors, hardware failure, incorrect or invalid data, computer viruses, software application errors or natural or man-made disasters. As it's not possible to prevent all TPS failures, a TPS must be able to cope with failures. The TPS must be able to detect and correct errors when they occur. A TPS will go through a recovery of the database to cope when the system fails, it involves the backup, journal, checkpoint, and recovery manager:

- Journal: A journal maintains an audit trail of transactions and database changes. Transaction logs and Database change logs are used, a transaction log records all the essential data for each transactions, including data values, time of transaction and terminal number. A database change log contains before and after copies of records that

have been modified by transactions.

- Checkpoint: *The purpose of checkpointing* is to provide a snapshot of the data within the database. A checkpoint, in general, is any identifier or other reference that identifies the state of the database at a point in time. Modifications to database pages are performed in memory and are not necessarily written to disk after every update. Therefore, periodically, the database system must perform a checkpoint to write these updates which are held in-memory to the storage disk. Writing these updates to storage disk creates a point in time in which the database system can apply changes contained in a transaction log during recovery after an unexpected shut down or crash of the database system.

If a checkpoint is interrupted and a recovery is required, then the database system must start recovery from a previous successful checkpoint. *Checkpointing can be either transaction-consistent or non-transaction-consistent* (called also fuzzy checkpointing). *Transaction-consistent checkpointing* produces a persistent database image that is sufficient to recover the database to the state that was externally perceived at the moment of starting the checkpointing. *A non-transaction-consistent checkpointing* results in a persistent database image that is insufficient to perform a recovery of the database state. To perform the database recovery, additional information is needed, typically contained in transaction logs. Transaction consistent checkpointing refers to a consistent database, which doesn't necessarily include all the latest committed transactions, but all modifications made by transactions, that were committed at the time checkpoint creation was started, are fully present. A non-consistent transaction refers to a checkpoint which is not necessarily a consistent database, and can't be recovered to one without all log records generated for open transactions included in the checkpoint. Depending on the type of database management system implemented *a checkpoint may incorporate indexes or storage pages (user data), indexes and storage pages*. If no indexes are incorporated into the checkpoint, indexes must be created when the database is restored from the checkpoint image.

- Recovery Manager: A recovery manager is a program which restores the database to a correct condition which can restart the transaction processing.

Depending on how the system failed, there can be two different recovery procedures used. Generally, the procedures involves restoring data that has been collected from a backup device and then running the transaction processing again. Two types of recovery are *backward recovery* and *forward recovery*:

- Backward recovery: used to undo unwanted changes to the database. It reverses the changes made by transactions which have been aborted.
- Forward recovery: it starts with a backup copy of the database. The transaction will then reprocess according to the transaction journal that occurred between the time the backup was made and the present time.

Types of back-up procedures

There are two main types of Back-up Procedures: **Grandfather-father-son** and **Partial backups**:

Grandfather-father-son

This procedure refers to at least three generations of backup master files. thus, the most recent backup is the son, the oldest backup is the grandfather. It's commonly used for a *batch transaction processing system* with a magnetic tape. If the system fails during a batch run, the master file is recreated by using the son backup and then restarting the batch. However if the son backup fails, is corrupted or destroyed, then the previous generation of backup (the father) is used. Likewise, if that fails, then the generation of backup previous to the father (i.e. the grandfather) is required. Of course the older the generation, the more the data may be out of date. Organizations can have many generations of backup.

Partial backups

This only occurs when parts of the master file are backed up. The master file is usually backed up to magnetic tape at regular times, this could be daily, weekly or monthly. Completed transactions since the last backup are stored separately and are called **journals**, or **journal files**. The master file can be recreated from the journal files on the backup tape if the system is to fail.

Updating in a batch

This is used when transactions are recorded on paper (such as bills and invoices) or when it's being stored on a magnetic tape. Transactions will be collected and updated as a batch when it's convenient or economical to process them. Historically, this was the most common method as the information technology did not exist to allow real-time processing.

The two stages in batch processing are:

- Collecting and storage of the transaction data into a transaction file - this involves sorting the data into sequential order.
- Processing the data by updating the master file - which can be difficult, this may involve data additions, updates and deletions which may require being performed in a certain order. If an error occurs, then the entire batch fails.

Updating in batch requires sequential access - since it uses a magnetic tape this is the only way to access data. A batch will start at the beginning of the tape, then reading it from the order it was stored; it's very time-consuming to locate specific transactions.

The information technology used includes a secondary storage medium which can store large quantities of data inexpensively (thus the common choice of a magnetic tape). The software used to collect data does not have to be online - it doesn't even need a user interface.

Updating in real-time

This is the immediate processing of data. It provides instant confirmation of a transaction. It may involve a large number of users who are simultaneously performing transactions which change data. Because of advances in technology (such as the increase in the speed of data transmission and larger bandwidth), real-time updating is possible.

Steps in a real-time update involve the sending of a transaction data to an online database in a master file. The person providing information is usually able to help with error correction and receives confirmation of the transaction completion.

Updating in real-time uses direct access of data. This occurs when data are accessed without accessing previous data items. The storage device stores data in a particular location based on a mathematical procedure. This will then be calculated to find an approximate location of the data. If data are not found at this location, it will search through successive locations until it's found.

The information technology used could be a secondary storage medium that can store large amounts of data and provide quick access (thus the common choice of a magnetic disk).

References

[1] http://en.wikipedia.org/wiki/Transaction_processing_system#endnote_HSC

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8

Transaction processing systems

Transaction processing is a style of computing that divides work into individual, indivisible operations, called transactions. A **transaction processing system (TPS)** or **transaction server** is a software system, or software/hardware combination, that supports transaction processing.

History

The first transaction processing systems was American Airlines SABRE system[*citation needed*], which became operational in 1960. Designed to process up to 83,000 transactions a day, the system ran on two IBM 7090 computers. SABRE was migrated to IBM System/360 computers in 1972, and became an IBM product first as *Airline control Program (ACP)* and later as *Transaction Processing Facility (TPF)*. In addition to airlines TPF is used by large banks, credit card companies, and hotel chains.

The Hewlett-Packard NonStop system (formerly Tandem NonStop) was a hardware and software system designed for *Online Transaction Processing (OLTP)* introduced in 1976. The systems were designed for transaction processing and provided an extreme level of availability and data integrity.

List of transaction processing systems

- IBM Transaction Processing Facility (TPF) - 1960. Unlike most other transaction processing systems TPF is a dedicated operating system for transaction processing on IBM System z mainframes. Originally Airline Control Program (ACP).
- IBM Information Management System (IMS) - 1966. A joint hierarchical database and information management system with extensive transaction processing capabilities. Runs on OS/360 and successors.
- IBM Customer Information Control System (CICS) - 1969. A transaction manager designed for rapid, high-volume online processing, CICS originally used standard system datasets, but now has a connection to IBM's DB/2 relational database system. Runs on OS/360 and successors and DOS/360 and successors, IBM AIX, VM, and OS/2. Non-mainframe versions are called *TXSeries*.
- Tuxedo - 1980s. Transactions for Unix, Extended for Distributed Operations developed by AT&T Corporation, now owned by Oracle Corporation. Tuxedo is a cross-platform TPS.
- UNIVAC Transaction Interface Package (TIP) - 1970s. A transaction processing monitor for UNIVAC 1100/2200 series computers.
- Burroughs Corporation supported transaction processing capabilities in its MCP operating systems. As of 2012 UNISYS ClearPath Enterprise Servers include Transaction Server, "an extremely flexible, high-performance message and application control system."
- Digital Equipment Corporation (DEC) Application Control and Management System (ACMS) - 1985. "Provides an environment for creating and controlling online transaction processing (OLTP) applications on the VMS operating system." Runs on VAX/VMS systems.
- Digital Equipment Corporation (DEC) Message Control System (MCS-10) for PDP-10 TOPS-10 systems.

- Honeywell Multics Transaction Processing. Feature (TP) - 1979.
- Transaction Management eXecutive (TMX) was NCR Corporation's proprietary transaction processing system running on NCR Tower 5000-series systems. This system was used mainly by financial institutions in the 1980s and 1990s.
- Hewlett-Packard NonStop system - 1976. NonStop is an integrated hardware and software system specifically designed for transaction processing. Originally from Tandem Computers.
- Transarc Encina - 1991. Transarc was purchased by IBM in 1994. Encina was discontinued as a product and folded into IBM's *TXSeries*. Encina support was discontinued in 2006.

Processing types

Transaction processing is distinct from other computer processing models — batch processing, time-sharing, and real-time processing.

Batch processing

Batch processing is execution of a series of programs (*jobs*) on a computer without manual intervention. Several transactions, called a *batch* are collected and processed at the same time. The results of each transaction are not immediately available when the transaction is being entered;[1] there is a time delay.

Real-time processing

"Real time systems attempt to guarantee an appropriate response to a stimulus or request quickly enough to affect the conditions that caused the stimulus." Each transaction in real-time processing is unique; it is not part of a group of transactions.

Time-sharing

Time sharing is the sharing of a computer system among multiple users, usually giving each user the illusion that they have exclusive control of the system. The users may be working on the same project or different projects, but there are usually few restrictions on the type of work each user is doing.

Transaction processing

Transaction processing systems also attempt to provide predictable response times to requests, although this is not as critical as for real-time systems. Rather than allowing the user to run arbitrary programs as time-sharing, transaction processing allows only predefined, structured transactions. Each transaction is usually short duration and the processing activity for each transaction is programmed in advance.

Transaction processing system features

The following features are considered important in evaluating transaction processing systems.

Performance

Fast performance with a rapid response time is critical. Transaction processing systems are usually measured by the number of transactions they can process in a given period of time.

Continuous availability

The system must be available during the time period when the users are entering transactions. Many organizations rely heavily on their TPS; a breakdown will disrupt operations or even stop the business.

Data integrity

The system must be able to handle hardware or software problems without corrupting data. Multiple users must be protected from attempting to change the same piece of data at the same time, for example two operators cannot sell the same seat on an airplane.

Ease of use

Often users of transaction processing systems are casual users. The system should be simple for them to understand, protect them from data-entry errors as much as possible, and allow them to easily correct their errors.

Modular growth

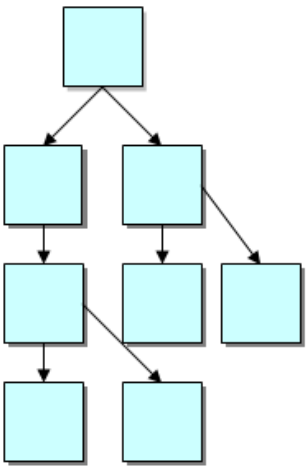
The system should be capable of growth at incremental costs, rather than requiring a complete replacement. It should be possible to add, replace, or update hardware and software components without shutting down the system.

Databases and files

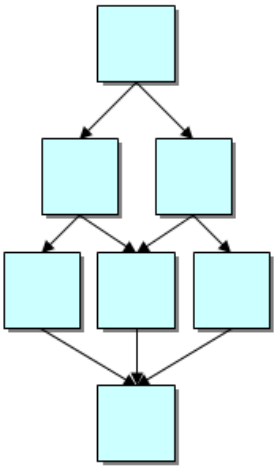
A database is an organized collection of data. Databases offer fast retrieval times for non-structured requests as in a typical transaction processing application.

Databases may be constructed using hierarchical, network, or relational structures.

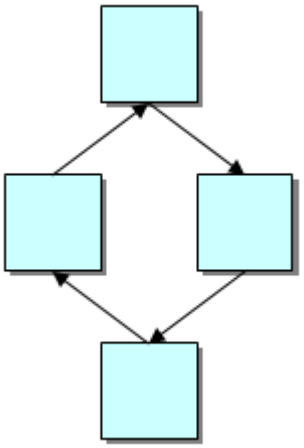
- Hierarchical structure: organizes data in a series of levels. Its top to bottom like structure consists of nodes and branches; each child node has branches and is only linked to one higher level parent node.
- Network structure: network structures also organizes data using nodes and branches. But, unlike hierarchical, each child node can be linked to multiple, higher parent nodes.
- Relational structure: a relational database organizes its data in a series of related tables. This gives flexibility as relationships between the tables are built.



A hierarchical structure.



A network structure.



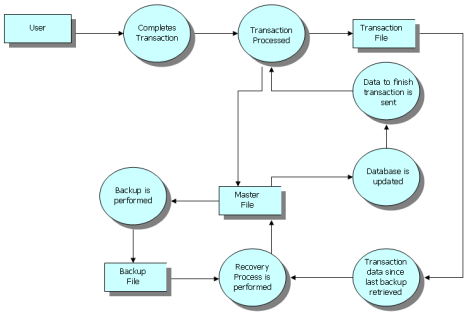
A relational structure.

The following features are desirable in a database system used in transaction processing systems:

- **Good data placement**: The database should be designed to access patterns of data from many simultaneous users.
- **Short transactions**: Short transactions enables quick processing. This avoids concurrency and paces the systems.
- **Real-time backup**: Backup should be scheduled between low times of activity to prevent lag of the server.
- **High normalization**: This lowers redundant information to increase the speed and improve concurrency, this also improves backups.
- **Archiving of historical data**: Uncommonly used data are moved into other databases or backed up tables. This keeps tables small and also improves backup times.
- **Good hardware configuration**: Hardware must be able to handle many users and provide quick response times.

Backup procedures

Since business organizations have become very dependent on TPSs, a breakdown in their TPS may stop the business' regular routines and thus stopping its operation for a certain amount of time. In order to prevent data loss and minimize disruptions when a TPS breaks down a well-designed backup and recovery procedure is put into use. The recovery process can rebuild the system when it goes down.



A Dataflow Diagram of backup and recovery procedures.

Recovery process

A TPS may fail for many reasons. These reasons could include a system failure, human errors, hardware failure, incorrect or invalid data, computer viruses, software application errors or natural or man-made disasters. As it's not possible to prevent all TPS failures, a TPS must be able to cope with failures. The TPS must be able to detect and correct errors when they occur. A TPS will go through a recovery of the database to cope when the system fails, it involves the backup, journal, checkpoint, and recovery manager:

- Journal: A journal maintains an audit trail of transactions and database changes. Transaction logs and Database change logs are used, a transaction log records all the essential data for each transactions, including data values, time of transaction and terminal number. A database change log contains before and after copies of records that

have been modified by transactions.

- Checkpoint: *The purpose of checkpointing* is to provide a snapshot of the data within the database. A checkpoint, in general, is any identifier or other reference that identifies the state of the database at a point in time. Modifications to database pages are performed in memory and are not necessarily written to disk after every update. Therefore, periodically, the database system must perform a checkpoint to write these updates which are held in-memory to the storage disk. Writing these updates to storage disk creates a point in time in which the database system can apply changes contained in a transaction log during recovery after an unexpected shut down or crash of the database system.

If a checkpoint is interrupted and a recovery is required, then the database system must start recovery from a previous successful checkpoint. *Checkpointing can be either transaction-consistent or non-transaction-consistent* (called also fuzzy checkpointing). *Transaction-consistent checkpointing* produces a persistent database image that is sufficient to recover the database to the state that was externally perceived at the moment of starting the checkpointing. *A non-transaction-consistent checkpointing* results in a persistent database image that is insufficient to perform a recovery of the database state. To perform the database recovery, additional information is needed, typically contained in transaction logs. Transaction consistent checkpointing refers to a consistent database, which doesn't necessarily include all the latest committed transactions, but all modifications made by transactions, that were committed at the time checkpoint creation was started, are fully present. A non-consistent transaction refers to a checkpoint which is not necessarily a consistent database, and can't be recovered to one without all log records generated for open transactions included in the checkpoint. Depending on the type of database management system implemented *a checkpoint may incorporate indexes or storage pages (user data), indexes and storage pages*. If no indexes are incorporated into the checkpoint, indexes must be created when the database is restored from the checkpoint image.

- Recovery Manager: A recovery manager is a program which restores the database to a correct condition which can restart the transaction processing.

Depending on how the system failed, there can be two different recovery procedures used. Generally, the procedures involves restoring data that has been collected from a backup device and then running the transaction processing again. Two types of recovery are *backward recovery* and *forward recovery*:

- Backward recovery: used to undo unwanted changes to the database. It reverses the changes made by transactions which have been aborted.
- Forward recovery: it starts with a backup copy of the database. The transaction will then reprocess according to the transaction journal that occurred between the time the backup was made and the present time.

Types of back-up procedures

There are two main types of Back-up Procedures: **Grandfather-father-son** and **Partial backups**:

Grandfather-father-son

This procedure refers to at least three generations of backup master files. thus, the most recent backup is the son, the oldest backup is the grandfather. It's commonly used for a *batch transaction processing system* with a magnetic tape. If the system fails during a batch run, the master file is recreated by using the son backup and then restarting the batch. However if the son backup fails, is corrupted or destroyed, then the previous generation of backup (the father) is used. Likewise, if that fails, then the generation of backup previous to the father (i.e. the grandfather) is required. Of course the older the generation, the more the data may be out of date. Organizations can have many generations of backup.

Partial backups

This only occurs when parts of the master file are backed up. The master file is usually backed up to magnetic tape at regular times, this could be daily, weekly or monthly. Completed transactions since the last backup are stored separately and are called **journals**, or **journal files**. The master file can be recreated from the journal files on the backup tape if the system is to fail.

Updating in a batch

This is used when transactions are recorded on paper (such as bills and invoices) or when it's being stored on a magnetic tape. Transactions will be collected and updated as a batch when it's convenient or economical to process them. Historically, this was the most common method as the information technology did not exist to allow real-time processing.

The two stages in batch processing are:

- Collecting and storage of the transaction data into a transaction file - this involves sorting the data into sequential order.
- Processing the data by updating the master file - which can be difficult, this may involve data additions, updates and deletions which may require being performed in a certain order. If an error occurs, then the entire batch fails.

Updating in batch requires sequential access - since it uses a magnetic tape this is the only way to access data. A batch will start at the beginning of the tape, then reading it from the order it was stored; it's very time-consuming to locate specific transactions.

The information technology used includes a secondary storage medium which can store large quantities of data inexpensively (thus the common choice of a magnetic tape). The software used to collect data does not have to be online - it doesn't even need a user interface.

Updating in real-time

This is the immediate processing of data. It provides instant confirmation of a transaction. It may involve a large number of users who are simultaneously performing transactions which change data. Because of advances in technology (such as the increase in the speed of data transmission and larger bandwidth), real-time updating is possible.

Steps in a real-time update involve the sending of a transaction data to an online database in a master file. The person providing information is usually able to help with error correction and receives confirmation of the transaction completion.

Updating in real-time uses direct access of data. This occurs when data are accessed without accessing previous data items. The storage device stores data in a particular location based on a mathematical procedure. This will then be calculated to find an approximate location of the data. If data are not found at this location, it will search through successive locations until it's found.

The information technology used could be a secondary storage medium that can store large amounts of data and provide quick access (thus the common choice of a magnetic disk).

References

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8

Transaction server

Transaction processing is a style of computing that divides work into individual, indivisible operations, called transactions. A **transaction processing system (TPS)** or **transaction server** is a software system, or software/hardware combination, that supports transaction processing.

History

The first transaction processing systems was American Airlines SABRE system[*citation needed*], which became operational in 1960. Designed to process up to 83,000 transactions a day, the system ran on two IBM 7090 computers. SABRE was migrated to IBM System/360 computers in 1972, and became an IBM product first as *Airline control Program (ACP)* and later as *Transaction Processing Facility (TPF)*. In addition to airlines TPF is used by large banks, credit card companies, and hotel chains.

The Hewlett-Packard NonStop system (formerly Tandem NonStop) was a hardware and software system designed for *Online Transaction Processing (OLTP)* introduced in 1976. The systems were designed for transaction processing and provided an extreme level of availability and data integrity.

List of transaction processing systems

- IBM Transaction Processing Facility (TPF) - 1960. Unlike most other transaction processing systems TPF is a dedicated operating system for transaction processing on IBM System z mainframes. Originally Airline Control Program (ACP).
- IBM Information Management System (IMS) - 1966. A joint hierarchical database and information management system with extensive transaction processing capabilities. Runs on OS/360 and successors.
- IBM Customer Information Control System (CICS) - 1969. A transaction manager designed for rapid, high-volume online processing, CICS originally used standard system datasets, but now has a connection to IBM's DB/2 relational database system. Runs on OS/360 and successors and DOS/360 and successors, IBM AIX, VM, and OS/2. Non-mainframe versions are called *TXSeries*.
- Tuxedo - 1980s. Transactions for Unix, Extended for Distributed Operations developed by AT&T Corporation, now owned by Oracle Corporation. Tuxedo is a cross-platform TPS.
- UNIVAC Transaction Interface Package (TIP) - 1970s. A transaction processing monitor for UNIVAC 1100/2200 series computers.
- Burroughs Corporation supported transaction processing capabilities in its MCP operating systems. As of 2012 UNISYS ClearPath Enterprise Servers include Transaction Server, "an extremely flexible, high-performance message and application control system."
- Digital Equipment Corporation (DEC) Application Control and Management System (ACMS) - 1985. "Provides an environment for creating and controlling online transaction processing (OLTP) applications on the VMS operating system." Runs on VAX/VMS systems.
- Digital Equipment Corporation (DEC) Message Control System (MCS-10) for PDP-10 TOPS-10 systems.
- Honeywell Multics Transaction Processing. Feature (TP) - 1979.
- Transaction Management eXecutive (TMX) was NCR Corporation's proprietary transaction processing system running on NCR Tower 5000-series systems. This system was used mainly by financial institutions in the 1980s and 1990s.
- Hewlett-Packard NonStop system - 1976. NonStop is an integrated hardware and software system specifically designed for transaction processing. Originally from Tandem Computers.
- Transarc Encina - 1991. Transarc was purchased by IBM in 1994. Encina was discontinued as a product and folded into IBM's *TXSeries*. Encina support was discontinued in 2006.

Processing types

Transaction processing is distinct from other computer processing models — batch processing, time-sharing, and real-time processing.

Batch processing

Batch processing is execution of a series of programs (*jobs*) on a computer without manual intervention. Several transactions, called a *batch* are collected and processed at the same time. The results of each transaction are not immediately available when the transaction is being entered;[1] there is a time delay.

Real-time processing

"Real time systems attempt to guarantee an appropriate response to a stimulus or request quickly enough to affect the conditions that caused the stimulus." Each transaction in real-time processing is unique; it is not part of a group of transactions.

Time-sharing

Time sharing is the sharing of a computer system among multiple users, usually giving each user the illusion that they have exclusive control of the system. The users may be working on the same project or different projects, but there are usually few restrictions on the type of work each user is doing.

Transaction processing

Transaction processing systems also attempt to provide predictable response times to requests, although this is not as critical as for real-time systems. Rather than allowing the user to run arbitrary programs as time-sharing, transaction processing allows only predefined, structured transactions. Each transaction is usually short duration and the processing activity for each transaction is programmed in advance.

Transaction processing system features

The following features are considered important in evaluating transaction processing systems.

Performance

Fast performance with a rapid response time is critical. Transaction processing systems are usually measured by the number of transactions they can process in a given period of time.

Continuous availability

The system must be available during the time period when the users are entering transactions. Many organizations rely heavily on their TPS; a breakdown will disrupt operations or even stop the business.

Data integrity

The system must be able to handle hardware or software problems without corrupting data. Multiple users must be protected from attempting to change the same piece of data at the same time, for example two operators cannot sell the same seat on an airplane.

Ease of use

Often users of transaction processing systems are casual users. The system should be simple for them to understand, protect them from data-entry errors as much as possible, and allow them to easily correct their errors.

Modular growth

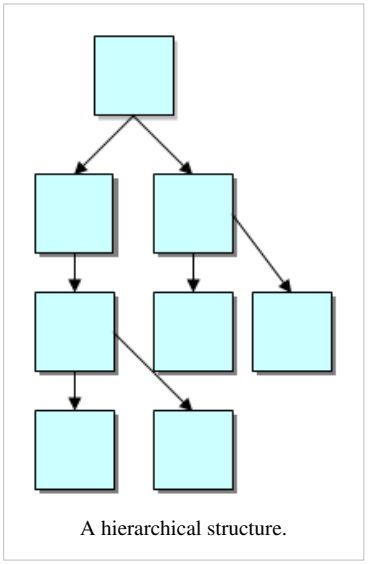
The system should be capable of growth at incremental costs, rather than requiring a complete replacement. It should be possible to add, replace, or update hardware and software components without shutting down the system.

Databases and files

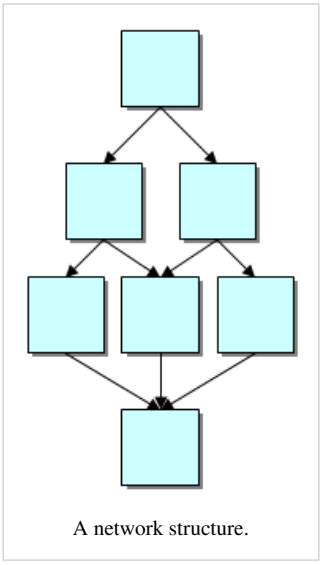
A database is an organized collection of data. Databases offer fast retrieval times for non-structured requests as in a typical transaction processing application.

Databases may be constructed using hierarchical, network, or relational structures.

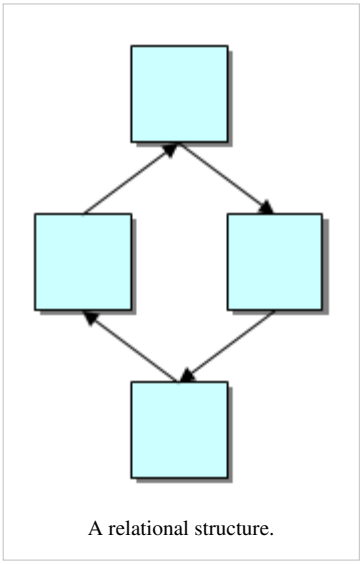
- Hierarchical structure: organizes data in a series of levels. Its top to bottom like structure consists of nodes and branches; each child node has branches and is only linked to one higher level parent node.
- Network structure: network structures also organizes data using nodes and branches. But, unlike hierarchical, each child node can be linked to multiple, higher parent nodes.
- Relational structure: a relational database organizes its data in a series of related tables. This gives flexibility as relationships between the tables are built.



A hierarchical structure.



A network structure.



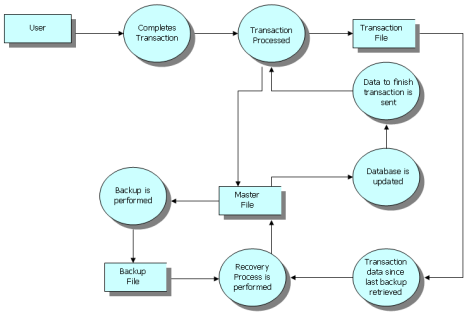
A relational structure.

The following features are desirable in a database system used in transaction processing systems:

- **Good data placement**: The database should be designed to access patterns of data from many simultaneous users.
- **Short transactions**: Short transactions enables quick processing. This avoids concurrency and paces the systems.
- **Real-time backup**: Backup should be scheduled between low times of activity to prevent lag of the server.
- **High normalization**: This lowers redundant information to increase the speed and improve concurrency, this also improves backups.
- **Archiving of historical data**: Uncommonly used data are moved into other databases or backed up tables. This keeps tables small and also improves backup times.
- **Good hardware configuration**: Hardware must be able to handle many users and provide quick response times.

Backup procedures

Since business organizations have become very dependent on TPSs, a breakdown in their TPS may stop the business' regular routines and thus stopping its operation for a certain amount of time. In order to prevent data loss and minimize disruptions when a TPS breaks down a well-designed backup and recovery procedure is put into use. The recovery process can rebuild the system when it goes down.



A Dataflow Diagram of backup and recovery procedures.

Recovery process

A TPS may fail for many reasons. These reasons could include a system failure, human errors, hardware failure, incorrect or invalid data, computer viruses, software application errors or natural or man-made disasters. As it's not possible to prevent all TPS failures, a TPS must be able to cope with failures. The TPS must be able to detect and correct errors when they occur. A TPS will go through a recovery of the database to cope when the system fails, it involves the backup, journal, checkpoint, and recovery manager:

- Journal: A journal maintains an audit trail of transactions and database changes. Transaction logs and Database change logs are used, a transaction log records all the essential data for each transactions, including data values, time of transaction and terminal number. A database change log contains before and after copies of records that have been modified by transactions.
- Checkpoint: *The purpose of checkpointing* is to provide a snapshot of the data within the database. A checkpoint, in general, is any identifier or other reference that identifies the state of the database at a point in time. Modifications to database pages are performed in memory and are not necessarily written to disk after every update. Therefore, periodically, the database system must perform a checkpoint to write these updates which are held in-memory to the storage disk. Writing these updates to storage disk creates a point in time in which the database system can apply changes contained in a transaction log during recovery after an unexpected shut down or crash of the database system.

If a checkpoint is interrupted and a recovery is required, then the database system must start recovery from a previous successful checkpoint. *Checkpointing can be either transaction-consistent or non-transaction-consistent* (called also fuzzy checkpointing). *Transaction-consistent checkpointing* produces a persistent database image that is sufficient to recover the database to the state that was externally perceived at the moment of starting the checkpointing. *A non-transaction-consistent checkpointing* results in a persistent database image that is insufficient to perform a recovery of the database state. To perform the database recovery, additional information is needed, typically contained in transaction logs. Transaction consistent checkpointing refers to a consistent database, which doesn't necessarily include all the latest committed transactions, but all modifications made by transactions, that were committed at the time checkpoint creation was started, are fully present. A non-consistent transaction refers to a checkpoint which is not necessarily a consistent database, and can't be recovered to one without all log records generated for open transactions included in the checkpoint. Depending on the type of database management system implemented *a checkpoint may incorporate indexes or storage pages (user data), indexes and storage pages*. If no indexes are incorporated into the checkpoint, indexes must be created when the database is restored from the checkpoint image.

- Recovery Manager: A recovery manager is a program which restores the database to a correct condition which can restart the transaction processing.

Depending on how the system failed, there can be two different recovery procedures used. Generally, the procedures involves restoring data that has been collected from a backup device and then running the transaction processing

again. Two types of recovery are *backward recovery* and *forward recovery*:

- Backward recovery: used to undo unwanted changes to the database. It reverses the changes made by transactions which have been aborted.
- Forward recovery: it starts with a backup copy of the database. The transaction will then reprocess according to the transaction journal that occurred between the time the backup was made and the present time.

Types of back-up procedures

There are two main types of Back-up Procedures: **Grandfather-father-son** and **Partial backups**:

Grandfather-father-son

This procedure refers to at least three generations of backup master files. thus, the most recent backup is the son, the oldest backup is the grandfather. It's commonly used for a *batch transaction processing system* with a magnetic tape. If the system fails during a batch run, the master file is recreated by using the son backup and then restarting the batch. However if the son backup fails, is corrupted or destroyed, then the previous generation of backup (the father) is used. Likewise, if that fails, then the generation of backup previous to the father (i.e. the grandfather) is required. Of course the older the generation, the more the data may be out of date. Organizations can have many generations of backup.

Partial backups

This only occurs when parts of the master file are backed up. The master file is usually backed up to magnetic tape at regular times, this could be daily, weekly or monthly. Completed transactions since the last backup are stored separately and are called **journals**, or **journal files**. The master file can be recreated from the journal files on the backup tape if the system is to fail.

Updating in a batch

This is used when transactions are recorded on paper (such as bills and invoices) or when it's being stored on a magnetic tape. Transactions will be collected and updated as a batch when it's convenient or economical to process them. Historically, this was the most common method as the information technology did not exist to allow real-time processing.

The two stages in batch processing are:

- Collecting and storage of the transaction data into a transaction file - this involves sorting the data into sequential order.
- Processing the data by updating the master file - which can be difficult, this may involve data additions, updates and deletions which may require being performed in a certain order. If an error occurs, then the entire batch fails.

Updating in batch requires sequential access - since it uses a magnetic tape this is the only way to access data. A batch will start at the beginning of the tape, then reading it from the order it was stored; it's very time-consuming to locate specific transactions.

The information technology used includes a secondary storage medium which can store large quantities of data inexpensively (thus the common choice of a magnetic tape). The software used to collect data does not have to be online - it doesn't even need a user interface.

Updating in real-time

This is the immediate processing of data. It provides instant confirmation of a transaction. It may involve a large number of users who are simultaneously performing transactions which change data. Because of advances in technology (such as the increase in the speed of data transmission and larger bandwidth), real-time updating is possible.

Steps in a real-time update involve the sending of a transaction data to an online database in a master file. The person providing information is usually able to help with error correction and receives confirmation of the transaction completion.

Updating in real-time uses direct access of data. This occurs when data are accessed without accessing previous data items. The storage device stores data in a particular location based on a mathematical procedure. This will then be calculated to find an approximate location of the data. If data are not found at this location, it will search through successive locations until it's found.

The information technology used could be a secondary storage medium that can store large amounts of data and provide quick access (thus the common choice of a magnetic disk).

References

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8

Priority inversion

In computer science, **priority inversion** is a problematic scenario in scheduling in which a high priority task is indirectly preempted by a medium priority task effectively "inverting" the relative priorities of the two tasks.

This violates the priority model that high priority tasks can only be prevented from running by higher priority tasks and briefly by low priority tasks which will quickly complete their use of a resource shared by the high and low priority tasks.

Example of a priority inversion

Consider two tasks H and L, of high and low priority respectively, either of which can acquire exclusive use of a shared resource R. If H attempts to acquire R after L has acquired it, then H becomes unrunnable until L relinquishes the resource. The use of the shared exclusive-use resource when properly designed is such that L relinquishes R promptly enough that H's priority use is not hindered excessively. In spite of the good design of these two cooperating tasks, the surprising behavior, priority inversion, occurs when any third task M of medium priority becomes runnable during L's use of R. Once H becomes unrunnable, M is the highest priority runnable task, thus it runs and while it does L cannot relinquish R. So in this scenario, the medium priority task preempts the high priority task, resulting in a priority inversion.

Consequences

In some cases, priority inversion can occur without causing immediate harm—the delayed execution of the high priority task goes unnoticed, and eventually the low priority task releases the shared resource. However, there are also many situations in which priority inversion can cause serious problems. If the high priority task is left starved of the resources, it might lead to a system malfunction or the triggering of pre-defined corrective measures, such as a watch dog timer resetting the entire system. The trouble experienced by the Mars lander "Mars Pathfinder"[1][2] is a classic example of problems caused by priority inversion in realtime systems.

Priority inversion can also reduce the perceived performance of the system. Low priority tasks usually have a low priority because it is not important for them to finish promptly (for example, they might be a batch job or another non-interactive activity). Similarly, a high priority task has a high priority because it is more likely to be subject to strict time constraints—it may be providing data to an interactive user, or acting subject to realtime response guarantees. Because priority inversion results in the execution of a lower priority task blocking the high priority task, it can lead to reduced system responsiveness, or even the violation of response time guarantees.

A similar problem called deadline interchange can occur within earliest deadline first scheduling (EDF).

Solutions

The existence of this problem has been known since the 1970s. Lampson and Redell published one of the first papers to point out the priority inversion problem. Systems such as the UNIX kernel were already addressing the problem with the splx() primitive. There is no fool-proof method to predict the situation. There are however many existing solutions, of which the most common ones are:

Disabling all interrupts to protect critical sections

> When disabled interrupts are used to prevent priority inversion, there are only two priorities: *preemptible*, and *interrupts disabled.* With no third priority, inversion is impossible. Since there's only one piece of lock data (the interrupt-enable bit), misordering locking is impossible, and so deadlocks cannot occur. Since the critical regions always run to completion, hangs do not occur. Note that this only works if all interrupts are disabled. If only a particular hardware device's interrupt is disabled, priority inversion is reintroduced by the hardware's prioritization of interrupts. In early versions of UNIX, a series of primitives named splx(0) ... splx(7) disabled all interrupts up through the given priority. By properly choosing the highest priority of any interrupt that ever entered the critical section, the priority inversion problem could be solved without locking out all of the interrupts. Ceilings were assigned in rate-monotonic order, i.e. the slower devices had lower priorities.
>
> In multiple CPU systems, a simple variation, "single shared-flag locking" is used. This scheme provides a single flag in shared memory that is used by all CPUs to lock all inter-processor critical sections with a busy-wait. Interprocessor communications are expensive and slow on most multiple CPU systems. Therefore, most such systems are designed to minimize shared resources. As a result, this scheme actually works well on many practical systems. These methods are widely used in simple embedded systems, where they are prized for their reliability, simplicity and low resource use. These schemes also require clever programming to keep the critical sections very brief. Many software engineers consider them impractical in general-purpose computers.

A priority ceiling

> With priority ceilings, the shared mutex process (that runs the operating system code) has a characteristic (high) priority of its own, which is assigned to the task locking the mutex. This works well, provided the other high priority task(s) that tries to access the mutex does not have a priority higher than the ceiling priority.

Priority inheritance

Under the policy of priority inheritance, whenever a high priority task has to wait for some resource shared with an executing low priority task, the low priority task is temporarily assigned the priority of the highest waiting priority task for the duration of its own use of the shared resource, thus keeping medium priority tasks from pre-empting the (originally) low priority task, and thereby affecting the waiting high priority task as well. Once the resource is released, the low priority task continues at its original priority level.

Random boosting

Ready tasks holding locks are randomly boosted in priority until they exit the critical section. This solution is used in Microsoft Windows.

Avoid blocking

Because priority inversion involves a low-priority task blocking a high-priority task, one way to avoid priority inversion is to avoid blocking, for example by using Non-blocking synchronization or Read-copy-update.

References

[1] What Really Happened on Mars (http://research.microsoft.com/~mbj/Mars_Pathfinder/Authoritative_Account.html) by Glenn Reeves of the JPL Pathfinder team

[2] Explanation of priority inversion problem experienced by Mars Pathfinder (http://research.microsoft.com/~mbj/Mars_Pathfinder/)

External links

- Description from FOLDOC (http://foldoc.org/priority+inversion)
- Citations from CiteSeer (http://citeseer.org/cs?q=priority+inversion)
- IEEE Priority Inheritance Paper by Sha, Rajkumar, Lehoczky (http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=626613)
- Introduction to Priority Inversion by Michael Barr (http://www.eetimes.com/discussion/other/4023947/Introduction-to-Priority-Inversion)

Priority ceiling protocol

In real-time computing, the **priority ceiling protocol** is a synchronization protocol for shared resources to avoid unbounded priority inversion and mutual deadlock due to wrong nesting of critical sections. In this protocol each resource is assigned a priority ceiling, which is a priority equal to the highest priority of any task which may lock the resource.

For example, with priority ceilings, the shared mutex process (that runs the operating system code) has a characteristic (high) priority of its own, which is assigned to the task locking the mutex. This works well, provided the other high priority task(s) that tries to access the mutex does not have a priority higher than the ceiling priority.

In the **Immediate Ceiling Priority Protocol** (ICPP) when a task locks the resource its priority is temporarily raised to the priority ceiling of the resource, thus no task that may lock the resource is able to get scheduled. This allows a low priority task to defer execution of higher-priority tasks.

The **Original Ceiling Priority Protocol** (OCPP) has the same worst-case performance, but is subtly different in the implementation which can provide finer grained priority inheritance control mechanism than ICPP. Under this approach, a task can lock a resource only if its dynamic priority is higher than the current system priority. (The dynamic priority of a task is the maximum of its own static priority and any it inherits due to it blocking higher-priority processes. The system current priority is the ceiling of the resource having the highest ceiling among those locked by *other* tasks currently.) Otherwise the task is blocked, and its priority is inherited by the task currently holding the resource that gives the current system priority.

A task will not get scheduled if any resource it may lock actually has been locked by another task, and therefore the priority ceiling protocol prevents deadlocks.

ICPP is called "Ceiling Locking" in Ada, "Priority Protect Protocol" in POSIX and "Priority Ceiling Emulation" in RTSJ. It is also known as "Highest Locker's Priority Protocol" (HLP). [1]

ICPP versus OCPP

The worst-case behaviour of the two ceiling schemes is identical from a scheduling view point. However, there are some differences:[2]

- ICPP is easier to implement than OCPP, as blocking relationships need not be monitored
- ICPP leads to fewer context switches as blocking is prior to first execution
- ICPP requires more priority movements as this happens with all resource usage
- OCPP changes priority only if an actual block has occurred

References

[1] http://user.it.uu.se/~yi/courses/rts/dvp-rts-08/notes/synchronization-resource-sharing.pdf
[2] http://rtsys.informatik.uni-kiel.de/teaching/ss05/v-rt2/lectures/lecture13-handout4.pdf

External links

- How to use priority inheritance (http://www.embedded.com/showArticle.jhtml?articleID=20600062), by Kyle Renwick and Bill Renwick (2004-05-18)

Priority inheritance

In real-time computing, **priority inheritance** is a method for eliminating priority inversion problems. Using this programming method, a process scheduling algorithm will increase the priority of a process to the maximum priority of any process waiting for any resource on which the process has a resource lock.

The basic idea of the priority inheritance protocol is that when a job blocks one or more high-priority jobs, it ignores its original priority assignment and executes its critical section at the highest priority level over all the jobs it blocks. After executing its critical section, the job returns to its original priority level.

Example

Consider three jobs:

Job Name	Priority
H	High
M	Medium
L	Low

Suppose H is blocked by L for some shared resource. The priority inheritance protocol requires that L executes its critical section at the (high) priority of H. As a result, M will be unable to preempt L and will be blocked. That is, the higher-priority job M must wait for the critical section of the lower priority job L to be executed, because L now inherits the priority of H. When L exits its critical section, it regains its original (low) priority and awakens H (which was blocked by L). H, having high priority, immediately preempts L and runs to completion. This enables M and L to resume in succession and run to completion.

References

External links

- Article " Priority Inheritance: The Real Story (http://archive.is/20130127194409/http://www.linuxdevices.com/articles/AT5698775833.html)" by Doug Locke
- Article " Against Priority Inheritance (http://archive.is/20130128022320/http://www.linuxdevices.com/articles/AT7168794919.html)" by Victor Yodaiken
- Article " Implementing Concurrency Control With Priority Inheritance in Real-Time CORBA (http://rtdoc.cs.uri.edu/downloads/wohlever_thesis.pdf)" by Steven Wohlever, Victor Fay Wolfe and Russell Johnston
- Article " Priority Inheritance Spin Locks for Multiprocessor Real-Time Systems (http://citeseer.ist.psu.edu/108383.html)" by Cai-Dong Wang, Hiroaki Takada and Ken Sakamura
- Article " Hardware Support for Priority Inheritance (http://doi.ieeecomputersociety.org/10.1109/REAL.2003.1253271)" by Bilge E. S. Akgul, Vincent J. Mooney, Henrik Thane and Pramote Kuacharoen

Query Optimization and Indexing

Query optimization

Query optimization is a function of many relational database management systems. The **query optimizer** attempts to determine the most efficient way to execute a given query by considering the possible query plans.

Generally, the query optimizer cannot be accessed directly by users: once queries are submitted to database server, and parsed by the parser, they are then passed to the query optimizer where optimization occurs. However, some database engines allow guiding the query optimizer with hints.

A query is a request for information from a database. It can be as simple as "finding the address of a person with SS# 123-45-6789," or more complex like "finding the average salary of all the employed married men in California between the ages 30 to 39, that earn less than their wives." Queries results are generated by accessing relevant database data and manipulating it in a way that yields the requested information. Since database structures are complex, in most cases, and especially for not-very-simple queries, the needed data for a query can be collected from a database by accessing it in different ways, through different data-structures, and in different orders. Each different way typically requires different processing time. Processing times of a same query may have large variance, from a fraction of a second to hours, depending on the way selected. The purpose of query optimization, which is an automated process, is to find the way to process a given query in minimum time. The large possible variance in time justifies performing query optimization, though finding the exact optimal way to execute a query, among all possibilities, is typically very complex, time consuming by itself, may be too costly, and often practically impossible. Thus query optimization typically tries to approximate the optimum by comparing several common-sense alternatives to provide in a reasonable time a "good enough" plan which typically does not deviate much from the best possible result.

General considerations

There is a trade-off between the amount of time spent figuring out the best query plan and the quality of the choice; the optimizer may not choose the best answer on its own. Different qualities of database management systems have different ways of balancing these two. Cost-based query optimizers evaluate the resource footprint of various query plans and use this as the basis for plan selection. These assign an estimated "cost" to each possible query plan, and choose the plan with the smallest cost. Costs are used to estimate the runtime cost of evaluating the query, in terms of the number of I/O operations required, CPU path length, amount of disk buffer space, disk storage service time, and interconnect usage between units of parallelism, and other factors determined from the data dictionary. The set of query plans examined is formed by examining the possible access paths (e.g., primary index access, secondary index access, full file scan) and various relational table join techniques (e.g., merge join, hash join, product join). The search space can become quite large depending on the complexity of the SQL query. There are two types of optimization. These consist of logical optimization—which generates a sequence of relational algebra to solve the query—and physical optimization—which is used to determine the means of carrying out each operation.

Implementation

Most query optimizers represent query plans as a tree of "plan nodes". A plan node encapsulates a single operation that is required to execute the query. The nodes are arranged as a tree, in which intermediate results flow from the bottom of the tree to the top. Each node has zero or more child nodes—those are nodes whose output is fed as input to the parent node. For example, a join node will have two child nodes, which represent the two join operands, whereas a sort node would have a single child node (the input to be sorted). The leaves of the tree are nodes which produce results by scanning the disk, for example by performing an index scan or a sequential scan.

Join ordering

The performance of a query plan is determined largely by the order in which the tables are joined. For example, when joining 3 tables A, B, C of size 10 rows, 10,000 rows, and 1,000,000 rows, respectively, a query plan that joins B and C first can take several orders-of-magnitude more time to execute than one that joins A and C first. Most query optimizers determine join order via a dynamic programming algorithm pioneered by IBM's System R database project [*citation needed*]. This algorithm works in two stages:

1. First, all ways to access each relation in the query are computed. Every relation in the query can be accessed via a sequential scan. If there is an index on a relation that can be used to answer a predicate in the query, an index scan can also be used. For each relation, the optimizer records the cheapest way to scan the relation, as well as the cheapest way to scan the relation that produces records in a particular sorted order.
2. The optimizer then considers combining each pair of relations for which a join condition exists. For each pair, the optimizer will consider the available join algorithms implemented by the DBMS. It will preserve the cheapest way to join each pair of relations, in addition to the cheapest way to join each pair of relations that produces its output according to a particular sort order.
3. Then all three-relation query plans are computed, by joining each two-relation plan produced by the previous phase with the remaining relations in the query.

sort order can avoid a redundant sort operation later on in processing the query. Second, a particular sort order can speed up a subsequent join because it clusters the data in a particular way.

Query planning for nested SQL queries

A SQL query to a modern relational DBMS does more than just selections and joins. In particular, SQL queries often nest several layers of SPJ blocks (Select-Project-Join), by means of group by, exists, and not exists operators. In some cases such nested SQL queries can be flattened into a select-project-join query, but not always. Query plans for nested SQL queries can also be chosen using the same dynamic programming algorithm as used for join ordering, but this can lead to an enormous escalation in query optimization time. So some database management systems use an alternative rule-based approach that uses a query graph model.

Cost estimation

One of the hardest problems in query optimization is to accurately estimate the costs of alternative query plans. Optimizers cost query plans using a mathematical model of query execution costs that relies heavily on estimates of the cardinality, or number of tuples, flowing through each edge in a query plan. Cardinality estimation in turn depends on estimates of the selection factor of predicates in the query. Traditionally, database systems estimate selectivities through fairly detailed statistics on the distribution of values in each column, such as histograms. This technique works well for estimation of selectivities of individual predicates. However many queries have conjunctions of predicates such as `select count(*) from R where R.make='Honda' and R.model='Accord'`. Query predicates are often highly correlated (for example, `model='Accord'` implies `make='Honda'`), and it is very hard to estimate the selectivity of the conjunct in general. Poor cardinality estimates and uncaught correlation are one of the main reasons why query optimizers pick poor query plans. This is

one reason why a database administrator should regularly update the database statistics, especially after major data loads/unloads.

References

- Chaudhuri, Surajit (1998). "An Overview of Query Optimization in Relational Systems" [1]. *Proceedings of the ACM Symposium on Principles of Database Systems*. pp. pages 34–43. doi:10.1145/275487.275492 [2].
- Ioannidis, Yannis (March 1996). "Query optimization" [3]. *ACM Computing Surveys* **28** (1): 121–123. doi:10.1145/234313.234367 [4].
- Selinger, P. G.; Astrahan, M. M.; Chamberlin, D. D.; Lorie, R. A.; Price, T. G. (1979). "Access Path Selection in a Relational Database Management System". *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*. pp. 23–34. doi:10.1145/582095.582099 [5]. ISBN 089791001X

References

[1] http://citeseer.ist.psu.edu/chaudhuri98overview.html
[2] http://dx.doi.org/10.1145%2F275487.275492
[3] http://citeseer.ist.psu.edu/487912.html
[4] http://dx.doi.org/10.1145%2F234313.234367
[5] http://dx.doi.org/10.1145%2F582095.582099

Query optimizer

Query optimization is a function of many relational database management systems. The **query optimizer** attempts to determine the most efficient way to execute a given query by considering the possible query plans.

Generally, the query optimizer cannot be accessed directly by users: once queries are submitted to database server, and parsed by the parser, they are then passed to the query optimizer where optimization occurs. However, some database engines allow guiding the query optimizer with hints.

A query is a request for information from a database. It can be as simple as "finding the address of a person with SS# 123-45-6789," or more complex like "finding the average salary of all the employed married men in California between the ages 30 to 39, that earn less than their wives." Queries results are generated by accessing relevant database data and manipulating it in a way that yields the requested information. Since database structures are complex, in most cases, and especially for not-very-simple queries, the needed data for a query can be collected from a database by accessing it in different ways, through different data-structures, and in different orders. Each different way typically requires different processing time. Processing times of a same query may have large variance, from a fraction of a second to hours, depending on the way selected. The purpose of query optimization, which is an automated process, is to find the way to process a given query in minimum time. The large possible variance in time justifies performing query optimization, though finding the exact optimal way to execute a query, among all possibilities, is typically very complex, time consuming by itself, may be too costly, and often practically impossible. Thus query optimization typically tries to approximate the optimum by comparing several common-sense alternatives to provide in a reasonable time a "good enough" plan which typically does not deviate much from the best possible result.

General considerations

There is a trade-off between the amount of time spent figuring out the best query plan and the quality of the choice; the optimizer may not choose the best answer on its own. Different qualities of database management systems have different ways of balancing these two. Cost-based query optimizers evaluate the resource footprint of various query plans and use this as the basis for plan selection. These assign an estimated "cost" to each possible query plan, and choose the plan with the smallest cost. Costs are used to estimate the runtime cost of evaluating the query, in terms of the number of I/O operations required, CPU path length, amount of disk buffer space, disk storage service time, and interconnect usage between units of parallelism, and other factors determined from the data dictionary. The set of query plans examined is formed by examining the possible access paths (e.g., primary index access, secondary index access, full file scan) and various relational table join techniques (e.g., merge join, hash join, product join). The search space can become quite large depending on the complexity of the SQL query. There are two types of optimization. These consist of logical optimization—which generates a sequence of relational algebra to solve the query—and physical optimization—which is used to determine the means of carrying out each operation.

Implementation

Most query optimizers represent query plans as a tree of "plan nodes". A plan node encapsulates a single operation that is required to execute the query. The nodes are arranged as a tree, in which intermediate results flow from the bottom of the tree to the top. Each node has zero or more child nodes—those are nodes whose output is fed as input to the parent node. For example, a join node will have two child nodes, which represent the two join operands, whereas a sort node would have a single child node (the input to be sorted). The leaves of the tree are nodes which produce results by scanning the disk, for example by performing an index scan or a sequential scan.

Join ordering

The performance of a query plan is determined largely by the order in which the tables are joined. For example, when joining 3 tables A, B, C of size 10 rows, 10,000 rows, and 1,000,000 rows, respectively, a query plan that joins B and C first can take several orders-of-magnitude more time to execute than one that joins A and C first. Most query optimizers determine join order via a dynamic programming algorithm pioneered by IBM's System R database project [*citation needed*]. This algorithm works in two stages:

1. First, all ways to access each relation in the query are computed. Every relation in the query can be accessed via a sequential scan. If there is an index on a relation that can be used to answer a predicate in the query, an index scan can also be used. For each relation, the optimizer records the cheapest way to scan the relation, as well as the cheapest way to scan the relation that produces records in a particular sorted order.
2. The optimizer then considers combining each pair of relations for which a join condition exists. For each pair, the optimizer will consider the available join algorithms implemented by the DBMS. It will preserve the cheapest way to join each pair of relations, in addition to the cheapest way to join each pair of relations that produces its output according to a particular sort order.
3. Then all three-relation query plans are computed, by joining each two-relation plan produced by the previous phase with the remaining relations in the query.

sort order can avoid a redundant sort operation later on in processing the query. Second, a particular sort order can speed up a subsequent join because it clusters the data in a particular way.

Query planning for nested SQL queries

A SQL query to a modern relational DBMS does more than just selections and joins. In particular, SQL queries often nest several layers of SPJ blocks (Select-Project-Join), by means of group by, exists, and not exists operators. In some cases such nested SQL queries can be flattened into a select-project-join query, but not always. Query plans for nested SQL queries can also be chosen using the same dynamic programming algorithm as used for join ordering, but this can lead to an enormous escalation in query optimization time. So some database management systems use an alternative rule-based approach that uses a query graph model.

Cost estimation

One of the hardest problems in query optimization is to accurately estimate the costs of alternative query plans. Optimizers cost query plans using a mathematical model of query execution costs that relies heavily on estimates of the cardinality, or number of tuples, flowing through each edge in a query plan. Cardinality estimation in turn depends on estimates of the selection factor of predicates in the query. Traditionally, database systems estimate selectivities through fairly detailed statistics on the distribution of values in each column, such as histograms. This technique works well for estimation of selectivities of individual predicates. However many queries have conjunctions of predicates such as `select count(*) from R where R.make='Honda' and R.model='Accord'` . Query predicates are often highly correlated (for example, `model='Accord'` implies `make='Honda'`), and it is very hard to estimate the selectivity of the conjunct in general. Poor cardinality estimates and uncaught correlation are one of the main reasons why query optimizers pick poor query plans. This is one reason why a database administrator should regularly update the database statistics, especially after major data loads/unloads.

References

- Chaudhuri, Surajit (1998). "An Overview of Query Optimization in Relational Systems" [1]. *Proceedings of the ACM Symposium on Principles of Database Systems*. pp. pages 34–43. doi:10.1145/275487.275492 [2].
- Ioannidis, Yannis (March 1996). "Query optimization" [3]. *ACM Computing Surveys* **28** (1): 121–123. doi:10.1145/234313.234367 [4].
- Selinger, P. G.; Astrahan, M. M.; Chamberlin, D. D.; Lorie, R. A.; Price, T. G. (1979). "Access Path Selection in a Relational Database Management System". *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*. pp. 23–34. doi:10.1145/582095.582099 [5]. ISBN 089791001X

Query plan

A **query plan** (or **query execution plan**) is an ordered set of steps used to access data in a SQL relational database management system. This is a specific case of the relational model concept of access plans.

Since SQL is declarative, there are typically a large number of alternative ways to execute a given query, with widely varying performance. When a query is submitted to the database, the query optimizer evaluates some of the different, correct possible plans for executing the query and returns what it considers the best alternative. Because query optimizers are imperfect, database users and administrators sometimes need to manually examine and tune the plans produced by the optimizer to get better performance.

Generating query plans

A given database management system may offer one or more mechanisms for returning the plan for a given query. Some packages feature tools which will generate a graphical representation of a query plan. Other tools allow a special mode to be set on the connection to cause the DBMS to return a textual description of the query plan. Another mechanism for retrieving the query plan involves querying a virtual database table after executing the query to be examined. In Oracle, for instance, this can be achieved using the EXPLAIN PLAN statement.

Graphical plans

The SQL Server Management Studio tool which ships with Microsoft SQL Server, for example, shows this graphical plan when executing this two-table join against a sample database:

```
SELECT *
FROM HumanResources.Employee AS e
    INNER JOIN Person.Contact AS c
    ON e.ContactID = c.ContactID
ORDER BY c.LastName
```

The UI allows exploration of various attributes of the operators involved in the query plan, including the operator type, the number of rows each operator consumes or produces, and the expected cost of each operator's work.

Textual plans

The textual plan given for the same query in the screenshot is shown here:

```
StmtText
----
  |--Sort(ORDER BY:([c].[LastName] ASC))
       |--Nested Loops(Inner Join, OUTER REFERENCES:([e].[ContactID],
[Expr1004]) WITH UNORDERED PREFETCH)
            |--Clustered Index
Scan(OBJECT:([AdventureWorks].[HumanResources].[Employee].[PK_Employee_EmployeeID]
 AS [e]))
            |--Clustered Index
Seek(OBJECT:([AdventureWorks].[Person].[Contact].[PK_Contact_ContactID]
 AS [c]),

SEEK:([c].[ContactID]=[AdventureWorks].[HumanResources].[Employee].[ContactID]
 as [e].[ContactID]) ORDERED FORWARD)
```

It indicates that the query engine will do a scan over the primary key index on the Employee table and a matching seek through the primary key index (the ContactID column) on the Contact table to find matching rows. The resulting rows from each side will be shown to a nested loops join operator, sorted, then returned as the result set to the connection.

In order to tune the query, the user must understand the different operators that the database may use, and which ones might be more efficient than others while still providing semantically correct query results.

Database tuning

Reviewing the query plan can present opportunities for new indexes or changes to existing indexes. It can also show that the database is not properly taking advantage of existing indexes (see query optimizer).

Query tuning

The query optimizer will not always choose the best query plan for a given query. In some databases the query plan can be reviewed, problems found, and then the query optimizer given hints on how to improve it. In other databases alternatives to express the same query (other queries that return the same results) can be tried. Some query tools can generate embedded hints in the query, for use by the optimizer.

Some databases like Oracle provide a Plan table for query tuning. This plan table will return the cost and time for executing a Query. In Oracle there are 2 optimization techniques:

1. CBO or Cost Based Optimization
2. RBO or Rule Based Optimization

The RBO is slowly being deprecated. For CBO to be used, all the tables referenced by the query must be analyzed. To analyze a table, a package DBMS_STATS can be made use of.

The others methods for query optimization include:

1. SQL Trace
2. Oracle Trace
3. TKPROF

- Video tutorial on how to perform SQL performance tuning with reference to Oracle [1]

References

[1] http://seeingwithc.org/sqltuning.html

Index (database)

A **database index** is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and the use of more storage space to maintain the extra copy of data. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.

An index is a copy of select columns of data from a table that can be searched very efficiently that also includes a low level disk block address or direct link to the complete row of data it was copied from. Some databases extend the power of indexing by allowing indices to be created on functions or expressions. For example, an index could be created on `upper(last_name)`, which would only store the upper case versions of the last_name field in the index. Another option sometimes supported is the use of "filtered" indices, where index entries are created only for those records that satisfy some conditional expression. A further aspect of flexibility is to permit indexing on user-defined functions, as well as expressions formed from an assortment of built-in functions.

Usage

Support for fast lookup

Most database software includes indexing technology that enables sub-linear time lookup to improve performance, as linear search is inefficient for large databases.

Suppose a database contains N data items and it is desired to retrieve one of them based on the value of one of the fields. A naïve implementation would retrieve and examine each item until a match was not found. A successful lookup would retrieve half the objects on average; an unsuccessful lookup all of them for each attempt. This means that the number of operations in the worst case is O(N) or linear time. Since databases commonly contain millions of objects and since lookup is a common operation, it is often desirable to improve on this performance.

An index is any data structure that improves the performance of lookup. There are many different data structures used for this purpose, and in fact a substantial proportion of the field of computer science is devoted to the design and analysis of index data structures. [*citation needed*] There are complex design trade-offs involving lookup performance, index size, and index update performance. Many index designs exhibit logarithmic (O(log(N))) lookup performance and in some applications it is possible to achieve flat (O(1)) performance.

Policing the database constraints

Indexes are used to police database constraints, such as UNIQUE, EXCLUSION, PRIMARY KEY and FOREIGN KEY. An index may be declared as UNIQUE which creates an implicit constraint on the underlying table. Database systems usually implicitly create an index on a set of columns declared PRIMARY KEY, and some are capable of using an already existing index to police this constraint. Many database systems require that both referencing and referenced sets of columns in a FOREIGN KEY constraint are indexed, thus improving performance of inserts, updates and deletes to the tables participating in the constraint.

Some database systems support EXCLUSION constraint which ensures that for a newly inserted or updated record a certain predicate would hold for no other record. This may be used to implement a UNIQUE constraint (with equality predicate) or more complex constraints, like ensuring that no overlapping time ranges or no intersecting geometry objects would be stored in the table. An index supporting fast searching for records satisfying the predicate is required to police such a constraint.[1]

Index architecture

Non-clustered

The data is present in arbitrary order, but the **logical ordering** is specified by the index. The data rows may be spread throughout the table regardless of the value of the indexed column or expression. The non-clustered index tree contains the index keys in sorted order, with the leaf level of the index containing the pointer to the record (page and the row number in the data page in page-organized engines; row offset in file-organized engines).

In a non-clustered index

- The physical order of the rows is not the same as the index order.
- The indexed columns are typically non-primary key columns used in JOIN, WHERE, and ORDER BY clauses.

There can be more than one non-clustered index on a database table.

Clustered

Clustering alters the data block into a certain distinct order to match the index, resulting in the row data being stored in order. Therefore, only one clustered index can be created on a given database table. Clustered indices can greatly increase overall speed of retrieval, but usually only where the data is accessed sequentially in the same or reverse order of the clustered index, or when a range of items is selected.

Since the physical records are in this sort order on disk, the next row item in the sequence is immediately before or after the last one, and so fewer data block reads are required. The primary feature of a clustered index is therefore the ordering of the physical data rows in accordance with the index blocks that point to them. Some databases separate the data and index blocks into separate files, others put two completely different data blocks within the same physical file(s).

Cluster

When multiple databases and multiple tables are joined, it's referred to as a **cluster** (not to be confused with clustered index described above). The records for the tables sharing the value of a cluster key shall be stored together in the same or nearby data blocks. This may improve the joins of these tables on the cluster key, since the matching records are stored together and less I/O is required to locate them.[2] The data layout in the tables which are parts of the cluster is defined by the cluster configuration. A cluster can be keyed with a B-Tree index or a hash table. The data block in which the table record will be stored is defined by the value of the cluster key.

Column order

The order in which columns are listed in the index definition is important. It is possible to retrieve a set of row identifiers using only the first indexed column. However, it is not possible or efficient (on most databases) to retrieve the set of row identifiers using only the second or greater indexed column.

For example, imagine a phone book that is organized by city first, then by last name, and then by first name. If you are given the city, you can easily extract the list of all phone numbers for that city. However, in this phone book it would be very tedious to find all the phone numbers for a given last name. You would have to look within each city's section for the entries with that last name. Some databases can do this, others just won't use the index.

In the phone book example with an composite index created on the columns (*city, last_name, first_name*), if we search by giving exact values for all the three fields, the search time will be minimal. But if we provide the values for *city* and *first_name* only, the search will use only the *city* field to retrieve all the records matched. And then a sequential lookup will be performed for checking the matching with *first_name*. So, to improve the performance, one must ensure that the index is created on the order of search columns.

Applications and limitations

Indices are useful for many applications but come with some limitations. Consider the following SQL statement: `SELECT first_name FROM people WHERE last_name = 'Smith';`. To process this statement without an index the database software must look at the last_name column on every row in the table (this is known as a full table scan). With an index the database simply follows the B-tree data structure until the Smith entry has been found; this is much less computationally expensive than a full table scan.

Consider this SQL statement: `SELECT email_address FROM customers WHERE email_address LIKE '%@yahoo.com';`. This query would yield an email address for every customer whose email address ends with "@yahoo.com", but even if the email_address column has been indexed the database must perform a full index scan. This is because the index is built with the assumption that words go from left to right. With a wildcard at the beginning of the search-term, the database software is unable to use the underlying B-tree data structure (in other words, the WHERE-clause is *not sargable*). This problem can be solved through the addition of another index created on `reverse(email_address)` and a SQL query like this: `SELECT email_address FROM customers WHERE reverse(email_address) LIKE reverse('%@yahoo.com');`. This puts the wild-card at the right-most part of the query (now moc.oohay@%) which the index on reverse(email_address) can satisfy.

But when the wildcard characters are used on both sides of the search word as "%yahoo.com%", the index available on this field will not be used. Rather only a sequential search will be performed which will take O(N) time. So, index must be available on the columns on which the lookup is performed.

Types of indexes

Bitmap index

A bitmap index is a special kind of index that stores the bulk of its data as bit arrays (bitmaps) and answers most queries by performing bitwise logical operations on these bitmaps. The most commonly used indexes, such as B+trees, are most efficient if the values they index do not repeat or repeat a smaller number of times. In contrast, the bitmap index is designed for cases where the values of a variable repeat very frequently. For example, the gender field in a customer database usually contains two distinct values: male or female. For such variables, the bitmap index can have a significant performance advantage over the commonly used trees.

Dense index

A dense index in databases is a file with pairs of keys and pointers for every record in the data file. Every key in this file is associated with a particular pointer to *a record* in the sorted data file. In clustered indices with duplicate keys, the dense index points *to the first record* with that key.[3]

Sparse index

A sparse index in databases is a file with pairs of keys and pointers for every block in the data file. Every key in this file is associated with a particular pointer *to the block* in the sorted data file. In clustered indices with duplicate keys, the sparse index points *to the lowest search key* in each block.

Reverse index

A reverse key index reverses the key value before entering it in the index. E.g., the value 24538 becomes 83542 in the index. Reversing the key value is particularly useful for indexing data such as sequence numbers, where new key values monotonically increase.

Index implementations

Indices can be implemented using a variety of data structures. Popular indices include balanced trees, B+ trees and hashes.

In Microsoft SQL Server, the leaf node of the clustered index corresponds to the actual data, not simply a pointer to data that resides elsewhere, as is the case with a non-clustered index. Each relation can have a single clustered index and many unclustered indices.

Index concurrency control

An index is typically being accessed concurrently by several transactions and processes, and thus needs concurrency control. While in principle indexes can utilize the common database concurrency control methods, specialized concurrency control methods for indexes exist, which are applied in conjunction with the common methods for a substantial performance gain.

Covering index

In most cases, an index is used to quickly locate the data record(s) from which the required data is read. In other words, the index is only used to locate data records in the table and not to return data.

A covering index is a special case where the index itself contains the required data field(s) and can return the data.

Consider the following table (other fields omitted):

ID	Name	Other Fields
12	Plug	...
13	Lamp	...
14	Fuse	...

To find the Name for ID 13, an index on (ID) will be useful, but the record must still be read to get the Name. However, an index on (ID, Name) contains the required data field and eliminates the need to look up the record.

A covering index can dramatically speed up data retrieval but may itself be large due to the additional keys, which slow down data insertion & update. To reduce such index size, some systems allow non-key fields to be included in the index. Non-key fields are not themselves part of the index ordering but only included at the leaf level, allowing for a covering index with less overall index size.

Standardization

There is no standard about creating indexes because the ISO SQL Standard does not cover physical aspects. Indexes are one of the physical parts of database conception among others like storage (tablespace or filegroups). RDBMS vendors all give a CREATE INDEX syntax with some specific options which depends on functionalities they provide to customers.

References

[1] PostgreSQL 9.1.2 Documentation: CREATE TABLE (http://www.postgresql.org/docs/9.1/static/sql-createtable.html)

[2] Overview of Clusters (http://download.oracle.com/docs/cd/B12037_01/server.101/b10743/schema.htm#sthref1069) Oracle® Database Concepts 10g Release 1 (10.1)

[3] Database Systems: The Complete Book. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer D. Widom

Partial index

In databases, a **partial index**, also known as **filtered index** is an index which has some condition applied to it so that it includes a subset of rows in the table.

This allows the index to remain small, even though the table may be rather large, and have extreme selectivity.

Suppose you have a transaction table where entries start out with STATUS = 'A' (active), and then may pass through other statuses ('P' for pending, 'W' for "being worked on") before reaching a final status, 'F', at which point it is no longer likely to be processed again.

In PostgreSQL, a useful partial index might be defined as:

```
create index partial_status on txn_table (status)
where status in ('A', 'P', 'W');
```

This index would not bother storing any of the millions of rows that have reached "final" status, 'F', and would allow queries looking for transactions that still "need work" to efficiently search via this index.

Similarly, a partial index can be used to index only those rows where a column is not null, which will be of benefit when the column usually is null.

```
create index partial_object_update on object_table (updated_on)
where updated_on is not null;
```

This index would allow the following query to read only the updated tuples:

```
select * from object_table
where updated_on is not null
order by updated_on;
```

It is not necessary that the condition be the same as the index criterion; Stonebraker's paper below presents a number of examples with indexes similar to the following:

```
create index partial_salary on employee(age)
where salary > 2100;
```

Support

In SQL Server, this type of index is called a *filtered index*. Partial indexes have been supported in PostgreSQL since version 7.2, released in February 2002. SQLite supports partial indexes since version 3.8.0.

MySQL as of version 5.4 does not support partial indexes. In MySQL, the term "partial index" is sometimes used to refer to prefix indexes, where only a truncated prefix of each value is stored in the index. This is another technique for reducing index size.

References

External links

- The Case For Partial Indexes (http://db.cs.berkeley.edu/papers/ERL-M89-17.pdf)

Expression index

An **expression index** is a database index that is built on a generic expression, rather than on a list of columns. This allows indexes to be defined for common query conditions that depend on data in a table, but are not actually stored in that table.

A common use for an expression index is to support case-insensitive searching or constraints. For example, if a web site wants to make user names case-insensitive, but still preserve the case as originally entered by the user, a unique index can be created on the upper- or lower-case representation of the user name:

```
CREATE UNIQUE INDEX site_user__user_name_lower ON site_user( lower( user_name ) );
```

That will create a unique index on "lower(user_name)". Any queries that search on "lower(user_name)" could then make use of that index:

```
SELECT user_id FROM site_user WHERE lower(user_name) = lower('Decibel');
```

Reverse index

Database management systems provide multiple types of indexes to improve performance and data integrity across diverse application. Index types include b-trees, bitmaps, and r-trees.

In database management systems, a **reverse key index** strategy reverses the key value before entering it in the index.[1] E.g., the value 24538 becomes 83542 in the index. Reversing the key value is particularly useful for indexing data such as sequence numbers, where each new key value is greater than the prior value, i.e., values monotonically increase. Reverse key indexes have become particularly important in high volume transaction processing systems because they reduce contention for index blocks.

Creating data

Reversed key indexes use b-tree structures, but preprocess key values before inserting them. Simplifying, b-trees place similar values on a single index block, e.g., storing 24538 on the same block as 24539. This makes them efficient both for looking up a specific value and for finding values within a range. However if the application inserts values in sequence, each insert must have access to the newest block in the index in order to add the new value. If many users attempt to insert at the same time, they all must write to that block and have to get in line, slowing down the application. This is particularly a problem in clustered databases, which may require the block to be copied from one computer's memory to another's to allow the next user to perform their insert.

Reversing the key spreads similar new values across the entire index instead of concentrating them in any one leaf block. This means that 24538 appears on the same block as 14538 while 24539 goes to a different block, eliminating this cause of contention. (Since 14538 would have been created long before 24538, their inserts don't interfere with each other.)

Querying data

Reverse indexes are just as efficient as unreversed indexes for finding specific values, although they aren't helpful for range queries. Range queries are uncommon for artificial values such as sequence numbers. When searching the index, the query processor simply reverses the search target before looking it up.

Deleting data

Typically, applications delete data that is older on average before deleting newer data. Thus, data with lower sequence numbers generally go before those with higher values. As time passes, in standard b-trees, index blocks for lower values end up containing few values, with a commensurate increase in unused space, referred to as "rot". Rot not only wastes space, but slows query speeds, because a smaller fraction of a rotten index's blocks fit in memory at any one time. In a b-tree, if 14538 gets deleted, its index space remains empty. In a reverse index, if 14538 goes before 24538 arrives, 24538 can reuse 14538's space.

Footnotes

[1] http://richardfoote.wordpress.com/2008/01/14/introduction-to-reverse-key-indexes-part-i/

External links

- http://download.oracle.com/docs/cd/A87860_01/doc/paraserv.817/a76970/design.htm

Bitmap index

A **bitmap index** is a special kind of database index that uses bitmaps.

Bitmap indexes have traditionally been considered to work well for *low-cardinality columns*, which have a modest number of distinct values, either absolutely, or relative to the number of records that contain the data. The extreme case of low cardinality is Boolean data (e.g.,Does a resident in a city have internet access?), which has two values, True and False. Bitmap indexes use bit arrays (commonly called bitmaps) and answer queries by performing bitwise logical operations on these bitmaps. Bitmap indexes have a significant space and performance advantage over other structures for query of such data. Their drawback is they are less efficient than the traditional B-tree indexes for columns whose data is frequently updated: consequently, they are more often employed in read-only systems that are specialized for fast query - e.g., data warehouses, and generally unsuitable for online transaction processing applications. However, this drawback appears to apply only to their implementation in relational database management systems: certain non-relational DBMSs, notably Intersystems Cache, a hierarchical database, use bitmap indexes for low-cardinality columns in transactional systems.

Some researchers argue that Bitmap indexes are also useful for moderate or even high-cardinality data (e.g., unique-valued data) which is accessed in a read-only manner, and queries access multiple bitmap-indexed columns using the AND, OR or XOR operators extensively.[1]

Bitmap indexes are also useful in data warehousing applications for joining a large fact table to smaller dimension tables such as those arranged in a star schema.

Example

Continuing the internet access example, a bitmap index may be logically viewed as follows:

Identifier	HasInternet	Bitmaps	
		Y	N
1	Yes	1	0
2	No	0	1
3	No	0	1
4	Unspecified	0	0
5	Yes	1	0

On the left, Identifier refers to the unique number assigned to each resident, HasInternet is the data to be indexed, the content of the bitmap index is shown as two columns under the heading *bitmaps*. Each column in the left illustration is a *bitmap* in the bitmap index. In this case, there are two such bitmaps, one for "has internet" *Yes* and one for "has internet" *No*. It is easy to see that each bit in bitmap *Y* shows whether a particular row refers to a person who has internet access. This is the simplest form of bitmap index. Most columns will have more distinct values. For example, the sales amount is likely to have a much larger number of distinct values. Variations on the bitmap index can effectively index this data as well. We briefly review three such variations.

Note: many of the references cited here are reviewed at. For those who might be interested in experimenting with some of the ideas mentioned here, many of them are implemented in open source software such as FastBit,[2] the Lemur Bitmap Index C++ Library,[3] the Roaring Bitmap Java library,[4] the Apache Hive Data Warehouse system and LucidDB.

Compression

Software can compress each bitmap in a bitmap index to save space. There has been considerable amount of work on this subject. Bitmap compression algorithms typically employ run-length encoding, such as the Byte-aligned Bitmap Code, the Word-Aligned Hybrid code, the Partitioned Word-Aligned Hybrid (PWAH) compression, the Position List Word Aligned Hybrid, the Compressed Adaptive Index (COMPAX), Enhanced Word-Aligned Hybrid (EWAH) and the COmpressed 'N' Composable Integer SEt.[5] These compression methods require very little effort to compress and decompress. More importantly, bitmaps compressed with BBC, WAH, COMPAX, PLWAH, EWAH and CONCISE can directly participate in bitwise operations without decompression. This gives them considerable advantages over generic compression techniques such as LZ77. BBC compression and its derivatives are used in a commercial database management system. BBC is effective in both reducing index sizes and maintaining query performance. BBC encodes the bitmaps in bytes, while WAH encodes in words, better matching current CPUs. "On both synthetic data and real application data, the new word aligned schemes use only 50% more space, but perform logical operations on compressed data 12 times faster than BBC." PLWAH bitmaps were reported to take 50% of the storage space consumed by WAH bitmaps and offer up to 20% faster performance on logical operations. Similar considerations can be done for CONCISE and Enhanced Word-Aligned Hybrid.

The performance of schemes such as BBC, WAH, PLWAH, EWAH, COMPAX and CONCISE is dependent on the order of the rows. A simple lexicographical sort can divide the index size by 9 and make indexes several times faster. The larger the table, the more important it is to sort the rows. Reshuffling techniques have also been proposed to achieve the same results of sorting when indexing streaming data.

Encoding

Basic bitmap indexes use one bitmap for each distinct value. It is possible to reduce the number of bitmaps used by using a different encoding method. For example, it is possible to encode C distinct values using log(C) bitmaps with binary encoding.

This reduces the number of bitmaps, further saving space, but to answer any query, most of the bitmaps have to be accessed. This makes it potentially not as effective as scanning a vertical projection of the base data, also known as a materialized view or projection index. Finding the optimal encoding method that balances (arbitrary) query performance, index size and index maintenance remains a challenge.

Without considering compression, Chan and Ioannidis analyzed a class of multi-component encoding methods and came to the conclusion that two-component encoding sits at the kink of the performance vs. index size curve and therefore represents the best trade-off between index size and query performance.

Binning

For high-cardinality columns, it is useful to bin the values, where each bin covers multiple values and build the bitmaps to represent the values in each bin. This approach reduces the number of bitmaps used regardless of encoding method. However, binned indexes can only answer some queries without examining the base data. For example, if a bin covers the range from 0.1 to 0.2, then when the user asks for all values less than 0.15, all rows that fall in the bin are possible hits and have to be checked to verify whether they are actually less than 0.15. The process of checking the base data is known as the candidate check. In most cases, the time used by the candidate check is significantly longer than the time needed to work with the bitmap index. Therefore, binned indexes exhibit irregular performance. They can be very fast for some queries, but much slower if the query does not exactly match a bin.

History

The concept of bitmap index was first introduced by Professor Israel Spiegler and Rafi Maayan in their research "Storage and Retrieval Considerations of Binary Data Bases", published in 1985. The first commercial database product to implement a bitmap index was Computer Corporation of America's Model 204. Patrick O'Neil published a paper about this implementation in 1987. This implementation is a hybrid between the basic bitmap index (without compression) and the list of Row Identifiers (RID-list). Overall, the index is organized as a B+tree. When the column cardinality is low, each leaf node of the B-tree would contain long list of RIDs. In this case, it requires less space to represent the RID-lists as bitmaps. Since each bitmap represents one distinct value, this is the basic bitmap index. As the column cardinality increases, each bitmap becomes sparse and it may take more disk space to store the bitmaps than to store the same content as RID-lists. In this case, it switches to use the RID-lists, which makes it a B+tree index.

In-memory bitmaps

One of the strongest reasons for using bitmap indexes is that the intermediate results produced from them are also bitmaps and can be efficiently reused in further operations to answer more complex queries. Many programming languages support this as a bit array data structure. For example, Java has the `BitSet` [6] class.

Some database systems that do not offer persistent bitmap indexes use bitmaps internally to speed up query processing. For example, PostgreSQL versions 8.1 and later implement a "bitmap index scan" optimization to speed up arbitrarily complex logical operations between available indexes on a single table.

For tables with many columns, the total number of distinct indexes to satisfy all possible queries (with equality filtering conditions on either of the fields) grows very fast, being defined by this formula:

$$\mathbf{C}_n^{\left[\frac{n}{2}\right]} \equiv \frac{n!}{\left(n - \left[\frac{n}{2}\right]\right)! \left[\frac{n}{2}\right]!}.$$

A bitmap index scan combines expressions on different indexes, thus requiring only one index per column to support all possible queries on a table.

Applying this access strategy to B-tree indexes can also combine range queries on multiple columns. In this approach, a temporary in-memory bitmap is created with one bit for each row in the table (1 MiB can thus store over 8 million entries). Next, the results from each index are combined into the bitmap using bitwise operations. After all conditions are evaluated, the bitmap contains a "1" for rows that matched the expression. Finally, the bitmap is traversed and matching rows are retrieved. In addition to efficiently combining indexes, this also improves locality of reference of table accesses, because all rows are fetched sequentially from the main table. The internal bitmap is discarded after the query. If there are too many rows in the table to use 1 bit per row, a "lossy" bitmap is created instead, with a single bit per disk page. In this case, the bitmap is just used to determine which pages to fetch; the filter criteria are then applied to all rows in matching pages.

References

Notes

[1] Bitmap Index vs. B-tree Index: Which and When? (http://www.oracle.com/technetwork/articles/sharma-indexes-093638.html), Vivek Sharma, Oracle Technical Network.
[2] FastBit (http://codeforge.lbl.gov/projects/fastbit/)
[3] Lemur Bitmap Index C++ Library (http://code.google.com/p/lemurbitmapindex/)
[4] Roaring bitmaps (http://roaringbitmap.org/)
[5] Concise: Compressed 'n' Composable Integer Set (http://ricerca.mat.uniroma3.it/users/colanton/concise.html)
[6] http://download.oracle.com/javase/6/docs/api/java/util/BitSet.html

Bibliography

- O'Connell, S. (2005). *Advanced Databases Course Notes*. Southampton: University of Southampton
- O'Neil, P.; O'Neil, E. (2001). *Database Principles, Programming, and Performance*. San Francisco: Morgan Kaufmann Publishers
- Zaker, M.; Phon-Amnuaisuk, S.; Haw, S.C. (2008). "An Adequate Design for Large Data Warehouse Systems: Bitmap index versus B-tree index" (http://www.universitypress.org.uk/journals/cc/cc-21.pdf). *International Journal of Computers and Communications* **2** (2). Retrieved 2010-01-07

Inverted index

In computer science, an **inverted index** (also referred to as **postings file** or **inverted file**) is an index data structure storing a mapping from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents. The purpose of an inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database. The inverted file may be the database file itself, rather than its index. It is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines. Several significant general-purpose mainframe-based database management systems have used inverted list architectures, including ADABAS, DATACOM/DB, and Model 204.

There are two main variants of inverted indexes: A **record level inverted index** (or **inverted file index** or just **inverted file**) contains a list of references to documents for each word. A **word level inverted index** (or **full inverted index** or **inverted list**) additionally contains the positions of each word within a document. The latter form offers more functionality (like phrase searches), but needs more time and space to be created.

Example

Given the texts

```
T[0] = "it is what it is"
T[1] = "what is it"
T[2] = "it is a banana"
```

we have the following inverted file index (where the integers in the set notation brackets refer to the indexes (or keys) of the text symbols, `T[0]`, `T[1]` etc.):

```
"a":      {2}
"banana": {2}
"is":     {0, 1, 2}
"it":     {0, 1, 2}
"what":   {0, 1}
```

A term search for the terms `"what"`, `"is"` and `"it"` would give the set $\{0,1\} \cap \{0,1,2\} \cap \{0,1,2\} = \{0,1\}$.

With the same texts, we get the following full inverted index, where the pairs are document numbers and local word numbers. Like the document numbers, local word numbers also begin with zero. So, `"banana": {(2, 3)}` means the word "banana" is in the third document (`T[2]`), and it is the fourth word in that document (position 3).

```
"a":      {(2, 2)}
"banana": {(2, 3)}
"is":     {(0, 1), (0, 4), (1, 1), (2, 1)}
"it":     {(0, 0), (0, 3), (1, 2), (2, 0)}
"what":   {(0, 2), (1, 0)}
```

If we run a phrase search for `"what is it"` we get hits for all the words in both document 0 and 1. But the terms occur consecutively only in document 1.

Applications

The inverted index data structure is a central component of a typical search engine indexing algorithm. A goal of a search engine implementation is to optimize the speed of the query: find the documents where word X occurs. Once a forward index is developed, which stores lists of words per document, it is next inverted to develop an inverted index. Querying the forward index would require sequential iteration through each document and to each word to verify a matching document. The time, memory, and processing resources to perform such a query are not always technically realistic. Instead of listing the words per document in the forward index, the inverted index data structure is developed which lists the documents per word.

With the inverted index created, the query can now be resolved by jumping to the word id (via random access) in the inverted index.

In pre-computer times, concordances to important books were manually assembled. These were effectively inverted indexes with a small amount of accompanying commentary that required a tremendous amount of effort to produce.

In bioinformatics, inverted indexes are very important in the sequence assembly of short fragments of sequenced DNA. One way to find the source of a fragment is to search for it against a reference DNA sequence. A small number of mismatches (due to differences between the sequenced DNA and reference DNA, or errors) can be accounted for by dividing the fragment into smaller fragments—at least one subfragment is likely to match the reference DNA sequence. The matching requires constructing an inverted index of all substrings of a certain length from the reference DNA sequence. Since the human DNA contains more than 3 billion base pairs, and we need to store a DNA substring for every index, and a 32-bit integer for index itself, the storage requirement for such an inverted index would probably be in the tens of gigabytes.

Bibliography

- Knuth, D. E. (1997) [1973]. "6.5. Retrieval on Secondary Keys". *The Art of Computer Programming* (Third ed.). Reading, Massachusetts: Addison-Wesley. ISBN 0-201-89685-0.
- Zobel, Justin; Moffat, Alistair; Ramamohanarao, Kotagiri (December 1998). "Inverted files versus signature files for text indexing". *ACM Transactions on Database Systems* (New York: Association for Computing Machinery) **23** (4): pp. 453–490. doi:10.1145/296854.277632 [1].
- Zobel, Justin RMIT University, Australia; Moffat, Alistair The University of Melbourne, Australia (July 2006). "Inverted Files for Text Search Engines". *ACM Computing Surveys* (New York: Association for Computing Machinery) **38** (2): 6. doi:10.1145/1132956.1132959 [2].

- Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier (1999). *Modern information retrieval*. Reading, Massachusetts: Addison-Wesley Longman. p. 192. ISBN 0-201-39829-X.
- Luk, Robert; W. Lam (2007). "Efficient in-memory extensible inverted file". *Information Systems* **32** (5): 733–754. doi:10.1016/j.is.2006.06.001 [3].
- Salton, Gerard; Fox, Edward A.; Wu, Harry (1983). "Extended Boolean information retrieval". *Commun. ACM* (ACM) **26** (11): 1022. doi:10.1145/182.358466 [4].
- *Information Retrieval: Implementing and Evaluating Search Engines* [5]. Cambridge, Massachusetts: MIT Press. 2010. ISBN 978-0-262-02651-2.

References

- Knuth 1997, pp. 560–563 of section 6.5: *Retrieval on Secondary Keys*

[1] http://dx.doi.org/10.1145%2F296854.277632
[2] http://dx.doi.org/10.1145%2F1132956.1132959
[3] http://dx.doi.org/10.1016%2Fj.is.2006.06.001
[4] http://dx.doi.org/10.1145%2F182.358466
[5] http://www.ir.uwaterloo.ca/book/

External links

- NIST's Dictionary of Algorithms and Data Structures: inverted index (http://www.nist.gov/dads/HTML/invertedIndex.html)
- Managing Gigabytes for Java (http://mg4j.dsi.unimi.it) a free full-text search engine for large document collections written in Java.
- Lucene (http://lucene.apache.org/java/docs/) - Apache Lucene is a full-featured text search engine library written in Java.
- Sphinx Search (http://sphinxsearch.com/) - Open source high-performance, full-featured text search engine library used by craigslist and others employing an inverted index.
- Example implementations (http://rosettacode.org/wiki/Inverted_Index) on Rosetta Code
- Caltech Large Scale Image Search Toolbox (http://www.vision.caltech.edu/malaa/software/research/image-search/): a Matlab toolbox implementing Inverted File Bag-of-Words image search.

Sargable

In relational databases, a condition (or predicate) in a query is said to be **sargable** if the DBMS engine can take advantage of an index to speed up the execution of the query. The term is derived from a contraction of *Search ARGument ABLE*.

A query failing to be sargable is known as ***Non-Sargable*** query and has an effect in query time, so one of the steps in query optimization is convert them to be sargable.

The typical situation that will make a sql query non-sargable is to include in the WHERE clause a function operating on a column value. Note that the WHERE clause is not the only clause where sargability can matter; it can also have an effect on ORDER BY, GROUP BY and HAVING clauses. The SELECT clause, on the other hand, can contain non-sargable expressions without adversely affecting the performance.

- **Sargable operators**: =,>,<,>=,<=,BETWEEN,LIKE without leading %
- **Sargable operators** that rarely improve performance: <>,IN,OR,NOT IN, NOT EXISTS, NOT LIKE
- **Non-sargable operators**: LIKE with leading %

Rules of thumb

- Avoid functions using table values in a sql condition.
- Avoid non-sargable predicates and replace them with sargable equivalents.

Examples

- **Non-Sargable**: Select ... WHERE Year(date) = 2012
- **Sargable**: Select ... WHERE date >= '01-01-2012' AND date < '01-01-2013'
- **Non-Sargable**: Select ... WHERE isNull(FullName,'John Smith') = 'John Smith'
- **Sargable**: Select ... WHERE ((FullName = 'John Smith') OR (FullName IS NULL))
- **Non-Sargable**: Select ... WHERE SUBSTRING(DealerName,6) = 'Toyota'
- **Sargable**: Select ... WHERE DealerName Like 'Toyota%'
- **Non-Sargable**: Select ... WHERE DateDiff(mm,Date,GetDate()) >= 20
- **Sargable**: Select ... WHERE Date < DateAdd(mm,-20,GetDate())

References

- *SQL Performance Tuning* by Peter Gulutzan, Trudy Pelzer (Addison Wesley, 2002) ISBN 0-201-79169-2 (Chapter 2, *Simple "Searches"* [1])

References

[1] http://www.informit.com/articles/article.aspx?p=30247

V-optimal histograms

Histograms are most commonly used as visual representations of data. However, database systems use histograms to summarize data internally and provide size estimates for queries. These histograms are not presented to users or displayed visually, so a wider range of options are available for their construction. Simple or exotic histograms are defined by four parameters, Sort Value, Source Value, Partition Class and Partition Rule. The most basic histogram is the equi-width histogram, where each bucket represents the same range of values. That histogram would be defined as having a Sort Value of Value, a Source Value of Frequency, be in the Serial Partition Class and have a Partition Rule stating that all buckets have the same range.

V-optimal histograms are an example of a more "exotic" histogram. V-optimality is a Partition Rule which states that the bucket boundaries are to be placed as to minimize the cumulative weighted variance of the buckets. Implementation of this rule is a complex problem and construction of these histograms is also a complex process.

Definition

A v-optimal histogram is based on the concept of minimizing a quantity which is called the *weighted variance* in this context.[1] This is defined as

$$W = \sum_{j=1}^{J} n_j V_j \,,$$

where the histogram consists of J bins or buckets, n_j is the number of items contained in the jth bin and where V_j is the variance between the values associated with the items in the jth bin.

Examples

The following example will construct a V-optimal histogram having a Sort Value of Value, a Source Value of Frequency, and a Partition Class of Serial. In practice, almost all histograms used in research or commercial products are of the Serial class, meaning that sequential sort values are placed in either the same bucket, or sequential buckets. For example, values 1, 2, 3 and 4 will be in buckets 1 and 2, or buckets 1, 2 and 3, but never in buckets 1 and 3. That will be taken as an assumption in any further discussion.

Take a simple set of data, for example, a list of integers:

1, 3, 4, 7, 2, 8, 3, 6, 3, 6, 8, 2, 1, 6, 3, 5, 3, 4, 7, 2, 6, 7, 2

Compute the value and frequency pairs (1, 2), (2, 4), (3, 5), (4, 2), (5, 1), (6, 4), (7, 3), (8, 2)

Our V-optimal histogram will have two buckets. Since one bucket must end at the data point for 8, we must decide where to put the other bucket boundary. The V-optimality rule states that the cumulative weighted variance of the buckets must be minimized. We will look at two options and compute the cumulative variance of those options.

Option 1: Bucket 1 contains values 1 through 4. Bucket 2 contains values 5 through 8.

Bucket 1:
Average frequency 3.25
Weighted variance 2.28

Bucket 2:
Average frequency 2.5
Weighted variance 2.19

Sum of Weighted Variance 4.47

Option 2: Bucket 1 contains values 1 through 2. Bucket 2 contains values 3 through 8.

Bucket 1:
Average frequency 3
Weighted variance 1.41

Bucket 2:
Average frequency 2.83
Weighted variance 3.29

Sum of Weighted Variance 4.70

The first choice is better, so the histogram that would wind up being stored is Bucket 1: Range(1 - 4), Average Frequency 3.25 Bucket 2: Range(5 - 8), Average Frequency 2.5

Advantages of V-optimality vs. equi-width or equi-depth

V-optimal histograms do a better job of estimating the bucket contents. A histogram is an estimation of the base data, and any histogram will have errors. The partition rule used in VOptimal histograms attempts to have the smallest variance possible among the buckets, which provides for a smaller error. Research done by Poosala and Ionnaidis 1 [2] has demonstrated that the most accurate estimation of data is done with a VOptimal histogram using value as a sort parameter and frequency as a source parameter.

Disadvantages of V-optimality vs. equi-width or equi-depth

While the V-optimal histogram is more accurate, it does have drawbacks. It is a difficult structure to update. Any changes to the source parameter could potentially result in having to re-build the histogram entirely, rather than updating the existing histogram. An equi-width histogram does not have this problem. Equi-depth histograms will experience this issue to some degree, but because the equi-depth construction is simpler, there is a lower cost to maintain it. The difficulty in updating VOptimal histograms is an outgrowth of the difficulty involved in constructing these histograms.

Construction issues

The above example is a simple one. There are only 7 choices of bucket boundaries. One could compute the cumulative variance for all 7 options easily and choose the absolute best placement. However, as the range of values gets larger and the number of buckets gets larger, the set of possible histograms grows exponentially and it becomes a dauntingly complex problem to find the set of boundaries that provide the absolute minimum variance. A solution is to give up on finding the absolute best solution and attempt to find a good solution instead. By creating random solutions, using those as a starting point and improving upon them, one can find a solution that is a fair approximation of the "best" solution. One construction method used to get around this problem is the Iterative Improvement algorithm. Another is Simulated Annealing. The two may be combined in Two Phase Optimization, or 2PO. These algorithms are put forth in "Randomized Algorithms..." (cited below) as a method to optimize queries, but the general idea may be applied to construction of V-optimal Histograms.

Iterative improvement

Iterative Improvement (II) is a fairly naive greedy algorithm. Starting from a random state, iterative steps in many directions are considered. The step that offers the best improvement of cost (in this case Total Variance) is taken. The process is repeated until one settles at the local minimum, where no further improvement is possible. Applied to the construction of V-optimal histograms, the initial random state would be a set of values representing the bucket boundary placements. The iterative improvement steps would involve moving each boundary until it was at its local minimum, then moving to the next boundary and adjusting it accordingly.

Simulated Annealing

A basic explanation of Simulated Annealing is that it is a lot like II, only instead of taking the greedy step each time, it will sometimes accept a step that results in an increase in cost. In theory, SA will be less likely to stop at a very local minimum, and more likely to find a more global one. A useful piece of imagery is an "M" shaped graph, representing overall cost on the Y axis. If the initial state is on the "V" shaped part of the "M", II will settle into the high valley, the local minimum. Because SA will accept uphill moves, it is more likely to climb up the slope of the "V" and wind up at the foot of the "M", the global minimum.

Two Phase Optimization

Two Phase Optimization, or 2PO, combines the II and SA methods. II is run until a local minimum is reached, then SA is run on that solution in an attempt to find less obvious improvements.

Variations of V-optimal Histograms

The idea behind V-optimal histograms is to minimize the variance inside each bucket. In considering this, a thought occurs that the variance of any set with one member is 0. This is the idea behind "End-Biased" V-optimal Histograms. The value with the highest frequency is always placed in its own bucket. This ensures that the estimate for that value (which is likely to be the most frequently requested estimate, since it is the most frequent value) will always be accurate and also removes the value most likely to cause a high variance from the data set.

Another thought that might occur is that variance would be reduced if one were to sort by frequency, instead of value. This would naturally tend to place like values next to each other. Such a histogram can be constructed by using a Sort Value of Frequency and a Source Value of Frequency. At this point, however, the buckets must carry additional information indicating what data values are present in the bucket. These histograms have been shown to be less accurate, due to the additional layer of estimation required.

Notes

[1] Poosala at al. (1996)

[2] http://citeseer.ist.psu.edu/poosala96improved.html

References and external links

- Poosala, V.; Haas, P. J.; Ioannidis, Y. E.; Shekita, E. J. (1996). "Improved histograms for selectivity estimation of range predicates". *Proceedings of the 1996 ACM SIGMOD international conference on Management of data - SIGMOD '96*. p. 294. doi: 10.1145/233269.233342 (http://dx.doi.org/10.1145/233269.233342). ISBN 0897917944. Download PDF (http://www.cs.cmu.edu/~natassa/courses/15-823/current/papers/poosala96improved.pdf)
- Ioannidis, Y. E.; Poosala, V. (1995). "Balancing histogram optimality and practicality for query result size estimation". *Proceedings of the 1995 ACM SIGMOD international conference on Management of data - SIGMOD '95*. p. 233. doi: 10.1145/223784.223841 (http://dx.doi.org/10.1145/223784.223841). ISBN 0897917316. Download PDF (http://reference.kfupm.edu.sa/content/b/a/balancing_histogram_optimality_and_pract_46453.pdf)
- Ioannidis, Y. E.; Kang, Y. (1990). "Randomized algorithms for optimizing large join queries". *ACM SIGMOD Record* **19** (2): 312. doi: 10.1145/93605.98740 (http://dx.doi.org/10.1145/93605.98740). Download PDF (http://www-static.cc.gatech.edu/computing/Database/readinggroup/articles/p312-ioannidis.pdf)

Cardinality (SQL statements)

In SQL (Structured Query Language), the term **cardinality** refers to the uniqueness of data values contained in a particular column (attribute) of a database table. The lower the cardinality, the more duplicated elements in a column. Thus, a column with the lowest possible cardinality would have the same value for every row. SQL databases use cardinality to help determine the optimal query plan for a given query.

Values of Cardinality

When dealing with columnar value sets, there are 3 types of cardinality: high-cardinality, normal-cardinality, and low-cardinality.

High-cardinality refers to columns with values that are very uncommon or unique. High-cardinality column values are typically identification numbers, email addresses, or user names. An example of a data table column with high-cardinality would be a USERS table with a column named USER_ID. This column would contain unique values of 1-*n*. Each time a new user is created in the USERS table, a new number would be created in the USER_ID column to identify them uniquely. Since the values held in the USER_ID column are unique, this column's cardinality type would be referred to as high-cardinality.

Normal-cardinality refers to columns with values that are somewhat uncommon. Normal-cardinality column values are typically names, street addresses, or vehicle types. An example of a data table column with normal-cardinality would be a CUSTOMER table with a column named LAST_NAME, containing the last names of customers. While some people have common last names, such as Smith, others have uncommon last names. Therefore, an examination of all of the values held in the LAST_NAME column would show "clumps" of names in some places (e.g.: a lot of Smith's) surrounded on both sides by a long series of unique values. Since there is a variety of possible values held in this column, its cardinality type would be referred to as normal-cardinality.

Low-cardinality refers to columns with few unique values. Low-cardinality column values are typically status flags, Boolean values, or major classifications such as gender. An example of a data table column with low-cardinality would be a CUSTOMER table with a column named NEW_CUSTOMER. This column would contain only 2 distinct values: Y or N, denoting whether the customer was new or not. Since there are only 2 possible values held in this column, its cardinality type would be referred to as low-cardinality.

Online aggregation

Online aggregation is a technique for improving the interactive behavior of database systems processing expensive analytical queries. Almost all database operations are performed in batch mode, i.e. the user issues a query and waits till the database has finished processing the entire query. On the contrary, using online aggregation, the user gets estimates of an aggregate query in an online fashion as soon as the query is issued. For example, if the final answer is 1000, after k seconds, the user gets the estimates in form of a confidence interval like [990, 1020] with 95% probability. This confidence keeps on shrinking as the system gets more and more samples.

Online aggregation was proposed in 1997 by Hellerstein, Haas and Wang for group-by aggregation queries over a single table. Later, the authors showed how to evaluate joins in an online fashion. In 2007, Jermaine et al. designed and implemented a prototype database system called Database-Online (or DBO) that computes group-by aggregate query over multiple tables in an online and more importantly in a scalable fashion. All the approaches for online aggregation use random sampling, which is non-trivial in a distributed environment due to inspection paradox of renewal reward theory. In 2011, Pansare et al. proposed a Bayesian model to deal with the inspection paradox and implemented online aggregation for a MapReduce-like environment.

References

Distributed DB

Very large database

A **very large database**, or **VLDB**, is a database that contains an extremely high number of tuples (database rows), or occupies an extremely large physical filesystem storage space. The most common definition of VLDB is a database that occupies more than 1 terabyte or contains several billion rows, although naturally this definition changes over time.[*citation needed*]

Since the year 2011, this term is now referred to as big data by industry.

References

Big data

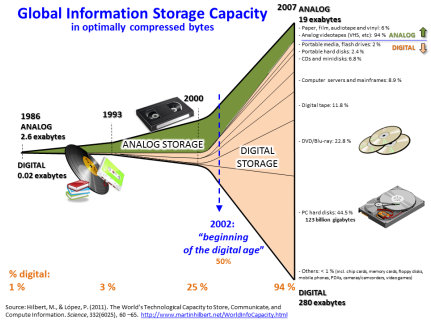
Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, curation, storage, search, sharing, transfer, analysis and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions."[1]

A visualization created by IBM of Wikipedia edits. At multiple terabytes in size, the text and images of Wikipedia are a classic example of big data.

As of 2012[2], limits on the size of data sets that are feasible to process in a reasonable amount of time were on the order of exabytes of data. Scientists regularly encounter limitations due to large data sets in many areas, including meteorology, genomics, connectomics, complex physics simulations, and biological and environmental research. The limitations also affect Internet search, finance and business informatics. Data sets grow in size in part because they are increasingly being gathered by ubiquitous information-sensing mobile devices, aerial sensory technologies (remote sensing), software logs, cameras, microphones, radio-frequency identification readers, and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012[2], every day 2.5 exabytes (2.5×10^{18}) of data were created. The challenge for large enterprises is determining who should own big data initiatives that straddle the entire organization.[3]

Big data is difficult to work with using most relational database management systems and desktop statistics and visualization packages, requiring instead "massively parallel software running on tens,

hundreds, or even thousands of servers". What is considered "big data" varies depending on the capabilities of the organization managing the set, and on the capabilities of the applications that are traditionally used to process and analyze the data set in its domain. "For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration."



Growth of and Digitization of Global Information Storage Capacity; source: http://www.martinhilbert.net/WorldInfoCapacity.html

Definition

Big Data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process the data within a tolerable elapsed time.[4] Big data sizes are a constantly moving target, as of 2012[2] ranging from a few dozen terabytes to many petabytes of data in a single data set.

In a 2001 research report and related lectures, META Group (now Gartner) analyst Doug Laney defined data growth challenges and opportunities as being three-dimensional, i.e. increasing volume (amount of data), velocity (speed of data in and out), and variety (range of data types and sources). Gartner, and now much of the industry, continue to use this "3Vs" model for describing big data. In 2012, Gartner updated its definition as follows: "Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization." Additionally, a new V "Veracity" is added by some organizations to describe it.

If Gartner's definition (the 3Vs) is still widely used, the growing maturity of the concept fosters a more sound difference between big data and Business Intelligence, regarding data and their use:

- Business Intelligence uses descriptive statistics with data with high information density to measure things, detect trends etc.;
- Big data uses inductive statistics and concepts from nonlinear system identification [5] to infer laws (regressions, nonlinear relationships, and causal effects) from large data sets [6] to reveal relationships, dependencies, and to perform predictions of outcomes and behaviors.[7]

Big science

The Large Hadron Collider experiments represent about 150 million sensors delivering data 40 million times per second. There are nearly 600 million collisions per second. After filtering and refraining from recording more than 99.999% of these streams, there are 100 collisions of interest per second.

- As a result, only working with less than 0.001% of the sensor stream data, the data flow from all four LHC experiments represents 25 petabytes annual rate before replication (as of 2012). This becomes nearly 200 petabytes after replication.
- If all sensor data were to be recorded in LHC, the data flow would be extremely hard to work with. The data flow would exceed 150 million petabytes annual rate, or nearly 500 exabytes per day, before replication. To put the number in perspective, this is equivalent to 500 quintillion (5×10^{20}) bytes per day, almost 200 times higher than all the other sources combined in the world.

Science and research

- When the Sloan Digital Sky Survey (SDSS) began collecting astronomical data in 2000, it amassed more in its first few weeks than all data collected in the history of astronomy. Continuing at a rate of about 200 GB per night, SDSS has amassed more than 140 terabytes of information. When the Large Synoptic Survey Telescope, successor to SDSS, comes online in 2016 it is anticipated to acquire that amount of data every five days.
- Decoding the human genome originally took 10 years to process, now it can be achieved in less than a week : the DNA sequencers have divided the sequencing cost by 10,000 in the last ten years, which is 100 times faster than the reduction in cost predicted by Moore's Law.[8]
- Computational social science ― Tobias Preis *et al.* used Google Trends data to demonstrate that Internet users from countries with a higher per capita gross domestic product (GDP) are more likely to search for information about the future than information about the past. The findings suggest there may be a link between online behaviour and real-world economic indicators. The authors of the study examined Google queries logs made by ratio of the volume of searches for the coming year ('2011') to the volume of searches for the previous year ('2009'), which they call the 'future orientation index'. They compared the future orientation index to the per capita GDP of each country and found a strong tendency for countries in which Google users enquire more about the future to exhibit a higher GDP. The results hint that there may potentially be a relationship between the economic success of a country and the information-seeking behavior of its citizens captured in big data.
- The [[Nasa|NASA [9]] Center for Climate Simulation (NCCS)] stores 32 petabytes of climate observations and simulations on the Discover supercomputing cluster.
- Tobias Preis and his colleagues Helen Susannah Moat and H. Eugene Stanley introduced a method to identify online precursors for stock market moves, using trading strategies based on search volume data provided by Google Trends. Their analysis of Google search volume for 98 terms of varying financial relevance, published in *Scientific Reports*, suggests that increases in search volume for financially relevant search terms tend to precede large losses in financial markets.

Government

- In 2012, the Obama administration announced the Big Data Research and Development Initiative, which explored how big data could be used to address important problems faced by the government. The initiative was composed of 84 different big data programs spread across six departments.
- Big data analysis played a large role in Barack Obama's successful 2012 re-election campaign.
- The United States Federal Government owns six of the ten most powerful supercomputers in the world.
- The Utah Data Center is a data center currently being constructed by the United States National Security Agency. When finished, the facility will be able to handle a large amount of information collected by the NSA over the Internet. The exact amount of storage space is unknown, but more recent sources claim it will be on the order of a few Exabytes.

Private sector

Bus wrapped with SAP Big data parked outside IDF13.

- eBay.com uses two data warehouses at 7.5 petabytes and 40PB as well as a 40PB Hadoop cluster for search, consumer recommendations, and merchandising. Inside eBay's 90PB data warehouse [10]
- Amazon.com handles millions of back-end operations every day, as well as queries from more than half a million third-party sellers. The core technology that keeps Amazon running is Linux-based and as of 2005 they had the world's three largest Linux databases, with capacities of 7.8 TB, 18.5 TB, and 24.7 TB.
- Walmart handles more than 1 million customer transactions every hour, which is imported into databases estimated to contain more than 2.5 petabytes (2560 terabytes) of data – the equivalent of 167 times the information contained in all the books in the US Library of Congress.
- Facebook handles 50 billion photos from its user base.
- FICO Falcon Credit Card Fraud Detection System protects 2.1 billion active accounts world-wide.
- The volume of business data worldwide, across all companies, doubles every 1.2 years, according to estimates.[11]
- Windermere Real Estate uses anonymous GPS signals from nearly 100 million drivers to help new home buyers determine their typical drive times to and from work throughout various times of the day.

International development

Research on the effective usage of information and communication technologies for development (also known as ICT4D) suggests that big data technology can make important contributions but also present unique challenges to International development.[12][13] Advancements in big data analysis offer cost-effective opportunities to improve decision-making in critical development areas such as health care, employment, economic productivity, crime, security, and natural disaster and resource management. However, longstanding challenges for developing regions such as inadequate technological infrastructure and economic and human resource scarcity exacerbate existing concerns with big data such as privacy, imperfect methodology, and interoperability issues.

Market

"Big Data" has increased the demand of information management specialists in that Software AG, Oracle Corporation, IBM, Microsoft, SAP, EMC, HP and Dell have spent more than $15 billion on software firms only specializing in data management and analytics. In 2010, this industry on its own was worth more than $100 billion and was growing at almost 10 percent a year: about twice as fast as the software business as a whole.

Developed economies make increasing use of data-intensive technologies. There are 4.6 billion mobile-phone subscriptions worldwide and there are between 1 billion and 2 billion people accessing the internet. Between 1990 and 2005, more than 1 billion people worldwide entered the middle class which means more and more people who gain money will become more literate which in turn leads to information growth. The world's effective capacity to exchange information through telecommunication networks was 281 petabytes in 1986, 471 petabytes in 1993, 2.2 exabytes in 2000, 65 exabytes in 2007 and it is predicted that the amount of traffic flowing over the internet will reach 667 exabytes annually by 2013.

Big Data Softwares:

- Hadoop - Apache Foundation
- MongoDB - MongoDB, Inc
- Splunk - Splunk Inc

Architecture

In 2004, Google published a paper on a process called MapReduce that used such an architecture. MapReduce framework provides a parallel processing model and associated implementation to process huge amount of data. With MapReduce, queries are split and distributed across parallel nodes and processed in parallel (the Map step). The results are then gathered and delivered (the Reduce step). The framework was incredibly successful,[14] so others wanted to replicate the algorithm. Therefore, an implementation of MapReduce framework was adopted by an Apache open source project named Hadoop.[15]

MIKE2.0 is an open approach to information management that acknowledges the need for revisions due to big data implications in an article title Big Data Solution Offering. The methodology addresses handling big data in terms of useful permutations of data sources, complexity in interrelationships, and difficulty in deleting (or modifying) individual records.

Recent studies show the use of multiple layer architecture as an option to Big Data. The Distributed Parallel architecture distributes data across multiple processing units and parallel processing units provide data much faster, by improving processing speeds. This type of architecture inserts data into parallel DBMS, which implements the use of MapReduce and Hadoop frameworks. This type of framework looks to make the processing power transparent to the end user by using a front end application server.

Technologies

Big data requires exceptional technologies to efficiently process large quantities of data within tolerable elapsed times. A 2011 McKinsey report suggests suitable technologies include A/B testing, crowdsourcing, data fusion and integration, genetic algorithms, machine learning, natural language processing, signal processing, simulation, time series analysis and visualisation. Multidimensional big data can also be represented as tensors, which can be more efficiently handled by tensor-based computation, such as multilinear subspace learning. Additional technologies being applied to big data include massively parallel-processing (MPP) databases, search-based applications, data-mining grids, distributed file systems, distributed databases, cloud based infrastructure (applications, storage and computing resources) and the Internet.[*citation needed*]

Some but not all MPP relational databases have the ability to store and manage petabytes of data. Implicit is the ability to load, monitor, back up, and optimize the use of the large data tables in the RDBMS.

DARPA's Topological Data Analysis program seeks the fundamental structure of massive data sets and in 2008 the technology went public with the launch of a company called Ayasdi.

The practitioners of big data analytics processes are generally hostile to slower shared storage, preferring direct-attached storage (DAS) in its various forms from solid state drive (SSD) to high capacity SATA disk buried inside parallel processing nodes. The perception of shared storage architectures—Storage area network (SAN) and Network-attached storage (NAS) —is that they are relatively slow, complex, and expensive. These qualities are not consistent with big data analytics systems that thrive on system performance, commodity infrastructure, and low cost.

Real or near-real time information delivery is one of the defining characteristics of big data analytics. Latency is therefore avoided whenever and wherever possible. Data in memory is good—data on spinning disk at the other end of a FC SAN connection is not. The cost of a SAN at the scale needed for analytics applications is very much higher than other storage techniques.

There are advantages as well as disadvantages to shared storage in big data analytics, but big data analytics practitioners as of 2011[2] did not favour it.

Research activities

Encrypted search & cluster formation in Big Data is set for the demonstration in March 2014 at American Society of Engineering Education. Gautam Siwach engaged at Tackling the challenges of Big Data by MIT Computer Science and Artificial Intelligence Laboratory and Dr. Amir Esmailpour at UNH Research Group investigated the key features of big data as formation of clusters and their interconnections. They focused on the security of big data and the actual orientation of the term towards the presence of different type of data in an encrypted form at cloud interface by providing the raw definitions and real time examples within the technology. Moreover, they proposed an approach for identifying the encoding technique to advance towards an expedited search over encrypted text leading to the security enhancements in big data. [16]

In March 2012, The White House announced a national "Big Data Initiative" that consisted of six Federal departments and agencies committing more than $200 million to big data research projects.

The initiative included a National Science Foundation "Expeditions in Computing" grant of $10 million over 5 years to the AMPLab at the University of California, Berkeley. The AMPLab also received funds from DARPA, and over a dozen industrial sponsors and uses big data to attack a wide range of problems from predicting traffic congestion to fighting cancer.

The White House Big Data Initiative also included a commitment by the Department of Energy to provide $25 million in funding over 5 years to establish the Scalable Data Management, Analysis and Visualization (SDAV) Institute, led by the Energy Department's Lawrence Berkeley National Laboratory. The SDAV Institute aims to bring together the expertise of six national laboratories and seven universities to develop new tools to help scientists manage and visualize data on the Department's supercomputers.

The U.S. state of Massachusetts announced the Massachusetts Big Data Initiative in May 2012, which provides funding from the state government and private companies to a variety of research institutions. The Massachusetts Institute of Technology hosts the Intel Science and Technology Center for Big Data in the MIT Computer Science and Artificial Intelligence Laboratory, combining government, corporate, and institutional funding and research efforts.

The European Commission is funding a 2-year-long Big Data Public Private Forum [17] through their Seventh Framework Program to engage companies, academics and other stakeholders in discussing big data issues. The project aims to define a strategy in terms of research and innovation to guide supporting actions from the European Commission in the successful implementation of the Big Data economy. Outcomes of this project will be used as input for Horizon 2020 [18], their next framework program.

The IBM sponsored 37th annual "Battle of the Brains" student Big Data championship will be held in July 2013. The inaugural professional 2014 Big Data World Championship is to be held in Dallas, Texas.

In order to make manufacturing more competitive in the United States (and globe), there is a need to integrate more American ingenuity and innovation into manufacturing ; Therefore, National Science Foundation has granted the Industry University cooperative research center for Intelligent Maintenance Systems (IMS) [19] at university of Cincinnati to focus on developing advanced predictive tools and techniques to be applicable in Big Data environment. In May 2013, IMS Center held an industry advisory board meeting focusing on Big Data where presenters from various industrial companies discussed their concerns, issues and future goals in Big Data environment.

Applications

Manufacturing

Based on TCS 2013 Global Trend Study, huge improvements in supply planning and boost product quality is the greatest benefit of Big Data for manufacturing. Big Data provides an infrastructure for transparency in manufacturing industry, which is the ability to unravel uncertainties such as inconsistent component performance and availability. Predictive manufacturing as an applicable approach toward near-zero downtime and transparency requires vast amount of data and advanced prediction tools for a systematic process of data into useful information. A conceptual framework of predictive manufacturing begins with data acquisition where different type of sensory data is available to acquire such as acoustics, vibration, pressure, current, voltage and controller data. Vast amount of sensory data in addition to historical data construct the “Big Data” in manufacturing. The generated Big Data acts as the input into predictive tools and preventive strategies such as Prognostics and Health Management (PHM).

Critique

Critiques of the big data paradigm come in two flavors, those that question the implications of the approach itself, and those that question the way it is currently done.

Critiques of the big data paradigm

Cartoon critical of big data application

"A crucial problem is that we do not know much about the underlying empirical micro-processes that lead to the emergence of the[se] typical network characteristics of Big Data". In their critique, Snijders, Matzat, and Reips point out that often very strong assumptions are made about mathematical properties that may not at all reflect what is really going on at the level of micro-processes. Mark Graham has leveled broad critiques at Chris Anderson's assertion that big data will spell the end of theory: focusing in particular on the notion that big data will always need to be contextualized in their social, economic and political contexts. Even as companies invest eight- and nine-figure sums to derive insight from information streaming in from suppliers and customers, less than 40% of employees have sufficiently mature processes and skills to do so. To overcome this insight deficit, "big data", no matter how comprehensive or well analyzed, needs to be complemented by "big judgment", according to an article in the Harvard Business Review.

Much in the same line, it has been pointed out that the decisions based on the analysis of big data are inevitably "informed by the world as it was in the past, or, at best, as it currently is".[] Fed by a large number of data on past experiences, algorithms can predict future development if the future is similar to the past. If the systems dynamics of the future change, the past can say little about the future. For this, it would be necessary to have a thorough understanding of the systems dynamic, which implies theory.[20] As a response to this critique it has been suggested to combine big data approaches with computer simulations, such as agent-based models, for example. Agent-based models are increasingly getting better in predicting the outcome of social complexities of even unknown future scenarios through computer simulations that are based on a collection of mutually interdependent algorithms.[21][22] In addition, use of multivariate methods that probe for the latent structure of the data, such as factor analysis and cluster analysis, have proven useful as analytic approaches that go well beyond the bi-variate approaches (cross-tabs) typically employed with smaller data sets.

In Health and biology, conventional scientific approaches are based on experimentation. For these approaches, the limiting factor are the relevant data that can confirm or refute the initial hypothesis.[23] A new postulate is accepted now in biosciences : the information provided by the data in huge volumes (omics) without prior hypothesis is complementary and sometimes necessary to conventional approaches based on experimentation. In the massive approaches it is the formulation of a relevant hypothesis to explain the data that is the limiting factor. The search logic is reversed and the limits of induction ("Glory of Science and Philosophy scandal", C. D. Broad, 1926) to be considered.

Privacy advocates are concerned about the threat to privacy represented by increasing storage and integration of personally identifiable information; expert panels have released various policy recommendations to conform practice to expectations of privacy.[24][25]

Critiques of big data execution

Researcher Danah Boyd has raised concerns about the use of big data in science neglecting principles such as choosing a representative sample by being too concerned about actually handling the huge amounts of data. This approach may lead to results bias in one way or another. Integration across heterogeneous data resources — some that might be considered "big data" and others not — presents formidable logistical as well as analytical challenges, but many researchers argue that such integrations are likely to represent the most promising new frontiers in science.

References

[1] by Cat Casey and Alejandra Perez

[2] http://en.wikipedia.org/w/index.php?title=Big_data&action=edit

[3] Oracle and FSN, "Mastering Big Data: CFO Strategies to Transform Insight into Opportunity" (http://www.fsn.co.uk/channel_bi_bpm_cpm/mastering_big_data_cfo_strategies_to_transform_insight_into_opportunity#.UO2Ac-TTuys), December 2012

[4] Snijders, C., Matzat, U., & Reips, U.-D. (2012). 'Big Data': Big gaps of knowledge in the field of Internet. *International Journal of Internet Science, 7*, 1-5. http://www.ijis.net/ijis7_1/ijis7_1_editorial.html

[5] Billings S.A. "Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains". Wiley, 2013

[6] Delort P., Big data Paris 2013 http://www.andsi.fr/tag/dsi-big-data/

[7] Delort P., Big Data car Low-Density Data ? La faible densité en information comme facteur discriminant http://lecercle.lesechos.fr/entrepreneur/tendances-innovation/221169222/big-data-low-density-data-faible-densite-information-com

[8] Delort P., OECD ICCP Technology Foresight Forum, 2012. http://www.oecd.org/sti/ieconomy/Session_3_Delort.pdf#page=6

[9] http://www.nasa.gov/centers/goddard/news/releases/2010/10-051.html

[10] http://www.itnews.com.au/News/342615,inside-ebay8217s-90pb-data-warehouse.aspx

[11] Leading Priorities for Big Data for Business and IT (http://www.statista.com/statistics/280444/global-leading-priorities-for-big-data-according-to-business-and-it-executives/). eMarketer. October 2013. Retrieved January 2014.

[12] UN GLobal Pulse (2012). Big Data for Development: Opportunities and Challenges (White p. by Letouzé, E.). New York: United Nations. Retrieved from http://www.unglobalpulse.org/projects/BigDataforDevelopment

[13] WEF (World Economic Forum), & Vital Wave Consulting. (2012). Big Data, Big Impact: New Possibilities for International Development. World Economic Forum. Retrieved August 24, 2012, from http://www.weforum.org/reports/big-data-big-impact-new-possibilities-international-development

[14] Bertolucci, Jeff "Hadoop: From Experiment To Leading Big Data Platform" (http://www.informationweek.com/big-data/news/software-platforms/hadoop-from-experiment-to-leading-big-d/240157176), "Information Week", 2013. Retrieved on 14 November 2013.

[15] Webster, John. "MapReduce: Simplified Data Processing on Large Clusters" (http://research.google.com/archive/mapreduce-osdi04.pdf), "Search Storage", 2004. Retrieved on 25 March 2013.

[16] http://ubconferences.org/March 2014.

[17] http://big-project.eu

[18] http://ec.europa.eu/research/horizon2020/index_en.cfm?pg=h2020

[19] http://www.imscenter.net

[20] Anderson, C. (2008, June 23). The End of Theory: The Data Deluge Makes the Scientific Method Obsolete. Wired Magazine, (Science: Discoveries). http://www.wired.com/science/discoveries/magazine/16-07/pb_theory

[21] Rauch, J. (2002). Seeing Around Corners. The Atlantic, (April), 35–48. http://www.theatlantic.com/magazine/archive/2002/04/seeing-around-corners/302471/

[22] Epstein, J. M., & Axtell, R. L. (1996). Growing Artificial Societies: Social Science from the Bottom Up. A Bradford Book.

[23] Delort P., Big data in Biosciences, Big Data Paris, 2012 http://www.bigdataparis.com/documents/Pierre-Delort-INSERM.pdf#page=5

[24] Darwin Bond-Graham, *Iron Cagebook - The Logical End of Facebook's Patents* (http://www.counterpunch.org/2013/12/03/iron-cagebook/), Counterpunch.org, 2013.12.03

[25] Darwin Bond-Graham, *Inside the Tech industry's Startup Conference* (http://www.counterpunch.org/2013/09/11/inside-the-tech-industrys-startup-conference/), Counterpunch.org, 2013.09.11

Further reading

- Gautam Siwach and Dr. Amir Esmailpour - Encrypted search & cluster formation in Big Data |publisher ASEE
- "Big Data for Good" (http://www.odbms.org/download/BigDataforGood.pdf). ODBMS.org. June 5, 2012. Retrieved 2013-11-12.
- Hilbert, Martin; López, Priscila (2011). "The World's Technological Capacity to Store, Communicate, and Compute Information" (http://martinhilbert.net/WorldInfoCapacity.html). *Science* **332** (6025): 60–65. doi: 10.1126/science.1200970 (http://dx.doi.org/10.1126/science.1200970). PMID 21310967 (http://www.ncbi.nlm.nih.gov/pubmed/21310967).
- "The Rise of Industrial Big Data" (http://www.ge-ip.com/library/detail/13476/?cid=wiki_Rise_of_Industrial_Big_Data). GE Intelligent Platforms. Retrieved 2013-11-12.

XLDB

XLDB refers to **eXtremely Large Data Bases**. The definition of *extremely large* refers to data sets that are too big in terms of volume (too much), and/or velocity (too fast), and/of variety (too many places, too many formats) to be handled using conventional solutions.

History

In October 2007 the XLDB experts gathered at SLAC for the First Workshop on Extremely Large Databases [1]. As a result, the XLDB research community was formed. to meet rapidly growing demands, in addition to the original invitational workshop, an open conference, tutorials, and annual satellite events on different continents were added. The main event, held annually at Stanford gathers over 300 technically savvy attendees. XLDB is one of the premier database events catered towards both academic and industrial communities.

Goals

The main goals of this community include:

- Identify trends, commonalities and major roadblocks related to building extremely large databases
- Bridge the gap between users trying to build extremely large databases and database solution providers worldwide
- Facilitate development and growth of practical technologies for extremely large data stores

XLDB Community

As of 2013, the community consisted of about a thousand members including:

1. Scientists who develop, use, or plan to develop or use XLDB for their research, from laboratories.
2. Commercial users of XLDB.
3. Providers of database products, including commercial vendors and representatives from open source database communities.
4. Academic database researchers.

XLDB Conferences, Workshops and Tutorials

The community meets annually at Stanford where the main event is held each fall, usually in September. These who live too far from California to attend have the opportunity to attend satellite events, organized annually around May/June either in Asia or in Europe.

A detailed report is produced after each workshop.

Year	Place	Link	Report	Comments
2013	Stanford	[2]		7th XLDB Conference
2013	CERN, Geneva/Switzerland	[3]		Satellite XLDB Workshop in Europe
2012	Stanford	[4]	[5]	6th XLDB Conference, Workshop & Tutorials
2012	Beijing, China	[6]	[7]	Satellite XLDB Conference in Asia
2011	SLAC	[8]	[9]	5th XLDB Conference and Workshop
2011	Edinburgh, UK	[10]	not available	Satellite XLDB Workshop in Europe
2010	SLAC	[11]	[12]	4th XLDB Conference and Workshop
2009	Lyon, France	[13]	[14]	3rd XLDB Workshop
2008	SLAC	[15]	[16]	2nd XLDB Workshop
2007	SLAC	[17]	[18]	1st XLDB Workshop

Tangible results

The XLDB events led to initiating the effort of building a new open source, science database, SciDB [19].

The XLDB organizers started defining a science benchmark [20] for scientific data management systems called SS-DB.

At 2012 [21] the XLDB organizers announced that two major databases that support arrays as first-class objects (MonetDB SciQL and SciDB) have formed a working group in conjunction with XLDB. This working group is proposing a common syntax (provisionally named “ArrayQL”) for manipulating arrays, including array creation and query.

References

[1] http://www-conf.slac.stanford.edu/xldb07/
[2] https://conf-slac.stanford.edu/xldb-2013/
[3] http://xldb-europe-workshop-2013.web.cern.ch/
[4] http://www-conf.slac.stanford.edu/xldb2012/
[5] http://www.jstage.jst.go.jp/article/dsj/12/0/12_12_023/_pdf
[6] http://idke.ruc.edu.cn/xldb/www.xldb-asia.org/home.html
[7] http://www.xldb.org/wp-content/uploads/2012/09/XLDBAsia2012Report.pdf
[8] http://www-conf.slac.stanford.edu/xldb2011/
[9] http://www.jstage.jst.go.jp/article/dsj/11/0/37/_pdf
[10] http://xldb.eu/xldb_europe_2011/
[11] http://www-conf.slac.stanford.edu/xldb2010/
[12] http://www.jstage.jst.go.jp/article/dsj/9/0/9_MR1/_article
[13] http://www-conf.slac.stanford.edu/xldb2009/
[14] http://www.jstage.jst.go.jp/article/dsj/8/0/MR1/_pdf
[15] http://www-conf.slac.stanford.edu/xldb2008/
[16] http://www.jstage.jst.go.jp/article/dsj/7/0/196/_pdf

[17] http://www-conf.slac.stanford.edu/xldb2007/
[18] http://www.jstage.jst.go.jp/article/dsj/7/0/1/_pdf
[19] http://scidb.org
[20] http://www.xldb.org/science-benchmark/
[21] http://xldb.org/2012|XLDB

Further reading

- Pavlo A., Paulson E., Rasin A., Abadi D. J., Dewitt D. J., Madden S., and Stonebraker M., *A Comparison of Approaches to Large-Scale Data Analysis," Proceedings of the 2009 ACM SIGMOD,* http://database.cs.brown.edu/sigmod09/benchmarks-sigmod09.pdf
- Becla, J., et al. 2006, *Designing a multi-petabyte database for LSST,* http://arxiv.org/abs/cs/0604112.
- Becla, J., & Wang, D. L. 2005, *Lessons Learned from Managing a Petabyte*, downloaded from http://www.slac.stanford.edu/pubs/slacpubs/10750/slac-pub-10963.pdf on 2007-11-25.
- Bell, G., Gray, J., & Szalay, A. 2005, *Petascale computations systems: Balanced cyberinfrastructure in a data-centric world,* http://arxiv.org/abs/cs/0701165.
- Duellmann, D. 1999, *Petabyte Databases*, ACM SIGMOD Record, vol. 28, p. 506, http://www.sigmod.org/sigmod/record/issues/9906/index.html#TutorialSessions.
- Hanushevsky, A., & Nowak, M. 1999, *Pursuit of a Scalable High Performance Multi-Petabyte Database*, 16th IEEE Symposium on Mass Storage Systems, pp. 169–175, http://citeseer.ist.psu.edu/217883.html.
- Shiers, J., *Building Very Large, Distributed Object Databases*, downloaded from http://wwwasd.web.cern.ch/wwwasd/cernlib/rd45/papers/dbprog.html on 2007-11-25.

Secondary database server

A **secondary database server** or SDS is a database server that is kept in synchronisation with the main database so that if the main database goes offline, the secondary database server can be used instead. This is important for a live system that must be online 24 hours a day, and most DBMS systems that support SDSs are capable of automatically detecting and switching to the secondary server within seconds of the main server going offline. The secondary database server can also be used to run read-only queries to take load off the first database server and increase system performance.

Centralized database

A **centralized database** is a database located and maintained in one location, unlike a distributed database. One main advantage is that all data is located in one place. The disadvantage is that bottlenecks may occur.

Distributed database

A **distributed database** is a database in which storage devices are not all attached to a common processing unit such as the CPU, controlled by a distributed database management system (together sometimes called a distributed database system). It may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely-coupled sites that share no physical components.

System administrators can distribute collections of data (e.g. in a database) across multiple physical locations. A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. Because they store data across multiple computers, distributed databases can improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one.[1]

Two processes ensure that the distributed databases remain up-to-date and current: replication and duplication.

1. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex and time-consuming depending on the size and number of the distributed databases. This process can also require a lot of time and computer resources.
2. Duplication, on the other hand, has less complexity. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten.

Both replication and duplication can keep the data current in all distributive locations.

Besides distributed database replication and fragmentation, there are many other distributed database design technologies. For example, local autonomy, synchronous and asynchronous distributed database technologies. These technologies' implementation can and does depend on the needs of the business and the sensitivity/confidentiality of the data stored in the database, and hence the price the business is willing to spend on ensuring data security, consistency and integrity.

When discussing access to distributed databases, Microsoft favors the term **distributed query**, which it defines in protocol-specific manner as "[a]ny SELECT, INSERT, UPDATE, or DELETE statement that references tables and rowsets from one or more external OLE DB data sources". Oracle provides a more language-centric view in which distributed queries and distributed transactions form part of **distributed SQL**.

Architecture

A database user accesses the distributed database through:

Local applications

: applications which do not require data from other sites.

Global applications

: applications which do require data from other sites.

A **homogeneous distributed database** has identical software and hardware running all databases instances, and may appear through a single interface as if it were a single database. A **heterogeneous distributed database** may have different hardware, operating systems, database management systems, and even data models for different databases.

Homogeneous DDBMS

In a homogeneous distributed database all sites have identical software and are aware of each other and agree to cooperate in processing user requests. Each site surrenders part of its autonomy in terms of right to change schema or software. A homogeneous DDBMS appears to the user as a single system. The homogeneous system is much easier to design and manage. The following conditions must be satisfied for homogeneous database:

- The operating system used, at each location must be same or compatible.Wikipedia:Avoid weasel wordsWikipedia:Please clarify
- The data structures used at each location must be same or compatible.
- The database application (or DBMS) used at each location must be same or compatible.

Heterogeneous DDBMS

In a heterogeneous distributed database, different sites may use different schema and software. Difference in schema is a major problem for query processing and transaction processing. Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing. In heterogeneous systems, different nodes may have different hardware & software and data structures at various nodes or locations are also incompatible. Different computers and operating systems, database applications or data models may be used at each of the locations. For example, one location may have the latest relational database management technology, while another location may store data using conventional files or old version of database management system. Similarly, one location may have the Windows NT operating system, while another may have UNIX. Heterogeneous systems are usually used when individual sites use their own hardware and software. On heterogeneous system, translations are required to allow communication between different sites (or DBMS). In this system, the users must be able to make requests in a database language at their local sites. Usually the SQL database language is used for this purpose. If the hardware is different, then the translation is straightforward, in which computer codes and word-length is changed. The heterogeneous system is often not technically or economically feasible. In this system, a user at one location may be able to read but not update the data at another location.

Important considerations

Care with a distributed database must be taken to ensure the following:

- The distribution is transparent — users must be able to interact with the system as if it were one logical system. This applies to the system's performance, and methods of access among other things.
- Transactions are transparent — each transaction must maintain database integrity across multiple databases. Transactions must also be divided into sub-transactions, each sub-transaction affecting one database system.

There are two principal approaches to store a relation r in a distributed database system:

A) Replication

B) Fragmentation/Partitioning

A) Replication: In replication, the system maintains several identical replicas of the same relation r in different sites.

- Data is more available in this scheme.
- Parallelism is increased when read request is served.
- Increases overhead on update operations as each site containing the replica needed to be updated in order to maintain consistency.
- Multi-datacenter replication provides geographical diversity: http://basho.com/tag/multi-datacenter-replication/

B) Fragmentation: The relation r is fragmented into several relations r_1, r_2, r_3....r_n in such a way that the actual relation could be reconstructed from the fragments and then the fragments are scattered to different locations. There are basically two schemes of fragmentation:

- Horizontal fragmentation - splits the relation by assigning each tuple of r to one or more fragments.
- Vertical fragmentation - splits the relation by decomposing the schema R of relation r.

Advantages

- Management of distributed data with different levels of transparency like network transparency, fragmentation transparency, replication transparency, etc.
- Increase reliability and availability
- Easier expansion
- Reflects organizational structure — database fragments potentially stored within the departments they relate to
- Local autonomy or site autonomy — a department can control the data about them (as they are the ones familiar with it)
- Protection of valuable data — if there were ever a catastrophic event such as a fire, all of the data would not be in one place, but distributed in multiple locations
- Improved performance — data is located near the site of greatest demand, and the database systems themselves are parallelized, allowing load on the databases to be balanced among servers. (A high load on one module of the database won't affect other modules of the database in a distributed database)
- Economics — it may cost less to create a network of smaller computers with the power of a single large computer
- Modularity — systems can be modified, added and removed from the distributed database without affecting other modules (systems)
- Reliable transactions - due to replication of the database
- Hardware, operating-system, network, fragmentation, DBMS, replication and location independence
- Continuous operation, even if some nodes go offline (depending on design)
- Distributed query processing can improve performance
- Distributed transaction management
- Single-site failure does not affect performance of system.
- All transactions follow A.C.I.D. property:

- A-atomicity, the transaction takes place as a whole or not at all
- C-consistency, maps one consistent DB state to another
- I-isolation, each transaction sees a consistent DB
- D-durability, the results of a transaction must survive system failures

The Merge Replication Method is popularly used to consolidate the data between databases.[*citation needed*]

Disadvantages

- Complexity — DBAs may have to do extra work to ensure that the distributed nature of the system is transparent. Extra work must also be done to maintain multiple disparate systems, instead of one big one. Extra database design work must also be done to account for the disconnected nature of the database — for example, joins become prohibitively expensive when performed across multiple systems.
- Economics — increased complexity and a more extensive infrastructure means extra labour costs
- Security — remote database fragments must be secured, and they are not centralized so the remote sites must be secured as well. The infrastructure must also be secured (for example, by encrypting the network links between remote sites).
- Difficult to maintain integrity — but in a distributed database, enforcing integrity over a network may require too much of the network's resources to be feasible
- Inexperience — distributed databases are difficult to work with, and in such a young field there is not much readily available experience in "proper" practice
- Lack of standards — there are no tools or methodologies yet to help users convert a centralized DBMS into a distributed DBMS[*citation needed*]
- Database design more complex — In addition to traditional database design challenges, the design of a distributed database has to consider fragmentation of data, allocation of fragments to specific sites and data replication
- Additional software is required
- Operating system should support distributed environment
- Concurrency control poses a major issue. It can be solved by locking and timestamping.
- Distributed access to data
- Analysis of distributed data

References

[1] O'Brien, J. & Marakas, G.M.(2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin

- M. T. Özsu and P. Valduriez, *Principles of Distributed Databases* (3rd edition) (2011), Springer, ISBN 978-1-4419-8833-1
- Elmasri and Navathe, *Fundamentals of database systems* (3rd edition), Addison-Wesley Longman, ISBN 0-201-54263-3
- *Oracle Database Administrator's Guide 10g* (Release 1), http://docs.oracle.com/cd/B14117_01/server.101/b10739/ds_concepts.htm

Distributed database management system

A **distributed database** is a database in which storage devices are not all attached to a common processing unit such as the CPU, controlled by a distributed database management system (together sometimes called a distributed database system). It may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely-coupled sites that share no physical components.

System administrators can distribute collections of data (e.g. in a database) across multiple physical locations. A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. Because they store data across multiple computers, distributed databases can improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one.[1]

Two processes ensure that the distributed databases remain up-to-date and current: replication and duplication.

1. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex and time-consuming depending on the size and number of the distributed databases. This process can also require a lot of time and computer resources.
2. Duplication, on the other hand, has less complexity. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten.

Both replication and duplication can keep the data current in all distributive locations.

Besides distributed database replication and fragmentation, there are many other distributed database design technologies. For example, local autonomy, synchronous and asynchronous distributed database technologies. These technologies' implementation can and does depend on the needs of the business and the sensitivity/confidentiality of the data stored in the database, and hence the price the business is willing to spend on ensuring data security, consistency and integrity.

When discussing access to distributed databases, Microsoft favors the term **distributed query**, which it defines in protocol-specific manner as "[a]ny SELECT, INSERT, UPDATE, or DELETE statement that references tables and rowsets from one or more external OLE DB data sources". Oracle provides a more language-centric view in which distributed queries and distributed transactions form part of **distributed SQL**.

Architecture

A database user accesses the distributed database through:

Local applications

: applications which do not require data from other sites.

Global applications

: applications which do require data from other sites.

A **homogeneous distributed database** has identical software and hardware running all databases instances, and may appear through a single interface as if it were a single database. A **heterogeneous distributed database** may have different hardware, operating systems, database management systems, and even data models for different databases.

Homogeneous DDBMS

In a homogeneous distributed database all sites have identical software and are aware of each other and agree to cooperate in processing user requests. Each site surrenders part of its autonomy in terms of right to change schema or software. A homogeneous DDBMS appears to the user as a single system. The homogeneous system is much easier to design and manage. The following conditions must be satisfied for homogeneous database:

- The operating system used, at each location must be same or compatible.Wikipedia:Avoid weasel wordsWikipedia:Please clarify
- The data structures used at each location must be same or compatible.
- The database application (or DBMS) used at each location must be same or compatible.

Heterogeneous DDBMS

In a heterogeneous distributed database, different sites may use different schema and software. Difference in schema is a major problem for query processing and transaction processing. Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing. In heterogeneous systems, different nodes may have different hardware & software and data structures at various nodes or locations are also incompatible. Different computers and operating systems, database applications or data models may be used at each of the locations. For example, one location may have the latest relational database management technology, while another location may store data using conventional files or old version of database management system. Similarly, one location may have the Windows NT operating system, while another may have UNIX. Heterogeneous systems are usually used when individual sites use their own hardware and software. On heterogeneous system, translations are required to allow communication between different sites (or DBMS). In this system, the users must be able to make requests in a database language at their local sites. Usually the SQL database language is used for this purpose. If the hardware is different, then the translation is straightforward, in which computer codes and word-length is changed. The heterogeneous system is often not technically or economically feasible. In this system, a user at one location may be able to read but not update the data at another location.

Important considerations

Care with a distributed database must be taken to ensure the following:

- The distribution is transparent — users must be able to interact with the system as if it were one logical system. This applies to the system's performance, and methods of access among other things.
- Transactions are transparent — each transaction must maintain database integrity across multiple databases. Transactions must also be divided into sub-transactions, each sub-transaction affecting one database system.

There are two principal approaches to store a relation r in a distributed database system:

A) Replication

B) Fragmentation/Partitioning

A) Replication: In replication, the system maintains several identical replicas of the same relation r in different sites.

- Data is more available in this scheme.
- Parallelism is increased when read request is served.
- Increases overhead on update operations as each site containing the replica needed to be updated in order to maintain consistency.
- Multi-datacenter replication provides geographical diversity: http://basho.com/tag/multi-datacenter-replication/

B) Fragmentation: The relation r is fragmented into several relations r_1, r_2, r_3....r_n in such a way that the actual relation could be reconstructed from the fragments and then the fragments are scattered to different locations. There are basically two schemes of fragmentation:

- Horizontal fragmentation - splits the relation by assigning each tuple of r to one or more fragments.
- Vertical fragmentation - splits the relation by decomposing the schema R of relation r.

Advantages

- Management of distributed data with different levels of transparency like network transparency, fragmentation transparency, replication transparency, etc.
- Increase reliability and availability
- Easier expansion
- Reflects organizational structure — database fragments potentially stored within the departments they relate to
- Local autonomy or site autonomy — a department can control the data about them (as they are the ones familiar with it)
- Protection of valuable data — if there were ever a catastrophic event such as a fire, all of the data would not be in one place, but distributed in multiple locations
- Improved performance — data is located near the site of greatest demand, and the database systems themselves are parallelized, allowing load on the databases to be balanced among servers. (A high load on one module of the database won't affect other modules of the database in a distributed database)
- Economics — it may cost less to create a network of smaller computers with the power of a single large computer
- Modularity — systems can be modified, added and removed from the distributed database without affecting other modules (systems)
- Reliable transactions - due to replication of the database
- Hardware, operating-system, network, fragmentation, DBMS, replication and location independence
- Continuous operation, even if some nodes go offline (depending on design)
- Distributed query processing can improve performance
- Distributed transaction management
- Single-site failure does not affect performance of system.
- All transactions follow A.C.I.D. property:
  - A-atomicity, the transaction takes place as a whole or not at all
  - C-consistency, maps one consistent DB state to another
  - I-isolation, each transaction sees a consistent DB
  - D-durability, the results of a transaction must survive system failures

The Merge Replication Method is popularly used to consolidate the data between databases.[*citation needed*]

Disadvantages

- Complexity — DBAs may have to do extra work to ensure that the distributed nature of the system is transparent. Extra work must also be done to maintain multiple disparate systems, instead of one big one. Extra database design work must also be done to account for the disconnected nature of the database — for example, joins become prohibitively expensive when performed across multiple systems.
- Economics — increased complexity and a more extensive infrastructure means extra labour costs
- Security — remote database fragments must be secured, and they are not centralized so the remote sites must be secured as well. The infrastructure must also be secured (for example, by encrypting the network links between remote sites).
- Difficult to maintain integrity — but in a distributed database, enforcing integrity over a network may require too much of the network's resources to be feasible
- Inexperience — distributed databases are difficult to work with, and in such a young field there is not much readily available experience in "proper" practice

- Lack of standards — there are no tools or methodologies yet to help users convert a centralized DBMS into a distributed DBMS[*citation needed*]
- Database design more complex — In addition to traditional database design challenges, the design of a distributed database has to consider fragmentation of data, allocation of fragments to specific sites and data replication
- Additional software is required
- Operating system should support distributed environment
- Concurrency control poses a major issue. It can be solved by locking and timestamping.
- Distributed access to data
- Analysis of distributed data

References

[1] O'Brien, J. & Marakas, G.M.(2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin

- M. T. Özsu and P. Valduriez, *Principles of Distributed Databases* (3rd edition) (2011), Springer, ISBN 978-1-4419-8833-1
- Elmasri and Navathe, *Fundamentals of database systems* (3rd edition), Addison-Wesley Longman, ISBN 0-201-54263-3
- *Oracle Database Administrator's Guide 10g* (Release 1), http://docs.oracle.com/cd/B14117_01/server.101/b10739/ds_concepts.htm

Distributed file system

A **clustered file system** is a file system which is shared by being simultaneously mounted on multiple servers. There are several approaches to clustering, most of which do not employ a clustered file system (only direct attached storage for each node). Clustered file systems can provide features like location-independent addressing and redundancy which improve reliability or reduce the complexity of the other parts of the cluster. **Parallel file systems** are a type of clustered file system that spread data across multiple storage nodes, usually for redundancy or performance.[1]

Shared-disk / storage area network

A **shared-disk filesystem** uses a storage-area network (SAN) to provide direct disk access from multiple computers at the block level. Access control and translation from file-level operations that applications use to block-level operations used by the SAN must take place on the client node. The most common type of clustered filesystems is shared-disk filesystem, which—by adding mechanisms for concurrency control—provides a consistent and serializable view of the file system, avoiding corruption and unintended data loss even when multiple clients try to access the same files at the same time. It is a common practice for shared-disk filesystems to employ some sort of a fencing mechanism to prevent data corruption in case of node failures, because an unfenced device can cause data corruption if it loses communication with its sister nodes, and tries to access the same information other nodes are accessing.

The underlying storage area network may use any of a number of block-level protocols, including SCSI, iSCSI, HyperSCSI, ATA over Ethernet (AoE), Fibre Channel, network block device, and InfiniBand.

There are different architectural approaches to a shared-disk filesystem. Some distribute file information across all the servers in a cluster (fully distributed). Others utilize a centralized metadata server. Both achieve the same result of enabling all servers to access all the data on a shared storage device.[*citation needed*]

Examples

- Silicon Graphics (SGI) clustered file system (CXFS)
- Veritas Cluster File System
- DataPlow Nasan File System
- DataPlow SAN File System (SFS)
- IBM General Parallel File System (GPFS)
- Microsoft Cluster Shared Volumes (CSV)
- Oracle Cluster File System (OCFS)
- PolyServe storage solutions
- Quantum StorNext FileSystem (SNFS), ex ADIC, ex CentraVision FileSystem (CVFS)
- Blue Whale Clustered file system (BWFS)
- Red Hat Global File System (GFS)
- Sanbolic Melio FS clustered file system
- Sun QFS
- TerraScale Technologies TerraFS
- VMware VMFS
- Xsan

Distributed file systems

Distributed file systems do not share block level access to the same storage but use a network protocol.[2] These are commonly known as *network file systems*, even though they are not the only file systems that use the network to send data.[*citation needed*] Distributed file systems can restrict access to the file system depending on access lists or capabilities on both the servers and the clients, depending on how the protocol is designed.

The difference between a distributed file system and a distributed data store is that a distributed file system allows files to be accessed using the same interfaces and semantics as local files - e.g. mounting/unmounting, listing directories, read/write at byte boundaries, system's native permission model. Distributed data stores, by contrast, require using a different API or library and have different semantics (most often those of a database).

Design goals

Distributed file systems may aim for "transparency" in a number of aspects. That is, they aim to be "invisible" to client programs, which "see" a system which is similar to a local file system. Behind the scenes, the distributed file system handles locating files, transporting data, and potentially providing other features listed below.

- *Access transparency* is that clients are unaware that files are distributed and can access them in the same way as local files are accessed.
- *Location transparency*; a consistent name space exists encompassing local as well as remote files. The name of a file does not give its location.
- *Concurrency transparency*; all clients have the same view of the state of the file system. This means that if one process is modifying a file, any other processes on the same system or remote systems that are accessing the files will see the modifications in a coherent manner.
- *Failure transparency*; the client and client programs should operate correctly after a server failure.
- *Heterogeneity*; file service should be provided across different hardware and operating system platforms.
- *Scalability*; the file system should work well in small environments (1 machine, a dozen machines) and also scale gracefully to huge ones (hundreds through tens of thousands of systems).
- *Replication transparency*; to support scalability, we may wish to replicate files across multiple servers. Clients should be unaware of this.
- *Migration transparency*; files should be able to move around without the client's knowledge.

History

The Incompatible Timesharing System used virtual devices for transparent inter-machine filesystem access in the 1960s. More file servers were developed in the 1970s. In 1976 Digital Equipment Corporation created the File Access Listener (FAL), an implementation of the Data Access Protocol as part of DECnet Phase II which became the first widely used network file system. In 1985 Sun Microsystems created the file system called "Network File System" (NFS) which became the first widely used Internet Protocol based network file system. Other notable network file systems are Andrew File System (AFS), Apple Filing Protocol (AFP), NetWare Core Protocol (NCP), and Server Message Block (SMB) which is also known as Common Internet File System (CIFS).

Examples

- GFS (Google Inc.)
- Ceph (Inktank)
- MooseFS (Core Technology / Gemius)
- Windows Distributed File System (DFS) (Microsoft)
- FhGFS (Fraunhofer)
- GlusterFS (Red Hat)
- Lustre
- Ibrix

Network attached storage

Network attached storage (NAS) provides both storage and a file system, like a shared disk file system on top of a storage area network (SAN). NAS typically uses file-based protocols (as opposed to block-based protocols a SAN would use) such as NFS (popular on UNIX systems), SMB/CIFS (Server Message Block/Common Internet File System) (used with MS Windows systems), AFP (used with Apple Macintosh computers), or NCP (used with OES and Novell NetWare).

Design considerations

Avoiding single point of failure

The failure of disk hardware or a given storage node in a cluster can create a single point of failure that can result in data loss or unavailability. Fault tolerance and high availability can be provided through data replication of one sort or another, so that data remains intact and available despite the failure of any single piece of equipment. For examples, see the lists of distributed fault-tolerant file systems and distributed parallel fault-tolerant file systems.

Performance

A common performance measurement of a clustered file system is the amount of time needed to satisfy service requests. In conventional systems, this time consists of a disk-access time and a small amount of CPU-processing time. But in a clustered file system, a remote access has additional overhead due to the distributed structure. This includes the time to deliver the request to a server, the time to deliver the response to the client, and for each direction, a CPU overhead of running the communication protocol software.

Concurrency

Concurrency control becomes an issue when more than one person or client is accessing the same file or block and want to update it. Hence updates to the file from one client should not interfere with access and updates from other clients. This problem is more complex with file systems due to concurrent overlapping writes, where different writers write to overlapping regions of the file concurrently.[3] This problem is usually handled by concurrency control or locking which may either be built into the file system or provided by an add-on protocol.

History

IBM mainframes in the 1970s could share physical disks and file systems if each machine had its own channel connection to the drives' control units. In the 1980s, Digital Equipment Corporation's TOPS-20 and VAX/VMS clusters included shared disk filesystems.[*citation needed*]

References

[1] http://www.dell.com/downloads/global/power/ps2q05-20040179-Saify-OE.pdf

[2] Silberschatz, Galvin (1994). *Operating System concepts*, chapter 17 *Distributed file systems*. Addison-Wesley Publishing Company. ISBN 0-201-59292-4.

[3] Pessach, Yaniv (2013). *Distributed Storage: Concepts, Algorithms, and Implementations*. ISBN 978-1482561043.

Further reading

- A Taxonomy of Distributed Storage Systems (http://www.cloudbus.org/reports/DistributedStorageTaxonomy.pdf)
- A Taxonomy and Survey on Distributed File Systems (http://trac.nchc.org.tw/grid/raw-attachment/wiki/jazz/09-05-22/A_Taxonomy_and_Survey_on_Distributed_File_Systems.pdf)
- A survey of distributed file systems (http://www.cis.upenn.edu/~bcpierce/courses/dd/papers/satya89survey.ps)
- The Evolution of File Systems (http://www.snia-europe.org/objects_store/Christian_Bandulet_SNIATutorialBasics_EvolutionFileSystems.pdf)

Distributed data store

A **distributed data store** is a computer network where information is stored on more than one node, often in a replicated fashion. It is usually specifically used to refer to either a distributed database where users store information on a *number of nodes*, or a computer network in which users store information on a *number of peer network nodes*.

Distributed databases

Distributed databases are usually non-relational databases that make a quick access to data over a large number of nodes possible. Some distributed databases expose rich query abilities while others are limited to a key-value store semantics. Examples of limited distributed databases are Google's BigTable, which is much more than a distributed file system or a peer-to-peer network, Amazon's Dynamo and Windows Azure Storage.

As the ability of arbitrary querying is not as important as the availability, designers of distributed data stores have increased the latter at an expense of consistency. But the high-speed read/write access results in reduced consistency, as it is not possible to have both consistency, availability, and partition tolerance of the network, as it has been proven by the CAP theorem.

Peer network node data stores

In peer network data stores, the user can usually reciprocate and allow other users to use their computer as a storage node as well. Information may or may not be accessible to other users depending on the design of the network.

Most peer-to-peer networks do not have distributed data stores in that the user's data is only available when their node is on the network. However, this distinction is somewhat blurred in a system such as BitTorrent, where it is possible for the originating node to go offline but the content to continue to be served. Still, this is only the case for individual files requested by the redistributors, as contrasted with a network such as Freenet where all computers are made available to serve all files.

Distributed data stores typically use an error detection and correction technique. Some distributed data stores (such as Parchive over NNTP) use forward error correction techniques to recover the original file when parts of that file are damaged or unavailable. Others try again to download that file from a different mirror.

Examples

Distributed non-relational databases

- Apache Cassandra, former data store of Facebook
- BigTable, the data store of Google
- Druid (open-source data store), used by Netflix
- Dynamo of Amazon
- HBase, current data store of Facebook's Messaging Platform
- Riak
- Voldemort, data store used by LinkedIn

Peer network node data stores

- BitTorrent
- Chord project
- GNUnet
- Freenet
- Unity, of the software Perfect Dark
- Mnet
- NNTP (the distributed data storage protocol used for Usenet news)
- Storage@home
- Tahoe-LAFS

References

Heterogeneous Database System

A **heterogeneous database system** is an automated (or semi-automated) system for the integration of heterogeneous, disparate database management systems to present a user with a single, unified query interface.

Heterogeneous database systems (HDBs) are computational models and software implementations that provide heterogeneous database integration.

Problems of heterogeneous database integration

This article does not contain details of distributed database management systems (sometimes known as federated database systems).

Technical heterogeneity

Different file formats, access protocols, query languages etc. Often called syntactic heterogeneity from the point of view of data.

Data model heterogeneity

Different ways of representing and storing the same data. Table decompositions may vary, column names (data labels) may be different (but have the same semantics), data encoding schemes may vary (i.e., should a measurement scale be explicitly included in a field or should it be implied elsewhere). Also referred as schematic heterogeneity.

Semantic heterogeneity

Data across constituent databases may be related but different. Perhaps a database system must be able to integrate genomic and proteomic data. They are related—a gene may have several protein products—but the data are different (nucleotide sequences and amino acid sequences, or hydrophilic or -phobic amino acid sequence and positively or negatively charged amino acids). There may be many ways of looking at semantically similar, but distinct, datasets.

The system may also be required to present "new" knowledge to the user. Relationships may be inferred between data according to rules specified in domain ontologies.

References

Simple Sloppy Semantic Database

Simple Sloppy Semantic Database (S3DB) is a distributed infrastructure that relies on Semantic Web concepts for management of heterogeneous data. This distributed data management system was first proposed in 2006, following the argumentation the previous year that omics data sets would be more easily managed if fragmented in RDF triples. That first version, 1.0, was focused on the support of an indexing engine for triplestore management. The second version, made available in October 2007, added cross-referencing between triples in distinct S3DB deployments to support it as a distributed infrastructure. The third version was released in July 2008 and exposes its API through a specialized query language[1] accessible as a REST web service. An update of that release (version 3.5) also includes a RESTful SPARQL endpoint. This update introduced a self-update feature which replaces version numbers by date of update. The rationale, core data model, and usage in National Cancer Institute (NIH/NCI) SPORE awards [2] are described and illustrated in a PLoS ONE manuscript published August 13, 2008 and a BMC Bioinformatics manuscript published on July 20, 2010. The following year, S3DB's API was published and was put to use in the management of a clinical trial at MDAnderson Cancer Center using a web application with a self-assembled interface.[3] In 2012, in "Semantic Web meets Integrative Biology: a survey" the features of S3DB were highlighted in an independent assessment. The summary of this assessment was that "S3QL supports a permission control mechanism that allows users to specify contextual minutia such as provenance and access control on the semantic level. The effectiveness of S3QL was illustrated through use cases of IB, such as genomic characterization of cancer and molecular epidemiology of infectious diseases. We expect S3QL or its variations to be accepted as the standard access control mechanism by the SW community". The API's specification is kept at s3ql.org [4]. S3DB is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License.

External links

Documentation and wiki are maintained at www.s3db.org [5]. This application is publicly available with open source in PHP code. The use of S3DB's web-services formatted as JSON is demonstrated in a youtube video [6]. Code development at github.com/s3db/s3db [7].

References

[1] S3QL API Basics (http://sites.google.com/a/s3db.org/s3db/documentation/api-basics)
[2] http://trp.cancer.gov/
[3] Correa MC, HF Deus, AT Vasconcelos, Y Hayashi, JA Ajani, SV Patnana, JS Almeida (2010) AGUIA: autonomous graphical user interface assembly for clinical tirals semantic data services. BMC Medical Informatics and Decision Making, (10)35 PMID 20977768 (http://www.ncbi.nlm.nih.gov/pubmed/20977768).
[4] https://sites.google.com/a/s3db.org/s3db/documentation/s3qlsyntax
[5] http://www.s3db.org
[6] http://www.youtube.com/watch?v=LZOLNT3_KbI
[7] http://github.com/s3db/s3db

Distributed transaction

A **distributed transaction** is an operations bundle, in which two or more network hosts are involved. Usually, hosts provide **transactional resources**, while the **transaction manager** is responsible for creating and managing a global transaction that encompasses all operations against such resources. Distributed transactions, as any other transactions, must have all four ACID (atomicity, consistency, isolation, durability) properties, where atomicity guarantees all-or-nothing outcomes for the unit of work (operations bundle).

Open Group, a vendor consortium, proposed the X/Open Distributed Transaction Processing (DTP) Model (X/Open XA), which became a de facto standard for behavior of transaction model components.

Databases are common transactional resources and, often, transactions span a couple of such databases. In this case, a distributed transaction can be seen as a database transaction that must be synchronized (or provide ACID properties) among multiple participating databases which are distributed among different physical locations. The isolation property (the I of ACID) poses a special challenge for multi database transactions, since the (global) serializability property could be violated, even if each database provides it (see also global serializability). In practice most commercial database systems use strong strict two phase locking (SS2PL) for concurrency control, which ensures global serializability, if all the participating databases employ it. (see also commitment ordering for multidatabases.)

A common algorithm for ensuring correct completion of a distributed transaction is the two-phase commit (2PC). This algorithm is usually applied for updates able to commit in a short period of time, ranging from couple of milliseconds to couple of minutes.

There are also long-lived distributed transactions, for example a transaction to book a trip, which consists of booking a flight, a rental car and a hotel. Since booking the flight might take up to a day to get a confirmation, two-phase commit is not applicable here, it will lock the resources for this long. In this case more sophisticated techniques that involve multiple undo levels are used. The way you can undo the hotel booking by calling a desk and cancelling the reservation, a system can be designed to undo certain operations (unless they are irreversibly finished).

In practice, long-lived distributed transactions are implemented in systems based on Web Services. Usually these transactions utilize principles of Compensating transactions, Optimism and Isolation Without Locking. X/Open standard does not cover long-lived DTP.

Several modern technologies, including Enterprise Java Beans (EJBs) and Microsoft Transaction Server (MTS) fully support distributed transaction standards.

References

- "Web-Services Transactions" [1]. *Web-Services Transactions*. Retrieved May 2, 2005.
- "Nuts And Bolts Of Transaction Processing" [2]. *Article about Transaction Management*. Retrieved May 3, 2005.
- "A Detailed Comparison of Enterprise JavaBeans (EJB) & The Microsoft Transaction Server (MTS) Models" [3].

Further reading

- Gerhard Weikum, Gottfried Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Morgan Kaufmann, 2002, ISBN 1-55860-508-8

References

[1] http://xml.sys-con.com/read/43755.htm
[2] http://www.subbu.org/articles/transactions/NutsAndBoltsOfTP.html
[3] http://gsraj.tripod.com/misc/ejbmtscomp.html

Network transparency

Network transparency in its most general sense refers to the ability of a protocol to transmit data over the network in a manner which is transparent (invisible) to those using the applications that are using the protocol.

The term is often incorrectly applied in the context of the X Window System, which is able to transmit graphical data over the network and integrate it seamlessly with applications running and displaying locally.

Databases

In a centralized database system, the only available resource that needs to be shielded from the user is the data (that is, the storage system). In a distributed DBMS, a second resource needs to be managed in much the same manner: the network. Preferably, the user should be protected from the network operational details. Then there would be no difference between database applications that would run on the centralized database and those that would run on a distributed one. This kind of transparency is referred to as **network transparency** or **distribution transparency**. From a database management system (DBMS) perspective, distribution transparency requires that users do not have to specify where data is located.

Some have separated distribution transparency into location transparency and naming transparency.

Location transparency in commands used to perform a task is independent both of the locations of the data, and of the system on which an operation is carried out.

Naming transparency means that a unique name is provided for each object in the database.

Firewalls

Transparency in firewall technology can be defined at the networking (IP or Internet layer) or at the application layer.

Transparency at the IP layer means the client targets the real IP address of the server. If a connection is non-transparent, then the client targets an intermediate host (address), which could be a proxy or a caching server. IP layer transparency could be also defined from the point of server's view. If the connection is transparent, the server sees the real client IP. If it is non-transparent, the server sees the IP of the intermediate host.

Transparency at the application layer means the client application uses the protocol in a different way. An example of a transparent HTTP request for a server:

```
GET / HTTP/1.1
Host: example.org
Connection: Keep-Alive
```

An example non-transparent HTTP request for a proxy (cache):

```
GET http://foo.bar/HTTP/1.1
Proxy-Connection: Keep-Alive
```

Application layer transparency is symmetric when the same working mode is used on both the sides. The transparency is asymmetric when the firewall (usually a proxy) converts server type requests to proxy type or vice versa.

Transparency at the IP layer does not mean automatically application layer transparency.

References

Long-lived transaction

A **long-lived transaction** is a transaction that spans multiple database transactions. The transaction is considered "long-lived" because its boundaries must, by necessity of business logic, extend past a single database transaction. A long-lived transaction can be thought of as a sequence of database transactions grouped to achieve a single atomic result..

A common example is a multi-step sequence of requests and responses of an interaction with a user through a web client.

A long-lived transaction creates challenges of concurrency control and scalability.

A chief strategy in designing long-lived transactions is optimistic concurrency control with versioning.

So much research work related to these long lived transactions was carried out by several professors from the Oxford University and Michigan State University and the Central University of Hyderabad. Dr. James from the Oxford University created several hypotheses for long-lived transactions. Dr Copperfield of the Michigan State University was regarded highly for his contributions in this field. Dr A B Sagar of Hyderabad Central University has also done very creative work in relating long-lived transactions with financial transactions in Microfinance.

However the study is not complete and is still open to challenges and research issues.

Distributed concurrency control

Distributed concurrency control is the concurrency control of a system distributed over a computer network (Bernstein et al. 1987, Weikum and Vossen 2001).

In *database systems* and *transaction processing* (*transaction management*) distributed concurrency control refers primarily to the concurrency control of a distributed database. It also refers to the concurrency control in a multidatabase (and other multi-transactional object) environment (e.g., federated database, grid computing, and cloud computing environments. A major goal for distributed concurrency control is distributed serializability (or global serializability for multidatabase systems). Distributed concurrency control poses special challenges beyond centralized one, primarily due to communication and computer latency. It often requires special techniques, like distributed lock manager over fast computer networks with low latency, like switched fabric (e.g., InfiniBand). commitment ordering (or commit ordering) is a general serializability technique that achieves distributed serializability (and global serializability in particular) effectively on a large scale, without concurrency control information distribution (e.g., local precedence relations, locks, timestamps, or tickets), and thus without performance penalties that are typical to other serializability techniques (Raz 1992).

The most common distributed concurrency control technique is *strong strict two-phase locking* (SS2PL, also named *rigorousness*), which is also a common centralized concurrency control technique. SS2PL provides both the *serializability*, *strictness*, and *commitment ordering* properties. Strictness, a special case of recoverability, is utilized for effective recovery from failure, and commitment ordering allows participating in a general solution for global serializability. For large-scale distribution and complex transactions, distributed locking's typical heavy performance penalty (due to delays, latency) can be saved by using the atomic commitment protocol, which is needed in a distributed database for (distributed) transactions' atomicity (e.g., two-phase commit, or a simpler one in a reliable

system), together with some local commitment ordering variant (e.g., local SS2PL) instead of distributed locking, to achieve global serializability in the entire system. All the commitment ordering theoretical results are applicable whenever atomic commitment is utilized over partitioned, distributed recoverable (transactional) data, including automatic *distributed deadlock* resolution. Such technique can be utilized also for a large-scale parallel database, where a single large database, residing on many nodes and using a distributed lock manager, is replaced with a (homogeneous) multidatabase, comprising many relatively small databases (loosely defined; any process that supports transactions over partitioned data and participates in atomic commitment complies), fitting each into a single node, and using commitment ordering (e.g., SS2PL, strict CO) together with some appropriate atomic commitment protocol (without using a distributed lock manager).

References

- Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* [2], Addison Wesley Publishing Company, 1987, ISBN 0-201-10715-5
- Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* [3], Elsevier, ISBN 1-55860-508-8
- Yoav Raz (1992): "The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Managers Using Atomic Commitment." [5] *Proceedings of the Eighteenth International Conference on Very Large Data Bases* (VLDB), pp. 292-312, Vancouver, Canada, August 1992. (also DEC-TR 841, Digital Equipment Corporation, November 1990)

Consistency model

In computer science, **consistency models** are used in distributed systems like distributed shared memory systems or distributed data stores (such as a filesystems, databases, optimistic replication systems or Web caching). The system supports a given model if operations on memory follow specific rules. The data consistency model specifies a contract between programmer and system, wherein the system guarantees that if the programmer follows the rules, memory will be consistent and the results of memory operations will be predictable.

High level languages, such as C, C++, and Java, partially maintain the contract by translating memory operations into low-level operations in a way that preserves memory semantics. To hold to the contract, compilers may reorder some memory instructions, and library calls such as `pthread_mutex_lock()` encapsulate required synchronization.

Verifying sequential consistency is undecidable in general, even for finite-state cache-coherence protocols.

Consistency models define rules for the apparent order and visibility of updates, and it is a continuum with tradeoffs.

Example

Assume that the following case occurs:

- The row X is replicated on nodes M and N
- The client A writes row X to node N
- After a period of time t, client B reads row X from node M

The consistency model has to determine whether client B sees the write from client A or not.

Types

A non-exhaustive list of consistency models are

- causal consistency
- delta consistency
- entry consistency
- eventual consistency
- fork consistency
- linearizability (also known as strict or atomic consistency)
- one-copy serializability
- PRAM consistency (also known as FIFO consistency)
- release consistency
- sequential consistency
- serializability
- vector-field consistency
- weak consistency
- strong consistency

References

Further reading

- Ali Sezgin (2004). *Formalization and verification of shared memory* (http://www.atilim.edu.tr/~asezgin/diss_finalcopy.pdf) (PDF). (contains many valuable references)
- Kathy Yelick; Dan Bonachea, Chuck Wallace (2004). *A Proposal for a UPC Memory Consistency Model (v1.0)* (http://www.gwu.edu/~upc/downloads/upcmem.pdf) (PDF).
- Mosberger, David (1993). "Memory Consistency Models" (http://citeseer.ist.psu.edu/mosberger93memory.html). *Operating Systems Review* **27** (1): 18–26. doi: 10.1145/160551.160553 (http://dx.doi.org/10.1145/160551.160553).
- Sarita V. Adve, Kourosh Gharachorloo (December 1996). "Shared Memory Consistency Models: A Tutorial" (http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-95-7.pdf). *IEEE Computer* **29** (12): 66–76. Retrieved 2008-05-28.
- Steinke, Robert C.; Gary J. Nutt (2004). "A unified theory of shared memory consistency". *Journal of the ACM* **51** (5): 800–849. arXiv: cs.DC/0208027 (http://arxiv.org/abs/cs.DC/0208027). doi: 10.1145/1017460.1017464 (http://dx.doi.org/10.1145/1017460.1017464).

External links

- Consistency Models (http://cs.gmu.edu/cne/modules/dsm/green/memcohe.html)
- IETF slides (http://www.ietf.org/old/2009/proceedings/01mar/slides/webi-1/sld006.htm)
- Memory Ordering in Modern Microprocessors, Part I (http://www.linuxjournal.com/article/8211) and Part II (http://www.linuxjournal.com/article/8212), by Paul E. McKenney (2005). *Linux Journal*

Distributed Transaction Coordinator

The **Distributed Transaction Coordinator** (MSDTC) service is a component of modern versions of Microsoft Windows that is responsible for coordinating transactions that span multiple resource managers, such as databases, message queues, and file systems. MSDTC is included in Windows 2000 and later operating systems, and is also available for Windows NT 4.0.

MSDTC performs the transaction coordination role for components, usually with COM and .NET architectures. In MSDTC terminology, the director is called the transaction manager.

By default, the Microsoft Distributed Transaction Coordinator (MSDTC) service is installed with Windows 2000. It cannot be uninstalled through Add/Remove Programs.

External links

- Distributed Transaction Coordinator [1] on the Microsoft Developer Network
- New functionality in the Distributed Transaction Coordinator service in Windows Server 2003 Service Pack 1 and in Windows XP Service Pack 2 [2]
- Florin Lazar's weblog [3]; a Microsoft developer blog with extensive discussions on MSDTC and transaction processing

[4]; Mohsen Agsen is a Technical Fellow who formed the core transaction group, which designed and delivered the Distributed Transaction Coordinator (DTC)

References

[1] http://msdn.microsoft.com/en-us/library/ms684146.aspx
[2] http://support.microsoft.com/?kbid=899191
[3] http://blogs.msdn.com/florinlazar/
[4] http://www.microsoft.com/presspass/exec/techfellow/Agsen/default.mspx

Two-phase commit protocol

In transaction processing, databases, and computer networking, the **two-phase commit protocol** (2PC) is a type of atomic commitment protocol (ACP). It is a distributed algorithm that coordinates all the processes that participate in a distributed atomic transaction on whether to *commit* or *abort* (*roll back*) the transaction (it is a specialized type of consensus protocol). The protocol achieves its goal even in many cases of temporary system failure (involving either process, network node, communication, etc. failures), and is thus widely utilized.[1][2][3] However, it is not resilient to all possible failure configurations, and in rare cases user (e.g., a system's administrator) intervention is needed to remedy an outcome. To accommodate recovery from failure (automatic in most cases) the protocol's participants use logging of the protocol's states. Log records, which are typically slow to generate but survive failures, are used by the protocol's recovery procedures. Many protocol variants exist that primarily differ in logging strategies and recovery mechanisms. Though usually intended to be used infrequently, recovery procedures compose a substantial portion of the protocol, due to many possible failure scenarios to be considered and supported by the protocol.

In a "normal execution" of any single distributed transaction, i.e., when no failure occurs, which is typically the most frequent situation, the protocol consists of two phases:

1. The *commit-request phase* (or *voting phase*), in which a *coordinator* process attempts to prepare all the transaction's participating processes (named *participants*, *cohorts*, or *workers*) to take the necessary steps for either committing or aborting the transaction and to *vote*, either "Yes": commit (if the transaction participant's local portion execution has ended properly), or "No": abort (if a problem has been detected with the local portion), and
2. The *commit phase*, in which, based on *voting* of the cohorts, the coordinator decides whether to commit (only if *all* have voted "Yes") or abort the transaction (otherwise), and notifies the result to all the cohorts. The cohorts then follow with the needed actions (commit or abort) with their local transactional resources (also called *recoverable resources*; e.g., database data) and their respective portions in the transaction's other output (if applicable).

Note that the two-phase commit (2PC) protocol should not be confused with the two-phase locking (2PL) protocol, a concurrency control protocol.

Assumptions

The protocol works in the following manner: one node is designated the **coordinator**, which is the master site, and the rest of the nodes in the network are designated the **cohorts**. The protocol assumes that there is stable storage at each node with a write-ahead log, that no node crashes forever, that the data in the write-ahead log is never lost or corrupted in a crash, and that any two nodes can communicate with each other. The last assumption is not too restrictive, as network communication can typically be rerouted. The first two assumptions are much stronger; if a node is totally destroyed then data can be lost.

The protocol is initiated by the coordinator after the last step of the transaction has been reached. The cohorts then respond with an **agreement** message or an **abort** message depending on whether the transaction has been processed successfully at the cohort.

Basic algorithm

Commit request phase

or **voting phase**

1. The coordinator sends a **query to commit** message to all cohorts and waits until it has received a reply from all cohorts.
2. The cohorts execute the transaction up to the point where they will be asked to commit. They each write an entry to their *undo log* and an entry to their *redo log*.
3. Each cohort replies with an **agreement** message (cohort votes **Yes** to commit), if the cohort's actions succeeded, or an **abort** message (cohort votes **No**, not to commit), if the cohort experiences a failure that will make it impossible to commit.

Commit phase

or **Completion phase**

Success

If the coordinator received an **agreement** message from *all* cohorts during the commit-request phase:

1. The coordinator sends a **commit** message to all the cohorts.
2. Each cohort completes the operation, and releases all the locks and resources held during the transaction.
3. Each cohort sends an **acknowledgment** to the coordinator.
4. The coordinator completes the transaction when all acknowledgments have been received.

Failure

If *any* cohort votes **No** during the commit-request phase (or the coordinator's timeout **expires**):

1. The coordinator sends a **rollback** message to all the cohorts.
2. Each cohort undoes the transaction using the undo log, and releases the resources and locks held during the transaction.
3. Each cohort sends an **acknowledgement** to the coordinator.
4. The coordinator undoes the transaction when all acknowledgements have been received.

Message flow

```
Coordinator                                         Cohort
                        QUERY TO COMMIT
                -------------------------------->
                        VOTE YES/NO                 prepare*/abort*
                <--------------------------------
commit*/abort*          COMMIT/ROLLBACK
                -------------------------------->
                        ACKNOWLEDGMENT              commit*/abort*
                <--------------------------------
end
```

An * next to the record type means that the record is forced to stable storage.[4]

Disadvantages

The greatest disadvantage of the two-phase commit protocol is that it is a blocking protocol. If the coordinator fails permanently, some cohorts will never resolve their transactions: After a cohort has sent an **agreement** message to the coordinator, it will block until a **commit** or **rollback** is received.

Implementing the two-phase commit protocol

Common architecture

In many cases the 2PC protocol is distributed in a computer network. It is easily distributed by implementing multiple dedicated 2PC components similar to each other, typically named *Transaction managers* (TMs; also referred to as *2PC agents*), that carry out the protocol's execution for each transaction (e.g., The Open Group's X/Open XA). The databases involved with a distributed transaction, the *participants*, both the coordinator and cohorts, *register* to close TMs (typically residing on respective same network nodes as the participants) for terminating that transaction using 2PC. Each distributed transaction has an ad hoc set of TMs, the TMs to which the transaction participants register. A leader, the coordinator TM, exists for each transaction to coordinate 2PC for it, typically the TM of the coordinator database. However, the coordinator role can be transferred to another TM for performance or reliability reasons. Rather than exchanging 2PC messages among themselves, the participants exchange the messages with their respective TMs. The relevant TMs communicate among themselves to execute the 2PC protocol schema above, "representing" the respective participants, for terminating that transaction. With this architecture the protocol is fully distributed (does not need any central processing component or data structure), and scales up with number of network nodes (network size) effectively.

This common architecture is also effective for the distribution of other atomic commitment protocols besides 2PC, since all such protocols use the same voting mechanism and outcome propagation to protocol participants.

Protocol optimizations

Database research has been done on ways to get most of the benefits of the two-phase commit protocol while reducing costs by *protocol optimizations* and protocol operations saving under certain system's behavior assumptions.

Presume abort and Presume commit

Presumed abort or *Presumed commit* are common such optimizations.[5] An assumption about the outcome of transactions, either commit, or abort, can save both messages and logging operations by the participants during the 2PC protocol's execution. For example, when presumed abort, if during system recovery from failure no logged evidence for commit of some transaction is found by the recovery procedure, then it assumes that the transaction has been aborted, and acts accordingly. This means that it does not matter if aborts are logged at all, and such logging can be saved under this assumption. Typically a penalty of additional operations is paid during recovery from failure, depending on optimization type. Thus the best variant of optimization, if any, is chosen according to failure and transaction outcome statistics.

Tree two-phase commit protocol

The **Tree 2PC protocol** (also called *Nested 2PC*, or *Recursive 2PC*) is a common variant of 2PC in a computer network, which better utilizes the underlying communication infrastructure. The participants in a distributed transaction are typically invoked in an order which defines a tree structure, the *invocation tree*, where the participants are the nodes and the edges are the invocations (communication links). The same tree is commonly utilized to complete the transaction by a 2PC protocol, but also another communication tree can be utilized for this, in principle. In a tree 2PC the coordinator is considered the root ("top") of a communication tree (inverted tree),

while the cohorts are the other nodes. The coordinator can be the node that originated the transaction (invoked recursively (transitively) the other participants), but also another node in the same tree can take the coordinator role instead. 2PC messages from the coordinator are propagated "down" the tree, while messages to the coordinator are "collected" by a cohort from all the cohorts below it, before it sends the appropriate message "up" the tree (except an **abort** message, which is propagated "up" immediately upon receiving it or if the current cohort initiates the abort).

The **Dynamic two-phase commit** (Dynamic two-phase commitment, D2PC) **protocol**[6] is a variant of Tree 2PC with no predetermined coordinator. It subsumes several optimizations that have been proposed earlier. **Agreement** messages (**Yes** votes) start to propagate from all the leaves, each leaf when completing its tasks on behalf of the transaction (becoming *ready*). An intermediate (non leaf) node sends when *ready* an **agreement** message to the last (single) neighboring node from which **agreement** message has not yet been received. The coordinator is determined dynamically by racing **agreement** messages over the transaction tree, at the place where they collide. They collide either at a transaction tree node, to be the coordinator, or on a tree edge. In the latter case one of the two edge's nodes is elected as a coordinator (any node). D2PC is time optimal (among all the instances of a specific transaction tree, and any specific Tree 2PC protocol implementation; all instances have the same tree; each instance has a different node as coordinator): By choosing an optimal coordinator D2PC commits both the coordinator and each cohort in minimum possible time, allowing the earliest possible release of locked resources in each transaction participant (tree node).

References

[1] Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman (1987): *Concurrency Control and Recovery in Database Systems* (http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx), Chapter 7, Addison Wesley Publishing Company, ISBN 0-201-10715-5

[2] Gerhard Weikum, Gottfried Vossen (2001): *Transactional Information Systems* (http://www.elsevier.com/wps/find/bookdescription.cws_home/677937/description#description), Chapter 19, Elsevier, ISBN 1-55860-508-8

[3] Philip A. Bernstein, Eric Newcomer (2009): *Principles of Transaction Processing*, 2nd Edition (http://www.elsevierdirect.com/product.jsp?isbn=9781558606234), Chapter 8, Morgan Kaufmann (Elsevier), ISBN 978-1-55860-623-4

[4] C. Mohan, Bruce Lindsay and R. Obermarck (1986): "Transaction management in the R* distributed database management system" (http://dl.acm.org/citation.cfm?id=7266),*ACM Transactions on Database Systems (TODS)*, Volume 11 Issue 4, Dec. 1986, Pages 378 - 396

[5] C. Mohan, Bruce Lindsay (1985): "Efficient commit protocols for the tree of processes model of distributed transactions" (http://portal.acm.org/citation.cfm?id=850772),*ACM SIGOPS Operating Systems Review*, 19(2),pp. 40-52 (April 1985)

[6] Yoav Raz (1995): "The Dynamic Two Phase Commitment (D2PC) protocol " (http://www.springerlink.com/content/pv12p828kk616258/),*Database Theory — ICDT '95*, *Lecture Notes in Computer Science*, Volume 893/1995, pp. 162-176, Springer, ISBN 978-3-540-58907-5

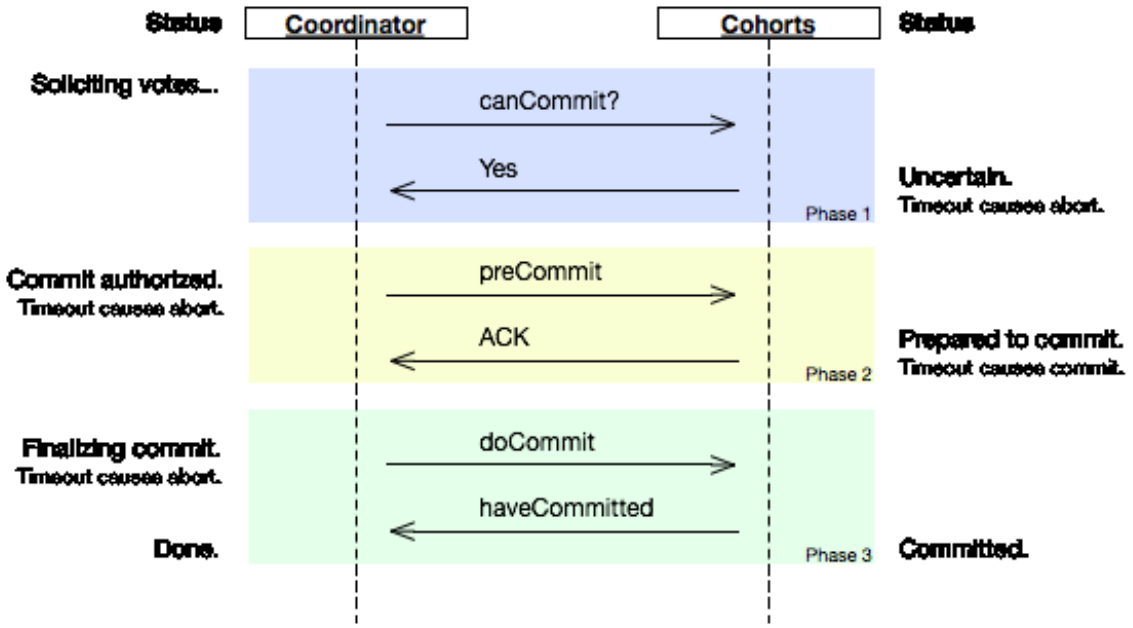
Three-phase commit protocol

In computer networking and databases, the **three-phase commit protocol** (3PC) is a distributed algorithm which lets all nodes in a distributed system agree to commit a transaction. Unlike the two-phase commit protocol (2PC) however, 3PC is non-blocking. Specifically, 3PC places an upper bound on the amount of time required before a transaction either commits or aborts. This property ensures that if a given transaction is attempting to commit via 3PC and holds some resource locks, it will release the locks after the timeout.

3PC was originally described by Dale Skeen and Michael Stonebraker in their paper, “A Formal Model of Crash Recovery in a Distributed System”. In that work, they modeled 2PC as a system of non-deterministic finite state automata and proved that it is not resilient to a random single site failure. The basic observation is that in 2PC, while one site is in the “prepared to commit” state, the other may be in either the “commit” or the “abort” state. From this analysis, they developed 3PC to avoid such states and it is thus resilient to such failures.

Protocol Description

In describing the protocol, we use terminology similar to that used in the two-phase commit protocol. Thus we have a single coordinator site leading the transaction and a set of one or more cohorts being directed by the coordinator.



Coordinator

1. The coordinator receives a transaction request. If there is a failure at this point, the coordinator aborts the transaction (i.e. upon recovery, it will consider the transaction aborted). Otherwise, the coordinator sends a **canCommit?** message to the cohorts and moves to the waiting state.
2. If there is a failure, timeout, or if the coordinator receives a **No** message in the waiting state, the coordinator aborts the transaction and sends an **abort** message to all cohorts. Otherwise the coordinator will receive **Yes** messages from all cohorts within the time window, so it sends **preCommit** messages to all cohorts and moves to the prepared state.
3. If the coordinator succeeds in the prepared state, it will move to the commit state. However if the coordinator times out while waiting for an acknowledgement from a cohort, it will abort the transaction. In the case where all acknowledgements are received, the coordinator moves to the commit state as well.

Cohort

1. The cohort receives a **canCommit?** message from the coordinator. If the cohort agrees it sends a **Yes** message to the coordinator and moves to the prepared state. Otherwise it sends a **No** message and aborts. If there is a failure, it moves to the abort state.
2. In the prepared state, if the cohort receives an **abort** message from the coordinator, fails, or times out waiting for a commit, it aborts. If the cohort receives a **preCommit** message, it sends an **ACK** message back and awaits a final **commit** or **abort**.
3. If, after a cohort member receives a **preCommit** message, the coordinator fails or times out, the cohort member goes forward with the commit.

Motivation

A Two-phase commit protocol cannot dependably recover from a failure of both the **coordinator** and a cohort member during the **Commit phase**. If only the **coordinator** had failed, and no cohort members had received a **commit** message, it could safely be inferred that no **commit** had happened. If, however, both the **coordinator** and a cohort member failed, it is possible that the failed cohort member was the first to be notified, and had actually done the **commit**. Even if a new **coordinator** is selected, it cannot confidently proceed with the operation until it has received an agreement from **all** cohort members ... and hence must block until all cohort members respond.

The Three-phase commit protocol eliminates this problem by introducing the **Prepared to commit** state. If the **coordinator** fails before sending **preCommit** messages, the **cohort** will unanimously agree that the operation was **aborted**. The **coordinator** will not send out a **doCommit** message until **all** cohort members have **ACK**ed that they are **Prepared to commit**. This eliminates the possibility that **any** cohort member actually completed the transaction before **all** cohort members were aware of the decision to do so (an ambiguity that necessitated indefinite blocking in the Two-phase commit protocol).

Disadvantages

The main disadvantage to this algorithm is that it cannot recover in the event the network is segmented in any manner. The original 3PC algorithm assumes a fail-stop model, where processes fail by crashing and crashes can be accurately detected, and does not work with network partitions or asynchronous communication.

Keidar and Dolev's E3PC algorithm eliminates this disadvantage.

The protocol requires at least 3 round trips to complete, needing a minimum of 3 round trip times (RTTs). This is potentially a long latency to complete each transaction.

References

Xeround

Xeround is a provider of cloud database software, launched in 2005.[1] The company was founded by Sharon Barkai and Gilad Zlotkin. Zlotkin, a former research fellow at MIT Sloan School of Management,[2] founded five other startups including Radview (NASDAQ:RDVW). Israeli financial newspaper Globes ranked the company as one of Israel's most promising start-ups in 2006.[3]

Xeround's product was initially used by Telecom providers, including T-Mobile; in 2009 the company added a MySQL front end to its product,[4] making it applicable to a mass market of 12 million MySQL applications.[5] The product allows MySQL users to scale their database and achieve high availability on cloud platforms like Amazon EC2.[] The beta version of the service was reported to be used by 2000 organizations;[6][7][8][9] General Availability was announced in June 2011.[10] According to CNET blogger Dave Rosenberg, Xeround's MySQL support makes it "well positioned to take a leadership position in the database market".[11]

On May 1, 2013 Xeround announced to its paid customers that they were shutting down the Cloud Database Service and all data must be migrated before being dropped on May 15, 2013

Product

Xeround provides a cloud database service for applications based on the open source edition of the MySQL database (MySQL is currently owned by Oracle). The product addresses two related problems: it is complex to run databases on the cloud, especially if high availability is needed; and databases in general are difficult to scale, as data throughput and volumes grow.[12] A cloud database service solves both problems, by managing the database on the cloud and taking care of scalability and high availability, in a way that is transparent to the application. Instead of connecting to a local instance of MySQL, applications can connect to Xeround's cloud database, and are then free to scale as needed. Because Xeround is an in-memory distributed database, it is currently limited up to 50 Gigabytes of data.[13] Xeround gives a no downtime SLA guarantee [14]. The service offers pay-per-use pricing, calculated per Gigabyte per hour, with an additional charge for data transfer for large databases.[15]

Xeround offers its service on several cloud platforms - as of September 2011, Xeround supported Amazon EC2, RackSpace,[16] and Heroku,[17] and is planning to support additional providers. As of March 2011, Xeround was the only commercially available product which supports more than one cloud provider, allowing users to move their databases freely between cloud platforms without being locked in.[18]

While Xeround uses the open source version of MySQL, the cloud database software itself is not open source. Another distinction is that while Xeround offers MySQL as a front-end, on the back-end it is a NoSQL data storage system distributed on a large number of physical nodes - so it is not subject to the scalability limitations of regular MySQL databases.

On 1 May 2013 Xeround announced via an e-mail to customers that they would no longer be providing their service. The service is to end on 15 May 2013. [19]

Company timeline

- 2005 - Xeround is founded by Sharon Barkai and Gilad Zlotkin, raises $6.5 million in Series A funding, focuses initially on distributed database software for Telecom providers.
- 2006 - Xeround ranked as one of Israel's most promising start-ups by Israeli financial newspaper Globes.
- 2008 - Xeround raises an additional $16 million in Series B funding.[20]
- 2009 - Xeround recruits Razi Sharir as CEO and repositions its product as a cloud database service with a MySQL front-end.
- 2010 - Xeround launches beta version of its database service.[21]
- 2011 - Xeround announces General Availability of its cloud database service, and raises an additional $4 million in its final financing round.
- 2011 - Xeround Raises $9.3M From Benchmark And Others.[22]

Competitors and alternatives

Xeround's primary competitors are database services offered by the large cloud vendors, Amazon Relational Database Service and Database.com by SalesForce. Other cloud database providers mentioned by industry sources[23] are Microsoft Azure SQL Database, NimbusDB, ClearDB, ParAccel, as well as NoSQL key-value data stores such as Amazon SimpleDB, Google AppEngine Data Store, Couchbase Server, and Cloudant.

Database users running their applications on the cloud also have the option of installing databases in a "do it yourself" manner instead of paying for a cloud database service. This involves purchasing a machine instance on a cloud computing platform like Amazon EC2, and manually installing a database. This method is considered to be less expensive, but more complex, than using an "out of the box" database service like Xeround.[24]

Next Generation

Currently the website is showing a message:

- We are working on the next generation of data management solution for today's complex distributed infrastructure needs. Register below if you wish to be notified.

References

[1] "Xeround" (http://investing.businessweek.com/research/stocks/private/snapshot.asp?privcapId=21008964), *Bloomberg.com* (http://www.bloomberg.com), Retrieved 25-8-2011.

[2] Gilad Zlotkin's Scientific Publications (http://dl.acm.org/author_page.cfm?id=81100061819&coll=DL&dl=ACM&trk=0&cfid=41956890&cftoken=15167202), *ACM Digital Library* (http://dl.acm.org), Retrieved 12-9-2011.

[3] Batya Feldman, Cloud database co Xeround raises $4m (http://www.globes.co.il/serveen/globes/docview.asp?did=1000636459&fid=1725), *Globes* (http://www.globes.co.il), Retrieved 25-8-2011

[4] Derrick Harris, "For Xeround, MySQL in the Cloud Knows No Bounds" (http://gigaom.com/cloud/for-xeround-mysql-in-the-cloud-knows-no-bounds/), *GigaOM* (http://gigaom.com/), Retrieved 25-8-2011.

[5] Victoria Barret, "Why Oracle Won't Kill MySQL" (http://archive.is/20130123095359/http://www.forbes.com/2009/04/20/mysql-marten-mickos-technology-enterprise-tech-mysql.html), *Forbes.com* (http://www.forbes.com), Retrieved 12-9-2011.

[6] James Niccolai, "Four companies rethink databases for the cloud" (http://www.computerworld.com.au/article/391370/four_companies_rethink_databases_cloud), *Computerworld.com.au* (http://www.computerworld.com.au), Retrieved 12-9-2011

[7] "Xeround Announces Cloud Service for MySQL" (http://www.devshed.com/c/a/MySQL/Xeround-Announces-Cloud-Service-for-MySQL/), *DevShed.com* (http://www.devshed.com), Retrieved 25-8-2011.

[8] Derrick Harris, "Xeround enters GA, tests the SQL-in-the-cloud water (http://gigaom.com/cloud/xeround-enters-ga-tests-the-sql-in-the-cloud-water/), *GigaOM* (http://www.gigaom.com), Retrieved 25-8-2011

[9] Maria Deutcher, "Xeround Releases Cloud Service for MySQL Applications" (http://siliconangle.com/blog/2011/06/13/xeround-releases-cloud-service-for-mysql-applications/), *Silicon Angle* (http://siliconangle.com), Retrieved 15-9-2011.

[10] Sean Michael Kerner, "Xeround MySQL Cloud Database Goes GA" (http://www.databasejournal.com/news/xeround-mysql-cloud-database-ga.html), *Database Journal* (http://www.databasejournal.com), Retrieved 25-8-2011

[11] Dave Rosenberg, Xeround scales MySQL for the cloud (http://news.cnet.com/8301-13846_3-20016428-62.html), *CNET* (http://news.cnet.com), Retrieved 25-8-2011

[12] Dave Rosenberg, Are databases in the cloud really all that different? (http://news.cnet.com/8301-13846_3-20022794-62.html,), *CNET* (http://news.cnet.com), Retrieved 8-9-2011

[13] Razi Sharir, "Xeround Pay-Per-Use Pricing Explained" (http://xeround.com/blog/2011/07/xeround-pay-per-use-pricing-explained), *Xeround.com* (http://xeround.com), Retrieved 18-9-2011.

[14] http://xeround.com/mysql-cloud-db-overview/mysql-high-availability/

[15] "Xeround Pricing" (http://xeround.com/mysql-cloud-db-overview/pay-per-use-cloud-database/), *Xeround.com* (http://xeround.com), Retrieved 15-9-2011

[16] Angela Bartels, "Xeround Provides Auto-Scaling & High-Availability for your MySQL Database in the Cloud" (http://www.rackspace.com/cloud/blog/2011/04/12/xeround-provides-auto-scaling-high-availability-for-your-mysql-database-in-the-cloud/), *RackSpace Cloud Blog* (http://www.rackspace.com/cloud/blog/), Retrieved 25-8-2011

[17] "Xeround Cloud Database for MySQL applications" (http://devcenter.heroku.com/articles/xeround), *Heroku Dev Center* (http://devcenter.heroku.com), Retrieved 25-8-2011

[18] Dan Kusnetzky, "Xeround Add-on for Heroku Cloud Platform" (http://www.zdnet.com/blog/virtualization/xeround-add-on-for-heroku-cloud-platform/2920), *ZDNet* (http://www.zdnet.com), Retrieved 25-8-2011

[19] Xeround website, " Discontinuing of Xeround Cloud Database Public Service (http://xeround.com/blog/2013/05/discontinuing-of-xeround-cloud-database-public-service)"

[20] Database Virtualization Company Xeround Secures $16 Million in Series B Financing (http://vmblog.com/archive/2008/07/11/database-virtualization-company-xeround-secures-16-million-in-series-b-financing.aspx), *VMblog.com* (http://vmblog.com), Retrieved 15-9-2011.

[21] "Out of Stealth, Xeround Launches MySQL as a Service" (http://gigaom.com/cloud/out-of-stealth-xeround-launches-mysql-as-a-service/), GigaOM (http://gigaom.com), Retrieved 15-9-2011.

[22] "Cloud Database Startup Xeround Raises $9.3M From Benchmark And Others" (http://techcrunch.com/2011/12/14/cloud-database-startup-xeround-raises-9-3m-from-benchmark-and-others/), TechCrunch (http://techcrunch.com), Retrieved 12-18-2011.

[23] Klint Finley, "7 Cloud-Based Database Services" (http://www.readwriteweb.com/cloud/2011/01/7-cloud-based-database-service.php), *ReadWriteWeb* (http://www.readwriteweb.com), Retrieved 18-9-2011.

[24] "Cloud Database: Do-It-Yourself or Database-as-a-Service?" (http://xeround.com/cloud-database-comparison/), Xeround.com (http://xeround.com), Retrieved 15-9-2011

External links

- Xeround Official Website (http://xeround.com)
- Xeround Blog (http://xeround.com/blog)

Vector-field consistency

Vector-Field Consistency[1] is a consistency model for replicated data (for example, objects), initially described in a paper which was awarded the best-paper prize in the ACM/IFIP/Usenix Middleware Conference 2007. It has since been enhanced for increased scalability and fault-tolerance in a recent paper.

Description

This consistency model was initially designed for replicated data management in adhoc gaming in order to minimize bandwidth usage without sacrificing playability. Intuitively, it captures the notion that although players require, wish, and take advantage of information regarding the whole of the game world (as opposed to a restricted view to rooms, arenas, etc. of limited size employed in many multiplayer games), they need to know information with greater freshness, frequency, and accuracy as other game entities are located closer and closer to the player's position.

It prescribes a multidimensional divergence bounding scheme, based on a vector field that employs consistency vectors k=(θ,σ,ν), standing for maximum allowed **t**ime - or replica staleness, **s**equence - or missing updates, and **v**alue[2] - or user-defined measured replica divergence, applied to all space coordinates in game scenario or world.

The consistency vector-fields emanate from field-generators designated as pivots (for example, players) and field intensity attenuates as distance grows from these pivots in concentric or square-like regions. This consistency model unifies locality-awareness techniques employed in message routing and consistency enforcement for multiplayer games, with divergence bounding techniques traditionally employed in replicated database and web scenarios.

Notes

[1] Designation coined by L. Veiga.

[2] Since in the Greek alphabet there was no letter for the *vee* sound, the *nu* letter was preferred for its resemblance with the roman V, for *v*alue, instead of β (*beta*) for the *vee* sound in contemporary Greek speaking.

References

Storage area network

Computer network types by spatial scope

• Near field Communication (NFC)
• Body (BAN)
• Personal (PAN)
• Car/Electronics (CAN)
• Near-me (NAN)
• Local (LAN)
  • Home (HAN)
  • Storage (SAN)
• Campus (CAN)
• Backbone
• Metropolitan (MAN)
• Wide (WAN)
• Cloud (IAN)
• Internet
• Interplanetary Internet
• Intergalactic Computer Network

• v • t • e [1]

A **storage area network** (**SAN**) is a dedicated network that provides access to consolidated, block level data storage. SANs are primarily used to enhance storage devices, such as disk arrays, tape libraries, and optical jukeboxes, accessible to servers so that the devices appear like locally attached devices to the operating system. A SAN typically has its own network of storage devices that are generally not accessible through the local area network by other devices. The cost and complexity of SANs dropped in the early 2000s to levels allowing wider adoption across both enterprise and small to medium sized business environments.

A SAN does not provide file abstraction, only block-level operations. However, file systems built on top of SANs do provide file-level access, and are known as *SAN filesystems* or shared disk file systems.

Storage

Historically, data centers first created "islands" of SCSI disk arrays as direct-attached storage (DAS), each dedicated to an application, and visible as a number of "virtual hard drives" (i.e. LUNs). Essentially, a SAN consolidates such storage islands together using a high-speed network.

Operating systems maintain their own file systems on their own dedicated, non-shared LUNs, as though they were local to themselves. If multiple systems were simply to attempt to share a LUN, these would interfere with each other and quickly corrupt the data. Any planned sharing of data on different computers within a LUN requires advanced solutions, such as SAN file systems or clustered computing.

Despite such issues, SANs help to increase storage capacity utilization, since multiple servers consolidate their private storage space onto the disk arrays.

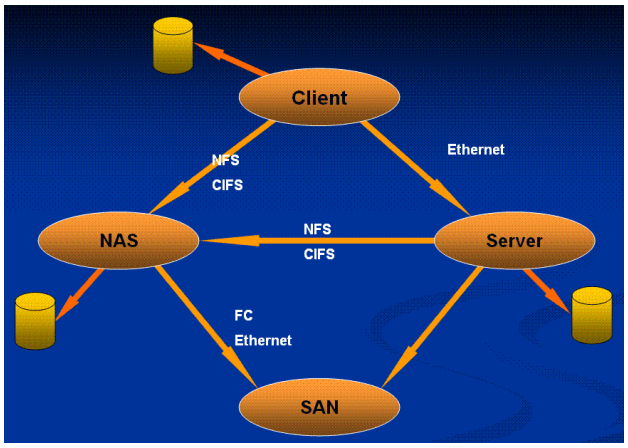
Common uses of a SAN include provision of transactionally accessed data that require high-speed block-level access to the hard drives such as email servers, databases, and high usage file servers.

SAN and NAS

NAS, Network Attached Storage was a solution to the problems of Direct Attached Storage (DAS). The solution was that servers would share a connection to the storage devices via a network connection through the LAN. This set up allows the server to be used loaded with software and applications instead of being split between two duties. With the old DAS setup the server would be split between application use and storage use. With NAS there is no longer a need for support of the traditional storage interface (SCSI) and now the server or client may access NAS storage with a network connection. The drawback to this is that there is no longer a high-speed connection between the CPU and storage units – they still must use the LAN to communicate and this creates bandwidth bottlenecks. In addition requests are processed using file access protocols and CPU cycles must be used to convert into block requests that a server may use to retrieve files and information. This has relegated NAS to be used as data backup more than anything else [*citation needed*].

SAN-NAS hybrid

Despite the differences between SAN and NAS, it is possible to create solutions that include both technologies.[*citation needed*]



Hybrid using DAS, NAS and SAN technologies.

Benefits

Sharing storage usually simplifies storage administration and adds flexibility since cables and storage devices do not have to be physically moved to shift storage from one server to another.

Other benefits include the ability to allow servers to boot from the SAN itself. This allows for a quick and easy replacement of faulty servers since the SAN can be reconfigured so that a replacement server can use the LUN of the faulty server. While this area of technology is still new, many view it as being the future of the enterprise datacenter.

SANs also tend to enable more effective disaster recovery processes. A SAN could span a distant location containing a secondary storage array. This enables storage replication either implemented by disk array controllers, by server software, or by specialized SAN devices. Since IP WANs are often the least costly method of long-distance transport, the Fibre Channel over IP (FCIP) and iSCSI protocols have been developed to allow SAN extension over IP networks. The traditional physical SCSI layer could only support a few meters of distance - not nearly enough to ensure business continuance in a disaster.

The economic consolidation of disk arrays has accelerated the advancement of several features including I/O caching, snapshotting, and volume cloning (Business Continuance Volumes or BCVs).

Network types

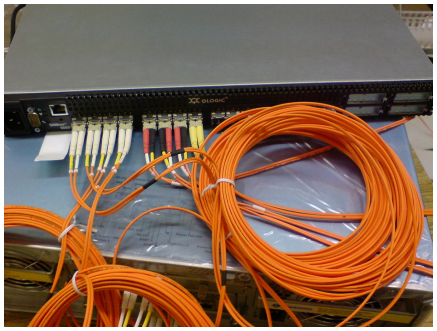
Most storage networks use the SCSI protocol for communication between servers and disk drive devices. A mapping layer to other protocols is used to form a network:

- ATA over Ethernet (AoE), mapping of ATA over Ethernet
- Fibre Channel Protocol (FCP), the most prominent one, is a mapping of SCSI over Fibre Channel
- Fibre Channel over Ethernet (FCoE)
- ESCON over Fibre Channel (FICON), used by mainframe computers
- HyperSCSI, mapping of SCSI over Ethernet
- iFCP or SANoIP mapping of FCP over IP
- iSCSI, mapping of SCSI over TCP/IP
- iSCSI Extensions for RDMA (iSER), mapping of iSCSI over InfiniBand

Storage networks may also be built using SAS and SATA technologies. SAS evolved from SCSI direct-attached storage. SATA evolved from IDE direct-attached storage. SAS and SATA devices can be networked using SAS Expanders.

SAN infrastructure

SANs often use a Fibre Channel fabric topology - an infrastructure specially designed to handle storage communications. It provides faster and more reliable access than higher-level protocols used in NAS. A fabric is similar in concept to a network segment in a local area network. A typical Fibre Channel SAN fabric is made up of a number of Fibre Channel switches.



Qlogic SAN-switch with optical Fibre Channel connectors installed.

Today, all major SAN equipment vendors also offer some form of Fibre Channel routing solution, and these bring substantial scalability benefits to the SAN architecture by allowing data to cross between different fabrics without merging them. These offerings use proprietary protocol elements, and the top-level architectures being promoted are radically different. They often enable mapping Fibre Channel traffic over IP or over SONET/SDH.

Compatibility

One of the early problems with Fibre Channel SANs was that the switches and other hardware from different manufacturers were not compatible. Although the basic storage protocols FCP were always quite standard, some of the higher-level functions did not interoperate well. Similarly, many host operating systems would react badly to other operating systems sharing the same fabric. Many solutions were pushed to the market before standards were finalized and vendors have since innovated around the standards[*citation needed*].

SANs in media and entertainment

Video editing workgroups require very high data transfer rates and very low latency. Outside of the enterprise market, this is one area that greatly benefits from SANs.

SANs in Media and Entertainment are often referred to as Serverless SANs due to the nature of the configuration which places the video workflow (ingest, editing, playout) clients directly on the SAN rather than attaching to servers. Control of data flow is managed by a distributed file system such as StorNext by Quantum.

Per-node bandwidth usage control, sometimes referred to as Quality of Service (QoS), is especially important in video workgroups as it ensures fair and prioritized bandwidth usage across the network, if there is insufficient open bandwidth available.

Storage virtualization

Storage virtualization is the process of abstracting logical storage from physical storage. The physical storage resources are aggregated into storage pools, from which the logical storage is created. It presents to the user a logical space for data storage and transparently handles the process of mapping it to the physical location, a concept called location transparency. This is implemented in modern disk arrays, often using vendor proprietary solutions. However, the goal of storage virtualization is to group multiple disk arrays from different vendors, scattered over a network, into a single storage device. The single storage device can then be managed uniformly. [*citation needed*]

SAN Storage QoS (Quality of Service)

SAN Storage QoS (Quality of Service) is the coordination of capacity and performance in a dedicated storage area network. This enables the desired storage performance to be calculated and maintained for network customers accessing the device.

Key factors that affect Storage Area Network QoS(Quality of Service) are:

- Bandwidth – The rate of data throughput available on the system.
- Latency – The time delay for a read/write operation to execute.
- Queue depth – The number of outstanding operations waiting to execute to the underlying disks (Traditional or SSD).

QoS can be impacted in a SAN storage system by unexpected increase in data traffic (usage spike) from one network user that can cause performance to decrease for other users on the same network. This can be known as the “Noisy Neighbor Effect.” When QoS services are enabled in a SAN storage system, the “Noisy Neighbor Effect” can be prevented and network storage performance can be accurately predicted.

Using SAN storage QoS is in contrast to using disk over-provisioning in a SAN environment. Over-provisioning can be used to provide additional capacity to compensate for peak network traffic loads. However, where network loads are not predictable, over-provisioning can eventually cause all bandwidth to be fully consumed and latency to increase significantly resulting in SAN performance degradation.

References

[1] http://en.wikipedia.org/w/index.php?title=Template:Area_networks&action=edit

External links

- Introduction to Storage Area Networks Exhaustive Introduction into SAN, [[IBM Redbooks|IBM Redbook (https://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf)]]
- SAN vs. DAS: A Cost Analysis of Storage in the Enterprise (http://capitalhead.com/articles/san-vs-das-a-cost-analysis-of-storage-in-the-enterprise.aspx)
- SAS and SATA, solid-state storage lower data center power consumption (http://searchstorage.techtarget.co.uk/generic/0,295582,sid181_gci1516893,00.html)
- SAN NAS Videos (http://www.youtube.com/playlist?list=PLivYD7W2z2HMGGRIwRoRcqLL4HMpR1dIe)
- Storage Area Network Info (http://www.storageareanetworkinfo.blogspot.com.ar/)

Partition (database)

A **partition** is a division of a logical database or its constituent elements into distinct independent parts. Database partitioning is normally done for manageability, performance or availability reasons.

Benefits of multiple partitions

A popular and favourable application of partitioning is in a distributed database management system. Each partition may be spread over multiple nodes, and users at the node can perform local transactions on the partition. This increases performance for sites that have regular transactions involving certain views of data, whilst maintaining availability and security.

Partitioning criteria

Current high end relational database management systems provide for different criteria to split the database. They take a *partitioning key* and assign a partition based on certain criteria. Common criteria are:

Range partitioning

> Selects a partition by determining if the partitioning key is inside a certain range. An example could be a partition for all rows where the column `zipcode` has a value between `70000` and `79999`.

List partitioning

> A partition is assigned a list of values. If the partitioning key has one of these values, the partition is chosen. For example all rows where the column `Country` is either `Iceland`, `Norway`, `Sweden`, `Finland` or `Denmark` could build a partition for the Nordic countries.

Hash partitioning

> The value of a hash function determines membership in a partition. Assuming there are four partitions, the hash function could return a value from 0 to 3.

Composite partitioning allows for certain combinations of the above partitioning schemes, by for example first applying a range partitioning and then a hash partitioning. Consistent hashing could be considered a composite of hash and list partitioning where the hash reduces the key space to a size that can be listed.

Partitioning methods

The partitioning can be done by either building separate smaller databases (each with its own tables, indices, and transaction logs), or by splitting selected elements, for example just one table.

Horizontal partitioning (also see ***shard***) involves putting different rows into different tables. Perhaps customers with ZIP codes less than 50000 are stored in CustomersEast, while customers with ZIP codes greater than or equal to 50000 are stored in CustomersWest. The two partition tables are then CustomersEast and CustomersWest, while a view with a union might be created over both of them to provide a complete view of all customers.

Vertical partitioning involves creating tables with fewer columns and using additional tables to store the remaining columns.[1] Normalization also involves this splitting of columns across tables, but vertical partitioning goes beyond that and partitions columns even when already normalized. Different physical storage might be used to realize vertical partitioning as well; storing infrequently used or very wide columns on a different device, for example, is a method of vertical partitioning. Done explicitly or implicitly, this type of partitioning is called "row splitting" (the row is split by its columns). A common form of vertical partitioning is to split dynamic data (slow to find) from static data (fast to find) in a table where the dynamic data is not used as often as the static. Creating a view across the two newly created tables restores the original table with a performance penalty, however performance will increase

when accessing the static data e.g. for statistical analysis.

References

[1] Vertical Partitioning Algorithms for Database Design, by Shamkant Navathe, Stefano Ceri, Gio Wiederhold, and Jinglie Dou, Stanford University 1984 (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.8306)

External links

- IBM DB2 partitioning (http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/admin/c0004885.htm)
- MySQL partitioning (http://dev.mysql.com/doc/refman/5.5/en/partitioning.html)
- Oracle partitioning (http://www.oracle.com/us/products/database/options/partitioning/index.htm)
- SQL Server partitions (http://msdn.microsoft.com/en-us/library/ms190787.aspx)
- PostgreSQL partitioning (http://www.postgresql.org/docs/current/interactive/ddl-partitioning.html)
- Sybase ASE 15.0 partitioning (http://www.sybase.com/detail?id=1036923)
- MongoDB partitioning (http://www.mongodb.org/display/DOCS/Sharding)
- ScimoreDB partitioning (http://scimore.com/wiki/Distributed_schema)
- VoltDB partitioning (http://community.voltdb.com/docs/UsingVoltDB/ChapAppDesign#DesignPartition)

Shared nothing architecture

A **shared nothing architecture** (**SN**) is a distributed computing architecture in which each node is independent and self-sufficient, and there is no single point of contention across the system. More specifically, none of the nodes share memory or disk storage.

People typically contrast SN with systems that keep a large amount of centrally-stored state information, whether in a database, an application server, or any other similar single point of contention. While SN is best known in the context of web development, the concept predates the web: Michael Stonebraker at the University of California, Berkeley used the term in a 1986 database paper.[1] In it he mentions existing commercial implementations of the architecture (although none are named explicitly). Teradata, which delivered its first system in 1983, was probably one of those commercial implementations. Tandem Computers officially released NonStop SQL, a shared nothing database, in 1984. [2]

Shared nothing is popular for web development because of its scalability. As Google has demonstrated, a pure SN system can scale almost infinitely simply by adding nodes in the form of inexpensive computers, since there is no single bottleneck to slow the system down. Google calls this *sharding*. A SN system typically partitions its data among many nodes on different databases (assigning different computers to deal with different users or queries), or may require every node to maintain its own copy of the application's data, using some kind of coordination protocol. This is often referred to as *database sharding*.

There is some doubt about whether a web application with many independent web nodes but a single, shared database (clustered or otherwise) should be counted as SN. One of the approaches to achieve SN architecture for stateful applications (which typically maintain state in a centralized database) is the use of a data grid, also known as distributed caching. This still leaves the centralized database as a single point of failure.

Shared nothing architectures have become prevalent in the data warehousing space. There is much debate as to whether the shared nothing approach is superior to shared Disk[3] with sound arguments presented by both camps. Shared nothing architectures certainly take longer to respond to queries that involve joins over large data sets from different partitions (machines). However the potential for scaling is huge.[4]

What is shared?

While there is no single point of contention within the software/hardware components of SN systems, it should be noted that information from disparate nodes may still need to be reintegrated at some point. Such points occur wherever an information system that is outside the SN architecture queries information from disparate nodes within the SN architecture for a single purpose. Examples of such external nodes might be:

1. persons (minds) who look at two SN nodes and decide that they hold or process data about the same thing (simply recognising that two nodes belong to the same SN system would be sufficient)
2. any software/hardware system that is written to query different nodes within the SN architecture

References

[1] The Case for Shared Nothing Architecture by Michael Stonebraker. [Originally published in *Database Engineering*, Volume 9, Number 1 (1986).] (http://db.cs.berkeley.edu/papers/hpts85-nothing.pdf)(PDF)

[2] NonStop SQL, A Distributed, High-Performance, High-Availability Implementation of SQL, Tandem Technical Report TR-87.4, http://www.hpl.hp.com/techreports/tandem/TR-87.4.pdf

[3] Independent article comparing Shared Nothing and Shared Disk (http://www.benstopford.com/2009/11/24/understanding-the-shared-nothing-architecture/)

[4] Article on Shared Nothing from the point of view of a Shared Nothing Vendor (http://db.csail.mit.edu/madden/high_perf.pdf)(PDF)

Shard (database architecture)

A **database shard** is a horizontal partition in a database or search engine. Each individual partition is referred to as a **shard** or **database shard**.

Database architecture

Horizontal partitioning is a database design principle whereby *rows* of a database table are held separately, rather than being split into columns (which is what normalization and vertical partitioning do, to differing extents). Each partition forms part of a **shard**, which may in turn be located on a separate database server or physical location.

There are numerous advantages to this partitioning approach. Since the tables are divided and distributed into multiple servers, the total number of rows in each table in each database is reduced. This reduces index size, which generally improves search performance. A database shard can be placed on separate hardware, and multiple shards can be placed on multiple machines. This enables a distribution of the database over a large number of machines, which means that the database performance can be spread out over multiple machines, greatly improving performance. In addition, if the database shard is based on some real-world segmentation of the data (e.g., European customers v. American customers) then it may be possible to infer the appropriate shard membership easily and automatically, and query only the relevant shard.

In practice, sharding is far more complex. Although it has been done for a long time by hand-coding (especially where rows have an obvious grouping, as per the example above), this is often inflexible. There is a desire to support sharding automatically, both in terms of adding code support for it, and for identifying candidates to be sharded separately. Consistent hashing is one form of automatic sharding to spread large loads across multiple smaller services and servers.

Where distributed computing is used to separate load between multiple servers (either for performance or reliability reasons), a shard approach may also be useful.

Shards compared to horizontal partitioning

Horizontal partitioning splits one or more tables by row, usually within a *single* instance of a schema and a database server. It may offer an advantage by reducing index size (and thus search effort) provided that there is some obvious, robust, implicit way to identify in which table a particular row will be found, without first needing to search the index, e.g., the classic example of the '`CustomersEast`' and '`CustomersWest`' tables, where their zip code already indicates where they will be found.

Sharding goes beyond this: it partitions the problematic table(s) in the same way, but it does this across potentially *multiple* instances of the schema. The obvious advantage would be that search load for the large partitioned table can now be split across multiple servers (logical or physical), not just multiple indexes on the same logical server.

Splitting shards across multiple isolated instances requires more than simple horizontal partitioning. The hoped-for gains in efficiency would be lost, if querying the database required *both* instances to be queried, just to retrieve a simple dimension table. Beyond partitioning, sharding thus splits large partitionable tables across the servers, while smaller tables are replicated as complete units.

This is also why sharding is related to a shared nothing architecture—once sharded, each shard can live in a totally separate logical schema instance / physical database server / data center / continent. There is no ongoing need to retain shared access (from between shards) to the other unpartitioned tables in other shards.

This makes replication across multiple servers easy (simple horizontal partitioning does not). It is also useful for worldwide distribution of applications, where communications links between data centers would otherwise be a bottleneck.

There is also a requirement for some notification and replication mechanism between schema instances, so that the unpartitioned tables remain as closely synchronized as the application demands. This is a complex choice in the architecture of sharded systems: approaches range from making these effectively read-only (updates are rare and batched), to dynamically replicated tables (at the cost of reducing some of the distribution benefits of sharding) and many options in between.

Support for shards

CUBRID

: CUBRID supports sharding from version 9.0

dbShards

: CodeFutures dbShards is a product dedicated to database shards.

eXtreme Scale

: eXtreme Scale is a cross-process in-memory key/value datastore (a variety of NoSQL datastore). It uses sharding to achieve scalability across processes for both data and MapReduce-style parallel processing.[1]

Hibernate ORM

: Hibernate Shards provides support for shards, although there has been little activity since 2007.

IBM Informix

: IBM supports sharding in Informix since version 12.1 xC1 as part of the MACH11 technology. Informix 12.10 xC2 added full compatibility with MongoDB drivers, allowing the mix of regular relational tables with NoSQL collections, still supporting sharding, failover and ACID properties.

MongoDB

: MongoDB supports sharding from version 1.6

MySQL Cluster

Auto-Sharding: Database is automatically and transparently partitioned across low cost commodity nodes, allowing scale-out of read and write queries, without requiring changes to the application.

Plugin for Grails

Grails supports sharding using the Grails Sharding Plugin.

Ruby ActiveRecord

Octopus works as a database sharding and replication extension for the ActiveRecord ORM.

ScaleBase's Data Traffic Manager

ScaleBase's Data Traffic Manager is a software product dedicated to automating MySQL database sharding without requiring changes to applications.

Solr Search Server

Solr enterprise search server provides sharding capabilities.

Spanner

Spanner is Google's global scale distributed database that shards data across multiple Paxos state machines to scale to "millions of machines across hundreds of datacenters and trillions of database rows".

SQLAlchemy ORM

SQLAlchemy is an object-relational mapper for the Python programming language that provides sharding capabilities.

SQL Azure

Microsoft supports sharding in SQL Azure through "federations".

Disadvantages of sharding

Sharding a database table before it has been optimized locally causes premature complexity. Sharding should be used only when all other options for optimization are inadequate. The introduced complexity of database sharding causes the following potential problems:

- **Increased complexity of SQL** - Increased bugs because the developers have to write more complicated SQL to handle sharding logic.
- **Sharding introduces complexity** - The sharding software that partitions, balances, coordinates, and ensures integrity can fail.
- **Single point of failure** - Corruption of one shard due to network/hardware/systems problems causes failure of the entire table.
- **Failover servers more complex** - Failover servers must themselves have copies of the fleets of database shards.
- **Backups more complex** - Database backups of the individual shards must be coordinated with the backups of the other shards.
- **Operational complexity added** - Adding/removing indexes, adding/deleting columns, modifying the schema becomes much more difficult.

These historical complications of do-it-yourself sharding are now being addressed by independent software vendors who provide autosharding solutions.

References

[1] http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=%2Fcom.ibm.websphere.extremescale.over.doc%2Fcxsovwork.html

Quorum (distributed computing)

A **quorum** is the minimum number of votes that a distributed transaction has to obtain in order to be allowed to perform an operation in a distributed system. A **quorum**-based technique is implemented to enforce consistent operation in a distributed system.

Quorum-based techniques in distributed database systems

Quorum-based voting can be used as a replica control method , as well as a commit method to ensure transaction atomicity in the presence of network partitioning.

Quorum-based voting in commit protocols

In a distributed database system, a transaction could be executing its operations at multiple sites. Since atomicity requires every distributed transaction to be atomic, the transaction must have the same fate (commit or abort) at every site. In case of network partitioning, sites are partitioned and the partitions may not be able to communicate with each other. This is where a quorum-based technique comes in. The fundamental idea is that a transaction is executed if the majority of sites vote to execute it.

Every site in the system is assigned a vote V_i. Let us assume that the total number of votes in the system is V and the abort and commit quorums are V_a and V_c, respectively. Then the following rules must be obeyed in the implementation of the commit protocol:

1. $V_a + V_c > V$, where $0 < V_c, V_a \leq V$.
2. Before a transaction commits, it must obtain a commit quorum V_c.
   The total of at least one site that is prepared to commit and zero or more sites waiting $\geq V_c$.
3. Before a transaction aborts, it must obtain an abort quorum V_a
   The total of zero or more sites that are prepared to abort or any sites waiting $\geq V_a$.

The first rule ensures that a transaction cannot be committed and aborted at the same time. The next two rules indicate the votes that a transaction has to obtain before it can terminate one way or the other.

Quorum-based voting for replica control

In replicated databases, a data object has copies present at several sites. To ensure serializability, no two transactions should be allowed to read or write a data item concurrently. In case of replicated databases, a quorum-based replica control protocol can be used to ensure that no two copies of a data item are read or written by two transactions concurrently.

The quorum-based voting for replica control is due to [Gifford, 1979] . Each copy of a replicated data item is assigned a vote. Each operation then has to obtain a *read quorum* (V_r) or a *write quorum* (V_w) to read or write a data item, respectively. If a given data item has a total of V votes, the quorums have to obey the following rules:

1. $V_r + V_w > V$
2. $V_w > V/2$

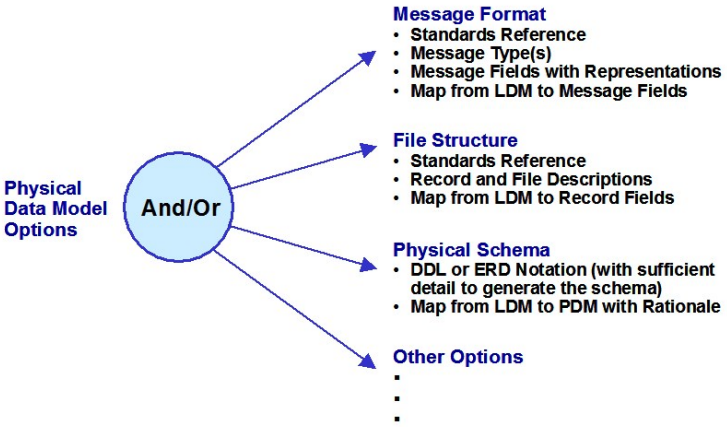
The first rule ensures that a data item is not read and written by two transactions concurrently. The second rule ensures that two write operations from two transactions cannot occur concurrently on the same data item. The two rules ensure that one-copy serializability is maintained.

References

Physical Design

Physical data model

A **physical data model** (or database design) is a representation of a data design which takes into account the facilities and constraints of a given database management system. In the lifecycle of a project it typically derives from a logical data model, though it may be reverse-engineered from a given database implementation. A complete physical data model will include all the database artifacts required to create relationships between tables or to achieve performance goals, such as indexes, constraint definitions, linking tables, partitioned tables or clusters. Analysts can usually use a physical data model to calculate storage estimates; it may include specific storage allocation details for a given database system.



Physical Data Model Options.[1]

As of 2012[2] seven main databases dominate the commercial marketplace: Informix, Oracle, Postgres, SQL Server, Sybase, DB2 and MySQL. Other RDBMS systems tend either to be legacy databases or used within academia such as universities or further education colleges. Physical data models for each implementation would differ significantly, not least due to underlying operating-system requirements that may sit underneath them. For example: SQL Server runs only on Microsoft Windows operating-systems, while Oracle and MySQL can run on Solaris, Linux and other UNIX-based operating-systems as well as on Windows. This means that the disk requirements, security requirements and many other aspects of a physical data model will be influenced by the RDBMS that a database administrator (or an organization) chooses to use.

References

[1] FEA Consolidated Reference Model Document (http://georgewbush-whitehouse.archives.gov/omb/egov/documents/CRM.PDF). whitehouse.gov May 2005. p.91.

[2] http://en.wikipedia.org/w/index.php?title=Physical_data_model&action=edit

Physical schema

Physical schema is a term used in data management to describe how data is to be represented and stored (files, indices, *et al.*) in secondary storage using a particular database management system (DBMS) (e.g., Oracle RDBMS, Sybase SQL Server, etc.).

In the ANSI/SPARC Architecture three schema approach, the *internal schema* is the view of data that involved data management technology. This is as opposed to an *external schema* that reflects an individual's view of the data, or the *conceptual schema* that is the integration of a set of external schemas.

Subsequently[*citation needed*] the internal schema was recognized to have two parts:

The logical schema was the way data were represented to conform to the constraints of a particular approach to database management. At that time the choices were hierarchical and network. Describing the logical schema, however, still did not describe how physically data would be stored on disk drives. That is the domain of the *physical schema*. Now logical schemas describe data in terms of relational *tables and columns*, object-oriented *classes*, and XML *tags*.

A single set of tables, for example, can be implemented in numerous ways, up to and including an architecture where table rows are maintained on computers in different countries.

Storage model

A **storage model** is a model that captures key *physical* aspects of data structure in a data store.

On the other hand, a data model is a model that captures key *logical* aspects of data structure in a database.

Storage block

A **storage block** is a physical sector on the surface of a disk or diskette. It is the smallest unit of transference between the main memory and a given disk drive.

In the IBM mainframe terminology, a **block** is the minimal physical division of data in a disk drive, either as used on a Fixed Block Architecture (FBA) disk or in the Cylinder-Head-Record (CCHHRR) addressing mode of a Count-Key-Data (CKD) disk.[*citation needed*] Blocks are separated on the track by "inter-record gaps" (approx. 6 bytes on IBM 3380 and IBM 3390 disk drive models). The number of blocks per tracks depends on the block size and the specific gap size associated to each block. Each block can contain a discrete quantity of logical records or it can be empty. Logical records are defined by the database designer or application designer; see Data set (IBM mainframe).

Tablespace

A **tablespace** is a storage location where the actual data underlying database objects can be kept. It provides a layer of abstraction between physical and logical data, and serves to allocate storage for all DBMS managed segments. (A database segment is a database object which occupies physical space such as table data and indexes.) Once created, a tablespace can be referred to by name when creating database segments.

Tablespaces specify only the database storage locations, not the logical database structure, or database schema. For instance, different objects in the same schema may have different underlying tablespaces. Similarly, a tablespace may service segments for more than one schema. Sometimes it can be used to specify schema as to form a bond between logical and physical data.

By using tablespaces, an administrator can control the disk layout of an installation. A common use of tablespaces is to optimize performance. For example, a heavily used index can be placed on a fast SCSI disk. On the other hand, a database table which contains archived data that is rarely accessed could be stored on a less expensive but slower IDE disk.

While it is common for tablespaces to store their data in a filesystem file, a single file must be part of a single tablespace. Some database management systems allow tablespaces to be configured directly over operating-system device entries, called raw devices, providing better performance by avoiding the OS filesystem overheads.

Oracle stores data logically in tablespaces and physically in datafiles associated with the corresponding tablespace.

References

Database tuning

Database tuning describes a group of activities used to optimize and homogenize the performance of a database. It usually overlaps with query tuning, but refers to design of the database files, selection of the database management system (DBMS) application, and configuration of the database's environment (operating system, CPU, etc.).

Database tuning aims to maximize use of system resources to perform work as efficiently and rapidly as possible. Most systems are designed to manage their use of system resources, but there is still much room to improve their efficiency by customizing their settings and configuration for the database and the DBMS.

I/O tuning

Hardware and software configuration of disk subsystems are examined: RAID levels and configuration, block and stripe size allocation, and the configuration of disks, controller cards, storage cabinets, and external storage systems such as SANs. Transaction logs and temporary spaces are heavy consumers of I/O, and affect performance for all users of the database. Placing them appropriately is crucial.

Frequently joined tables and indexes are placed so that as they are requested from file storage, they can be retrieved in parallel from separate disks simultaneously. Frequently accessed tables and indexes are placed on separate disks to balance I/O and prevent read queuing.

DBMS tuning

DBMS tuning refers to tuning of the DBMS and the configuration of the memory and processing resources of the computer running the DBMS. This is typically done through configuring the DBMS, but the resources involved are shared with the host system.

Tuning the DBMS can involve setting the recovery interval (time needed to restore the state of data to a particular point in time), assigning parallelism (the breaking up of work from a single query into tasks assigned to different processing resources), and network protocols used to communicate with database consumers.

Memory is allocated for data, execution plans, procedure cache, and work space[*clarify*]. It is much faster to access data in memory than data on storage, so maintaining a sizable cache of data makes activities perform faster. The same consideration is given to work space. Caching execution plans and procedures means that they are reused instead of recompiled when needed. It is important to take as much memory as possible, while leaving enough for other processes and the OS to use without excessive paging of memory to storage.

Processing resources are sometimes assigned to specific activities to improve concurrency. On a server with eight processors, six could be reserved for the DBMS to maximize available processing resources for the database.

Database maintenance

Database maintenance includes backups, column statistics updates, and defragmentation of data inside the database files.

On a heavily used database, the transaction log grows rapidly. Transaction log entries must be removed from the log to make room for future entries. Frequent transaction log backups are smaller, so they interrupt database activity for shorter periods of time.

DBMS use statistic histograms to find data in a range against a table or index. Statistics updates should be scheduled frequently and sample as much of the underlying data as possible. Accurate and updated statistics allow query engines to make good decisions about execution plans, as well as efficiently locate data.

Defragmentation of table and index data increases efficiency in accessing data. The amount of fragmentation depends on the nature of the data, how it is changed over time, and the amount of free space in database pages to accept inserts of data without creating additional pages.

References

Database dump

For information on obtaining the Wikipedia database, see Wikipedia:Database download.

A **database dump** contains a record of the table structure and/or the data from a database and is usually in the form of a list of SQL statements. A database dump is most often used for backing up a database so that its contents can be restored in the event of data loss. Corrupted databases can often be recovered by analysis of the dump. Database dumps are often published by free software and free content projects, to allow reuse or forking of the database.

Example

```
-- Database
CREATE DATABASE `example`;
USE `example`;

-- Table structure for table `users`
CREATE TABLE `users` (
  `id` int(8) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(16) NOT NULL,
  `password` varchar(16) NOT NULL,
  PRIMARY KEY (`id`)
);

-- Data for table `users`
INSERT INTO `users` VALUES (1,'alice','secret'),(2,'bob','secret');
```

References

expdp or impdp or rman in oracle

External links

- mysqldump — A Database Backup Program [1]
- PostgreSQL dump backup methods [2], for PostgreSQL databases.

References

[1] http://dev.mysql.com/doc/refman/5.0/en/mysqldump.html
[2] http://www.postgresql.org/docs/8.2/interactive/backup-dump.html

Spindling

In computers **spindling** is the allocation of different files (e.g., the data files and index files of a database) on different hard disks. This practice usually reduces contention for read or write resources, thus increasing the system's performance.

The word comes from spindle, the axis on which the hard disks spin.

Data Mapping and Integration Tasks

Data mapping

Data transformation/Source transformation
Concepts
• metadata • data mapping • data transformation • model transformation
Languages
• ATL • AWK • MOFM2T • QVT • TXL • XML languages
Techniques and transforms
• identity • synthesis • refinement
Applications
• data migration • data conversion • ETL • program transformation
Application fields
• Data warehouse • Software engineering Software languages macro preprocessing template
• v • t • e [1]

Data mapping is the process of creating data element mappings between two distinct data models. Data mapping is used as a first step for a wide variety of data integration tasks including:

- Data transformation or data mediation between a data source and a destination
- Identification of data relationships as part of data lineage analysis

- Discovery of hidden sensitive data such as the last four digits social security number hidden in another user id as part of a data masking or de-identification project
- Consolidation of multiple databases into a single data base and identifying redundant columns of data for consolidation or elimination

For example, a company that would like to transmit and receive purchases and invoices with other companies might use data mapping to create data maps from a company's data to standardized ANSI ASC X12 messages for items such as purchase orders and invoices.

Standards

X12 standards are generic Electronic Data Interchange (EDI) standards designed to allow a company to exchange data with any other company, regardless of industry. The standards are maintained by the Accredited Standards Committee X12 (ASC X12), with the American National Standards Institute (ANSI) accredited to set standards for EDI. The X12 standards are often called **ANSI ASC X12 standards**.

In the future, tools based on semantic web languages such as Resource Description Framework (RDF), the Web Ontology Language (OWL) and standardized metadata registry will make data mapping a more automatic process. This process will be accelerated if each application performed metadata publishing. Full automated data mapping is a very difficult problem (see Semantic translation).

Hand-coded, graphical manual

Data mappings can be done in a variety of ways using procedural code, creating XSLT transforms or by using graphical mapping tools that automatically generate executable transformation programs. These are graphical tools that allow a user to "draw" lines from fields in one set of data to fields in another. Some graphical data mapping tools allow users to "Auto-connect" a source and a destination. This feature is dependent on the source and destination data element name being the same. Transformation programs are automatically created in SQL, XSLT, Java programming language or C++. These kinds of graphical tools are found in most ETL Tools (Extract, Transform, Load Tools) as the primary means of entering data maps to support data movement.

Data-driven mapping

This is the newest approach in data mapping and involves simultaneously evaluating actual data values in two data sources using heuristics and statistics to automatically discover complex mappings between two data sets. This approach is used to find transformations between two data sets and will discover substrings, concatenations, arithmetic, case statements as well as other kinds of transformation logic. This approach also discovers data exceptions that do not follow the discover....

Semantic mapping

Semantic mapping is similar to the auto-connect feature of data mappers with the exception that a metadata registry can be consulted to look up data element synonyms. For example, if the source system lists FirstName but the destination lists PersonGivenName, the mappings will still be made if these data elements are listed as synonyms in the metadata registry. Semantic mapping is only able to discover exact matches between columns of data and will not discover any transformation logic or exceptions between columns.

References

[1] http://en.wikipedia.org/w/index.php?title=Template:Data_transformation&action=edit

Bibliography

- Bogdan Alexe, Laura Chiticariu, Renée J. Miller, Wang Chiew Tan: Muse: Mapping Understanding and deSign by Example (http://dx.doi.org/10.1109/ICDE.2008.4497409). ICDE 2008: 10-19
- Khalid Belhajjame, Norman W. Paton, Suzanne M. Embury, Alvaro A. A. Fernandes, Cornelia Hedeler: Feedback-Based Annotation, Selection and Refinement of Schema Mappings for Dataspaces (http://www.edbt.org/Proceedings/2010-Lausanne/edbt/papers/p0573-Belhajjame.pdf). EDBT 2010: 573-584
- Laura Chiticariu, Wang Chiew Tan: Debugging Schema Mappings with Routes (http://www.vldb.org/conf/2006/p79-chiticariu.pdf). VLDB 2006: 79-90
- Ronald Fagin, Laura M. Haas, Mauricio A. Hernández, Renée J. Miller, Lucian Popa, Yannis Velegrakis: Clio: Schema Mapping Creation and Data Exchange. Conceptual Modeling: Foundations and Applications 2009: 198-236 (http://dx.doi.org/10.1007/978-3-642-02463-4_12)
- Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, Lucian Popa: Data exchange: semantics and query answering (http://dx.doi.org/10.1016/j.tcs.2004.10.033). Theor. Comput. Sci. 336(1): 89-124 (2005)
- Maurizio Lenzerini: Data Integration: A Theoretical Perspective (http://www.acm.org/sigs/sigmod/pods/proc02/papers/233-Lenzerini.pdf). PODS 2002: 233-246
- Renée J. Miller, Laura M. Haas, Mauricio A. Hernández: Schema Mapping as Query Discovery (http://www.informatik.uni-trier.de/~ley/db/conf/vldb/MillerHH00.html). VLDB 2000: 77-88

Semantic integration

Semantic integration is the process of interrelating information from diverse sources, for example calendars and to do lists, email archives, presence information (physical, psychological, and social), documents of all sorts, contacts (including social graphs), search results, and advertising and marketing relevance derived from them. In this regard, semantics focuses on the organization of and action upon information by acting as a intermediary between heterogeneous data sources, which may conflict not only by structure but also context or value.

Applications and Methods

In enterprise application integration (EAI), semantic integration can facilitate or even automate the communication between computer systems using metadata publishing. Metadata publishing potentially offers the ability to automatically link ontologies. One approach to (semi-)automated ontology mapping requires the definition of a semantic distance or its inverse, semantic similarity and appropriate rules. Other approaches include so-called *lexical methods,* as well as methodologies that rely on exploiting the structures of the ontologies. For explicitly stating similarity/equality, there exist special properties or relationships in most ontology languages. OWL, for example has “sameIndividualAs” or “same-ClassAs”.

Eventually system designs may see the advent of composable architectures where published semantic-based interfaces are joined together to enable new and meaningful capabilities[*citation needed*]. These could predominately be described by means of design-time declarative specifications, that could ultimately be rendered and executed at run-time[*citation needed*].

Semantic integration can also be used to facilitate design-time activities of interface design and mapping. In this model, semantics are only explicitly applied to design and the run-time systems work at the syntax level[*citation needed*]. This "early semantic binding" approach can improve overall system performance while retaining the benefits of semantic driven design[*citation needed*].

Examples

The Pacific Symposium on Biocomputing has been a venue for the popularization of the ontology mapping task in the biomedical domain, and a number of papers on the subject can be found in its proceedings.

References

External links

- Semantic Integration: Loosely Coupling the Meaning of Data (http://www.zapthink.com/report.html?id=ZapFlash-08082003)
- Ontology Mapping: The State of the Art (http://drops.dagstuhl.de/opus/volltexte/2005/40/) (2005 paper)
- 2010 paper by Carl Hewitt (http://arxiv.org/ftp/arxiv/papers/0901/0901.4934.pdf)
- OpenCyc to Oracle Interface (http://wwwhome.portavita.nl/~yeb/ooi.pdf)

Semantic translation

Semantic translation is the process of using semantic information to aid in the translation of data in one representation or data model to another representation or data model. Semantic translation takes advantage of semantics that associate meaning with individual data elements in one dictionary to create an equivalent meaning in a second system.

An example of semantic translation is the conversion of XML data from one data model to a second data model using formal ontologies for each system such as the Web Ontology Language (OWL). This is frequently required by intelligent agents that wish to perform searches on remote computer systems that use different data models to store their data elements. The process of allowing a single user to search multiple systems with a single search request is also known as federated search.

Semantic translation should be differentiated from data mapping tools that do simple one-to-one translation of data from one system to another without actually associating meaning with each data element.

Semantic translation requires that data elements in the source and destination systems have "semantic mappings" to a central registry or registries of data elements. The simplest mapping is of course where there is equivalence. There are three types of Semantic equivalence:

- **Class Equivalence** - indicating that class or "concepts" are equivalent. For example: "Person" is the same as "Individual"
- **Property Equivalence** - indicating that two properties are equivalent. For example: "PersonGivenName" is the same as "FirstName"
- **Instance Equivalence** - indicating that two individual instances of objects are equivalent. For example: "Dan Smith" is the same person as "Daniel Smith"

Semantic translation is very difficult if the terms in a particular data model do not have direct one-to-one mappings to data elements in a foreign data model. In that situation an alternative approach must be used to find mappings from the original data to the foreign data elements. This problem can be alleviated by centralized metadata registries that use the ISO-11179 standards such as the National Information Exchange Model (NIEM).

Record linkage

Record linkage (RL) refers to the task of finding records in a data set that refer to the same entity across different data sources (e.g., data files, books, websites, databases). Record linkage is necessary when joining data sets based on entities that may or may not share a common identifier (e.g., database key, URI, National identification number), as may be the case due to differences in record shape, storage location, and/or curator style or preference. A data set that has undergone RL-oriented reconciliation may be referred to as being *cross-linked*. Record Linkage is called Data Linkage in many jurisdictions, but is the same process.

History

The initial idea of record linkage goes back to Halbert L. Dunn in his 1946 article titled "Record Linkage" published in the *American Journal of Public Health*. Howard Borden Newcombe laid the probabilistic foundations of modern record linkage theory in a 1959 article in *Science*, which were then formalized in 1969 by Ivan Fellegi and Alan Sunter who proved that the probabilistic decision rule they described was optimal when the comparison attributes were conditionally independent. Their pioneering work "A Theory For Record Linkage" remains the mathematical foundation for many record linkage applications even today.

Since the late 1990s, various machine learning techniques have been developed that can, under favorable conditions, be used to estimate the conditional probabilities required by the Fellegi-Sunter (FS) theory. Several researchers have reported that the conditional independence assumption of the FS algorithm is often violated in practice; however, published efforts to explicitly model the conditional dependencies among the comparison attributes have not resulted in an improvement in record linkage quality. [*citation needed*]

Record linkage can be done entirely without the aid of a computer, but the primary reasons computers are often used for record linkage are to reduce or eliminate manual review and to make results more easily reproducible. Computer matching has the advantages of allowing central supervision of processing, better quality control, speed, consistency, and better reproducibility of results.

Naming conventions

"Record linkage" is the term used by statisticians, epidemiologists, and historians, among others, to describe the process of joining records from one data source with another that describe the same entity. Commercial mail and database applications refer to it as "merge/purge processing" or "list washing". Computer scientists often refer to it as "data matching" or as the "object identity problem". Other names used to describe the same concept include: "coreference/entity/identity/name/record resolution", "entity disambiguation/linking", "duplicate detection", "deduplication", "record matching", "(reference) reconciliation", "object identification", "data/information integration", and "conflation".[1] This profusion of terminology has led to few cross-references between these research communities.[2]

While they share similar names, record linkage and Linked Data are two separate concepts. Whereas record linkage focuses on the more narrow task of identifying matching entities across different data sets, Linked Data focuses on the broader methods of structuring and publishing data to facilitate the discovery of related information.

Methods

Data preprocessing

Record linkage is highly sensitive to the quality of the data being linked, so all data sets under consideration (particularly their key identifier fields) should ideally undergo a data quality assessment prior to record linkage. Many key identifiers for the same entity can be presented quite differently between (and even within) data sets, which can greatly complicate record linkage unless understood ahead of time. For example, key identifiers for a man named William J. Smith might appear in three different data sets as so:

Data set	Name	Date of birth	City of residence
Data set 1	William J. Smith	1/2/73	Berkeley, California
Data set 2	Smith, W. J.	1973.1.2	Berkeley, CA
Data set 3	Bill Smith	Jan 2, 1973	Berkeley, Calif.

In this example, the different formatting styles lead to records that look different but in fact all refer to the same entity with the same logical identifier values. Most, if not all, record linkage strategies would result in more accurate linkage if these values were first *normalized* or *standardized* into a consistent format (e.g., all names are "Surname, Given name", all dates are "YYYY/MM/DD", and all cities are "Name, 2-letter state abbreviation"). Standardization can be accomplished through simple rule-based data transformations or more complex procedures such as lexicon-based tokenization and probabilistic hidden Markov models. Several of the packages listed in the *Software Implementations* section provide some of these features to simplify the process of data standardization.

Identity resolution

Identity resolution is an operational intelligence process, typically powered by an identity resolution engine or middleware, whereby organizations can connect disparate data sources with a view to understanding possible identity matches and non-obvious relationships across multiple data silos. It analyzes all of the information relating to individuals and/or entities from multiple sources of data, and then applies likelihood and probability scoring to determine which identities are a match and what, if any, non-obvious relationships exist between those identities.

Identity resolution engines are typically used to uncover risk, fraud, and conflicts of interest, but are also useful tools for use within Customer Data Integration (CDI) and Master Data Management (MDM) requirements. Typical uses for identity resolution engines include terrorist screening, insurance fraud detection, USA Patriot Act compliance, Organized retail crime ring detection and applicant screening.

For example: Across different data silos - employee records, vendor data, watch lists, etc. - an organization may have several variations of an identity named ABC, which may or may not be the same individual. These entries may, in fact, appear as ABC1, ABC2, or ABC3 within those data sources. By comparing similarities between underlying attributes such as address, date of birth, or social security number, the user can eliminate some possible matches and confirm others as very likely matches.

Identity resolution engines then apply rules, based on common sense logic, to identify hidden relationships across the data. In the example above, perhaps ABC1 and ABC2 are not the same individual, but rather two distinct people who share common attributes such as address or phone number.

Data Matching

While entity resolution solutions include data matching technology, many data matching offerings do not fit the definition of identity (or entity) resolution. Here are four factors that distinguish entity resolution from data matching, according to John Talburt, director of the UALR Center for Advanced Research in Entity Resolution and Information Quality:

- Works with both structured and unstructured records, and it entails the process of extracting references when the sources are unstructured or semi-structured
- Uses elaborate business rules and concept models to deal with missing, conflicting, and corrupted information
- Utilizes non-matching, asserted linking (associate) information in addition to direct matching
- Uncovers non-obvious relationships and association networks (i.e. who's associated with whom)

In contrast to data quality products, more powerful identity resolution engines also include a rules engine and workflow process, which apply business intelligence to the resolved identities and their relationships. These advanced technologies make automated decisions and impact business processes in real time, limiting the need for human intervention.

Deterministic record linkage

The simplest kind of record linkage, called *deterministic* or *rules-based record linkage*, generates links based on the number of individual identifiers that match among the available data sets. Two records are said to match via a deterministic record linkage procedure if all or some identifiers (above a certain threshold) are identical. Deterministic record linkage is a good option when the entities in the data sets are identified by a common identifier, or when there are several representative identifiers (e.g., name, date of birth, and sex when identifying a person) whose quality of data is relatively high.

As an example, consider two standardized data sets, Set A and Set B, that contain different bits of information about patients in a hospital system. The two data sets identify patients using a variety of identifiers: Social Security Number (SSN), name, date of birth (DOB), sex, and ZIP code (ZIP). The records in two data sets (identified by the "#" column) are shown below:

Data Set	#	SSN	Name	DOB	Sex	ZIP
Set A	1	000956723	Smith, William	1973/01/02	Male	94701
	2	000956723	Smith, William	1973/01/02	Male	94703
	3	000005555	Jones, Robert	1942/08/14	Male	94701
	4	123001234	Sue, Mary	1972/11/19	Female	94109
Set B	1	000005555	Jones, Bob	1942/08/14		
	2		Smith, Bill	1973/01/02	Male	94701

The most simple deterministic record linkage strategy would be to pick a single identifier that is assumed to be uniquely identifying, say SSN, and declare that records sharing the same value identify the same person while records not sharing the same value identify different people. In this example, deterministic linkage based on SSN would create entities based on A1 and A2; A3 and B1; and A4. While A1, A2, and B2 appear to represent the same entity, B2 would not be included into the match because it is missing a value for SSN.

Handling exceptions such as missing identifiers involves the creation of additional record linkage rules. One such rule in the case of missing SSN might be to compare name, date of birth, sex, and ZIP code with other records in hopes of finding a match. In the above example, this rule would still not match A1/A2 with B2 because the names are still slightly different: standardization put the names into the proper (Surname, Given name) format but could not discern "Bill" as a nickname for "William". Running names through a phonetic algorithm such as Soundex, NYSIIS,

or metaphone, can help to resolve these types of problems (though it may still stumble over surname changes as the result of marriage or divorce), but then B2 would be matched only with A1 since the ZIP code in A2 is different. Thus, another rule would need to be created to determine whether differences in particular identifiers are acceptable (such as ZIP code) and which are not (such as date of birth).

As this example demonstrates, even a small decrease in data quality or small increase in the complexity of the data can result in a very large increase in the number of rules necessary to link records properly. Eventually, these linkage rules will become too numerous and interrelated to build without the aid of specialized software tools. In addition, linkage rules are often specific to the nature of the data sets they are designed to link together. One study was able to link the Social Security Death Master File with two hospital registries from the Midwestern United States using SSN, NYSIIS-encoded first name, birth month, and sex, but these rules may not work as well with data sets from other geographic regions or with data collected on younger populations. Thus, continuous maintenance testing of these rules is necessary to ensure they continue to function as expected as new data enter the system and need to be linked. New data that exhibit different characteristics than was initially expected could require a complete rebuilding of the record linkage rule set, which could be a very time-consuming and expensive endeavor.

Probabilistic record linkage

Probabilistic record linkage, sometimes called *fuzzy matching* (also *probabilistic merging* or *fuzzy merging* in the context of merging of databases), takes a different approach to the record linkage problem by taking into account a wider range of potential identifiers, computing weights for each identifier based on its estimated ability to correctly identify a match or a non-match, and using these weights to calculate the probability that two given records refer to the same entity. Record pairs with probabilities above a certain threshold are considered to be matches, while pairs with probabilities below another threshold are considered to be non-matches; pairs that fall between these two thresholds are considered to be "possible matches" and can be dealt with accordingly (e.g., human reviewed, linked, or not linked, depending on the requirements). Whereas deterministic record linkage requires a series of potentially complex rules to be programmed ahead of time, probabilistic record linkage methods can be "trained" to perform well with much less human intervention.

Many probabilistic record linkage algorithms assign match/non-match weights to identifiers by means of u probabilities and m probabilities. The u probability is the probability that an identifier in two *non-matching* records will agree purely by chance. For example, the u probability for birth month (where there are twelve values that are approximately uniformly distributed) is $1/12 \approx 0.083$; identifiers with values that are not uniformly distributed will have different u probabilities for different values (possibly including missing values). The m probability is the probability that an identifier in *matching* pairs will agree (or be sufficiently similar, such as strings with high Jaro-Winkler distance or low Levenshtein distance). This value would be 1.0 in the case of perfect data, but given that this is rarely (if ever) true, it can instead be estimated. This estimation may be done based on prior knowledge of the data sets, by manually identifying a large number of matching and non-matching pairs to "train" the probabilistic record linkage algorithm, or by iteratively running the algorithm to obtain closer estimations of the m probability. If a value of 0.95 were to be estimated for the m probability, then the match/non-match weights for the birth month identifier would be:

Outcome	Proportion of links	Proportion of non-links	Frequency ratio	Weight
Match	$m = 0.95$	$u \approx 0.083$	$m/u \approx 11.4$	$\ln(m/u)/\ln(2) \approx 3.51$
Non-match	$1-m = 0.05$	$1-u \approx 0.917$	$(1-m)/(1-u) \approx 0.0545$	$\ln((1-m)/(1-u))/\ln(2) \approx -4.20$

The same calculations would be done for all other identifiers under consideration to find their match/non-match weights. Then, the identifiers of one record would be compared with the identifiers with every other record to compute the total weight: the *match* weight is added to the running total whenever a pair of identifiers agree, while the *non-match* weight is added (i.e. the running total decreases) whenever the pair of identifiers disagrees. The resulting total weight is then compared to the aforementioned thresholds to determine whether the pair should be linked, non-linked, or set aside for special consideration (e.g. manual validation).

Determining where to set the match/non-match thresholds is a balancing act between obtaining an acceptable sensitivity (or *recall*, the proportion of truly matching records that are linked by the algorithm) and positive predictive value (or *precision*, the proportion of records linked by the algorithm that truly do match). Various manual and automated methods are available to predict the best thresholds, and some record linkage software packages have built-in tools to help the user find the most acceptable values. Because this can be a very computationally demanding task, particularly for large data sets, a technique known as *blocking* is often used to improve efficiency. Blocking attempts to restrict comparisons to just those records for which one or more particularly discriminating identifiers agree, which has the effect of increasing the positive predictive value (precision) at the expense of sensitivity (recall). For example, blocking based on a phonetically coded surname and ZIP code would reduce the total number of comparisons required and would improve the chances that linked records would be correct (since two identifiers already agree), but would potentially miss records referring to the same person whose surname or ZIP code was different (due to marriage or relocation, for instance). Blocking based on birth month, a more stable identifier that would be expected to change only in the case of data error, would provide a more modest gain in positive predictive value and loss in sensitivity, but would create only twelve distinct groups which, for extremely large data sets, may not provide much net improvement in computation speed. Thus, robust record linkage systems often use multiple blocking passes to group data in various ways in order to come up with groups of records that should be compared to each other.

Machine learning

In recent years, a variety of machine learning techniques have been used in record linkage. It has been recognizedWikipedia:Manual of Style/Words to watch#Unsupported attributions that probabilistic record linkage is equivalent to the "Naive Bayes" algorithm in the field of machine learning,[3] and suffers from the same assumption of the independence of its features (an assumption that is typically not true).[4][5] Higher accuracy can often be achieved by using various other machine learning techniques, including a single-layer perceptron.[6]

Mathematical model

In an application with two files, A and B, denote the rows (*records*) by $\alpha(a)$ in file A and $\beta(b)$ in file B. Assign K *characteristics* to each record. The set of records that represent identical entities is defined by

$M = \{(a,b); a = b; a \in A; b \in B\}$

and the complement of set M, namely set U representing different entities is defined as

$U = \{(a,b); a \neq b; a \in A, b \in B\}$.

A vector, γ is defined, that contains the coded agreements and disagreements on each characteristic:

$\gamma[\alpha(a), \beta(b)] = \{\gamma^1[\alpha(a), \beta(b)], ..., \gamma^K[\alpha(a), \beta(b)]\}$

where K is a subscript for the characteristics (sex, age, marital status, etc.) in the files. The conditional probabilities of observing a specific vector γ given $(a,b) \in M$, $(a,b) \in U$ are defined as

$$m(\gamma) = P\left\{\gamma\left[\alpha(a), \beta(b)\right] | (a,b) \in M\right\} = \sum_{(a,b)\in M} P\left\{\gamma\left[\alpha(a), \beta(b)\right]\right\} \cdot P\left[(a,b)|M\right]$$

and

$$u(\gamma) = P\left\{\gamma\left[\alpha(a), \beta(b)\right] | (a,b) \in U\right\} = \sum_{(a,b)\in U} P\left\{\gamma\left[\alpha(a), \beta(b)\right]\right\} \cdot P\left[(a,b)|U\right], \text{respectively.}$$

Applications

Master data management

Most Master data management (MDM) products use a record linkage process to identify records from different sources representing the same real-world entity. This linkage is used to create a "golden master record" containing the cleaned, reconciled data about the entity. The techniques used in MDM are the same as for record linkage generally.

Data warehousing and business intelligence

Record linkage plays a key role in data warehousing and business intelligence. Data warehouses serve to combine data from many different operational source systems into one logical data model, which can then be subsequently fed into a business intelligence system for reporting and analytics. Each operational source system may have its own method of identifying the same entities used in the logical data model, so record linkage between the different sources becomes necessary to ensure that the information about a particular entity in one source system can be seamlessly compared with information about the same entity from another source system. Data standardization and subsequent record linkage often occur in the "transform" portion of the extract, transform, load (ETL) process.

Historical research

Record linkage is important to social history research since most data sets, such as census records and parish registers were recorded long before the invention of National identification numbers. When old sources are digitized, linking of data sets is a prerequisite for longitudinal study. This process is often further complicated by lack of standard spelling of names, family names that change according to place of dwelling, changing of administrative boundaries, and problems of checking the data against other sources. Record linkage was among the most prominent themes in the History and computing field in the 1980s, but has since been subject to less attention in research.[*citation needed*]

Medical practice and research

Record linkage is an important tool in creating data required for examining the health of the public and of the health care system itself. It can be used to improve data holdings, data collection, quality assessment, and the dissemination of information. Data sources can be examined to eliminate duplicate records, to identify under-reporting and missing cases (e.g., census population counts), to create person-oriented health statistics, and to generate disease registries and health surveillance systems. Some cancer registries link various data sources (e.g., hospital admissions, pathology and clinical reports, and death registrations) to generate their registries. Record linkage is also used to create health indicators. For example, fetal and infant mortality is a general indicator of a country's socioeconomic development, public health, and maternal and child services. If infant death records are matched to birth records, it is possible to use birth variables, such as birth weight and gestational age, along with mortality data, such as cause of death, in analyzing the data. Linkages can help in follow-up studies of cohorts or other groups to determine factors such as vital status, residential status, or health outcomes. Tracing is often needed for follow-up of industrial cohorts,

clinical trials, and longitudinal surveys to obtain the cause of death and/or cancer. An example of a successful and long-standing record linkage system allowing for population-based medical research is the Rochester Epidemiology Project based in Rochester, Minnesota.

Criticism of existing software implementations

The main reasons cited are:

- **Project costs**: costs typically in the hundreds of thousands of dollars
- **Time**: lack of enough time to deal with large-scale data-cleansing software
- **Security**: concerns over sharing information, giving an application access across systems, and effects on legacy systems

Notes and references

[1] http://homes.cs.washington.edu/~pedrod/papers/icdm06.pdf

[2] Cristen, P & T: Febrl - Freely extensible biomedical record linkage (Manual, release 0.3) p.9 (http://datamining.anu.edu.au/linkage.html)

[3] Quass, Dallan, and Starkey, Paul. “Record Linkage for Genealogical Databases,” ACM SIGKDD '03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation, August 24–27, 2003, Washington, D.C.

[4] Langley, Pat, Wayne Iba, and Kevin Thompson. “An Analysis of Bayesian Classifiers,” In Proceedings of the 10th National Conference on Artificial Intelligence, (AAAI-92), AAAI Press/MIT Press, Cambridge, MA, pp. 223-228, 1992.

[5] Michie, D., D. Spiegelhalter, and C. Taylor. Machine Learning, Neural and Statistical Classification, Ellis Horwood, Hertfordshire, England. Book 19, 1994.

[6] Wilson, D. Randall, "Beyond Probabilistic Record Linkage: Using Neural Networks and Complex Features to Improve Genealogical Record Linkage", Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011

External links

- Data Linkage Project at Penn State, USA (http://pike.psu.edu/linkage/)
- Datadecision - Data matching online tool (http://www.datadecision.com)

Metadata discovery

In metadata, **metadata discovery** is the process of using automated tools to discover the semantics of a data element in data sets. This process usually ends with a set of mappings between the data source elements and a centralized metadata registry. Metadata discovery is also known as metadata scanning.

Data source formats for metadata discovery

Data sets may be in a variety of different forms including:

1. Relational databases
2. Spreadsheets
3. XML files
4. Web services
5. Software source code such as Fortran, Jovial, COBOL, Assembler, RPG, PL/1, EasyTrieve, Java, C# or C++ classes, and thousands of other software languages
6. Unstructured text documents such as Microsoft Word or PDF files

A taxonomy of metadata matching algorithms

There are distinct categories of automated metadata discovery:

Lexical Matching

1. **Exact match** - where data element linkages are made based on the exact name of a column in a database, the name of an XML element or a label on a screen. For example if a database column has the name "PersonBirthDate" and a data element in a metadata registry also has the name "PersonBirthDate", automated tools can infer that the column of a database has the same semantics (meaning) as the data element in the metadata registry.
2. **Synonym match** - where the discovery tool is not just given a single name but a set of synonym.
3. **Pattern match** - in this case the tools is given a set of lexical patterns that it can match. For example the tools may search for "*gender*" or "*sex*"

Semantic Matching

Semantic matching attempts to use semantics to associate target data with registered data elements.

1. **Semantic Similarity** - In this algorithm that relies on a database of word conceptual nearness is used. For example the WordNet system can rank how close words are conceptually to each other. For example the terms "Person", "Individual" and "Human" may be highly similar concepts.

Statistical Matching

Statistical matching uses statistics about data sources data itself to derive similarities with registered data elements.

1. **Distinct Value Analysis** - By analyzing all the distinct values in a column the similarity to a registered data element may be made. For example if a column only has two distinct values of 'male' and 'female' this could be mapped to 'PersonGenderCode'.
2. **Data distribution analysis** - By analyzing the distribution of values within a single column and comparing this distribution with known data elements a semantic linkage could be inferred.

Vendors

The following vendors (listed in alphabetical order) provide metadata discovery and metadata mapping software and solutions

- Esquire Innovations (see [7 [1])
- IBM
- InfoLibrarian Corporation (see [2])
- Masai Technologies (see [3])
- MindHARBOR Metadata Database application (see www.mindharbor.com/metadata-database.asp [4])
- Revelytix (see [5])
- Sliver Creek Systems (see [6])
- Sypherlink: Harvester (see [7])
- Unicorn Systems (see [8])

Research

- INDUS project at the Iowa State University (see [9])
- **Mercury** - A Distributed Metadata Management and Data Discovery System developed at the Oak Ridge National Laboratory DAAC (see [10])

References

[1] http://www.esqinc.com/section/products/2/iscrub.html
[2] http://www.infolibcorp.com/scanners.html
[3] http://www.masaitechnologies.com/
[4] http://www.mindharbor.com/metadata-database.asp
[5] http://www.revelytix.com/
[6] http://www.silvercreeksystems.com/
[7] http://www.sypherlink.com/products/index.asp
[8] http://www.unicorn.com/products/unicornsystem/scanners.htm
[9] http://www.cild.iastate.edu/software/indus.html
[10] http://mercury.ornl.gov

- Massive Data Analysis Systems (http://www.sdsc.edu/MDAS/Reports/MDAS.Final.SciTech/techreport-97.1/techreport.html) by San Diego Supercomputer Center June 1997
- IBM Whitepaper on Enterprise Metadata Discovery (http://public.dhe.ibm.com/software/dw/library/j-emd/EnterpriseMetadataDiscovery_v0.12.pdf)
- White Paper on Metadata Management (http://esqinc.com/Content/WhitePapers/Managing-Metadata.php) - by Esquire Innovations (http://esqinc.com/)

Schema matching

The terms **schema matching** and *mapping* are often used interchangeably. For this article, we differentiate the two as follows: Schema matching is the process of identifying that two objects are semantically related (scope of this article) while mapping refers to the transformations between the objects. For example, in the two schemas DB1.Student (Name, SSN, Level, Major, Marks) and DB2.Grad-Student (Name, ID, Major, Grades); possible matches would be: DB1.Student ≈ DB2.Grad-Student; DB1.SSN = DB2.ID etc. and possible transformations or mappings would be: DB1.Marks to DB2.Grades (100-90 A; 90-80 B: etc.).

Automating these two approaches has been one of the fundamental tasks of data integration. In general it is not possible to determine fully automatically the different correspondences between two schemas, primarily because of the differing and often not explicated or documented semantics of the two schemas.

Impediments to Schema Matching

Among others, common challenges to automating matching and mapping have been previously classified in especially for relational DB schemas; and in - a fairly comprehensive list of heterogeneity not limited to the relational model recognizing schematic vs semantic differences/heterogeneity. Most of these heterogeneities exist because schemas use different representations or definitions to represent the same information (schema conflicts); OR different expressions, units, and precision result in conflicting representations of the same data (data conflicts). Research in schema matching seeks to provide automated support to the process of finding semantic matches between two schemas. This process is made harder due to heterogeneities at the following levels

- Syntactic heterogeneity - differences in the language used for representing the elements
- Structural heterogeneity - differences in the types, structures of the elements
- Model / Representational heterogeneity – differences in the underlying models (database, ontologies) or their representations (relational, object-oriented, RDF,OWL)
- Semantic heterogeneity - where the same real world entity is represented using different terms or vice-versa

Schema Matching

Methodology

Discusses a generic methodology for the task of schema integration or the activities involved. According to the authors, one can view the integration

- Preintegration - An analysis of schemas is carried out before integration to decide upon some integration policy. This governs the choice of schemas to be integrated, the order of integration, and a possible assignment of preferences to entire schemas or portions of schemas.
- Comparison of the Schemas - Schemas are analyzed and compared to determine the correspondences among concepts and detect possible conflicts. Interschema properties may be discovered while comparing schemas.
- Conforming the Schemas - Once conflicts are detected, an effort is made to resolve them so that the merging of various schemas is possible.
- Merging and Restructuring - Now the schemas are ready to be superimposed, giving rise to some intermediate integrated schema(s). The intermediate results are analyzed and, if necessary, restructured in order to achieve several desirable qualities.

Approaches

Approaches to schema integration can be broadly classified as ones that exploit either just schema information or schema and instance level information.

Schema-level matchers only consider schema information, not instance data. The available information includes the usual properties of schema elements, such as name, description, data type, relationship types (part-of, is-a, etc.), constraints, and schema structure. Working at the element (atomic elements like attributes of objects) or structure level (matching combinations of elements that appear together in a structure), these properties are used to identify matching elements in two schemas. Language-based or linguistic matchers use names and text (i.e., words or sentences) to find semantically similar schema elements. Constraint based matchers exploit constraints often contained in schemas. Such constraints are used to define data types and value ranges, uniqueness, optionality, relationship types and cardinalities, etc. Constraints in two input schemas are matched to determine the similarity of the schema elements.

Instance-level matchers use instance-level data to gather important insight into the contents and meaning of the schema elements. These are typically used in addition to schema level matches in order to boost the confidence in match results, more so when the information available at the schema level is insufficient. Matchers at this level use linguistic and constraint based characterization of instances. For example, using linguistic techniques, it might be possible to look at the Dept, DeptName and EmpName instances to conclude that DeptName is a better match candidate for Dept than EmpName. Constraints like zipcodes must be 5 digits long or format of phone numbers may allow matching of such types of instance data.

Hybrid matchers directly combine several matching approaches to determine match candidates based on multiple criteria or information sources.

Most of these techniques also employ additional information such as dictionaries, thesauri, and user-provided match or mismatch information

Reusing matching information Another initiative has been to re-use previous matching information as auxiliary information for future matching tasks. The motivation for this work is that structures or substructures often repeat, for example in schemas in the E-commerce domain. Such a reuse of previous matches however needs to be a careful choice. It is possible that such a reuse makes sense only for some part of a new schema or only in some domains. For example, Salary and Income may be considered identical in a payroll application but not in a tax reporting application. There are several open ended challenges in such reuse that deserves further work.

Sample Prototypes Typically, the implementation of such matching techniques can be classified as being either rule based or learner based systems. The complementary nature of these different approaches has instigated a number of applications using a combination of techniques depending on the nature of the domain or application under consideration.

Identified Relationships

The relationship types between objects that are identified at the end of a matching process are typically those with set semantics such as overlap, disjointness, exclusion, equivalence, subsumption. The logical encodings of these relationships are what they mean. Among others, an early attempt to use description logics for schema integration and identifying such relationships was presented. Several state of the art matching tools today and those benchmarked in the *Ontology Alignment Evaluation Initiative*[1] are capable of identifying many such simple (1:1 / 1:n / n:1 element level matches) and complex matches (n:1 / n:m element or structure level matches) between objects.

References

[1] Ontology Alignment Evaluation Initiative::2006 (http://oaei.ontologymatching.org/2006/)

External links

- Early work in schema matching (http://knoesis.wright.edu/library/download/S04-Dagstuhl-Early-Work.pdf)

Schema crosswalk

A **Schema crosswalk** is a table that shows equivalent elements (or "fields") in more than one database schema. It maps the elements in one schema to the equivalent elements in another schema.

Crosswalk tables are often employed within or in parallel to enterprise systems, especially when multiple systems are interfaced or when the system includes legacy system data. In the context of Interfaces, they function as a sort of internal ETL mechanism.

For example, this is a metadata crosswalk from MARC to Dublin Core:

MARC field		Dublin Core element
260$c (Date of publication, distribution, etc.)	→	Date.Created
522 (Geographic Coverage Note)	→	Coverage.Spatial
300$a (Physical Description)	→	Format.Extent

Crosswalks show people where to put the data from one scheme into a different scheme. They are often used by libraries, archives, museums, and other cultural institutions to translate data to or from MARC, Dublin Core, TEI, and other metadata schemes. For example, say an archive has a MARC record in their catalog describing a manuscript. If the archive makes a digital copy of that manuscript and wants to display it on the web along with the information from the catalog, it will have to translate the data from the MARC catalog record into a different format such as MODS that is viewable in a webpage. Because MARC has different fields than MODS, decisions must be made about where to put the data into MODS. This type of "translating" from one format to another is often called "metadata mapping" or "field mapping," and is related to "data mapping," and "semantic mapping."

Crosswalks also have several technical capabilities. They help databases using different metadata schemes to share information. They help metadata harvesters create union catalogs. They enable search engines to search multiple databases simultaneously with a single query.

Challenges for crosswalks

One of the biggest challenges for crosswalks is that no two metadata schemes are 100% equivalent. One scheme may have a field that doesn't exist in another scheme, or it may have a field that is split into two different fields in another scheme; this is why you often lose data when mapping from a complex scheme to a simpler one. For example, when mapping from MARC to Simple Dublin Core, you lose the distinction between types of titles:

MARC field		Dublin Core element
210 Abbreviated Title	→	Title
222 Key Title	→	Title
240 Uniform Title	→	Title
242 Translated Title	→	Title
245 Title Statement	→	Title
246 Variant Title	→	Title

Simple Dublin Core only has one single "Title" element so all of the different types of MARC titles get lumped together without any further distinctions. This is called "many-to-one" mapping. This is also why, once you've translated these titles into Simple Dublin Core you can't translate them back into MARC. Once they're Simple Dublin Core you've lost the MARC information about what types of titles they are so when you map from Simple Dublin Core back to MARC, all the data in the "Title" element maps to the basic MARC 245 Title Statement field.[1]

Dublin Core element		MARC field
Title	→	245 Title Statement
Title	→	245 Title Statement
Title	→	245 Title Statement
Title	→	245 Title Statement
Title	→	245 Title Statement
Title	→	245 Title Statement

This is why crosswalks are said to be "lateral" (one-way) mappings from one scheme to another. Separate crosswalks would be required to map from scheme A to scheme B and from scheme B to scheme A.[2]

Difficulties in mapping

Other mapping problems arise when:

- One scheme has one element that needs to be split up with different parts of it placed in multiple other elements in the second scheme ("one-to-many" mapping)
- One scheme allows an element to be repeated more than once while another only allows that element to appear once with multiple terms in it
- Schemes have different data formats (e.g. *John Doe* or *Doe, John*)
- An element in one scheme is indexed but the equivalent element in the other scheme is not
- Schemes may use different controlled vocabularies
- Schemes change their standards over time

Some of these problems are simply not fixable. As Karen Coyle says in "*Crosswalking Citation Metadata: The University of California's Experience,*"

> "The more metadata experience we have, the more it becomes clear that metadata perfection is not attainable, and anyone who attempts it will be sorely disappointed. When metadata is crosswalked between two or more unrelated sources, there will be data elements that cannot be reconciled in an ideal manner. The key to a successful metadata crosswalk is intelligent flexibility. It is essential to focus on the important goals and be willing to compromise in order to reach a practical conclusion to projects."[3]

Examples

MARC to Dublin Core (Library of Congress) http://loc.gov/marc/marc2dc.html

Dublin Core to MARC21 (Library of Congress) http://www.loc.gov/marc/dccross.html

Dublin Core to UNIMARC (UKOLN) http://www.ukoln.ac.uk/metadata/interoperability/dc_unimarc.html

TEI to and from MARC http://purl.oclc.org/NET/teiinlibraries

FGDC to USMARC (Alexandria) http://www.alexandria.ucsb.edu/public-documents/metadata/fgdc2marc.html

ONIX to MARC21 (LC) http://www.loc.gov/marc/onix2marc.html

VRA to MARC (Indiana University) http://php.indiana.edu/%7Efryp/marcmap.html

Metadata Mappings (MIT Library) http:/ / libraries. mit. edu/ guides/ subjects/ metadata/ mappings. html Wikipedia:Link rot

Mapping Between Metadata formats (UKOLN) http://www.ukoln.ac.uk/metadata/interoperability/

International Metadata Standard Mappings (Academia Sinica) http://www.sinica.edu.tw/%7Emetadata/standard/mapping-foreign_eng.htm

JATS to MARC http://webservices.itcs.umich.edu/mediawiki/jats/index.php/JATS-to-MARC_mapping

References

[1] "Dublin Core to MARC Crosswalk," (http://www.loc.gov/marc/dccross.html) Network Development and MARC Standards Office, Library of Congress

[2] "Metadata Fundamentals for All Librarians," Priscilla Caplan, American Library Association, Chicago, 2003, p.39

[3] in "Metadata in Practice" Diane I. Hillmann and Elaine L. Westbrooks, eds., American Library Association, Chicago, 2004, p. 91.

External links

- "Metadata Crosswalk Depository" (SchemaTrans) (http://www.oclc.org/research/researchworks/schematrans/default.htm)(OCLC)
- "Mapping Between Metadata Formats" (http://www.ukoln.ac.uk/metadata/interoperability) (UKOLN)
- "Crosswalks the Path to Universal Access?" (http://www.getty.edu/research/conducting_research/standards/intrometadata/path.html) (Getty)
- "Metadata Interoperability and Standardization - A Study of Methodology Part I" (http://www.dlib.org/dlib/june06/chan/06chan.html) (D-Lib)

Schema evolution

In computer science, **schema evolution** refers to the problem of evolving a database schema to adapt it to a change in the modeled reality. The problem is not limited to the modification of the schema. It, in fact, affects the data stored under the given schema and the queries (and thus the applications) posed on that schema.

Until recently the design of a database was expected to create a "one size fits all" schema capable of accepting every future change in the requirements, thus, schema evolution was not considered. This assumption, almost unrealistic in the context of traditional information systems, becomes unacceptable in the context of Web Information Systems, that due to the distributed and cooperative nature of their development and fruition are subject of an even stronger pressure toward change (from 39% to over 500% more intense than in traditional settings). Due to this historical heritage the process of schema evolution is nowadays a particularly taxing one. It is, in fact, widely acknowledged that the data management core of an applications is one of the most difficult and critical components to evolve. The key problem is the impact of the schema evolution on queries and applications. As shown in (which provides an analysis of the MediaWiki evolution) each evolution step might affect up to 70% of the queries operating on the schema, that must be manually reworked consequently.

The problem has been recognized as a very pressing one by the database community for more than 12 years. The support for Schema Evolution, is a difficult problem involving complex mapping among schema versions, the tool support has been so far very limited. The recent theoretical advances on mapping composition and mapping invertibility, which represent the core problems underlying the schema evolution remains almost inaccessible to the large public.

Related works

- A rich bibliography on Schema Evolution is collected at: http://se-pubs.dbs.uni-leipzig.de/pubs/results/taxonomy%3A100
- UCLA university carried out an analysis of the MediaWiki Schema Evolution: Schema Evolution Benchmark [1]
- PRISM, a tool to support graceful relational schema evolution: Prism: schema evolution tool [2]
- PRIMA, a tool supporting transaction time databases under schema evolution PRIMA: supporting transaction-time DB under schema evolution [3]
- Pario is a software development tool that includes fully automated schema evolution

References

[1] http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Schema_Evolution_Benchmark
[2] http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Schema_Evolution_Tool
[3] http://prima.schemaevolution.org

Data integration

Data integration involves combining data residing in different sources and providing users with a unified view of these data. This process becomes significant in a variety of situations, which include both commercial (when two similar companies need to merge their databases) and scientific (combining research results from different bioinformatics repositories, for example) domains. Data integration appears with increasing frequency as the volume and the need to share existing data explodes. It has become the focus of extensive theoretical work, and numerous open problems remain unsolved. In management circles, people frequently refer to data integration as "Enterprise Information Integration" (EII).

History

Issues with combining heterogeneous data sources under a single query interface have existed for some time. The rapid adoption of databases after the 1960s naturally led to the need to share or to merge existing repositories. This merging can take place at several levels in the database architecture.

One popular solution is implemented based on data warehousing (see figure 1). The warehouse system extracts, transforms, and loads data from heterogeneous sources into a single view schema so data becomes compatible with each other. This approach offers a tightly coupled architecture because the data is already physically reconciled in a single queriable repository, so it usually takes little time to resolve queries. However, problems lie in the data freshness, that is, information in warehouse is not always up-to-date. Thus updating an original data source may outdate the warehouse, accordingly, the ETL process needs re-execution for synchronization. Difficulties also arise in constructing data warehouses when one has only a query interface to summary data sources and no access to the full data. This problem frequently emerges when integrating several commercial query services like travel or classified advertisement web applications.

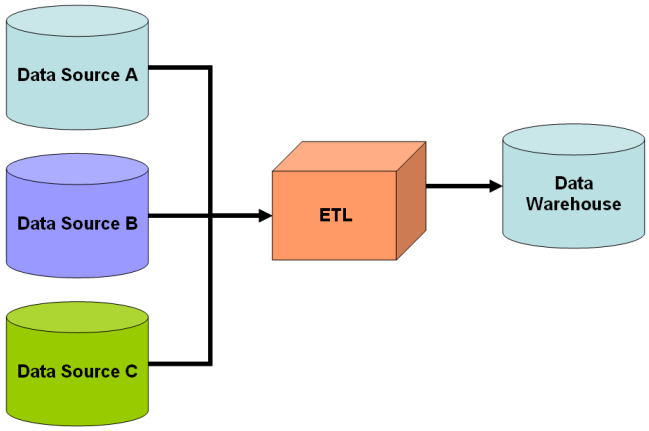


Figure 1: Simple schematic for a data warehouse. The ETL process extracts information from the source databases, transforms it and then loads it into the data warehouse.

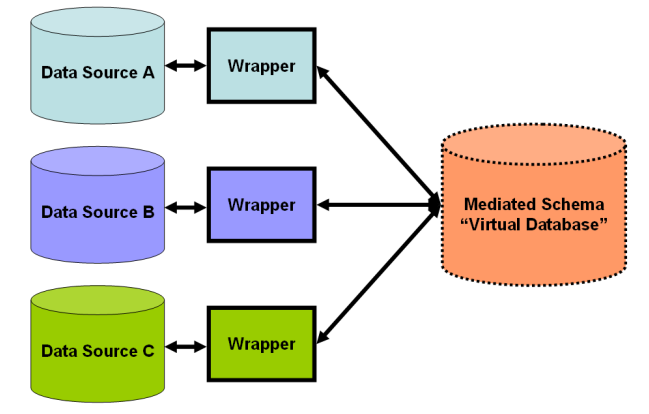


Figure 2: Simple schematic for a data-integration solution. A system designer constructs a mediated schema against which users can run queries. The virtual database interfaces with the source databases via wrapper code if required.

As of 2009[1] the trend in data integration has favored loosening the coupling between data[*citation needed*] and providing a unified query-interface to access real time data over a

mediated schema (see figure 2), which allows information to be retrieved directly from original databases. This approach relies on a mappings between the mediated schema and the schema of original sources, and transform a query into specialized queries to match the schema of the original databases. Such mappings can be specified in 2 ways : as a mapping from entities in the mediated schema to entities in the original sources (the "Global As View" (GAV) approach), or as a mapping from entities in the original sources to the mediated schema (the "Local As View" (LAV) approach). The latter approach requires more sophisticated inferences to resolve a query on the mediated schema, but makes it easier to add new data sources to a (stable) mediated schema.

As of 2010[1] some of the work in data integration research concerns the semantic integration problem. This problem addresses not the structuring of the architecture of the integration, but how to resolve semantic conflicts between heterogeneous data sources. For example if two companies merge their databases, certain concepts and definitions in their respective schemas like "earnings" inevitably have different meanings. In one database it may mean profits in dollars (a floating-point number), while in the other it might represent the number of sales (an integer). A common strategy for the resolution of such problems involves the use of ontologies which explicitly define schema terms and thus help to resolve semantic conflicts. This approach represents ontology-based data integration. On the other hand, the problem of combining research results from different bioinformatics repositories requires bench-marking of the similarities, computed from different data sources, on a single criterion such as positive predictive value. This enables the data sources to be directly comparable and can be integrated even when the natures of experiments are distinct.

As of 2011[1] it was determined that current data modeling methods were imparting data isolation into every data architecture in the form of islands of disparate data and information silos each of which represents a disparate system. This data isolation is an unintended artifact of the data modeling methodology that results in the development of disparate data models. Disparate data models, when instantiated as databases, form disparate databases. Enhanced data model methodologies have been developed to eliminate the data isolation artifact and to promote the development of integrated data models. One enhanced data modeling method recasts data models by augmenting them with structural metadata in the form of standardized data entities. As a result of recasting multiple data models, the set of recast data models will now share one or more commonality relationships that relate the structural metadata now common to these data models. Commonality relationships are a peer-to-peer type of entity relationships that relate the standardized data entities of multiple data models. Multiple data models that contain the same standard data entity may participate in the same commonality relationship. When integrated data models are instantiated as databases and are properly populated from a common set of master data, then these databases are integrated.

Example

Consider a web application where a user can query a variety of information about cities (such as crime statistics, weather, hotels, demographics, etc.). Traditionally, the information must be stored in a single database with a single schema. But any single enterprise would find information of this breadth somewhat difficult and expensive to collect. Even if the resources exist to gather the data, it would likely duplicate data in existing crime databases, weather websites, and census data.

A data-integration solution may address this problem by considering these external resources as materialized views over a virtual mediated schema, resulting in "virtual data integration". This means application-developers construct a virtual schema — the *mediated schema* — to best model the kinds of answers their users want. Next, they design "wrappers" or adapters for each data source, such as the crime database and weather website. These adapters simply transform the local query results (those returned by the respective websites or databases) into an easily processed form for the data integration solution (see figure 2). When an application-user queries the mediated schema, the data-integration solution transforms this query into appropriate queries over the respective data sources. Finally, the virtual database combines the results of these queries into the answer to the user's query.

This solution offers the convenience of adding new sources by simply constructing an adapter or an application software blade for them. It contrasts with ETL systems or with a single database solution, which require manual integration of entire new dataset into the system. The virtual ETL solutions leverage virtual mediated schema to implement data harmonization; whereby the data is copied from the designated "master" source to the defined targets, field by field. Advanced Data virtualization is also built on the concept of object-oriented modeling in order to construct virtual mediated schema or virtual metadata repository, using hub and spoke architecture.

Each data source is disparate and as such is not designed to support reliable joins between data sources. Therefore, data virtualization as well as data federation depends upon accidental data commonality to support combining data and information from disparate data sets. Because of this lack of data value commonality across data sources, the return set may be inaccurate, incomplete, and impossible to validate.

One solution is to recast disparate databases to integrate these databases without the need for ETL. The recast databases support commonality constraints where referential integrity may be enforced between databases. The recast databases provide designed data access paths with data value commonality across databases.

Theory of data integration

The theory of data integration forms a subset of database theory and formalizes the underlying concepts of the problem in first-order logic. Applying the theories gives indications as to the feasibility and difficulty of data integration. While its definitions may appear abstract, they have sufficient generality to accommodate all manner of integration systems.[*citation needed*]

Definitions

Data integration systems are formally defined as a triple $\langle G, S, M \rangle$ where G is the global (or mediated) schema, S is the heterogeneous set of source schemas, and M is the mapping that maps queries between the source and the global schemas. Both G and S are expressed in languages over alphabets composed of symbols for each of their respective relations. The mapping M consists of assertions between queries over G and queries over S. When users pose queries over the data integration system, they pose queries over G and the mapping then asserts connections between the elements in the global schema and the source schemas.

A database over a schema is defined as a set of sets, one for each relation (in a relational database). The database corresponding to the source schema S would comprise the set of sets of tuples for each of the heterogeneous data sources and is called the *source database*. Note that this single source database may actually represent a collection of disconnected databases. The database corresponding to the virtual mediated schema G is called the *global database*. The global database must satisfy the mapping M with respect to the source database. The legality of this mapping depends on the nature of the correspondence between G and S. Two popular ways to model this correspondence exist: *Global as View* or GAV and *Local as View* or LAV.

GAV systems model the global database as a set of views over S. In this case M associates to each element of G as a query over S. Query processing becomes a straightforward operation due to the well-defined associations between G and S. The burden of complexity falls on implementing mediator code instructing the data integration system exactly how to retrieve elements from the source databases. If any new sources join the system, considerable effort may be necessary to update the mediator, thus the GAV approach appears preferable when the sources seem unlikely to change.

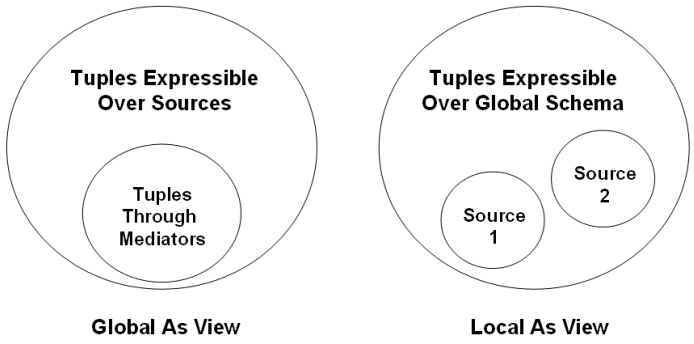


Figure 3: Illustration of tuple space of the GAV and LAV mappings. In GAV, the system is constrained to the set of tuples mapped by the mediators while the set of tuples expressible over the sources may be much larger and richer. In LAV, the system is constrained to the set of tuples in the sources while the set of tuples expressible over the global schema can be much larger. Therefore LAV systems must often deal with incomplete answers.

In a GAV approach to the example data integration system above, the system designer would first develop mediators for each of the city information sources and then design the global schema around these mediators. For example, consider if one of the sources served a weather website. The designer would likely then add a corresponding element for weather to the global schema. Then the bulk of effort concentrates on writing the proper mediator code that will transform predicates on weather into a query over the weather website. This effort can become complex if some other source also relates to weather, because the designer may need to write code to properly combine the results from the two sources.

On the other hand, in LAV, the source database is modeled as a set of views over G. In this case M associates to each element of S a query over G. Here the exact associations between G and S are no longer well-defined. As is illustrated in the next section, the burden of determining how to retrieve elements from the sources is placed on the query processor. The benefit of an LAV modeling is that new sources can be added with far less work than in a GAV system, thus the LAV approach should be favored in cases where the mediated schema is more stable and unlikely to change.

In an LAV approach to the example data integration system above, the system designer designs the global schema first and then simply inputs the schemas of the respective city information sources. Consider again if one of the sources serves a weather website. The designer would add corresponding elements for weather to the global schema only if none existed already. Then programmers write an adapter or wrapper for the website and add a schema description of the website's results to the source schemas. The complexity of adding the new source moves from the designer to the query processor.

Query processing

The theory of query processing in data integration systems is commonly expressed using conjunctive queries and Datalog, a purely declarative logic programming language. One can loosely think of a conjunctive query as a logical function applied to the relations of a database such as " $f(A, B)$ where $A < B$ ". If a tuple or set of tuples is substituted into the rule and satisfies it (makes it true), then we consider that tuple as part of the set of answers in the query. While formal languages like Datalog express these queries concisely and without ambiguity, common SQL queries count as conjunctive queries as well.

In terms of data integration, "query containment" represents an important property of conjunctive queries. A query A contains another query B (denoted $A \supset B$) if the results of applying B are a subset of the results of applying A for any database. The two queries are said to be equivalent if the resulting sets are equal for any database. This is important because in both GAV and LAV systems, a user poses conjunctive queries over a *virtual*

schema represented by a set of views, or "materialized" conjunctive queries. Integration seeks to rewrite the queries represented by the views to make their results equivalent or maximally contained by our user's query. This corresponds to the problem of answering queries using views (AQUV).

In GAV systems, a system designer writes mediator code to define the query-rewriting. Each element in the user's query corresponds to a substitution rule just as each element in the global schema corresponds to a query over the source. Query processing simply expands the subgoals of the user's query according to the rule specified in the mediator and thus the resulting query is likely to be equivalent. While the designer does the majority of the work beforehand, some GAV systems such as Tsimmis [2] involve simplifying the mediator description process.

In LAV systems, queries undergo a more radical process of rewriting because no mediator exists to align the user's query with a simple expansion strategy. The integration system must execute a search over the space of possible queries in order to find the best rewrite. The resulting rewrite may not be an equivalent query but maximally contained, and the resulting tuples may be incomplete. As of 2009[1] the MiniCon algorithm is the leading query rewriting algorithm for LAV data integration systems.

In general, the complexity of query rewriting is NP-complete. If the space of rewrites is relatively small this does not pose a problem — even for integration systems with hundreds of sources.

Data Integration in the Life Sciences

Large-scale questions in science, such as global warming, invasive species spread, and resource depletion, are increasingly requiring the collection of disparate data sets for meta-analysis. This type of data integration is especially challenging for ecological and environmental data because metadata standards are not agreed upon and there are many different data types produced in these fields. National Science Foundation initiatives such as Datanet are intended to make data integration easier for scientists by providing cyberinfrastructure and setting standards. The two funded Datanet initiatives are DataONE and the Data Conservancy.

References

[1] http://en.wikipedia.org/w/index.php?title=Data_integration&action=edit
[2] http://www-db.stanford.edu/tsimmis/

Further reading

- Ronald Schuldt (November 15, 2011). *UDEF – Six Steps to Cost Effective Data Integration* (https://www.createspace.com/3711806). CreateSpace. ISBN 978-1-4664-6762-0.
- Roberta Shauger (December 20, 2011). *UDEF Concepts Defined – Reference Guide* (https://www.createspace.com/3753707). CreateSpace. ISBN 978-1-4681-1483-6.

Ontology-based data integration

Ontology based Data Integration involves the use of ontology(s) to effectively combine data or information from multiple heterogeneous sources. It is one of the multiple data integration approaches and may be classified as Global-As-View (GAV). The effectiveness of ontology based data integration is closely tied to the consistency and expressivity of the ontology used in the integration process.

Background

Data from multiple sources are characterized by multiple types of heterogeneity. The following hierarchy is often used:[1]

- Syntactic Heterogeneity: is a result of differences in representation format of data
- Schematic or Structural Heterogeneity: the native model or structure to store data differ in data sources leading to structural heterogeneity. Schematic heterogeneity that particularly appears in structured databases is also an aspect of structural heterogeneity.
- Semantic Heterogeneity: differences in interpretation of the 'meaning' of data are source of semantic heterogeneity
- System Heterogeneity: use of different operating system, hardware platforms lead to system heterogeneity

Ontologies, as formal models of representation with explicitly defined concepts and named relationships linking them, are used to address the issue of semantic heterogeneity in data sources. In domains like bioinformatics and biomedicine, the rapid development, adoption and public availability of ontologies [2] has made it possible for the data integration community to leverage them for semantic integration of data and information.

The Role of Ontologies

Ontologies enable the unambiguous identification of entities in heterogeneous information systems and assertion of applicable named relationships that connect these entities together. Specifically, ontologies play the following roles:

- Content Explication

The ontology enables accurate interpretation of data from multiple sources through the explicit definition of terms and relationships in the ontology.

- Query Model

In some systems like SIMS, the query is formulated using the ontology as a global query schema.

- Verification

The ontology verifies the mappings used to integrate data from multiple sources. These mappings may either be user specified or generated by a system.

Approaches using ontologies for data Integration

There are three main architectures that are implemented in ontology-based data integration applications, namely,

Single ontology approach

> A single ontology is used as a global reference model in the system. This is the simplest approach as it can be simulated by other approaches. SIMS is a prominent example of this approach.

Multiple ontologies

> Multiple ontologies, each modeling an individual data source, are used in combination for integration. Though, this approach is more flexible than the single ontology approach, it requires creation of mappings between the multiple ontologies. Ontology mapping is a challenging issue and is focus of large number of research efforts

in computer science [3]. The OBSERVER system is an example of this approach.

Hybrid approaches

The hybrid approach involves the use of multiple ontologies that subscribe to a common, top-level vocabulary. The top-level vocabulary defines the basic terms of the domain. Thus, the hybrid approach makes it easier to use multiple ontologies for integration in presence of the common vocabulary.

References

[1] AHM02 Tutorial 5: Data Integration and Mediation; Contributors: B. Ludaescher, I. Altintas, A. Gupta, M. Martone, R. Marciano, X. Qian (http://daks.ucdavis.edu/~ludaesch/Paper/AHM02/tutorial5.html)

[2] http://www.bioontology.org/repositories.html#obo

[3] http://www.ontologymatching.org/

External links

- OBSERVER home page (http://sid.cps.unizar.es/OBSERVER/)

Ontology merging

Ontology merging defines the act of bringing together two conceptually divergent ontologies or the instance data associated to two ontologies. This is similar to work in database merging (schema matching). This merging process can be performed in a number of ways, manually, semi automatically, or automatically. Manual ontology merging although ideal is extremely labour intensive and current research attempts to find semi or entirely automated techniques to merge ontologies. These techniques are statistically driven often taking into account similarity of concepts and raw similarity of instances through textual string metrics and semantic knowledge. These techniques are similar to those used in information integration employing string metrics from open source similarity libraries.

Parallel

MapReduce

MapReduce is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.[1]

A MapReduce program is composed of a **Map()** procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a **Reduce()** procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

The model is inspired by the map and reduce functions commonly used in functional programming,[2] although their purpose in the MapReduce framework is not the same as in their original forms. Furthermore, the key contributions of the MapReduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once.

MapReduce libraries have been written in many programming languages, with different levels of optimization. A popular open-source implementation is Apache Hadoop. The name MapReduce originally referred to the proprietary Google technology but has since been genericized.

Overview

'MapReduce' is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogenous hardware). Computational processing can occur on data stored either in a filesystem (unstructured) or in a database (structured). MapReduce can take advantage of locality of data, processing data on or near the storage assets to decrease transmission of data.

"Map" step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.

"Reduce" step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve.

MapReduce allows for distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel – though in practice this is limited by the number of independent data sources and/or the number of CPUs near each source. Similarly, a set of 'reducers' can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. While this process can often appear inefficient compared to algorithms that are more sequential, MapReduce can be applied to significantly larger datasets than "commodity" servers can handle – a large server farm can use MapReduce to sort a petabyte of data in only a few hours.[*citation needed*] The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data is still available.

Another way to look at MapReduce is as a 5-step parallel and distributed computation:

1. **Prepare the Map() input** – the "MapReduce system" designates Map processors, assigns the K1 input key value each processor would work on, and provides that processor with all the input data associated with that key value.
2. **Run the user-provided Map() code** – Map() is run exactly once for each K1 key value, generating output organized by key values K2.
3. **"Shuffle" the Map output to the Reduce processors** – the MapReduce system designates Reduce processors, assigns the K2 key value each processor would work on, and provides that processor with all the Map-generated data associated with that key value.
4. **Run the user-provided Reduce() code** – Reduce() is run exactly once for each K2 key value produced by the Map step.
5. **Produce the final output** – the MapReduce system collects all the Reduce output, and sorts it by K2 to produce the final outcome.

Logically these 5 steps can be thought of as running in sequence – each step starts only after the previous step is completed – though in practice, of course, they can be intertwined, as long as the final result is not affected.

In many situations the input data might already be distributed ("sharded") among many different servers, in which case step 1 could sometimes be greatly simplified by assigning Map servers that would process the locally present input data. Similarly, step 3 could sometimes be sped up by assigning Reduce processors that are as much as possible local to the Map-generated data they need to process.

Logical view

The *Map* and *Reduce* functions of *MapReduce* are both defined with respect to data structured in (key, value) pairs. *Map* takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain:

`Map(k1,v1) → list(k2,v2)`

The *Map* function is applied in parallel to every pair in the input dataset. This produces a list of pairs for each call. After that, the MapReduce framework collects all pairs with the same key from all lists and groups them together, creating one group for each key.

The *Reduce* function is then applied in parallel to each group, which in turn produces a collection of values in the same domain:

`Reduce(k2, list (v2)) → list(v3)`

Each *Reduce* call typically produces either one value v3 or an empty return, though one call is allowed to return more than one value. The returns of all calls are collected as the desired result list.

Thus the MapReduce framework transforms a list of (key, value) pairs into a list of values. This behavior is different from the typical functional programming map and reduce combination, which accepts a list of arbitrary values and returns one single value that combines *all* the values returned by map.

It is necessary but not sufficient to have implementations of the map and reduce abstractions in order to implement MapReduce. Distributed implementations of MapReduce require a means of connecting the processes performing the Map and Reduce phases. This may be a distributed file system. Other options are possible, such as direct streaming from mappers to reducers, or for the mapping processors to serve up their results to reducers that query them.

Examples

The prototypical MapReduce example counts the appearance of each word in a set of documents:[3]

```
function map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
    emit (w, 1)

function reduce(String word, Iterator partialCounts):
  // word: a word
  // partialCounts: a list of aggregated partial counts
  sum = 0
  for each pc in partialCounts:
    sum += ParseInt(pc)
  emit (word, sum)
```

Here, each document is split into words, and each word is counted by the *map* function, using the word as the result key. The framework puts together all the pairs with the same key and feeds them to the same call to *reduce*, thus this function just needs to sum all of its input values to find the total appearances of that word.

As another example, imagine that for a database of 1.1 billion people, one would like to compute the average number of social contacts a person has according to age. In SQL such a query could be expressed as:

```
  SELECT age, AVG(contacts)
    FROM social.person
GROUP BY age
ORDER BY age
```

Using MapReduce, the `K1` key values could be the integers 1 through 1,100, each representing a batch of 1 million records, the `K2` key value could be a person's age in years, and this computation could be achieved using the following functions:

```
function Map is
    input: integer K1 between 1 and 1100, representing a batch of 1 million social.person records
    for each social.person record in the K1 batch do
        let Y be the person's age
        let N be the number of contacts the person has
        produce one output record (Y,(N,1))
    repeat
end function

function Reduce is
    input: age (in years) Y
    for each input record (Y,(N,C)) do
        Accumulate in S the sum of N*C
        Accumulate in C_new the sum of C
    repeat
    let A be S/C_new
    produce one output record (Y,(A,C_new))
```

```
end function
```

The MapReduce System would line up the 1,100 Map processors, and would provide each with its corresponding 1 million input records. The Map step would produce 1.1 billion `(Y,(N,1))` records, with `Y` values ranging between, say, 8 and 103. The MapReduce System would then line up the 96 Reduce processors by performing shuffling operation of the key/value pairs due to the fact that we need average per age, and provide each with its millions of corresponding input records. The Reduce step would result in the much reduced set of only 96 output records `(Y,A)`, which would be put in the final result file, sorted by `Y`.

The count info in the record is important if the processing is reduced more than one time. If we don't add the count of the records, the computed average would be wrong, for example:

```
-- map output #1: age, quantity of contacts
10, 9
10, 9
10, 9
```

```
-- map output #2: age, quantity of contacts
10, 9
10, 9
```

```
-- map output #3: age, quantity of contacts
10, 10
```

If we reduce files `#1` and `#2`, we will have a new file with an average of 9 contacts for a 10 year old person ((9+9+9+9+9)/5):

```
-- reduce step #1: age, average of contacts
10, 9
```

If we reduce it with file `#3`, we lost the count of how many records we've already seen, so we would end up with an average of 9.5 contacts for a 10 year old person ((9+10)/2), which is wrong. The correct answer is 9.17 ((9+9+9+9+9+10)/6).

Dataflow

The frozen part of the MapReduce framework is a large distributed sort. The hot spots, which the application defines, are:

- an *input reader*
- a *Map* function
- a *partition* function
- a *compare* function
- a *Reduce* function
- an *output writer*

Input reader

The *input reader* divides the input into appropriate size 'splits' (in practice typically 16 MB to 128 MB) and the framework assigns one split to each *Map* function. The *input reader* reads data from stable storage (typically a distributed file system) and generates key/value pairs.

A common example will read a directory full of text files and return each line as a record.

Map function

The *Map* function takes a series of key/value pairs, processes each, and generates zero or more output key/value pairs. The input and output types of the map can be (and often are) different from each other.

If the application is doing a word count, the map function would break the line into words and output a key/value pair for each word. Each output pair would contain the word as the key and the number of instances of that word in the line as the value.

Partition function

Each *Map* function output is allocated to a particular *reducer* by the application's *partition* function for sharding purposes. The *partition* function is given the key and the number of reducers and returns the index of the desired *reducer*.

A typical default is to hash the key and use the hash value modulo the number of *reducers*. It is important to pick a partition function that gives an approximately uniform distribution of data per shard for load-balancing purposes, otherwise the MapReduce operation can be held up waiting for slow reducers (reducers assigned more than their share of data) to finish.

Between the map and reduce stages, the data is *shuffled* (parallel-sorted / exchanged between nodes) in order to move the data from the map node that produced it to the shard in which it will be reduced. The shuffle can sometimes take longer than the computation time depending on network bandwidth, CPU speeds, data produced and time taken by map and reduce computations.

Comparison function

The input for each *Reduce* is pulled from the machine where the *Map* ran and sorted using the application's *comparison* function.

Reduce function

The framework calls the application's *Reduce* function once for each unique key in the sorted order. The *Reduce* can iterate through the values that are associated with that key and produce zero or more outputs.

In the word count example, the *Reduce* function takes the input values, sums them and generates a single output of the word and the final sum.

Output writer

The *Output Writer* writes the output of the *Reduce* to the stable storage, usually a distributed file system.

Distribution and reliability

MapReduce achieves reliability by parceling out a number of operations on the set of data to each node in the network. Each node is expected to report back periodically with completed work and status updates. If a node falls silent for longer than that interval, the master node (similar to the master server in the Google File System) records the node as dead and sends out the node's assigned work to other nodes. Individual operations use atomic operations

for naming file outputs as a check to ensure that there are not parallel conflicting threads running. When files are renamed, it is possible to also copy them to another name in addition to the name of the task (allowing for side-effects).

The reduce operations operate much the same way. Because of their inferior properties with regard to parallel operations, the master node attempts to schedule reduce operations on the same node, or in the same rack as the node holding the data being operated on. This property is desirable as it conserves bandwidth across the backbone network of the datacenter.

Implementations are not necessarily highly reliable. For example, in older versions of Hadoop the *NameNode* was a single point of failure for the distributed filesystem. Later versions of Hadoop have high availability with an active/passive failover for the "NameNode."

Uses

MapReduce is useful in a wide range of applications, including distributed pattern-based searching, distributed sorting, web link-graph reversal, term-vector per host, web access log stats, inverted index construction, document clustering, machine learning, and statistical machine translation. Moreover, the MapReduce model has been adapted to several computing environments like multi-core and many-core systems, desktop grids, volunteer computing environments, dynamic cloud environments, and mobile environments.

At Google, MapReduce was used to completely regenerate Google's index of the World Wide Web. It replaced the old *ad hoc* programs that updated the index and ran the various analyses.

MapReduce's stable inputs and outputs are usually stored in a distributed file system. The transient data is usually stored on local disk and fetched remotely by the reducers.

Criticism

Lack of novelty

David DeWitt and Michael Stonebraker, computer scientists specializing in parallel databases and shared-nothing architectures, have been critical of the breadth of problems that MapReduce can be used for. They called its interface too low-level and questioned whether it really represents the paradigm shift its proponents have claimed it is. They challenged the MapReduce proponents' claims of novelty, citing Teradata as an example of prior art that has existed for over two decades. They also compared MapReduce programmers to Codasyl programmers, noting both are "writing in a low-level language performing low-level record manipulation." MapReduce's use of input files and lack of schema support prevents the performance improvements enabled by common database system features such as B-trees and hash partitioning, though projects such as Pig (or PigLatin), Sawzall, Apache Hive, YSmart [4], HBase and BigTable are addressing some of these problems.

Greg Jorgensen wrote an article rejecting these views. Jorgensen asserts that DeWitt and Stonebraker's entire analysis is groundless as MapReduce was never designed nor intended to be used as a database.

DeWitt and Stonebraker have subsequently published a detailed benchmark study in 2009 comparing performance of Hadoop's MapReduce and RDBMS approaches on several specific problems. They concluded that relational databases offer real advantages for many kinds of data use, especially on complex processing or where the data is used across an enterprise, but that MapReduce may be easier for users to adopt for simple or one-time processing tasks.

Google has been granted a patent on MapReduce.[5] However, there have been claims that this patent should not have been granted because MapReduce is too similar to existing products. For example, map and reduce functionality can be very easily implemented in Oracle's PL/SQL database oriented language.

Restricted programming framework

MapReduce tasks must be written as acyclic dataflow programs, i.e. a stateless mapper followed by a stateless reducer, that are executed by a batch job scheduler. This paradigm makes repeated querying of datasets difficult and imposes limitations that are felt in fields such as machine learning, where iterative algorithms that revisit a single working set multiple times are the norm.

Conferences and users groups

- The First International Workshop on MapReduce and its Applications (MAPREDUCE'10) [6] was held with the HPDC conference and OGF'29 meeting in Chicago, IL.
- MapReduce Users Groups [7] around the world.

References

Specific references:

[1] Google spotlights data center inner workings | Tech news blog - CNET News.com (http://news.cnet.com/8301-10784_3-9955184-7.html)
[2] "Our abstraction is inspired by the map and reduce primitives present in Lisp and many other functional languages." - "MapReduce: Simplified Data Processing on Large Clusters" (http://research.google.com/archive/mapreduce.html), by Jeffrey Dean and Sanjay Ghemawat; from Google Research
[3] Example: Count word occurrences (http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0004.html). Research.google.com. Retrieved on 2013-09-18.
[4] http://ysmart.cse.ohio-state.edu/
[5] US Patent 7,650,331: "System and method for efficient large-scale data processing " (http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/PTO/srchnum.htm&r=1&f=G&l=50&s1=7,650,331.PN.&OS=PN/7,650,331&RS=PN/7,650,331)
[6] http://graal.ens-lyon.fr/mapreduce/
[7] http://mapreduce.meetup.com/

General references:

- Dean, Jeffrey & Ghemawat, Sanjay (2004). "MapReduce: Simplified Data Processing on Large Clusters" (http://research.google.com/archive/mapreduce.html). Retrieved Nov. 23, 2011.
- Matt WIlliams (2009). "Understanding Map-Reduce" (http://wordflows.com/matt/2009/01/18/understanding-mapreduce/). Retrieved Apr. 13, 2011.

External links

- MapReduce-MPI (http://mapreduce.sandia.gov/index.html) MapReduce-MPI Library

Papers

- "CloudSVM: Training an SVM Classifier in Cloud Computing Systems" (http://www.researchgate.net/publication/259226804_A_MapReduce_based_distributed_SVM_algorithm_for_binary_classification)-paper by F. Ozgur Catak, M. Erdal Balaban, Springer, Lecture Notes in Computer Science,Pervasive Computing and Networked World 2012 from TÜBİTAK and Istanbul University
- "A Hierarchical Framework for Cross-Domain MapReduce Execution" (http://dl.acm.org/citation.cfm?id=1996023.1996026) — paper by Yuan Luo, Zhenhua Guo, Yiming Sun, Beth Plale, Judy Qiu; from Indiana University and Wilfred Li; from University of California, San Diego
- "Interpreting the Data: Parallel Analysis with Sawzall" (http://research.google.com/archive/sawzall.html) — paper by Rob Pike, Sean Dorward, Robert Griesemer, Sean Quinlan; from Google Labs
- "Evaluating MapReduce for Multi-core and Multiprocessor Systems" (http://csl.stanford.edu/~christos/publications/2007.cmp_mapreduce.hpca.pdf) — paper by Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, and Christos Kozyrakis; from Stanford University

- "Why MapReduce Matters to SQL Data Warehousing" (http://www.dbms2.com/2008/08/26/why-mapreduce-matters-to-sql-data-warehousing/) — analysis related to the August, 2008 introduction of MapReduce/SQL integration by Aster Data Systems and Greenplum
- "MapReduce for the Cell B.E. Architecture" (http://pages.cs.wisc.edu/~dekruijf/docs/mapreduce-cell.pdf) — paper by Marc de Kruijf and Karthikeyan Sankaralingam; from University of Wisconsin–Madison
- "Mars: A MapReduce Framework on Graphics Processors" (http://www.cse.ust.hk/catalac/users/saven/GPGPU/MapReduce/PACT08/171.pdf) — paper by Bingsheng He, Wenbin Fang, Qiong Luo, Naga K. Govindaraju, Tuyong Wang; from Hong Kong University of Science and Technology; published in Proc. PACT 2008. It presents the design and implementation of MapReduce on graphics processors.
- "A Peer-to-Peer Framework for Supporting MapReduce Applications in Dynamic Cloud Environments" (http://www.springerlink.com/content/h17r882710314147/) — paper by Fabrizio Marozzo, Domenico Talia, Paolo Trunfio; from University of Calabria; published in Cloud Computing: Principles, Systems and Applications, N. Antonopoulos, L. Gillam (Editors), chapt. 7, pp. 113–125, Springer, 2010, ISBN 978-1-84996-240-7.
- "Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters" (http://portal.acm.org/citation.cfm?doid=1247480.1247602) — paper by Hung-Chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker; from Yahoo and UCLA; published in Proc. of ACM SIGMOD, pp. 1029–1040, 2007. (This paper shows how to extend MapReduce for relational data processing.)
- FLuX: the Fault-tolerant (http://citeseer.ist.psu.edu/647742.html), Load Balancing (http://citeseer.ist.psu.edu/546646.html) eXchange operator from UC Berkeley provides an integration of partitioned parallelism with process pairs. This results in a more pipelined approach than Google's MapReduce with instantaneous failover, but with additional implementation cost.
- "A New Computation Model for Rack-Based Computing" (http://infolab.stanford.edu/~ullman/pub/mapred.pdf) — paper by Foto N. Afrati; Jeffrey D. Ullman; from Stanford University; Not published as of Nov 2009. This paper is an attempt to develop a general model in which one can compare algorithms for computing in an environment similar to what map-reduce expects.
- FPMR: MapReduce framework on FPGA (http://portal.acm.org/beta/citation.cfm?id=1723112.1723129)—paper by Yi Shan, Bo Wang, Jing Yan, Yu Wang, Ningyi Xu, Huazhong Yang (2010), in FPGA '10, Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays.
- "Tiled-MapReduce: Optimizing Resource Usages of Data-parallel Applications on Multicore with Tiling" (http://ipads.se.sjtu.edu.cn/lib/exe/fetch.php?media=publications:ostrich-pact10.pdf)—paper by Rong Chen, Haibo Chen and Binyu Zang from Fudan University; published in Proc. PACT 2010. It presents the Tiled-MapReduce programming model which optimizes resource usages of MapReduce applications on multicore environment using tiling strategy.
- "Tiled MapReduce: Efficient and Flexible MapReduce Processing on Multicore with Tiling" (http://ipads.se.sjtu.edu.cn/lib/exe/fetch.php?media=publications:ostrich-taco13.pdf)—paper by Rong Chen, and Haibo Chen from Shanghai Jiao Tong University; published in ACM TACO, 10(1), 2013. It extends the earlier version of Ostrich to support several usage scenarios such as online and incremental computing on multicore machines.
- "Scheduling divisible MapReduce computations " (http://dx.doi.org/10.1016/j.jpdc.2010.12.004)—paper by Joanna Berlińska from Adam Mickiewicz University and Maciej Drozdowski from Poznan University of Technology; Journal of Parallel and Distributed Computing 71 (2011) 450-459, doi: 10.1016/j.jpdc.2010.12.004 (http://dx.doi.org/10.1016/j.jpdc.2010.12.004). It presents scheduling and performance model of MapReduce.
- "Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing" (http://stratosphere.eu/files/NephelePACTs_10.pdf)—paper by D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke from TU Berlin (http://www.tu-berlin.de/menue/home/parameter/en/) published in Proc. of ACM SoCC 2010. The paper introduces the PACT programming model, a generalization of MapReduce, developed in the Stratosphere (http://www.stratosphere.eu) research project.

- "MapReduce and PACT - Comparing Data Parallel Programming Models" (http://stratosphere.eu/files/ComparingMapReduceAndPACTs_11.pdf)—paper by A. Alexandrov, S. Ewen, M. Heimel, F. Hueske, O. Kao, V. Markl, E. Nijkamp, and D. Warneke from TU Berlin (http://www.tu-berlin.de/menue/home/parameter/en/) published in Proc. of BTW 2011.

Books

- Jimmy Lin and Chris Dyer. "Data-Intensive Text Processing with MapReduce" (http://www.umiacs.umd.edu/~jimmylin/book.html) (manuscript)

Educational courses

- Cluster Computing and MapReduce (http://code.google.com/edu/submissions/mapreduce-minilecture/listing.html) course from Google Code University (http://code.google.com/edu/) contains video lectures and related course materials from a series of lectures that was taught to Google software engineering interns during the Summer of 2007.
- MapReduce in a Week (http://code.google.com/edu/submissions/mapreduce/listing.html) course from Google Code University (http://code.google.com/edu/) contains a comprehensive introduction to MapReduce including lectures, reading material, and programming assignments.
- MapReduce course (http://mr.iap.2008.googlepages.com/), taught by engineers of Google Boston, part of 2008 Independent Activities Period at MIT.

Bibliography

- MapReduce bibliography by A. Kamil, 2010 (http://www.columbia.edu/~ak2834/mapreduce.html)

Apache Hadoop

Apache Hadoop

Developer(s)	Apache Software Foundation
Stable release	2.2 / October 15, 2013
Preview release	2.1.0-beta / August 25, 2013
Development status	Active
Written in	Java
Operating system	Cross-platform
Type	Distributed File System
License	Apache License 2.0
Website	hadoop.apache.org [1]

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. It is licensed under the Apache License 2.0.

The Apache Hadoop framework is composed of the following modules:

- Hadoop Common – contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop YARN – a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
- Hadoop MapReduce – a programming model for large scale data processing.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's MapReduce and HDFS components originally derived respectively from Google's MapReduce and Google File System (GFS) papers.

Beyond HDFS, YARN and MapReduce, the entire Apache Hadoop “platform” is now commonly considered to consist of a number of related projects as well – Apache Pig, Apache Hive, Apache HBase, Apache Spark, and others.

For the end-users, though MapReduce Java code is common, any programming language can be used with "Hadoop Streaming" to implement the "map" and "reduce" parts of the user's program. Apache Pig, Apache Hive, Apache Spark among other related projects expose higher level user interfaces like Pig latin and a SQL variant respectively. The Hadoop framework itself is mostly written in the Java programming language, with some native code in C and command line utilities written as shell-scripts.

Apache Hadoop is a registered trademark of the Apache Software Foundation.

History

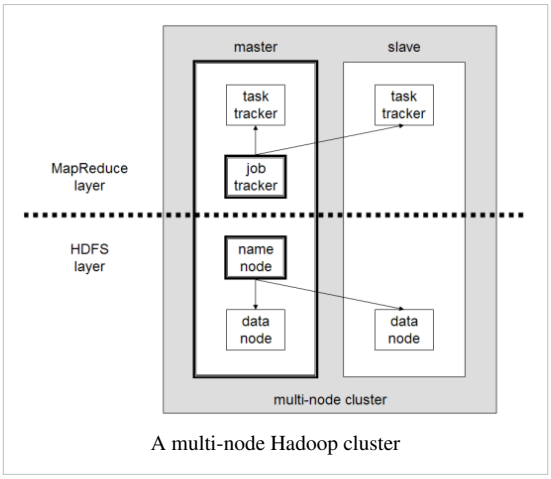
Hadoop was created by Doug Cutting and Mike Cafarella in 2005. Cutting, who was working at Yahoo! at the time,[2] named it after his son's toy elephant. It was originally developed to support distribution for the Nutch search engine project.[3]

Architecture

Hadoop consists of the *Hadoop Common* package, which provides filesystem and OS level abstractions, a MapReduce engine (either MapReduce/MR1 or YARN/MR2) and the Hadoop Distributed File System (HDFS). The Hadoop Common package contains the necessary Java ARchive (JAR) files and scripts needed to start Hadoop. The package also provides source code, documentation and a contribution section that includes projects from the Hadoop Community.[*citation needed*]

For effective scheduling of work, every Hadoop-compatible file system should provide location awareness: the name of the rack (more precisely, of the network switch) where a worker node is. Hadoop applications can use this information to run work on the node where the data is, and, failing that, on the same rack/switch, reducing backbone traffic. HDFS uses this method when replicating data to try to keep different copies of the data on different racks. The goal is to reduce the impact of a rack power outage or switch failure, so that even if these events occur, the data may still be readable.

A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a JobTracker, TaskTracker, NameNode and DataNode. A slave or *worker node* acts as both a DataNode and TaskTracker, though it is possible to have data-only worker nodes and compute-only worker nodes. These are normally used only in nonstandard applications. Hadoop requires Java Runtime Environment (JRE) 1.6 or higher. The standard start-up and shutdown scripts require Secure Shell (ssh) to be set up between nodes in the cluster.



A multi-node Hadoop cluster

In a larger cluster, the HDFS is managed through a dedicated NameNode server to host the file system index, and a secondary NameNode that can generate snapshots of the namenode's memory structures, thus preventing file-system corruption and reducing loss of data. Similarly, a standalone JobTracker server can manage job scheduling. In clusters where the Hadoop MapReduce engine is deployed against an alternate file system, the NameNode, secondary NameNode and DataNode architecture of HDFS is replaced by the file-system-specific equivalent.

File system

Hadoop distributed file system

The **Hadoop distributed file system** (**HDFS**) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single namenode; a cluster of datanodes form the HDFS cluster. The situation is typical because each node does not require a datanode to be present. Each datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses the TCP/IP layer for communication. Clients use Remote procedure call (RPC) to communicate between each other.

HDFS stores large files (typically in the range of gigabytes to terabytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence theoretically does not require RAID storage on hosts (but to increase I/O performance some RAID configurations are still useful). With the default replication value,

3, data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high. HDFS is not fully POSIX-compliant, because the requirements for a POSIX file-system differ from the target goals for a Hadoop application. The tradeoff of not having a fully POSIX-compliant file-system is increased performance for data throughput and support for non-POSIX operations such as Append.

HDFS added the high-availability capabilities, as announced for release 2.0 in May 2012, allowing the main metadata server (the NameNode) to be failed over manually to a backup in the event of failure. The project has also started developing automatic fail-over.

The HDFS file system includes a so-called *secondary namenode,* which misleads some people into thinking[*citation needed*] that when the primary namenode goes offline, the secondary namenode takes over. In fact, the secondary namenode regularly connects with the primary namenode and builds snapshots of the primary namenode's directory information, which the system then saves to local or remote directories. These checkpointed images can be used to restart a failed primary namenode without having to replay the entire journal of file-system actions, then to edit the log to create an up-to-date directory structure. Because the namenode is the single point for storage and management of metadata, it can become a bottleneck for supporting a huge number of files, especially a large number of small files. HDFS Federation, a new addition, aims to tackle this problem to a certain extent by allowing multiple name-spaces served by separate namenodes.

An advantage of using HDFS is data awareness between the job tracker and task tracker. The job tracker schedules map or reduce jobs to task trackers with an awareness of the data location. For example: if node A contains data (x,y,z) and node B contains data (a,b,c), the job tracker schedules node B to perform map or reduce tasks on (a,b,c) and node A would be scheduled to perform map or reduce tasks on (x,y,z). This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. When Hadoop is used with other file systems this advantage is not always available. This can have a significant impact on job-completion times, which has been demonstrated when running data-intensive jobs.

HDFS was designedWikipedia:Manual of Style/Words to watch#Unsupported attributions for mostly immutable files and may not be suitable for systems requiring concurrent write-operations.

Another limitation of HDFS is that it cannot be mounted directly by an existing operating system. Getting data into and out of the HDFS file system, an action that often needs to be performed before and after executing a job, can be inconvenient. A Filesystem in Userspace (FUSE) virtual file system has been developed to address this problem, at least for Linux and some other Unix systems.

File access can be achieved through the native Java API, the Thrift API to generate a client in the language of the users' choosing (C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk, and OCaml), the command-line interface, or browsed through the HDFS-UI webapp over HTTP.

Other file systems

By May 2011, the list of supported file systems included:

- Amazon S3 file system. This is targeted at clusters hosted on the Amazon Elastic Compute Cloud server-on-demand infrastructure. There is no rack-awareness in this file system, as it is all remote.
- CloudStore (previously Kosmos Distributed File System), which is rack-aware.
- FTP File system: this stores all its data on remotely accessible FTP servers.
- HDFS: Hadoop's own rack-aware file system. This is designed to scale to tens of petabytes of storage and runs on top of the file systems of the underlying operating systems.
- MapR's maprfs[4] file system. This system provides inherent high availability, transactionally correct snapshots and mirrors while offering higher scaling than HDFS while giving higher performance. Maprfs is available as part of the MapR distribution[5] and as a native option[6] on Elastic Map Reduce from Amazon's web services, as well as on Google Compute Engine.

- Read-only HTTP and HTTPS file systems.

Hadoop can work directly with any distributed file system that can be mounted by the underlying operating system simply by using a file:// URL; however, this comes at a price: the loss of locality. To reduce network traffic, Hadoop needs to know which servers are closest to the data; this is information that Hadoop-specific file system bridges can provide.

Out-of-the-box, this includes Amazon S3, and the CloudStore filestore, through s3:// and kfs:// URLs directly.

A number of third-party file system bridges have also been written, none of which are currently in Hadoop distributions.

- In 2009 IBM discussed running Hadoop over the IBM General Parallel File System. The source code was published in October 2009.
- In April 2010, Parascale published the source code to run Hadoop against the Parascale file system.
- In April 2010, Appistry released a Hadoop file system driver for use with its own CloudIQ Storage product.
- In June 2010, HP discussed a location-aware IBRIX Fusion file system driver.
- In May 2011, MapR Technologies, Inc. announced the availability of an alternative file system for Hadoop, which replaced the HDFS file system with a full random-access read/write file system, with advanced features like snaphots and mirrors, and got rid of the single point of failure issue of the default HDFS NameNode.

JobTracker and TaskTracker: the MapReduce engine

Above the file systems comes the MapReduce engine, which consists of one *JobTracker*, to which client applications submit MapReduce jobs. The JobTracker pushes work out to available *TaskTracker* nodes in the cluster, striving to keep the work as close to the data as possible. With a rack-aware file system, the JobTracker knows which node contains the data, and which other machines are nearby. If the work cannot be hosted on the actual node where the data resides, priority is given to nodes in the same rack. This reduces network traffic on the main backbone network. If a TaskTracker fails or times out, that part of the job is rescheduled. The TaskTracker on each node spawns off a separate Java Virtual Machine process to prevent the TaskTracker itself from failing if the running job crashes the JVM. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. The Job Tracker and TaskTracker status and information is exposed by Jetty and can be viewed from a web browser.

If the JobTracker failed on Hadoop 0.20 or earlier, all ongoing work was lost. Hadoop version 0.21 added some checkpointing to this process; the JobTracker records what it is up to in the file system. When a JobTracker starts up, it looks for any such data, so that it can restart work from where it left off.

Known limitations of this approach are:

- The allocation of work to TaskTrackers is very simple. Every TaskTracker has a number of available *slots* (such as "4 slots"). Every active map or reduce task takes up one slot. The Job Tracker allocates work to the tracker nearest to the data with an available slot. There is no consideration of the current system load of the allocated machine, and hence its actual availability.
- If one TaskTracker is very slow, it can delay the entire MapReduce job – especially towards the end of a job, where everything can end up waiting for the slowest task. With speculative execution enabled, however, a single task can be executed on multiple slave nodes.

Scheduling

By default Hadoop uses FIFO, and optional 5 scheduling priorities to schedule jobs from a work queue.[7] In version 0.19 the job scheduler was refactored out of the JobTracker, and added the ability to use an alternate scheduler (such as the *Fair scheduler* or the *Capacity scheduler*).

Fair scheduler

The fair scheduler was developed by Facebook. The goal of the fair scheduler is to provide fast response times for small jobs and QoS for production jobs. The fair scheduler has three basic concepts.[8]

1. Jobs are grouped into Pools.
2. Each pool is assigned a guaranteed minimum share.
3. Excess capacity is split between jobs.

By default, jobs that are uncategorized go into a default pool. Pools have to specify the minimum number of map slots, reduce slots, and a limit on the number of running jobs.

Capacity scheduler

The capacity scheduler was developed by Yahoo. The capacity scheduler supports several features that are similar to the fair scheduler.[9]

- Jobs are submitted into queues.
- Queues are allocated a fraction of the total resource capacity.
- Free resources are allocated to queues beyond their total capacity.
- Within a queue a job with a high level of priority has access to the queue's resources.

There is no preemption once a job is running.

Other applications

The HDFS file system is not restricted to MapReduce jobs. It can be used for other applications, many of which are under development at Apache. The list includes the HBase database, the Apache Mahout machine learning system, and the Apache Hive Data Warehouse system. Hadoop can in theory be used for any sort of work that is batch-oriented rather than real-time, that is very data-intensive, and able to work on pieces of the data in parallel. As of October 2009, commercial applications of Hadoop included:

- Log and/or clickstream analysis of various kinds
- Marketing analytics
- Machine learning and/or sophisticated data mining
- Image processing
- Processing of XML messages
- Web crawling and/or text processing
- General archiving, including of relational/tabular data, e.g. for compliance

Prominent users

Yahoo!

On February 19, 2008, Yahoo! Inc. launched what it claimed was the world's largest Hadoop production application. The Yahoo! Search Webmap is a Hadoop application that runs on a more than 10,000 core Linux cluster and produces data that is used in every Yahoo! Web search query.[10]

There are multiple Hadoop clusters at Yahoo! and no HDFS file systems or MapReduce jobs are split across multiple datacenters. Every Hadoop cluster node bootstraps the Linux image, including the Hadoop distribution. Work that the clusters perform is known to include the index calculations for the Yahoo! search engine.

On June 10, 2009, Yahoo! made the source code of the version of Hadoop it runs in production available to the public. Yahoo! contributes all the work it does on Hadoop to the open-source community. The company's developers also fix bugs, provide stability improvements internally and release this patched source code so that other users may benefit from their effort.

Facebook

In 2010 Facebook claimed that they had the largest Hadoop cluster in the world with 21 PB of storage. On June 13, 2012 they announced the data had grown to 100 PB. On November 8, 2012 they announced the data gathered in the warehouse grows by roughly half a PB per day.

Other users

As of 2013, Hadoop adoption is widespread. For example, more than half of the Fortune 50 uses Hadoop.

Hadoop on Amazon EC2/S3 services

It is possible to run Hadoop on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). As an example The New York Times used 100 Amazon EC2 instances and a Hadoop application to process 4 TB of raw image TIFF data (stored in S3) into 11 million finished PDFs in the space of 24 hours at a computation cost of about $240 (not including bandwidth).

There is support for the S3 file system in Hadoop distributions, and the Hadoop team generates EC2 machine images after every release. From a pure performance perspective, Hadoop on S3/EC2 is inefficient, as the S3 file system is remote and delays returning from every write operation until the data is guaranteed not to be lost. This removes the locality advantages of Hadoop, which schedules work near data to save on network load.

Amazon Elastic MapReduce

Elastic MapReduce (EMR)[11] was introduced by Amazon in April 2009. Provisioning of the Hadoop cluster, running and terminating jobs, and handling data transfer between EC2 and S3 are automated by Elastic MapReduce. Apache Hive, which is built on top of Hadoop for providing data warehouse services, is also offered in Elastic MapReduce.

Support for using Spot Instances[12] was later added in August 2011. Elastic MapReduce is fault tolerant for slave failures, and it is recommended to only run the Task Instance Group on spot instances to take advantage of the lower cost while maintaining availability.

In June 2012, premium options for EMR were added that replace ordinary Hadoop with MapR's M3 and M5 versions. These options provide additional capabilities over and above what the default EMR offering provides.

Industry support of academic clusters

IBM and Google announced an initiative in 2007 to use Hadoop to support university courses in distributed computer programming.

In 2008 this collaboration, the Academic Cloud Computing Initiative (ACCI), partnered with the National Science Foundation to provide grant funding to academic researchers interested in exploring large-data applications. This resulted in the creation of the Cluster Exploratory (CLuE) program.

Running Hadoop in compute farm environments

Hadoop can also be used in compute farms and high-performance computing environments. Instead of setting up a dedicated Hadoop cluster, an existing compute farm can be used if the resource manager of the cluster is aware of the Hadoop jobs, and thus Hadoop jobs can be scheduled like other jobs in the cluster.

Grid engine integration

Integration with Sun Grid Engine was released in 2008, and running Hadoop on Sun Grid (Sun's on-demand utility computing service) was possible. In the initial implementation of the integration, the CPU-time scheduler has no knowledge of the locality of the data. Unfortunately, this means that the processing is not always done on the same rack as the data; this was a key feature of the Hadoop Runtime. An improved integration with data-locality was announced during the Sun HPC Software Workshop '09.

In 2008-2009 Sun released the *Hadoop Live CD* OpenSolaris project, which allows running a fully functional Hadoop cluster using a live CD. This distribution includes Hadoop 0.19 – as of April 2010 there has not been an updated release.

Condor integration

The Condor High-Throughput Computing System integration was presented at the *Condor Week* conference in 2010.

Commercial support

A number of companies offer commercial implementations or support for Hadoop.

- Big Data Partnership, based in Europe, offers Hadoop consulting, implementation services and training.
- BMC Software provides BMC Control-M for Hadoop, which adds capabilities to monitor and manage Hadoop workflows with predictive analytics and automated alerts.
- Cloudera offers CDH (Cloudera's Distribution including Apache Hadoop) and Cloudera Enterprise.
- Dell added Pentaho Business Analytics to the Dell Apache Hadoop solution for big data analytics. This consists of Dell servers, Dell networking components, Dell's Crowbar cloud deployment framework open source software, and Cloudera Distribution including Apache Hadoop (CDH).
- EMC released *EMC Greenplum Community Edition* and *EMC Greenplum HD Enterprise Edition* in May 2011. The community edition, with optional for-fee technical support, consists of Hadoop, HDFS, HBase, Hive, and the ZooKeeper configuration service. The enterprise edition is an offering based on the MapR product, and offers proprietary features such as snapshots and wide area replication.
- Asia's Etu is owned by Systex Corporation and represented in Europe by GNR Corporation.
- EMC Isilon announced support for HDFS in its OneFS clustered file system.
- Google added *AppEngine-MapReduce* to support running Hadoop 0.20 programs on Google App Engine.
- Grand Logic's JobServer product allows developers and admins to deploy, manage and monitor their Hadoop infrastructure, with support for Hadoop job processing and HDFS file/content management.
- Hortonworks (formed by Yahoo! and Benchmark Capital focuses is on making Hadoop more robust and easier to install, manage and use for enterprise users.

- IBM offers WebSphere eXtreme Scale (formerly ObjectGrid), which includes two styles of the HADOOP map-reduce pattern in its "agents" a.k.a. DataGrid APIs. Together with its scalable distributed data cache capability, it gives both map-reduce's ability to parallelize function and the ability to store plenty of data (in memory) for the function to quickly access. It's transactional and highly available, too.
- IBM offers InfoSphere BigInsights based on Hadoop in both a basic and enterprise edition.
- Intel released its own Hadoop distribution that takes advantage of capabilities in Intel Xeon chips, such as its processor instructions for accelerating AES encryption.
- MapR Technologies, Inc. announced the availability of their distributed file system and MapReduce engine, the MapR Distribution for Apache Hadoop. The MapR product includes most Hadoop eco-system components and adds capabilities such as snapshots, mirrors, NFS access and full read-write file semantics. MapR's distribution was selected by Amazon to provide premium versions of the Elastic Map Reduce (EMR) service.
- MetaScale, a wholly owned subsidiary of Sears offers vendor neutral and platform independent Hadoop consulting and implementation services.
- Microsoft offers Windows Azure HDInsight, which is a 100% Apache compatible Hadoop distribution on Windows Azure.
- OceanSync Hadoop Management and Visualization Software allows users to control, monitor, and visualize all aspects of a Hadoop cluster including data analytic workflow management and output data processing visualization. The package is offered in three versions, OceanSync Free Desktop Edition, OceanSync Enterprise Edition with Visualization, and OceanSync Mobile for iPhone/Android devices, by Dovestech.
- Oracle announced the *Big Data Appliance*, which integrates Cloudera's Distribution Including Hadoop (CDH), Oracle Linux, the R programming language, and a NoSQL database with the Exadata hardware.
- Pentaho announced support for Apache Hadoop allowing companies to access data integration and business analytics directly against Apache Hadoop based distributions of Hadoop. In January 2012, Pentaho announced they made their big data integration capabilities freely under open source, and moved the entire Pentaho Kettle (data integration engine) project from the LGPL license to the Apache License.
- Pivotal offers Pivotal HD, a distribution of Hadoop that includes HAWQ, with 100% ANSI SQL compatibility.
- Platform Computing announced support for the Hadoop MapReduce API in its Symphony software.
- Silicon Graphics International offers Hadoop optimized solutions based on the SGI Rackable and CloudRack server lines with implementation services.
- Splunk offers a Hadoop integration product called Hadoop Connect, which is certified on MapR, Cloudera, Hortonworks, and Apache Hadoop. This integration allows users to search Hadoop data from Splunk and import data from Hadoop into Splunk and vice versa.
- sqrrl offers sqrrl enterprise, which extends hadoop with Apache accumulo and combines the features of several datastores (Column + Document + Graph).[13]
- Supermicro offers Hadoop servers and rack solutions based on standard servers, multi-nodes and fixed-disks designs.
- Syncsort provides an ETL Solution, which extends the capabilities of Hadoop, turning it into a highly scalable, affordable, and easy-to-use data integration environment.[14]
- Talend offers Talend Open Studio for Big Data, released under the Apache Software License, that includes native support for Apache Hadoop.
- Teradata offers Teradata Appliance for Hadoop, as well as customer support for software-only Hadoop deployments.[15]
- TIBCO supports Hadoop with their Businessworks Hadoop plugin and the TIBCO Spotfire Client.
- WANdisco offers Non Stop Hadoop for both Cloudera & Hortonworks Hadoop distributions.[16]
- Zettaset offers new version of its Big Data Mgt Platform based on Hadoop. Zettaset's Big Data Platform delivers High Availability via NameNode Failover, a streamlined UI, network Time Protocol and built in security via Kerberos Authentication

ASF's view on the use of "Hadoop" in product names

The Apache Software Foundation has stated that only software officially released by the Apache Hadoop Project can be called *Apache Hadoop* or *Distributions of Apache Hadoop*. The naming of products and derivative works from other vendors and the term "compatible" are somewhat controversial within the Hadoop developer community.

Papers

Some papers influenced the birth and growth of Hadoop and big data processing. Here is a partial list:

- 2004 MapReduce: Simplified Data Processing on Large Clusters] by Jeffrey Dean and Sanjay Ghemawat from Google Lab. This paper inspired Doug Cutting to develop an open-source implementation of the Map-Reduce framework. He named it Hadoop [1], after his son's toy elephant.
- 2005 From Databases to Dataspaces: A New Abstraction for Information Management [17], the authors highlight the need for storage systems to accept all data formats and to provide APIs for data access that evolve based on the storage system's understanding of the data.
- 2006 Bigtable: A Distributed Storage System for Structured Data [18] from Google Lab.
- 2008 H-store: a high-performance, distributed main memory transaction processing system [19]
- 2009 MAD Skills: New Analysis Practices for Big Data [20]
- 2011 Apache Hadoop Goes Realtime at Facebook [21]

References

[1] http://hadoop.apache.org/

[2] Hadoop creator goes to Cloudera (http://www.sdtimes.com/blog/post/2009/08/10/Hadoop-creator-goes-to-Cloudera.aspx)

[3] "Hadoop contains the distributed computing platform that was formerly a part of Nutch. This includes the Hadoop Distributed Filesystem (HDFS) and an implementation of MapReduce." About Hadoop (http://hadoop.apache.org/core/)

[4] http://www.slideshare.net/mcsrivas/design-scale-and-performance-of-maprs-distribution-for-hadoop

[5] http://www.mapr.com/products/mapr-editions

[6] http://aws.amazon.com/elasticmapreduce/mapr/

[7] job (http://hadoop.apache.org/common/docs/current/commands_manual.html#job)

[8] (http://svn.apache.org/repos/asf/hadoop/mapreduce/trunk/src/contrib/fairscheduler/designdoc/fair_scheduler_design_doc.pdf) Hadoop Fair Scheduler Design Document

[9] (http://hadoop.apache.org/common/docs/r0.20.2/capacity_scheduler.html) Capacity Scheduler Guide

[10] Yahoo! Launches World's Largest Hadoop Production Application (Hadoop and Distributed Computing at Yahoo!) (http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html)

[11] http://aws.amazon.com/elasticmapreduce/

[12] http://aws.amazon.com/ec2/spot-instances/

[13] Gallagher, Sean. (2013-06-19) Secret Sqrrl: NSA "spin-off" company releases data mining tool (http://arstechnica.com/information-technology/2013/06/secret-sqrrl-nsa-spin-off-company-releases-data-mining-tool/). Ars Technica. Retrieved on 2013-09-18.

[14] DMX-h Solutions: Hadoop ETL (http://www.syncsort.com/en/Data-Integration/Solutions/Hadoop-Solutions/Hadoop-ETL). Syncsort. Retrieved on 2013-09-18.

[15] Teradata Portfolio for Hadoop (http://www.teradata.com/Teradata-Portfolio-for-Hadoop)

[16] Non-Stop Hadoop for Cloudera & Hortonworks (http://www.wandisco.com/hadoop|title=)

[17] http://www.eecs.berkeley.edu/~franklin/Papers/dataspaceSR.pdf

[18] http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/archive/bigtable-osdi06.pdf

[19] http://www.vldb.org/pvldb/1/1454211.pdf

[20] http://db.cs.berkeley.edu/jmh/papers/madskills-032009.pdf

[21] http://borthakur.com/ftp/RealtimeHadoopSigmod2011.pdf

Bibliography

- Lam, Chuck (July 28, 2010). *Hadoop in Action* (1st ed.). Manning Publications. p. 325. ISBN 1-935182-19-6.
- Venner, Jason (June 22, 2009). *Pro Hadoop* (http://www.apress.com/book/view/1430219424) (1st ed.). Apress. p. 440. ISBN 1-4302-1942-4.
- White, Tom (June 16, 2009). *Hadoop: The Definitive Guide* (http://oreilly.com/catalog/9780596521974) (1st ed.). O'Reilly Media. p. 524. ISBN 0-596-52197-9.

External links

- Official Hadoop Homepage (http://hadoop.apache.org/)
- Official Hadoop Wiki (http://wiki.apache.org/hadoop/)
- Introducing Apache Hadoop: The Modern Data Operating System (http://www.stanford.edu/class/ee380/Abstracts/111116.html) — lecture given at Stanford University by Co-Founder and CTO of Cloudera, Amr Awadallah (video archive (http://ee380.stanford.edu/cgi-bin/videologger.php?target=111116-ee380-300.asx)).

Pig (programming language)

Pig is a high-level platform for creating MapReduce programs used with Hadoop. The language for this platform is called **Pig Latin**. Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for RDBMS systems. Pig Latin can be extended using UDF (User Defined Functions) which the user can write in Java, Python, JavaScript, Ruby or Groovy and then call directly from the language.

Pig was originally developed at Yahoo Research around 2006 for researchers to have an ad-hoc way of creating and executing map-reduce jobs on very large data sets. In 2007, it was moved into the Apache Software Foundation.

Example

Below is an example of a "Word Count" program in Pig Latin:

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS (line:chararray);

-- Extract words from each line and put them into a pig bag
-- datatype, then flatten the bag to get one word on each row
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;

-- filter out any words that are just white spaces
filtered_words = FILTER words BY word MATCHES '\\w+';

-- create a group for each word
word_groups = GROUP filtered_words BY word;

-- count the entries in each group
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS word;

-- order the records by count
ordered_word_count = ORDER word_count BY count DESC;
```

```
STORE ordered_word_count INTO '/tmp/number-of-words-on-internet';
```

The above program will generate parallel executable tasks which can be distributed across multiple machines in a Hadoop cluster to count the number of words in a dataset such as all the webpages on the internet.

Pig vs SQL

In comparison to SQL, Pig

1. uses lazy evaluation,
2. uses ETL,
3. is able to store data at any point during a pipeline,
4. declares execution plans,
5. supports pipeline splits.

On the other hand, it has been argued DBMSs are substantially faster than the MapReduce system once the data is loaded, but that loading the data takes considerably longer in the database systems. It has also been argued RDBMSs offer out of the box support for column-storage, working with compressed data, indexes for efficient random data access, and transaction- level fault tolerance.[1]

Pig Latin is procedural and fits very naturally in the pipeline paradigm while SQL is instead declarative. In SQL users can specify that data from two tables must be joined, but not what join implementation to use. Pig Latin allows users to specify an implementation or aspects of an implementation to be used in executing a script in several ways. In effect, Pig Latin programming is similar to specifying a query execution plan, making it easier for programmers to explicitly control the flow of their data processing task.[2]

SQL is oriented around queries that produce a single result. SQL handles trees naturally, but has no built in mechanism for splitting a data processing stream and applying different operators to each sub-stream. Pig Latin script describes a directed acyclic graph (DAG) rather than a pipeline.

Pig Latin's ability to include user code at any point in the pipeline is useful for pipeline development. If SQL is used, data must first be imported into the database, and then the cleansing and transformation process can begin.[]

References

[1] Communications of the ACM: MapReduce and Parallel DBMSs: Friends or Foes? (http://database.cs.brown.edu/papers/stonebraker-cacm2010.pdf)

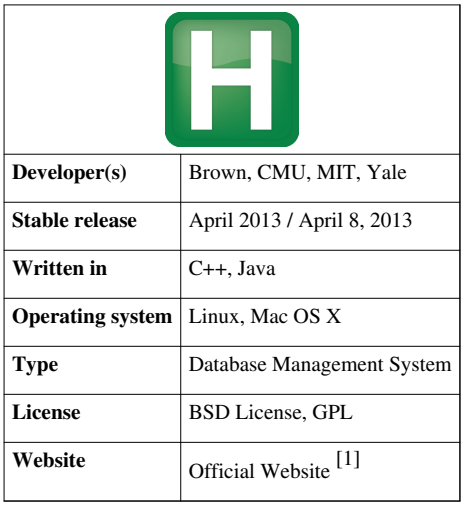
[2] ACM SigMod 08: Pig Latin: A Not-So-Foreign Language for Data Processing (http://infolab.stanford.edu/~olston/publications/sigmod08.pdf)

External links

- Official site (http://pig.apache.org/)

H-Store

H-Store



Developer(s)	Brown, CMU, MIT, Yale
Stable release	April 2013 / April 8, 2013
Written in	C++, Java
Operating system	Linux, Mac OS X
Type	Database Management System
License	BSD License, GPL
Website	Official Website [1]

H-Store is an experimental database management system (DBMS) designed for online transaction processing applications that is being developed by a team at Brown University, Carnegie Mellon University, the Massachusetts Institute of Technology, and Yale University. The system's design was developed in 2007 by database researchers Michael Stonebraker, Sam Madden, and Daniel Abadi.

The significance of the **H-Store** is that it is the first implementation of a new class of parallel database management systems, called NewSQL, that provide the high-throughput and high-availability of NoSQL systems, but without giving up the transactional guarantees of a traditional DBMS. Such systems are able to scale out horizontally across multiple machines to improve throughput, as opposed to moving to a more powerful, more expensive machine for a single-node system.

H-Store is able to execute transaction processing with high throughput by forgoing much of legacy architecture of System R-like systems. For example, **H-Store** was designed as a parallel, row-storage relational DBMS that runs on a cluster of shared-nothing, main memory executor nodes. The database is partitioned into disjoint subsets that are assigned to a single-threaded execution engine assigned to one and only one core on a node. Each engine has exclusive access to all of the data at its partition. Because it is single-threaded, only one transaction at a time is able to access the data stored at its partition. Thus, there are no physical locks or latches in the system, and no transaction will stall waiting for another transaction once it is started.

H-Store is licensed under the BSD license and GPL licenses. The commercial version of H-Store's design is VoltDB.

References

[1] http://hstore.cs.brown.edu

Information Security

Data control language

A **data control language** (**DCL**) is a syntax similar to a computer programming language used to control access to data stored in a database. In particular, it is a component of Structured Query Language (SQL).

Examples of DCL commands include:

- GRANT to allow specified users to perform specified tasks.
- REVOKE to cancel previously granted or denied permissions.

The operations for which privileges may be granted to or revoked from a user or role may include CONNECT, SELECT, INSERT, UPDATE, DELETE, EXECUTE, and USAGE.

In the Oracle database, executing a DCL command issues an implicit commit. Hence you cannot roll back the command.

In PostgreSQL, executing DCL is transactional, and can be rolled back.

SQL injection

SQL injection is a code injection technique, used to attack data driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

In a 2012 study, security company Imperva observed that the average web application received 4 attack campaigns per month, and retailers received twice as many attacks as other industries.

<table>
<tr><th>Classification parameters</th><th>Methods</th><th colspan="2">Techniques/ Implementation</th></tr>
<tr><td rowspan="8">Intent</td><td>Identifying injectable parameters</td><td colspan="2" rowspan="8">see 'Input type of attacks'</td></tr>
<tr><td>Extracting Data</td></tr>
<tr><td>Adding or Modifying Data</td></tr>
<tr><td>Performing Denial of Service</td></tr>
<tr><td>Evading detection</td></tr>
<tr><td>Bypassing Authentication</td></tr>
<tr><td>Executing remote commands</td></tr>
<tr><td>Performing privilege escalation</td></tr>
<tr><td rowspan="6">Input Source</td><td rowspan="2">Injection through user input</td><td>Malicious strings in Web forms</td><td>URL: GET- Method</td></tr>
<tr><td colspan="2">Input filed(s): POST- Method</td></tr>
<tr><td>Injection through cookies</td><td colspan="2">Modified cookie fields containing SQLIA</td></tr>
<tr><td>Injection through server variables</td><td colspan="2">Headers are manipulated to contain SQLIA</td></tr>
<tr><td rowspan="4">Second-order injection</td><td colspan="2">Frequency-based Primary Application</td></tr>
<tr><td colspan="2">Frequency-based Secondary Application</td></tr>
<tr><td colspan="2">Secondary Support Application</td></tr>
<tr><td colspan="2">Cascaded Submission Application</td></tr>
<tr><td rowspan="13">Input type of attacks, technical aspect</td><td rowspan="5">Classic SQLIA</td><td colspan="2">Piggy-Backed Queries</td></tr>
<tr><td colspan="2">Tautologies</td></tr>
<tr><td colspan="2">Alternate Encodings</td></tr>
<tr><td colspan="2">Illegal/ Logically Incorrect Queries</td></tr>
<tr><td colspan="2">UNION SQLIA</td></tr>
<tr><td rowspan="5">Inference</td><td colspan="2">Stored Procedures SQLIA</td></tr>
<tr><td rowspan="3">Classic Blind SQLIA</td><td>Conditional Responses</td></tr>
<tr><td>Conditional Errors</td></tr>
<tr><td>Out-Of-Band Channeling</td></tr>
<tr><td rowspan="2">Timing SQLIA</td><td>Double - Blind - SQLIA(Time-delays/ Benchmark attacks)</td></tr>
<tr><td>Deep Blind SQLIA (Multiple statements SQLIA)</td></tr>
<tr><td rowspan="2">DBMS specific SQLIA</td><td colspan="2">DB Fingerprinting</td></tr>
<tr><td colspan="2">DB Mapping</td></tr>
<tr><td>Compounded SQLIA</td><td colspan="2">Fast-Fluxing SQLIA</td></tr>
</table>

A Classification of SQL injection attacking vector until 2010.

Forms

SQL injection (SQLI) is considered one of the top 10 web application vulnerabilities of 2007 and 2010 by the Open Web Application Security Project. In 2013, SQLI was rated the number one attack on the OWASP top ten. There are five main sub-classes of SQL injection:

- Classic SQLI
- Blind or Inference SQL injection
- Database management system-specific SQLI
- Compounded SQLI
  - SQL injection + insufficient authentication
  - SQL injection + DDoS attacks
  - SQL injection + DNS hijacking

- SQL injection + XSS

The Storm Worm is one representation of Compounded SQLI.

This classification represents the state of SQLI, respecting its evolution until 2010—further refinement is underway.

Technical implementations

Incorrectly filtered escape characters

This form of SQL injection occurs when user input is not filtered for escape characters and is then passed into a SQL statement. This results in the potential manipulation of the statements performed on the database by the end-user of the application.

The following line of code illustrates this vulnerability:

```
statement = "
```

```
SELECT * FROM users WHERE name =
```

```
'" + userName + "';"
```

This SQL code is designed to pull up the records of the specified username from its table of users. However, if the "userName" variable is crafted in a specific way by a malicious user, the SQL statement may do more than the code author intended. For example, setting the "userName" variable as:

```
' or '1'='1'
```

or using comments to even block the rest of the query (there are three types of SQL comments):

```
' or '1'='1' -- '
' or '1'='1' ({ '
' or '1'='1' /* '
```

renders one of the following SQL statements by the parent language:

```
SELECT * FROM users WHERE name = '' or '1'='1';
```

```
SELECT * FROM users WHERE name = '' or '1'='1' -- ';
```

If this code were to be used in an authentication procedure then this example could be used to force the selection of a valid username because the evaluation of '1'='1' is always true.

The following value of "userName" in the statement below would cause the deletion of the "users" table as well as the selection of all data from the "userinfo" table (in essence revealing the information of every user), using an API that allows multiple statements:

```
a';
```

```
DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't
```

This input renders the final SQL statement as follows and specified:

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM
userinfo WHERE 't' = 't';
```

While most SQL server implementations allow multiple statements to be executed with one call in this way, some SQL APIs such as PHP's `mysql_query();` function do not allow this for security reasons. This prevents

attackers from injecting entirely separate queries, but doesn't stop them from modifying queries.

Incorrect type handling

This form of SQL injection occurs when a user-supplied field is not strongly typed or is not checked for type constraints. This could take place when a numeric field is to be used in a SQL statement, but the programmer makes no checks to validate that the user supplied input is numeric. For example:

```
statement := "
```

```
SELECT * FROM userinfo WHERE id =
```

```
" + a_variable + ";"
```

It is clear from this statement that the author intended a_variable to be a number correlating to the "id" field. However, if it is in fact a string then the end-user may manipulate the statement as they choose, thereby bypassing the need for escape characters. For example, setting a_variable to

```
1;DROP TABLE users
```

will drop (delete) the "users" table from the database, since the SQL becomes:

```
SELECT * FROM userinfo WHERE id=1;DROP TABLE users;
```

Blind SQL injection

Blind SQL Injection is used when a web application is vulnerable to an SQL injection but the results of the injection are not visible to the attacker. The page with the vulnerability may not be one that displays data but will display differently depending on the results of a logical statement injected into the legitimate SQL statement called for that page. This type of attack can become time-intensive because a new statement must be crafted for each bit recovered. There are several tools that can automate these attacks once the location of the vulnerability and the target information has been established.

Conditional responses

One type of blind SQL injection forces the database to evaluate a logical statement on an ordinary application screen. As an example, a book review website uses a query string to determine which book review to display. So the URL **`http://books.example.com/showReview.php?ID=5`** would cause the server to run the query

```
SELECT * FROM bookreviews WHERE ID = 'Value(ID)';
```

from which it would populate the review page with data from the review with ID 5, stored in the table bookreviews. The query happens completely on the server; the user does not know the names of the database, table, or fields, nor does the user know the query string. The user only sees that the above URL returns a book review. A hacker can load the URLs **`http://books.example.com/showReview.php?ID=5 OR 1=1`** and **`http://books.example.com/showReview.php?ID=5 AND 1=2`**, which may result in queries

```
SELECT * FROM bookreviews WHERE ID = '5' OR '1'='1';
SELECT * FROM bookreviews WHERE ID = '5' AND '1'='2';
```

respectively. If the original review loads with the "1=1" URL and a blank or error page is returned from the "1=2" URL, the site is likely vulnerable to a SQL injection attack. The hacker may proceed with this query string designed to reveal the version number of MySQL running on the server: **`http://books.example.com/showReview.php?ID=5 AND substring(@@version,1,1)=4`**, which would show the book review on a server running MySQL 4 and a blank or error page otherwise. The hacker

can continue to use code within query strings to glean more information from the server until another avenue of attack is discovered or his or her goals are achieved.

Second Order SQL Injection

Second Order SQLi happens when submitted values contain SQL injection attacks that are stored instead of executed immediately. In some cases, the application may correctly encode a SQL statement and store it as valid SQL. Then, another part of that application without controls to protect against SQL Injection might execute that stored SQL statement. This attack requires more knowledge of how submitted values are later used. Automated web application security scanners would not easily detect this type of SQL Injection and may need to be manually instructed where to check for evidence of second order SQLi.

Mitigation

Parameterized statements

With most development platforms, parameterized statements that work with parameters can be used (sometimes called placeholders or bind variables) instead of embedding user input in the statement. A placeholder can only store a value of the given type and not an arbitrary SQL fragment. Hence the SQL injection would simply be treated as a strange (and probably invalid) parameter value.

In many cases, the SQL statement is fixed, and each parameter is a scalar, not a table. The user input is then assigned (bound) to a parameter.

Enforcement at the coding level

Using object-relational mapping libraries avoids the need to write SQL code. The ORM library in effect will generate parameterized SQL statements from object-oriented code.

Escaping

A straightforward, though error-prone, way to prevent injections is to escape characters that have a special meaning in SQL. The manual for an SQL DBMS explains which characters have a special meaning, which allows creating a comprehensive blacklist of characters that need translation. For instance, every occurrence of a single quote (`'`) in a parameter must be replaced by two single quotes (`''`) to form a valid SQL string literal. For example, in PHP it is usual to escape parameters using the function `mysql_real_escape_string();` before sending the SQL query:

```
$mysqli = new mySqli('hostname', 'db_username', 'db_password',
'db_name');
$query = sprintf("SELECT * FROM `Users` WHERE UserName='%s' AND
Password='%s'",
                 $mysqli->real_escape_string($Username),
                 $mysqli->real_escape_string($Password));
$mysqli->query($query);
```

Note: 'mysql' is deprecated, and should be avoided. `mysqli [1]`, is preferred. This function, i.e. mysql_real_escape_string(), calls MySQL's library function mysql_real_escape_string, which prepends backslashes to the following characters: \x00, \n, \r, \, ', " and \x1a. This function must always (with few exceptions) be used to make data safe before sending a query to MySQL.

There are other functions for many database types in PHP such as pg_escape_string() for PostgreSQL. There is, however, one function that works for escaping characters, and is used especially for querying on databases that do

not have escaping functions in PHP. This function is: **addslashes(string $str)**. It returns a string with backslashes before characters that need to be quoted in database queries, etc. These characters are single quote ('), double quote ("), backslash (\) and NUL (the NULL byte).
Routinely passing escaped strings to SQL is error prone because it is easy to forget to escape a given string. Creating a transparent layer to secure the input can reduce this error-proneness, if not entirely eliminate it.

In .Net Framework just use parameterized query or stored procedure using parameters

```
SqlCommand cmd1= new SqlCommand("Select * from users where username ='"+@paramUser+"' and password='"+@paramPass+"'", conn1)
```

cmd1.CommandType= CommandType.Text cmd1.Parameters.Add("@paramUser", SqlDbType.Nvarchar,20).Value=Username1 cmd1.Parameters.Add("@paramPass", SqlDbType.Nvarchar,20).Value=pass1

In this case if for example Username1="' or 1=1--" this will not succeed.

Pattern check

Integer, float or boolean parameters can be checked if their value is valid representation for the given type. Strings that must follow some strict pattern (date, UUID, alphanumeric only, etc.) can be checked if they match this pattern.

Database permissions

Limiting the permissions on the database logon used by the web application to only what is needed may help reduce the effectiveness of any SQL injection attacks that exploit any bugs in the web application.

For example, on Microsoft SQL Server, a database logon could be restricted from selecting on some of the system tables which would limit exploits that try to insert JavaScript into all the text columns in the database.

```
deny select on sys.sysobjects to webdatabaselogon;
deny select on sys.objects to webdatabaselogon;
deny select on sys.tables to webdatabaselogon;
deny select on sys.views to webdatabaselogon;
deny select on sys.packages to webdatabaselogon;
```

Examples

- In September 1995, Andrew Plato, a technical writer for Microsoft discovered that he could send SQL queries through URL string of an early e-commerce site and directly query the database (SQL Server 6.0). Unaware of what this meant, Plato approached developers who dismissed the issue as irrelevant. [*citation needed*]
- In February 2002, Jeremiah Jacks discovered that Guess.com was vulnerable to an SQL injection attack, permitting anyone able to construct a properly-crafted URL to pull down 200,000+ names, credit card numbers and expiration dates in the site's customer database.
- On November 1, 2005, a teenage hacker used SQL injection to break into the site of a Taiwanese information security magazine from the Tech Target group and steal customers' information.
- On January 13, 2006, Russian computer criminals broke into a Rhode Island government web site and allegedly stole credit card data from individuals who have done business online with state agencies.
- On March 29, 2006, a hacker discovered an SQL injection flaw in an official Indian government's tourism site.
- On June 29, 2007, a computer criminal defaced the Microsoft UK website using SQL injection. UK website *The Register* quoted a Microsoft spokesperson acknowledging the problem.
- In January 2008, tens of thousands of PCs were infected by an automated SQL injection attack that exploited a vulnerability in application code that uses Microsoft SQL Server as the database store.

- In July 2008, Kaspersky's Malaysian site was hacked by a Turkish hacker going by the handle of "m0sted", who said to have used an SQL injection.
- In February 2013, a group of Maldivian hackers, hacked the website " UN-Maldives" using SQL Injection.
- In May 28, 2009 Anti-U.S. Hackers Infiltrate Army Servers Investigators believe the hackers used a technique called SQL injection to exploit a security vulnerability in Microsoft's SQL Server database to gain entry to the Web servers. "m0sted" is known to have carried out similar attacks on a number of other Web sites in the past—including against a site maintained by Internet security company Kaspersky Lab.
- On April 13, 2008, the Sexual and Violent Offender Registry of Oklahoma shut down its website for "routine maintenance" after being informed that 10,597 Social Security numbers belonging to sex offenders had been downloaded via an SQL injection attack
- In May 2008, a server farm inside China used automated queries to Google's search engine to identify SQL server websites which were vulnerable to the attack of an automated SQL injection tool.
- In 2008, at least April through August, a sweep of attacks began exploiting the SQL injection vulnerabilities of Microsoft's IIS web server and SQL Server database server. The attack does not require guessing the name of a table or column, and corrupts all text columns in all tables in a single request. A HTML string that references a malware JavaScript file is appended to each value. When that database value is later displayed to a website visitor, the script attempts several approaches at gaining control over a visitor's system. The number of exploited web pages is estimated at 500,000.
- On August 17, 2009, the United States Department of Justice charged an American citizen, Albert Gonzalez, and two unnamed Russians with the theft of 130 million credit card numbers using an SQL injection attack. In reportedly "the biggest case of identity theft in American history", the man stole cards from a number of corporate victims after researching their payment processing systems. Among the companies hit were credit card processor Heartland Payment Systems, convenience store chain 7-Eleven, and supermarket chain Hannaford Brothers.
- In December 2009, an attacker breached a RockYou plaintext database containing the unencrypted usernames and passwords of about 32 million users using an SQL injection attack.
- On July 2010, a South American security researcher who goes by the handle "Ch Russo" obtained sensitive user information from popular BitTorrent site The Pirate Bay. He gained access to the site's administrative control panel and exploited a SQL injection vulnerability that enabled him to collect user account information, including IP addresses, MD5 password hashes and records of which torrents individual users have uploaded.
- From July 24 to 26, 2010, attackers from Japan and China used an SQL injection to gain access to customers' credit card data from Neo Beat, an Osaka-based company that runs a large online supermarket site. The attack also affected seven business partners including supermarket chains Izumiya Co, Maruetsu Inc, and Ryukyu Jusco Co. The theft of data affected a reported 12,191 customers. As of August 14, 2010 it was reported that there have been more than 300 cases of credit card information being used by third parties to purchase goods and services in China.
- On September 19 during the 2010 Swedish general election a voter attempted a code injection by hand writing SQL commands as part of a write-in vote.
- On November 8, 2010 the British Royal Navy website was compromised by a Romanian hacker named TinKode using SQL injection.[2]
- On February 5, 2011 HBGary, a technology security firm, was broken into by LulzSec using a SQL injection in their CMS-driven website
- On March 27, 2011, mysql.com [3], the official homepage for MySQL, was compromised by a hacker using SQL blind injection
- On April 11, 2011, Barracuda Networks was compromised using an SQL injection flaw. Email addresses and usernames of employees were among the information obtained.
- Over a period of 4 hours on April 27, 2011, an automated SQL injection attack occurred on Broadband Reports website that was able to extract 8% of the username/password pairs: 8,000 random accounts of the 9,000 active

and 90,000 old or inactive accounts.

- On June 1, 2011, "hacktivists" of the group LulzSec were accused of using SQLI to steal coupons, download keys, and passwords that were stored in plaintext on Sony's website, accessing the personal information of a million users.
- In June 2011, PBS was hacked, mostly likely through use of SQL injection; the full process used by hackers to execute SQL injections was described in this Imperva [4] blog.
- In May 2012, the website for *Wurm Online*, a massively multiplayer online game, was shut down from an SQL injection while the site was being updated.
- In July 2012 a hacker group was reported to have stolen 450,000 login credentials from Yahoo!. The logins were stored in plain text and were allegedly taken from a Yahoo subdomain, Yahoo! Voices. The group breached Yahoo's security by using a "union-based SQL injection technique".[5][6]
- On October 1, 2012, a hacker group called "Team GhostShell" published the personal records of students, faculty, employees, and alumni from 53 universities including Harvard, Princeton, Stanford, Cornell, Johns Hopkins, and the University of Zurich on pastebin.com. The hackers claimed that they were trying to “raise awareness towards the changes made in today’s education”, bemoaning changing education laws in Europe and increases in tuition in the United States.
- On June 27, 2013, hacker group "RedHack" breached Istanbul Administration Site. They claimed that, they’ve been able to erase people's debts to water, gas, Internet, electricity, and telephone companies. Additionally, they published admin user name and password for other citizens to log in and clear their debts early morning. They announced the news from twitter.
- On November 4th, 2013, hacktivist group "RaptorSwag" allegedly compromised 71 Chinese government databases using an SQL injection attack on the Chinese Chamber of International Commerce. The leaked data was posted publicly in cooperation with Anonymous.[7]

In popular culture

- Unauthorized login to web sites (i.e. hacking) by means of SQL injection forms the basis of one of the subplots in J.K. Rowling's novel "The Casual Vacancy", published in 2012.
- The minor *xkcd* character "Robert'); DROP TABLE students;--" (also known as "Little Bobby Tables") was named to carry out a SQL injection.

References

[1] http://in3.php.net/manual/en/book.mysqli.php
[2] Royal Navy website attacked by Romanian hacker (http://www.bbc.co.uk/news/technology-11711478) *BBC News*, 8-11-10, Accessed November 2010
[3] http://www.mysql.com
[4] http://blog.imperva.com/2011/05/pbs-breached-how-hackers-probably-did-it.html
[5] Chenda Ngak. "Yahoo reportedly hacked: Is your account safe?" (http://www.cbsnews.com/8301-501465_162-57470956-501465/yahoo-reportedly-hacked-is-your-account-safe/), CBS News. July 12, 2012. Retrieved July 16, 2012.
[6] http://www.zdnet.com/450000-user-passwords-leaked-in-yahoo-breach-7000000772/
[7] http://news.softpedia.com/news/Hackers-Leak-Data-Allegedly-Stolen-from-Chinese-Chamber-of-Commerce-Website-396936.shtml

External links

- Complete Reference Guide to SQL Injection, Attack and Prevention Method of SQL Injection (http://www.worldofhacker.com/2013/09/complete-reference-guide-to-sqli-how-to.html) by WorldofHacker.
- SQL Injection Knowledge Base (http://www.websec.ca/kb/sql_injection), by Websec.
- SQL Injection Wiki (http://www.sqlinjectionwiki.com)
- Blind Sql injection with Regular Expression (http://www.ihteam.net/papers/blind-sqli-regexp-attack.pdf)
- WASC Threat Classification - SQL Injection Entry (http://projects.webappsec.org/SQL-Injection), by the Web Application Security Consortium.
- Why SQL Injection Won't Go Away (http://docs.google.com/leaf?id=0BykNNUTb95yzYTRjMjNjMWEtODBmNS00YzgwLTlmMGYtNWZmODI2MTNmZWYw&sort=name&layout=list&num=50), by Stuart Thomas.
- SQL Injection Attacks by Example (http://unixwiz.net/techtips/sql-injection.html), by Steve Friedl
- SQL Injection Prevention Cheat Sheet (http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet), by OWASP.
- SQL Injection Tutorial (http://www.breakthesecurity.com/2010/12/hacking-website-using-sql-injection.html), by BTS.
- sqlmap: automatic SQL injection and database takeover tool (http://sqlmap.org/)
- SDL Quick security references on SQL injection (http://go.microsoft.com/?linkid=9707610) by Bala Neerumalla.
- Backdoor Web-server using MySQL SQL Injection (http://www.greensql.com/articles/backdoor-webserver-using-mysql-sql-injection) By Yuli Stremovsky
- Defacing website with SQL injection (http://www.sploitswiki.com/2011/02/deface-website-sql-injection.html) by sploitswiki

Data Warehouseing

Business intelligence

Business intelligence (**BI**) is a set of theories, methodologies, architectures, and technologies that transform raw data into meaningful and useful information for business purposes. BI can handle enormous amounts of unstructured data to help identify, develop and otherwise create new opportunities. BI, in simple words, makes interpreting voluminous data friendly. Making use of new opportunities and implementing an effective strategy can provide a competitive market advantage and long-term stability.[1]

Generally, Business Intelligence is made up of an increasing number of components, these are:

- Multidimensional aggregation and allocation
- Denormalization, tagging and standardization
- Realtime reporting with analytical alert
- Interface with unstructured data source
- Group consolidation, budgeting and rolling forecast
- Statistical inference and probabilistic simulation
- Key performance indicators optimization
- Version control and process management
- Open item management

BI technologies provide historical, current and predictive views of business operations. Common functions of business intelligence technologies are reporting, online analytical processing, analytics, data mining, process mining, complex event processing, business performance management, benchmarking, text mining, predictive analytics and prescriptive analytics.

Though the term business intelligence is sometimes a synonym for competitive intelligence (because they both support decision making), BI uses technologies, processes, and applications to analyze mostly internal, structured data and business processes while competitive intelligence gathers, analyzes and disseminates information with a topical focus on company competitors. If understood broadly, business intelligence can include the subset of competitive intelligence.

History

The term Business Intelligence was originally first phrased by Richard Millar Devens' in the 'Cyclopædia of Commercial and Business Anecdotes' from 1865. Devens used the term to describe how the banker Sir Henry Furnese, gained profit by receiving and acting upon information about his environment, prior to his competitors. "*Throughout Holland, Flanders, France, and Germany, he maintained a complete and perfect train of business intelligence. The news of the many battles fought was thus received first by him, and the fall of Namur added to his profits, owing to his early receipt of the news*." (Devens, (1865), p. 210). The ability to collect and react accordingly based on the information retrieved, an ability that Furnese excelled in, is today still at the very heart of BI.

In a 1958 article, IBM researcher Hans Peter Luhn used the term business intelligence. He employed the Webster's dictionary definition of intelligence: "the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal."

Business intelligence as it is understood today is said to have evolved from the decision support systems (DSS) that began in the 1960s and developed throughout the mid-1980s. DSS originated in the computer-aided models created

to assist with decision making and planning. From DSS, data warehouses, Executive Information Systems, OLAP and business intelligence came into focus beginning in the late 80s.

In 1988, an Italian-Dutch-French-English consortium organized an international meeting on the Multiway Data Analysis in Rome.[2] The ultimate goal is to reduce the multiple dimensions down to one or two (by detecting the patterns within the data) that can then be presented to human decision-makers.

In 1989, Howard Dresner (later a Gartner Group analyst) proposed "business intelligence" as an umbrella term to describe "concepts and methods to improve business decision making by using fact-based support systems." It was not until the late 1990s that this usage was widespread.

Business intelligence and data warehousing

Often BI applications use data gathered from a data warehouse or a data mart. A data warehouse is a copy of analytical data that facilitates decision support. However, not all data warehouses are used for business intelligence, nor do all business intelligence applications require a data warehouse.

To distinguish between the concepts of business intelligence and data warehouses, Forrester Research often defines business intelligence in one of two ways:

Using a broad definition: "Business Intelligence is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making." When using this definition, business intelligence also includes technologies such as data integration, data quality, data warehousing, master data management, text and content analytics, and many others that the market sometimes lumps into the Information Management segment. Therefore, Forrester refers to *data preparation* and *data usage* as two separate, but closely linked segments of the business intelligence architectural stack.

Forrester defines the latter, narrower business intelligence market as, "...referring to just the top layers of the BI architectural stack such as reporting, analytics and dashboards."

Business intelligence and business analytics

Thomas Davenport argues that business intelligence should be divided into querying, reporting, OLAP, an "alerts" tool, and business analytics. In this definition, business analytics is the subset of BI based on statistics, prediction, and optimization.

Applications in an enterprise

Business intelligence can be applied to the following business purposes, in order to drive business value.[*citation needed*]

1. Measurement – program that creates a hierarchy of performance metrics (see also Metrics Reference Model) and benchmarking that informs business leaders about progress towards business goals (business process management).
2. Analytics – program that builds quantitative processes for a business to arrive at optimal decisions and to perform business knowledge discovery. Frequently involves: data mining, process mining, statistical analysis, predictive analytics, predictive modeling, business process modeling, complex event processing and prescriptive analytics.
3. Reporting/enterprise reporting – program that builds infrastructure for strategic reporting to serve the strategic management of a business, not operational reporting. Frequently involves data visualization, executive information system and OLAP.
4. Collaboration/collaboration platform – program that gets different areas (both inside and outside the business) to work together through data sharing and electronic data interchange.

5. Knowledge management – program to make the company data driven through strategies and practices to identify, create, represent, distribute, and enable adoption of insights and experiences that are true business knowledge. Knowledge management leads to learning management and regulatory compliance.

In addition to above, business intelligence also can provide a pro-active approach, such as ALARM function to alert immediately to end-user. There are many types of alerts, for example if some business value exceeds the threshold value the color of that amount in the report will turn RED and the business analyst is alerted. Sometimes an alert mail will be sent to the user as well. This end to end process requires data governance, which should be handled by the expert.[*citation needed*]

Prioritization of business intelligence projects

It is often difficult to provide a positive business case for business intelligence initiatives and often the projects must be prioritized through strategic initiatives. Here are some hints and advantages to increase the benefits for a BI project.

- As described by Kimball[3] you must determine the tangible benefits such as eliminated cost of producing legacy reports.
- Enforce access to data for the entire organization. In this way even a small benefit, such as a few minutes saved, makes a difference when multiplied by the number of employees in the entire organization.
- As described by Ross, Weil & Roberson for Enterprise Architecture,[4] consider letting the BI project be driven by other business initiatives with excellent business cases. To support this approach, the organization must have enterprise architects who can identify suitable business projects.
- Use a structured and quantitative methodology to create defensible prioritization in line with the actual needs of the organization, such as a weighted decision matrix.

Success factors of implementation

Before implementing a BI solution, it is worth taking different factors into consideration before proceeding. According to Kimball et al., these are the three critical areas that you need to assess within your organization before getting ready to do a BI project:[5]

1. The level of commitment and sponsorship of the project from senior management
2. The level of business need for creating a BI implementation
3. The amount and quality of business data available.

Business sponsorship

The commitment and sponsorship of senior management is according to Kimball *et al.*, the most important criteria for assessment.[6] This is because having strong management backing helps overcome shortcomings elsewhere in the project. However, as Kimball *et al.* state: “even the most elegantly designed DW/BI system cannot overcome a lack of business [management] sponsorship”.[7]

It is important that personnel who participate in the project have a vision and an idea of the benefits and drawbacks of implementing a BI system. The best business sponsor should have organizational clout and should be well connected within the organization. It is ideal that the business sponsor is demanding but also able to be realistic and supportive if the implementation runs into delays or drawbacks. The management sponsor also needs to be able to assume accountability and to take responsibility for failures and setbacks on the project. Support from multiple members of the management ensures the project does not fail if one person leaves the steering group. However, having many managers work together on the project can also mean that there are several different interests that attempt to pull the project in different directions, such as if different departments want to put more emphasis on their usage. This issue can be countered by an early and specific analysis of the business areas that benefit the most from

the implementation. All stakeholders in project should participate in this analysis in order for them to feel ownership of the project and to find common ground.

Another management problem that should be encountered before start of implementation is if the business sponsor is overly aggressive. If the management individual gets carried away by the possibilities of using BI and starts wanting the DW or BI implementation to include several different sets of data that were not included in the original planning phase. However, since extra implementations of extra data may add many months to the original plan, it's wise to make sure the person from management is aware of his actions.

Business needs

Because of the close relationship with senior management, another critical thing that must be assessed before the project begins is whether or not there is a business need and whether there is a clear business benefit by doing the implementation.[8] The needs and benefits of the implementation are sometimes driven by competition and the need to gain an advantage in the market. Another reason for a business-driven approach to implementation of BI is the acquisition of other organizations that enlarge the original organization it can sometimes be beneficial to implement DW or BI in order to create more oversight.

Companies that implement BI are often large, multinational organizations with diverse subsidiaries. A well-designed BI solution provides a consolidated view of key business data not available anywhere else in the organization, giving management visibility and control over measures that otherwise would not exist.

Amount and quality of available data

Without good data, it does not matter how good the management sponsorship or business-driven motivation is. Without proper data, or with too little quality data, any BI implementation fails. Before implementation it is a good idea to do data profiling. This analysis identifies the “content, consistency and structure [..]” of the data. This should be done as early as possible in the process and if the analysis shows that data is lacking, put the project on the shelf temporarily while the IT department figures out how to properly collect data.

When planning for business data and business intelligence requirements, it is always advisable to consider specific scenarios that apply to a particular organization, and then select the business intelligence features best suited for the scenario.

Often, scenarios revolve around distinct business processes, each built on one or more data sources. These sources are used by features that present that data as information to knowledge workers, who subsequently act on that information. The business needs of the organization for each business process adopted correspond to the essential steps of business intelligence. These essential steps of business intelligence include but are not limited to:

1. Go through business data sources in order to collect needed data
2. Convert business data to information and present appropriately
3. Query and analyze data
4. Act on those data collected

The **quality aspect** in business intelligence should cover all the process from the source data to the final reporting. At each step, the **quality gates** are different:

1. Source Data:
    - Data Standardization: make data comparable (same unit, same pattern..)
    - Master Data Management: unique referential
2. Operational Data Store (ODS):
    - Data Cleansing: detect & correct inaccurate data
    - Data Profiling: check inappropriate value, null/empty
3. Datawarehouse:

- Completeness: check that all expected data are loaded
- Referential integrity: unique and existing referential over all sources
- Consistency between sources: check consolidated data vs sources

4. Reporting:
   - Uniqueness of indicators: only one share dictionary of indicators
   - Formula accuracy: local reporting formula should be avoided or checked

User aspect

Some considerations must be made in order to successfully integrate the usage of business intelligence systems in a company. Ultimately the BI system must be accepted and utilized by the users in order for it to add value to the organization.[9][10] If the usability of the system is poor, the users may become frustrated and spend a considerable amount of time figuring out how to use the system or may not be able to really use the system. If the system does not add value to the users´ mission, they simply don't use it.

To increase user acceptance of a BI system, it can be advisable to consult business users at an early stage of the DW/BI lifecycle, for example at the requirements gathering phase. This can provide an insight into the business process and what the users need from the BI system. There are several methods for gathering this information, such as questionnaires and interview sessions.

When gathering the requirements from the business users, the local IT department should also be consulted in order to determine to which degree it is possible to fulfill the business's needs based on the available data.

Taking on a user-centered approach throughout the design and development stage may further increase the chance of rapid user adoption of the BI system.

Besides focusing on the user experience offered by the BI applications, it may also possibly motivate the users to utilize the system by adding an element of competition. Kimball suggests implementing a function on the Business Intelligence portal website where reports on system usage can be found. By doing so, managers can see how well their departments are doing and compare themselves to others and this may spur them to encourage their staff to utilize the BI system even more.

In a 2007 article, H. J. Watson gives an example of how the competitive element can act as an incentive. Watson describes how a large call centre implemented performance dashboards for all call agents, with monthly incentive bonuses tied to performance metrics. Also, agents could compare their performance to other team members. The implementation of this type of performance measurement and competition significantly improved agent performance.

BI chances of success can be improved by involving senior management to help make BI a part of the organizational culture, and by providing the users with necessary tools, training, and support. Training encourages more people to use the BI application.

Providing user support is necessary to maintain the BI system and resolve user problems. User support can be incorporated in many ways, for example by creating a website. The website should contain great content and tools for finding the necessary information. Furthermore, helpdesk support can be used. The help desk can be manned by power users or the DW/BI project team.

BI Portals

A **Business Intelligence portal** (BI portal) is the primary access interface for Data Warehouse (DW) and Business Intelligence (BI) applications. The BI portal is the users first impression of the DW/BI system. It is typically a browser application, from which the user has access to all the individual services of the DW/BI system, reports and other analytical functionality. The BI portal must be implemented in such a way that it is easy for the users of the DW/BI application to call on the functionality of the application.[11]

The BI portal's main functionality is to provide a navigation system of the DW/BI application. This means that the portal has to be implemented in a way that the user has access to all the functions of the DW/BI application.

The most common way to design the portal is to custom fit it to the business processes of the organization for which the DW/BI application is designed, in that way the portal can best fit the needs and requirements of its users.[12]

The BI portal needs to be easy to use and understand, and if possible have a look and feel similar to other applications or web content of the organization the DW/BI application is designed for (consistency).

The following is a list of desirable features for web portals in general and BI portals in particular:

Usable

: User should easily find what they need in the BI tool.

Content Rich

: The portal is not just a report printing tool, it should contain more functionality such as advice, help, support information and documentation.

Clean

: The portal should be designed so it is easily understandable and not over complex as to confuse the users

Current

: The portal should be updated regularly.

Interactive

: The portal should be implemented in a way that makes it easy for the user to use its functionality and encourage them to use the portal. Scalability and customization give the user the means to fit the portal to each user.

Value Oriented

: It is important that the user has the feeling that the DW/BI application is a valuable resource that is worth working on.

Marketplace

There are a number of business intelligence vendors, often categorized into the remaining independent "pure-play" vendors and consolidated "megavendors" that have entered the market through a recent trend of acquisitions in the BI industry.

Some companies adopting BI software decide to pick and choose from different product offerings (best-of-breed) rather than purchase one comprehensive integrated solution (full-service).

Industry-specific

Specific considerations for business intelligence systems have to be taken in some sectors such as governmental banking regulations. The information collected by banking institutions and analyzed with BI software must be protected from some groups or individuals, while being fully available to other groups or individuals. Therefore BI solutions must be sensitive to those needs and be flexible enough to adapt to new regulations and changes to existing law.

Semi-structured or unstructured data

Businesses create a huge amount of valuable information in the form of e-mails, memos, notes from call-centers, news, user groups, chats, reports, web-pages, presentations, image-files, video-files, and marketing material and news. According to Merrill Lynch, more than 85% of all business information exists in these forms. These information types are called either *semi-structured* or *unstructured* data. However, organizations often only use these documents once.

The management of semi-structured data is recognized as a major unsolved problem in the information technology industry. According to projections from Gartner (2003), white collar workers spend anywhere from 30 to 40 percent of their time searching, finding and assessing unstructured data. BI uses both structured and unstructured data, but the former is easy to search, and the latter contains a large quantity of the information needed for analysis and decision making. Because of the difficulty of properly searching, finding and assessing unstructured or semi-structured data, organizations may not draw upon these vast reservoirs of information, which could influence a particular decision, task or project. This can ultimately lead to poorly informed decision making.

Therefore, when designing a business intelligence/DW-solution, the specific problems associated with semi-structured and unstructured data must be accommodated for as well as those for the structured data.

Unstructured data vs. semi-structured data

Unstructured and semi-structured data have different meanings depending on their context. In the context of relational database systems, unstructured data cannot be stored in predictably ordered columns and rows. One type of unstructured data is typically stored in a BLOB (binary large object), a catch-all data type available in most relational database management systems. Unstructured data may also refer to irregularly or randomly repeated column patterns that vary from row to row within each file or document.

Many of these data types, however, like e-mails, word processing text files, PPTs, image-files, and video-files conform to a standard that offers the possibility of metadata. Metadata can include information such as author and time of creation, and this can be stored in a relational database. Therefore it may be more accurate to talk about this as semi-structured documents or data, but no specific consensus seems to have been reached.

Unstructured data can also simply be the knowledge that business users have about future business trends. Business forecasting naturally aligns with the BI system because business users think of their business in aggregate terms. Capturing the business knowledge that may only exist in the minds of business users provides some of the most important data points for a complete BI solution.

Problems with semi-structured or unstructured data

There are several challenges to developing BI with semi-structured data. According to Inmon & Nesavich,[13] some of those are:

1. Physically accessing unstructured textual data – unstructured data is stored in a huge variety of formats.
2. Terminology – Among researchers and analysts, there is a need to develop a standardized terminology.
3. Volume of data – As stated earlier, up to 85% of all data exists as semi-structured data. Couple that with the need for word-to-word and semantic analysis.
4. Searchability of unstructured textual data – A simple search on some data, e.g. apple, results in links where there is a reference to that precise search term. (Inmon & Nesavich, 2008) gives an example: "a search is made on the term felony. In a simple search, the term felony is used, and everywhere there is a reference to felony, a hit to an unstructured document is made. But a simple search is crude. It does not find references to crime, arson, murder, embezzlement, vehicular homicide, and such, even though these crimes are types of felonies."

The use of metadata

To solve problems with searchability and assessment of data, it is necessary to know something about the content. This can be done by adding context through the use of metadata. Many systems already capture some metadata (e.g. filename, author, size, etc.), but more useful would be metadata about the actual content – e.g. summaries, topics, people or companies mentioned. Two technologies designed for generating metadata about content are automatic categorization and information extraction.

Future

A 2009 Gartner paper predicted[14] these developments in the business intelligence market:

- Because of lack of information, processes, and tools, through 2012, more than 35 percent of the top 5,000 global companies regularly fail to make insightful decisions about significant changes in their business and markets.
- By 2012, business units will control at least 40 percent of the total budget for business intelligence.
- By 2012, one-third of analytic applications applied to business processes will be delivered through coarse-grained application mashups.

A 2009 *Information Management* special report predicted the top BI trends: "green computing, social networking, data visualization, mobile BI, predictive analytics, composite applications, cloud computing and multitouch."

Other business intelligence trends include the following:

- Third party SOA-BI products increasingly address ETL issues of volume and throughput.
- Companies embrace in-memory processing, 64-bit processing, and pre-packaged analytic BI applications.
- Operational applications have callable BI components, with improvements in response time, scaling, and concurrency.
- Near or real time BI analytics is a baseline expectation.
- Open source BI software replaces vendor offerings.

Other lines of research include the combined study of business intelligence and uncertain data.[15] In this context, the data used is not assumed to be precise, accurate and complete. Instead, data is considered uncertain and therefore this uncertainty is propagated to the results produced by BI.

According to a study by the Aberdeen Group, there has been increasing interest in Software-as-a-Service (SaaS) business intelligence over the past years, with twice as many organizations using this deployment approach as one year ago – 15% in 2009 compared to 7% in 2008.[*citation needed*]

An article by InfoWorld's Chris Kanaracus points out similar growth data from research firm IDC, which predicts the SaaS BI market will grow 22 percent each year through 2013 thanks to increased product sophistication, strained IT budgets, and other factors.[16]

References

[1] ()

[2] Pieter M. Kroonenberg, Applied Multiway Data Analysis, Wiley 2008, pp. xv.

[3] Kimball et al., 2008: 29

[4] Jeanne W. Ross, Peter Weill, David C. Robertson (2006) *Enterprise Architecture As Strategy*, p. 117 ISBN 1-59139-839-8.

[5] Kimball et al. 2008: p. 298

[6] Kimball et al., 2008: 16

[7] Kimball et al., 2008: 18

[8] Kimball et al., 2008: 17

[9] Kimball

[10] Swain Scheps *Business Intelligence for Dummies*, 2008, ISBN 978-0-470-12723-0

[11] *The Data Warehouse Lifecycle Toolkit (2nd ed.). Ralph Kimball (2008).*

[12] *Microsoft Data Warehouse Toolkit. Wiley Publishing. (2006)*

[13] Inmon, B. & A. Nesavich, "Unstructured Textual Data in the Organization" from "Managing Unstructured data in the organization", Prentice Hall 2008, pp. 1–13

[14] Gartner Reveals Five Business Intelligence Predictions for 2009 and Beyond (http://www.gartner.com/it/page.jsp?id=856714). gartner.com. 15 January 2009

[15] | conference = ICIQ'09 | year = 2009

[16] SaaS BI growth will soar in 2010 | Cloud Computing (http://infoworld.com/d/cloud-computing/saas-bi-growth-will-soar-in-2010-511). InfoWorld (2010-02-01). Retrieved on 17 January 2012.

Bibliography

- Ralph Kimball *et al.* "The Data warehouse Lifecycle Toolkit" (2nd ed.) Wiley ISBN 0-470-47957-4
- Peter Rausch, Alaa Sheta, Aladdin Ayesh : *Business Intelligence and Performance Management: Theory, Systems, and Industrial Applications*, Springer Verlag U.K., 2013, ISBN 978-1-4471-4865-4.

External links

- Chaudhuri, Surajit; Dayal, Umeshwar; Narasayya, Vivek (August 2011). "An Overview Of Business Intelligence Technology" (http://cacm.acm.org/magazines/2011/8/114953-an-overview-of-business-intelligence-technology/fulltext). *Communications of the ACM* **54** (8): 88–98. doi: 10.1145/1978542.1978562 (http://dx.doi.org/10.1145/1978542.1978562). Retrieved 26 October 2011.

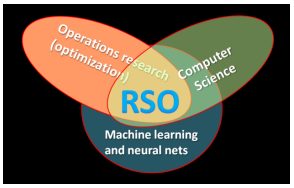
Reactive business intelligence

Reactive business intelligence (RBI) advocates an holistic approach that integrates data mining, modeling and interactive visualization, into an end-to-end discovery and continuous innovation process powered by human and automated learning.[1]

In the area of decision-making this approach has been used to adapt the decision method to the knowledge which is progressively acquired from the decision maker.

Relationships with reactive search optimization (RSO)

RSO is a multi-disciplinary research area between operations research (optimization), computer science, machine learning and neural networks that studies online learning schemes applied to problem-solving and optimization, according to a *learning while optimizing* principle . The word **reactive** hints at a ready response to events during the search through an internal feedback loop for online self-tuning and dynamic adaptation. Online adaptation and model revision is used in **reactive business intelligence** to adapt the data mining and interactive visualization techniques to the knowledge derived from a user in real time.



RSO is multi-disciplinary

References

[1] Roberto Battiti and Mauro Brunato, *Reactive Business Intelligence. From Data to Models to Insight* (http://www.reactivebusinessintelligence.com/), Reactive Search Srl, Italy, February 2011. ISBN 978-88-905795-0-9.

Business analytics

Business analytics (**BA**) refers to the skills, technologies, applications and practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning. Business analytics focuses on developing new insights and understanding of business performance based on data and statistical methods. In contrast, business intelligence traditionally focuses on using a consistent set of metrics to both measure past performance and guide business planning, which is also based on data and statistical methods.

Business analytics makes extensive use of data, statistical and quantitative analysis, explanatory and predictive modeling, and fact-based management to drive decision making. It is therefore closely related to management science. Analytics may be used as input for human decisions or may drive fully automated decisions. Business intelligence is querying, reporting, OLAP, and "alerts."

In other words, querying, reporting, OLAP, and alert tools can answer questions such as what happened, how many, how often, where the problem is, and what actions are needed. Business analytics can answer questions like why is this happening, what if these trends continue, what will happen next (that is, predict), what is the best that can happen (that is, optimize).

Examples of application

Banks, such as Capital One, use data analysis (or *analytics*, as it is also called in the business setting), to differentiate among customers based on credit risk, usage and other characteristics and then to match customer characteristics with appropriate product offerings. Harrah's, the gaming firm, uses analytics in its customer loyalty programs. E & J Gallo Winery quantitatively analyzes and predicts the appeal of its wines. Between 2002 and 2005, Deere & Company saved more than $1 billion by employing a new analytical tool to better optimize inventory.

Types of analytics

- Descriptive Analytics: Gain insight from historical data with reporting, scorecards, clustering etc.
- Predictive analytics (predictive modeling using statistical and machine learning techniques)
- Prescriptive analytics recommend decisions using optimization, simulation etc.
- Decisive analytics: supports human decisions with visual analytics the user models to reflect reasoning.

Basic domains within analytics

- Retail sales analytics
- Financial services analytics
- Risk & Credit analytics
- Talent analytics
- Marketing analytics
- Behavioral analytics
- Cohort Analysis
- Collections analytics
- Fraud analytics
- Pricing analytics
- Telecommunications
- Supply Chain analytics
- Transportation analytics
- Contextual data modeling - supports the human reasoning that occurs after viewing "executive dashboards" or any other visual analytics

History

Analytics have been used in business since the time management exercises that were initiated by Frederick Winslow Taylor in the late 19th century. Henry Ford measured pacing of assembly line. But analytics began to command more attention in the late 1960s when computers were used in decision support systems. Since then, analytics have evolved with the development of enterprise resource planning (ERP) systems, data warehouses, and a wide variety of other hardware and software tools and applications.

With the recent explosion of big data and intuitive BI tools, data is more accessible to business professionals and managers than ever before. Thus there is a big opportunity to make better decisions using that data to drive incremental revenue, decrease cost and loss by building better products, improving customer experience, catching fraud before it happens, improving customer engagement through targeting and customization- all with the power of data. More and more companies are now equipping their employees with the know-how of Business Analytics to drive efficiency in day-to-day decision making.

Challenges

Business analytics depends on sufficient volumes of high quality data. The difficulty in ensuring data quality is integrating and reconciling data across different systems, and then deciding what subsets of data to make available.

Previously, analytics was considered a type of after-the-fact method of forecasting consumer behavior by examining the number of units sold in the last quarter or the last year. This type of data warehousing required a lot more storage space than it did speed. Now business analytics is becoming a tool that can influence the outcome of customer interactions. When a specific customer type is considering a purchase, an analytics-enabled enterprise can modify the sales pitch to appeal to that consumer. This means the storage space for all that data must react extremely fast to provide the necessary data in real-time.

Competing on analytics

Davenport argues that businesses can optimize a distinct business capability via analytics and thus better compete. He identifies these characteristics of an organization that are apt to compete on analytics:

- One or more senior executives who strongly advocate fact-based decision making and, specifically, analytics
- Widespread use of not only descriptive statistics, but also predictive modeling and complex optimization techniques
- Substantial use of analytics across multiple business functions or processes
- Movement toward an enterprise level approach to managing analytical tools, data, and organizational skills and capabilities

References

Further reading

- Bartlett, Randy (February 2013). *A Practitioner's Guide To Business Analytics: Using Data Analysis Tools to Improve Your Organization's Decision Making and Strategy*. McGraw-Hill. ISBN 978-0071807593.
- Davenport, Thomas H.; Jeanne G. Harris (March 2007). *Competing on Analytics: The New Science of Winning*. Harvard Business School Press.
- McDonald, Mark; Tina Nunno (February 2007). *Creating Enterprise Leverage: The 2007 CIO Agenda*. Stamford, CT: Gartner, Inc.
- Stubbs, Evan (July 2011). *The Value of Business Analytics*. John Wiley & Sons.
- Ranadive, Vivek (2006-01-26). *The Power to Predict: How Real Time Businesses Anticipate Customer Needs, Create Opportunities, and Beat the Competition*. McGraw-Hill.
- Zabin, Jeffrey; Gresh Brebach (February 2004). *Precision Marketing*. John Wiley.
- Baker, Stephen (January 23, 2006). "Math Will Rock Your World" (http://www.businessweek.com/print/magazine/content/06_04/b3968001.htm?chan=gl). *BusinessWeek*. Retrieved 2007-09-19.
- Davenport, Thomas H. (January 1, 2006). "Competing on Analytics". *Harvard Business Review*.
- Pfeffer, Jeffrey; Robert I. Sutton (January 2006). "Evidence-Based Management". *Harvard Business Review*.
- Davenport, Thomas H.; Jeanne G. Harris (Summer 2005). "Automated Decision Making Comes of Age". *MIT Sloan Management Review*.
- Lewis, Michael (April 2004). *Moneyball: The Art of Winning an Unfair Game*. W.W. Norton & Co.
- Bonabeau, Eric (May 2003). "Don't Trust Your Gut". *Harvard Business Review*.
- Davenport, Thomas H.; Jeanne G. Harris, David W. De Long, Alvin L. Jacobson. "Data to Knowledge to Results: Building an Analytic Capability". *California Management Review* **43** (2): 117–138.

Sales intelligence

Sales intelligence (SI) refers to technologies, applications and practices for the collection, integration, analysis, and presentation of information to help salespeople keep up to date with clients, prospect data and drive business. In addition to providing metrics for win-loss and sales confidence,[1] SI can present contextually relevant customer and product information.

The 2008 survey of 300 companies by the Aberdeen Group [2] show that the recent economic downturn has lengthened traditional sales cycles. As businesses have been forced to reduce spending, sales representatives have been challenged to meet quotas. Top performing companies have implemented sales intelligence programs to improve the quality and quantity of sales leads. SI contextualizes opportunities by providing relevant industry, corporate and personal information. Frequently SI's fact-based information is integrated or includes Customer Relationship Management (CRM).

Although some aspects of sales intelligence overlaps business intelligence (BI), SI is specifically designed for the use of salespeople and sales managers. Unlike customer relationship management (CRM) and traditional business intelligence (BI) applications, SI provides real-time analysis of current sales data and assists with suggesting and delivering actionable, relevant information.

Sales intelligence solutions are predominantly designed for companies in the manufacturing, distribution and wholesale sectors. These are highly competitive markets, where volumes are high, margins are low. (SI) solutions provide unique insight into customer buying patterns. By automatically analysing and evaluating these patterns, Sales Intelligence pro-actively identifies and delivers up-sell, cross-sell and switch-sell opportunities.

References

[1] Metrics for sales intelligence (http://chapmanhq.com/solutions/strategic-account-management-sam/metrics-and-measurements)

[2] Sales Intelligence, Aberdeen Group study (2008) (http://www.aberdeen.com/Aberdeen-Library/5379/RA-sales-intelligence-nirvana.aspx)

External links

- Sales Intelligence A Short Primer (http://blog.findable.me/post/52963306183/sales-intelligence-a-short-primer)

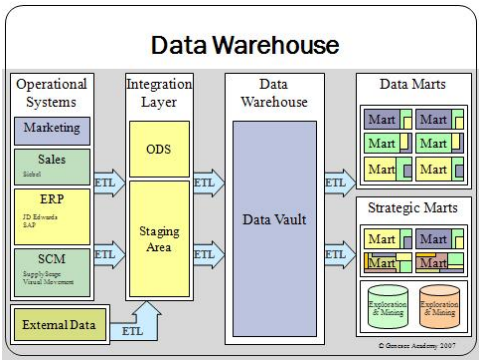
Performance intelligence

Performance Intelligence refers to technologies that utilize standard Business Intelligence techniques on performance-related IT metrics. Traditionally, the performance metrics are loaded into a set of tables that resemble the table structure of a data warehouse. Analysis is performed with standard Business Intelligence techniques such as slicing, dicing, drilling down and the trend analysis.

This type of analysis is useful, because it allows the user to quickly look at the data from several perspectives to see trends or anomalies that they might miss while looking at data in a more granular form.

Data warehouse

In computing, a **data warehouse** (**DW**, **DWH**), or an **enterprise data warehouse** (**EDW**), is a database used for reporting (1) and data analysis (2). Integrating data from one or more disparate sources creates a central repository of data, a data warehouse (DW). Data warehouses store current and historical data and are used for creating trending reports for senior management reporting such as annual and quarterly comparisons.



Data Warehouse Overview

The data stored in the warehouse is uploaded from the operational systems (such as marketing, sales, etc., shown in the figure to the right). The data may pass through an operational data store for additional operations before it is used in the DW for reporting.

The typical extract transform load (ETL)-based data warehouse uses staging, data integration, and access layers to house its key functions. The staging layer or staging database stores raw data extracted from each of the disparate source data systems. The integration layer integrates the disparate data sets by transforming the data from the staging layer often storing this transformed data in an operational data store (ODS) database. The integrated data are then moved to yet another database, often called the data warehouse database, where the data is arranged into hierarchical groups often called dimensions and into facts and aggregate facts. The combination of facts and dimensions is sometimes called a star schema. The access layer helps users retrieve data.

A data warehouse constructed from an integrated data source systems does not require ETL, staging databases, or operational data store databases. The integrated data source systems may be considered to be a part of a distributed operational data store layer. Data federation methods or data virtualization methods may be used to access the distributed integrated source data systems to consolidate and aggregate data directly into the data warehouse database tables. Unlike the ETL-based data warehouse, the integrated source data systems and the data warehouse are all integrated since there is no transformation of dimensional or reference data. This integrated data warehouse architecture supports the drill down from the aggregate data of the data warehouse to the transactional data of the integrated source data systems.

A data mart is a small data warehouse focused on a specific area of interest. Data warehouses can be subdivided into data marts for improved performance and ease of use within that area. Alternatively, an organization can create one or more data marts as first steps towards a larger and more complex enterprise data warehouse.

This definition of the data warehouse focuses on data storage. The main source of the data is cleaned, transformed, cataloged and made available for use by managers and other business professionals for data mining, online analytical processing, market research and decision support (Marakas & O'Brien 2009). However, the means to retrieve and analyze data, to extract, transform and load data, and to manage the data dictionary are also considered essential

components of a data warehousing system. Many references to data warehousing use this broader context. Thus, an expanded definition for data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.

Benefits of a data warehouse

A data warehouse maintains a copy of information from the source transaction systems. This architectural complexity provides the opportunity to :

- Congregate data from multiple sources into a single database so a single query engine can be used to present data.
- Mitigate the problem of database isolation level lock contention in transaction processing systems caused by attempts to run large, long running, analysis queries in transaction processing databases.
- Maintain data history, even if the source transaction systems do not.
- Integrate data from multiple source systems, enabling a central view across the enterprise. This benefit is always valuable, but particularly so when the organization has grown by merger.
- Improve data quality, by providing consistent codes and descriptions, flagging or even fixing bad data.
- Present the organization's information consistently.
- Provide a single common data model for all data of interest regardless of the data's source.
- Restructure the data so that it makes sense to the business users.
- Restructure the data so that it delivers excellent query performance, even for complex analytic queries, without impacting the operational systems.
- Add value to operational business applications, notably customer relationship management (CRM) systems.
- Making decision–support queries easier to write.

Generic data warehouse environment

The environment for data warehouses and marts includes the following:

- Source systems that provide data to the warehouse or mart;
- Data integration technology and processes that are needed to prepare the data for use;
- Different architectures for storing data in an organization's data warehouse or data marts;
- Different tools and applications for the variety of users;
- Metadata, data quality, and governance processes must be in place to ensure that the warehouse or mart meets its purposes.

In regards to source systems listed above, Rainer states, “A common source for the data in data warehouses is the company’s operational databases, which can be relational databases”.

Regarding data integration, Rainer states, “It is necessary to extract data from source systems, transform them, and load them into a data mart or warehouse”.

Rainer discusses storing data in an organization’s data warehouse or data marts. “There are a variety of possible architectures to store decision-support data”.

Metadata are data about data. “IT personnel need information about data sources; database, table, and column names; refresh schedules; and data usage measures“.

Today, the most successful companies are those that can respond quickly and flexibly to market changes and opportunities. A key to this response is the effective and efficient use of data and information by analysts and managers. A “data warehouse” is a repository of historical data that are organized by subject to support decision makers in the organization. Once data are stored in a data mart or warehouse, they can be accessed.

History

The concept of data warehousing dates back to the late 1980s when IBM researchers Barry Devlin and Paul Murphy developed the "business data warehouse". In essence, the data warehousing concept was intended to provide an architectural model for the flow of data from operational systems to decision support environments. The concept attempted to address the various problems associated with this flow, mainly the high costs associated with it. In the absence of a data warehousing architecture, an enormous amount of redundancy was required to support multiple decision support environments. In larger corporations it was typical for multiple decision support environments to operate independently. Though each environment served different users, they often required much of the same stored data. The process of gathering, cleaning and integrating data from various sources, usually from long-term existing operational systems (usually referred to as legacy systems), was typically in part replicated for each environment. Moreover, the operational systems were frequently reexamined as new decision support requirements emerged. Often new requirements necessitated gathering, cleaning and integrating new data from "data marts" that were tailored for ready access by users.

Key developments in early years of data warehousing were:

- 1960s — General Mills and Dartmouth College, in a joint research project, develop the terms *dimensions* and *facts*.[1]
- 1970s — ACNielsen and IRI provide dimensional data marts for retail sales.
- 1970s — Bill Inmon begins to define and discuss the term: Data Warehouse
- 1975 — Sperry Univac Introduce MAPPER (MAintain, Prepare, and Produce Executive Reports) is a database management and reporting system that includes the world's first 4GL. It was the first platform designed for building Information Centers (a forerunner of contemporary Enterprise Data Warehousing platforms)
- 1983 — Teradata introduces a database management system specifically designed for decision support.
- 1983 — Sperry Corporation Martyn Richard Jones defines the Sperry Information Center approach, which while not being a true DW in the Inmon sense, did contain many of the characteristics of DW structures and process as defined previously by Inmon, and later by Devlin. First used at the TSB England & Wales
- 1984 — Metaphor Computer Systems, founded by David Liddle and Don Massaro, releases Data Interpretation System (DIS). DIS was a hardware/software package and GUI for business users to create a database management and analytic system.
- 1988 — Barry Devlin and Paul Murphy publish the article *An architecture for a business and information system* [2] in *IBM Systems Journal* where they introduce the term "business data warehouse".
- 1990 — Red Brick Systems, founded by Ralph Kimball, introduces Red Brick Warehouse, a database management system specifically for data warehousing.
- 1991 — Prism Solutions, founded by Bill Inmon, introduces Prism Warehouse Manager, software for developing a data warehouse.
- 1992 — Bill Inmon publishes the book *Building the Data Warehouse*.
- 1995 — The Data Warehousing Institute, a for-profit organization that promotes data warehousing, is founded.
- 1996 — Ralph Kimball publishes the book *The Data Warehouse Toolkit*.
- 2000 — Daniel Linstedt releases the *Data Vault*, enabling real time auditable Data Warehouses warehouse.

Information storage

Facts

A fact is a value or measurement, which represents a fact about the managed entity or system.

Facts as reported by the reporting entity are said to be at raw level.

E.g. if a BTS received 1,000 requests for traffic channel allocation, it allocates for 820 and rejects the remaining then it would report 3 **facts** or measurements to a management system:

- tch_req_total = 1000
- tch_req_success = 820
- tch_req_fail = 180

Facts at raw level are further aggregated to higher levels in various dimensions to extract more service or business-relevant information out of it. These are called aggregates or summaries or aggregated facts.

E.g. if there are 3 BTSs in a city, then facts above can be aggregated from BTS to city level in network dimension. E.g.

- $tch\_req\_success\_city = tch\_req\_success\_bts1 + tch\_req\_success\_bts2 + tch\_req\_success\_bts3$
- $avg\_tch\_req\_success\_city = (tch\_req\_success\_bts1 + tch\_req\_success\_bts2 + tch\_req\_success\_bts3)/3$

Dimensional vs. normalized approach for storage of data

There are three or more leading approaches to storing data in a data warehouse — the most important approaches are the dimensional approach and the normalized approach.

The dimensional approach, whose supporters are referred to as “Kimballites”, believe in Ralph Kimball's approach in which it is stated that the data warehouse should be modeled using a Dimensional Model/star schema. The normalized approach, also called the 3NF model (Third Normal Form), whose supporters are referred to as “Inmonites”, believe in Bill Inmon's approach in which it is stated that the data warehouse should be modeled using an E-R model/normalized model.

In a dimensional approach, transaction data are partitioned into "facts", which are generally numeric transaction data, and "dimensions", which are the reference information that gives context to the facts. For example, a sales transaction can be broken up into facts such as the number of products ordered and the price paid for the products, and into dimensions such as order date, customer name, product number, order ship-to and bill-to locations, and salesperson responsible for receiving the order.

A key advantage of a dimensional approach is that the data warehouse is easier for the user to understand and to use. Also, the retrieval of data from the data warehouse tends to operate very quickly.[*citation needed*] Dimensional structures are easy to understand for business users, because the structure is divided into measurements/facts and context/dimensions. Facts are related to the organization’s business processes and operational system whereas the dimensions surrounding them contain context about the measurement (Kimball, Ralph 2008).

The main disadvantages of the dimensional approach are the following:

1. In order to maintain the integrity of facts and dimensions, loading the data warehouse with data from different operational systems is complicated.
2. It is difficult to modify the data warehouse structure if the organization adopting the dimensional approach changes the way in which it does business.

In the normalized approach, the data in the data warehouse are stored following, to a degree, database normalization rules. Tables are grouped together by *subject areas* that reflect general data categories (e.g., data on customers, products, finance, etc.). The normalized structure divides data into entities, which creates several tables in a relational database. When applied in large enterprises the result is dozens of tables that are linked together by a web

of joins. Furthermore, each of the created entities is converted into separate physical tables when the database is implemented (Kimball, Ralph 2008)[*citation needed*]. The main advantage of this approach is that it is straightforward to add information into the database. Some disadvantages of this approach are that, because of the number of tables involved, it can be difficult for users to join data from different sources into meaningful information and to access the information without a precise understanding of the sources of data and of the data structure of the data warehouse.

It should be noted that both normalized and dimensional models can be represented in entity-relationship diagrams as both contain joined relational tables. The difference between the two models is the degree of normalization (also known as Normal Forms).

These approaches are not mutually exclusive, and there are other approaches. Dimensional approaches can involve normalizing data to a degree (Kimball, Ralph 2008).

In Information-Driven Business, Robert Hillard proposes an approach to comparing the two approaches based on the information needs of the business problem. The technique shows that normalized models hold far more information than their dimensional equivalents (even when the same fields are used in both models) but this extra information comes at the cost of usability. The technique measures information quantity in terms of Information Entropy and usability in terms of the Small Worlds data transformation measure.

Top-down versus bottom-up design methodologies

Bottom-up design

Ralph Kimball,[3] designed an approach to data warehouse design known as *bottom-up*.

In the *bottom-up* approach, data marts are first created to provide reporting and analytical capabilities for specific business processes.

Data marts contain, primarily, dimensions and facts. Facts can contain either atomic data and, if necessary, summarized data. The single data mart often models a specific business area such as "Sales" or "Production." These data marts can eventually be integrated to create a comprehensive data warehouse. The data warehouse bus architecture is primarily an implementation of "the bus", a collection of conformed dimensions and conformed facts, which are dimensions that are shared (in a specific way) between facts in two or more data marts.

The integration of the data marts in the data warehouse is centered on the conformed dimensions (residing in "the bus") that define the possible integration "points" between data marts. The actual integration of two or more data marts is then done by a process known as "Drill across". A drill-across works by grouping (summarizing) the data along the keys of the (shared) conformed dimensions of each fact participating in the "drill across" followed by a join on the keys of these grouped (summarized) facts.

Maintaining tight management over the data warehouse bus architecture is fundamental to maintaining the integrity of the data warehouse. The most important management task is making sure dimensions among data marts are consistent.

Business value can be returned as quickly as the first data marts can be created, and the method lends itself well to an exploratory and iterative approach to building data warehouses. For example, the data warehousing effort might start in the "Sales" department, by building a Sales-data mart. Upon completion of the Sales-data mart, the business might then decide to expand the warehousing activities into the, say, "Production department" resulting in a Production data mart. The requirement for the Sales data mart and the Production data mart to be integrable, is that they share the same "Bus", that will be, that the data warehousing team has made the effort to identify and implement the conformed dimensions in the bus, and that the individual data marts links that information from the bus. The Sales-data mart is good as it is (assuming that the bus is complete) and the Production-data mart can be constructed virtually independent of the Sales-data mart (but not independent of the Bus).

If integration via the bus is achieved, the data warehouse, through its two data marts, will not only be able to deliver the specific information that the individual data marts are designed to do, in this example either "Sales" or "Production" information, but can deliver integrated Sales-Production information, which, often, is of critical business value.

Top-down design

Bill Inmon, has defined a data warehouse as a centralized repository for the entire enterprise.[4] The *top-down* approach is designed using a normalized enterprise data model. "Atomic" data, that is, data at the lowest level of detail, are stored in the data warehouse. Dimensional data marts containing data needed for specific business processes or specific departments are created from the data warehouse. In the Inmon vision, the data warehouse is at the center of the "Corporate Information Factory" (CIF), which provides a logical framework for delivering business intelligence (BI) and business management capabilities. Gartner released a research note confirming Inmon's definition in 2005[5] with additional clarity plus they added one additional attribute.

The data warehouse is:

Subject-oriented

: The data in the data warehouse is organized so that all the data elements relating to the same real-world event or object are linked together.

Non-volatile

: Data in the data warehouse are never over-written or deleted — once committed, the data are static, read-only, and retained for future reporting.

Integrated

: The data warehouse contains data from most or all of an organization's operational systems and these data are made consistent.

Time-variant

: For an **operational system**, the stored data contains the current value. The data warehouse, however, contains the history of data values.

No virtualization

: A data warehouse is a physical repository.

The top-down design methodology generates highly consistent dimensional views of data across data marts since all data marts are loaded from the centralized repository. Top-down design has also proven to be robust against business changes. Generating new dimensional data marts against the data stored in the data warehouse is a relatively simple task. The main disadvantage to the top-down methodology is that it represents a very large project with a very broad scope. The up-front cost for implementing a data warehouse using the top-down methodology is significant, and the duration of time from the start of project to the point that end users experience initial benefits can be substantial. In addition, the top-down methodology can be inflexible and unresponsive to changing departmental needs during the implementation phases.

Hybrid design

Data warehouse (DW) solutions often resemble the hub and spokes architecture. Legacy systems feeding the DW/BI solution often include customer relationship management (CRM) and enterprise resource planning solutions (ERP), generating large amounts of data. To consolidate these various data models, and facilitate the extract transform load (ETL) process, DW solutions often make use of an operational data store (ODS). The information from the ODS is then parsed into the actual DW. To reduce data redundancy, larger systems will often store the data in a normalized way. Data marts for specific reports can then be built on top of the DW solution.

It is important to note that the DW database in a hybrid solution is kept on third normal form to eliminate data redundancy. A normal relational database however, is not efficient for business intelligence reports where dimensional modelling is prevalent. Small data marts can shop for data from the consolidated warehouse and use the filtered, specific data for the fact tables and dimensions required. The DW effectively provides a single source of information from which the data marts can read, creating a highly flexible solution from a BI point of view. The hybrid architecture allows a DW to be replaced with a master data management solution where operational, not static information could reside.

The Data Vault Modeling components follow hub and spokes architecture. This modeling style is a hybrid design, consisting of the best practices from both 3rd normal form and star schema. The Data Vault model is not a true 3rd normal form, and breaks some of the rules that 3NF dictates be followed. It is however, a top-down architecture with a bottom up design. The Data Vault model is geared to be strictly a data warehouse. It is not geared to be end-user accessible, which when built, still requires the use of a data mart or star schema based release area for business purposes.

Data warehouses versus operational systems

Operational systems are optimized for preservation of data integrity and speed of recording of business transactions through use of database normalization and an entity-relationship model. Operational system designers generally follow the Codd rules of database normalization in order to ensure data integrity. Codd defined five increasingly stringent rules of normalization. Fully normalized database designs (that is, those satisfying all five Codd rules) often result in information from a business transaction being stored in dozens to hundreds of tables. Relational databases are efficient at managing the relationships between these tables. The databases have very fast insert/update performance because only a small amount of data in those tables is affected each time a transaction is processed. Finally, in order to improve performance, older data are usually periodically purged from operational systems.

Evolution in organization use

These terms refer to the level of sophistication of a data warehouse:

Offline operational data warehouse

> Data warehouses in this stage of evolution are updated on a regular time cycle (usually daily, weekly or monthly) from the operational systems and the data is stored in an integrated reporting-oriented data

Offline data warehouse

> Data warehouses at this stage are updated from data in the operational systems on a regular basis and the data warehouse data are stored in a data structure designed to facilitate reporting.

On time data warehouse

> Online Integrated Data Warehousing represent the real time Data warehouses stage data in the warehouse is updated for every transaction performed on the source data

Integrated data warehouse

> These data warehouses assemble data from different areas of business, so users can look up the information they need across other systems.

References

[1] Kimball 2002, pg. 16
[2] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5387658
[3] Kimball 2002, pg. 310
[4] Ericsson 2004, pp. 28–29
[5] Gartner, Of Data Warehouses, Operational Data Stores, Data Marts and Data Outhouses, Dec 2005

Further reading

- Davenport, Thomas H. and Harris, Jeanne G. *Competing on Analytics: The New Science of Winning* (2007) Harvard Business School Press. ISBN 978-1-4221-0332-6
- Ganczarski, Joe. *Data Warehouse Implementations: Critical Implementation Factors Study* (2009) VDM Verlag ISBN 3-639-18589-7 ISBN 978-3-639-18589-8
- Kimball, Ralph and Ross, Margy. *The Data Warehouse Toolkit* Second Edition (2002) John Wiley and Sons, Inc. ISBN 0-471-20024-7
- Linstedt, Graziano, Hultgren. *The Business of Data Vault Modeling* Second Edition (2010) Dan linstedt, ISBN 978-1-4357-1914-9
- William Inmon. *Building the Data Warehouse* 2005) John Wiley and Sons, ISBN 978-8-1265-0645-3

External links

- Ralph Kimball articles (http://www.kimballgroup.com/html/articles.html)
- International Journal of Computer Applications (http://www.ijcaonline.org/archives/number3/77-172)
- Data Warehouse Introduction (http://dwreview.com/DW_Overview.html)
- Time to Reconsider the Data Warehouse (Global Association of Risk Professionals) (http://www.garp.org/risk-news-and-resources/2013/june/time-to-reconsider-the-data-warehouse.aspx?)

Data warehouse architectures

–

The technical architecture of data warehouses is somewhat similar to other systems, but does have some special characteristics. There are two border areas in data warehouse architecture - the single-layer architecture and the N-layer architecture. The difference here is the number of middleware between the operational systems and the analytical tools. The data warehouse architecture described here is a high level architecture and the parts in the architectures mentioned are full bodied systems and not system-parts.

Single-layer architecture

A simple architecture is the single-layer architecture. There is no physical data warehouse or data mart between the operation data and the analytic tools. The middleware in this type of system should be considered a virtual data warehouse, which consists of a software layer and not a data based layer. The single-layer model is light weight as it minimises redundancies and thereby the amount of data stored. It has, however, no separation between analytical and operational processing. The analysis are based directly on the operational data.[1]

Three-layer architecture

The three-layer architecture consists of the source layer (containing multiple source systems), the reconciled layer and the data warehouse layer (containing both data warehouses and data marts). The reconciled layer sits between the source data and data warehouse. It is populated with data from the source systems through an ETL process and the data stored in it is published further through another ETL process. In the reconciled layer the data has been cleaned up once and integrated to a common standardised form from multiple different source systems. The ETL process that feeds the data warehouse then only gets already integrated data that has less need for transformation. This architecture is especially useful for the very large, enterprise-wide systems. A disadvantage of this architecture is the extra data storage space used through the extra redundant reconciled layer. It also makes the analytical tools a little further away from being real-time.

References

[1] Golfarelli, Matteo; Rizzi Stefano (2009). "Data Warehouse Design : Modern Principles and Methodologies", New York: McGraw-Hill.

Data mart

A **data mart** is the access layer of the data warehouse environment that is used to get data out to the users. The data mart is a subset of the data warehouse that is usually oriented to a specific business line or team. Data marts are small slices of the data warehouse. Whereas data warehouses have an enterprise-wide depth, the information in data marts pertains to a single department. In some deployments, each department or business unit is considered the *owner* of its data mart including all the *hardware*, *software* and *data*.[1] This enables each department to use, manipulate and develop their data any way they see fit; without altering information inside other data marts or the data warehouse. In other deployments where conformed dimensions are used, this business unit ownership will not hold true for shared dimensions like customer, product, etc.

The related term spreadmart is a derogatory label describing the situation that occurs when one or more business analysts develop a system of linked spreadsheets to perform a business analysis, then grow it to a size and degree of complexity that makes it nearly impossible to maintain.

The primary use for a data mart is business intelligence (BI) applications. BI is used to gather, store, access and analyze data. The data mart can be used by smaller businesses to utilize the data they have accumulated. A data mart can be less expensive than implementing a data warehouse, thus making it more practical for the small business. A data mart can also be set up in much less time than a data warehouse, being able to be set up in less than 90 days. Since most small businesses only have use for a small number of BI applications, the low cost and quick set up of the data mart makes it a suitable method for storing data.

Design schemas

- Star schema - fairly popular design choice; enables a relational database to emulate the analytical functionality of a multidimensional database
- Snowflake schema

Reasons for creating a data mart

- Easy access to frequently needed data
- Creates collective view by a group of users
- Improves end-user response time
- Ease of creation
- Lower cost than implementing a full data warehouse
- Potential users are more clearly defined than in a full data warehouse
- Contains only business essential data and is less cluttered.

Dependent data mart

According to the Inmon school of data warehousing, a **dependent data mart** is a logical subset (view) or a physical subset (extract) of a larger data warehouse, isolated for one of the following reasons:

- A need refreshment for a special data model or schema: e.g., to restructure for OLAP
- Performance: to offload the data mart to a separate computer for greater efficiency or to obviate the need to manage that workload on the centralized data warehouse.
- Security: to separate an authorized data subset selectively
- Expediency: to bypass the data governance and authorizations required to incorporate a new application on the Enterprise Data Warehouse

- Proving Ground: to demonstrate the viability and ROI (return on investment) potential of an application prior to migrating it to the Enterprise Data Warehouse
- Politics: a coping strategy for IT (Information Technology) in situations where a user group has more influence than funding or is not a good citizen on the centralized data warehouse.
- Politics: a coping strategy for consumers of data in situations where a data warehouse team is unable to create a usable data warehouse.

According to the Inmon school of data warehousing, tradeoffs inherent with data marts include limited scalability, duplication of data, data inconsistency with other silos of information, and inability to leverage enterprise sources of data.

The alternative school of data warehousing is that of Ralph Kimball. In his view, a data warehouse is nothing more than the union of all the data marts. This view helps to reduce costs and provides fast development, but can create an inconsistent data warehouse, especially in large organizations. Therefore, Kimball's approach is more suitable for small-to-medium corporations.[2]

References

[1] Data Mart Does Not Equal Data Warehouse (http://www.information-management.com/infodirect/19991120/1675-1.html)

[2] Paulraj Ponniah. *Data Warehousing Fundamentals for IT Professionals.* Wiley, 2010, pp. 29–32. ISBN 0470462078.

Bibliography

- Inmon, William (2000-07-18). "Data Mart Does Not Equal Data Warehouse" (http://csis.bits-pilani.ac.in/faculty/goel/Data Warehousing/Articles/Data Marts/dataWarehouse_com Article_DM VS DW.htm). DMReview.com.

External links

The Kimball Lifecycle

The Kimball Lifecycle is a methodology for developing data warehouses, and has been developed over many years by Ralph Kimball and a variety of colleagues.

Program/Project planning

According to Kimball et al., this phase is the start of the Lifecycle. It is a planning-phase in which project is a single iteration of the Lifecycle while program is the broader coordination of resources. When launching a project/program Kimball et al. suggests following three focus areas:

- Define and Scope project
- Plan project
- Manage project

Program/Project Management

This is an ongoing discipline in the project. The purpose is to keep the project/program on course, develop a communication plan and manage expectations.

Business Requirements Definition

This phase/milestone of the project is about making the project team understand the business requirements. Its purpose is to establish a foundation for all the following activities in the Lifecycle. Kimball et al. makes it very clear that it is extremely important for the project team to talk with the business users. It is important to be prepared, to focus on listening and to document the interview with the business users.

Technology Track

This is the top track in the diagram. It holds two milestones:

Technical Architecture Design is supposed to create a framework for the DW/BI system. The main focus in this phase is to create a plan for the application architecture, while considering business requirements, technical environment and the planned strategic technical directions.

Product Selection & Installation use the architecture plan to identify what components are needed to complete the DW/BI project. This phase then selects, installs and tests the products.

Data Track

Dimensional Modeling is a process in which the business requirements are used to design dimensional models for the system.

Physical Design is the phase where the database is designed. It involves the database environment as well as security.

ETL Design & Development is the design of some of the heavy procedures in the DW/BI-system. Kimball et al. suggests four parts to this process, which are further divided into 34 subsystems (Kimball et al., 2008):

- Extracting data
- Cleaning and conforming data
- Delivering data for Presentation
- Managing the ETL system

Business Intelligence Application Track

BI Application Design deals with designing and selecting some applications to support the business requirements. BI Application Development use the design to develop and validate applications to support the business requirements.

Deployment

When the three tracks are complete they all end up in the final deployment. This phase requires planning and should include pre-deployment testing, documentation, training and maintenance and support.

Maintenance

When the deployment has finished the system will need proper maintenance to stay alive. This includes data reconciliation, execution and monitoring and performance tuning.

Growth

As the project can be seen as part of the larger iterative program, it is very likely that the system will want to expand. There will be projects to add new data as well as reaching new segments of the business areas. The Lifecycle then starts over again.

References

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2008). The data warehouse lifecycle toolkit (2nd ed.). Wiley Publishing, Inc.

Time variance

Time variance is the ability to remember historic perspectives. The requirement is to be able to know how something was classified or who owned something and how this changed as time passed.

For the context of time and frequency and qualification of oscillators and amplifiers the technical terms time deviation and time variance is defined.

Understanding time variance

Future change, be it organizational, regulatory or geographical is hard to conceive. In 1988, who imagined that within a few years, Yugoslavia and East Germany would cease to exist? Enabling a data warehouse to report pre- and post-change information together in a meaningful context is very expensive and time-consuming. Couple that with the pressure to rapidly meet other business requirements, and with the inability for any of us to predict change (especially at system design time!), and you can see why the time-variant reporting requirement is often ignored. But at what expense? The real-life case below illustrates the value of time-variant reporting: A beverage company paid rebates to customers at year-end, based on ownership of customer sites at year end. The sales data warehouse did not reflect customers selling sites to one another throughout the year, resulting in mis-payments and a multi-million dollar customer service dilemma. For regulatory compliance and other reasons, data warehouses must remember how things were in the past because at some point business people will expect to be able to be report on them that way.So it is one of the important characteristic of data warehouse

Federated database system

A **federated database system** is a type of meta-database management system (DBMS), which transparently maps multiple autonomous database systems into a single **federated database**. The constituent databases are interconnected via a computer network and may be geographically decentralized. Since the constituent database systems remain autonomous, a federated database system is a contrastable alternative to the (sometimes daunting) task of merging several disparate databases. A federated database, or **virtual database**, is a composite of all constituent databases in a federated database system. There is no actual data integration in the constituent disparate databases as a result of data federation.

McLeod and Heimbigner were among the first to define a federated database system, as one which "define[s] the architecture and interconnect[s] databases that minimize central authority yet support partial sharing and coordination among database systems".

Through data abstraction, federated database systems can provide a uniform user interface, enabling users and clients to store and retrieve data in multiple noncontiguous databases with a single query -- even if the constituent databases are heterogeneous. To this end, a federated database system must be able to decompose the query into subqueries for submission to the relevant constituent DBMS's, after which the system must composite the result sets of the subqueries. Because various database management systems employ different query languages, federated database systems can apply wrappers to the subqueries to translate them into the appropriate query languages.

- Note: this description of federated databases does not accurately reflect the McLeod/Heimbigner definition of a federated database. Rather, this description fits what McLeod/Heimbinger called a *composite* database. McLeod/Heimbigner's federated database is a collection of autonomous components that make their data available to other members of the federation through the publication of an export schema and access operations; there is no unified, central schema that encompasses the information available from the members of the federation.

Among other surveys, defines a Federated Database as a collection of cooperating component systems which are autonomous and are possibly heterogeneous. The three important components of an FDBS as pointed out in are autonomy, heterogeneity and distribution. Another dimension which has also been considered is the Networking Environment Computer Network, e.g., many DBSs over a LAN or many DBSs over a WAN update related functions of participating DBSs (e.g., no updates, nonatomic transitions, atomic updates).

FDBS architecture

A DBMS can be classified as either centralized or distributed. A centralized system manages a single database while distributed manages multiple databases. A component DBS in a DBMS may be centralized or distributed. A multiple DBS (MDBS) can be classified into two types depending on the autonomy of the component DBS as federated and non federated. A nonfederated database system is an integration of component DBMS that are not autonomous. A federated database system consists of component DBS that are autonomous yet participate in a federation to allow partial and controlled sharing of their data.

Federated architectures differ based on levels of integration with the component database systems and the extent of services offered by the federation. A FDBS can be categorized as loosely or tightly coupled systems.

- Loosely Coupled require component databases to construct their own federated schema. A user will typically access other component database systems by using a multidatabase language but this removes any levels of location transparency, forcing the user to have direct knowledge of the federated schema. A user imports the data they require from other component databases and integrates it with their own to form a federated schema.
- Tightly coupled system consists of component systems that use independent processes to construct and publicize an integrated federated schema.

Multiple DBS of which FDBS are a specific type can be characterized along three dimensions: Distribution, Heterogeneity and Autonomy. Another characterization could be based on the dimension of networking, for example single databases or multiple databases in a LAN or WAN.

Distribution

Distribution of data in an FDBS is due to the existence of a multiple DBS before an FDBS is built. Data can be distributed among multiple DB which could be stored in a single computer or multiple computers. These computers could be geographically located in different places but interconnected by a network. The benefits of data distribution help in increased availability and reliability as well as improved access times.

Heterogeneity

Heterogeneities in databases arise due to factors such as differences in structures, semantics of data, the constraints supported or query language. Differences in structure occur when two data models provide different primitives such as object oriented (OO) models that support specialization and inheritance and relational models that do not. Differences due to constraints occur when two models support two different constraints. For example the set type in CODASYL schema may be partially modeled as a referential integrity constraint in a relationship schema. CODASYL supports insertion and retention that are not captured by referential integrity alone. The query language supported by one DBMS can also contribute to heterogeneity between other component DBMSs. For example, differences in query languages with the same data models or different versions of query languages could contribute to heterogeneity.

Semantic heterogeneities arise when there is a disagreement about meaning, interpretation or intended use of data. At the schema and data level, classification of possible heterogeneities include:

- Naming conflicts e.g. databases using different names to represent the same concept.
- Domain conflicts or data representation conflicts e.g. databases using different values to represent same concept.
- Precision conflicts e.g. databases using same data values from domains of different cardinalities for same data.
- Metadata conflicts e.g. same concepts are represented at schema level and instance level.
- Data conflicts e.g. missing attributes
- Schema conflicts e.g. table versus table conflict which includes naming conflicts, data conflicts etc.

In creating a federated schema, one has to resolve such heterogeneities before integrating the component DB schemas.

Schema matching, schema mapping

Dealing with incompatible data types or query syntax is not the only obstacle to a concrete implementation of an FDBS. In systems that are not planned top-down, a generic problem lies in matching semantically equivalent, but differently named parts from different schemas (=data models) (tables, attributes). A pairwise mapping between n attributes would result in $\frac{n(n-1)}{2}$ mapping rules (given equivalence mappings) - a number that quickly gets too large for practical purposes. A common way out is to provide a global schema that comprises the relevant parts of all member schemas and provide mappings in the form of database views. Two principal solutions can be realized, depending on the direction of the mapping:

1. Global as View (GaV): the global schema is defined in terms of the underlying schemas
2. Local as View (LaV): the local schemas are defined in terms of the global schema

Both are explained in more detail in the article Data integration. Alternate approaches to the schema matching problem and a classification of the same are explained in more detail in the article Schema Matching

Autonomy

Fundamental to the difference between an MDBS and an FDBS is the concept of autonomy. It is important to understand the aspects of autonomy for component databases and how they can be addressed when a component DBS participates in an FDBS. There are four kinds of autonomies addressed:

- Design Autonomy which refers to ability to choose its design irrespective of data, query language or conceptualization, functionality of the system implementation.

Heterogeneities in an FDBS are primarily due to design autonomy.

- Communication autonomy refers to the general operation of the DBMS to communicate with other DBMS or not.
- Execution autonomy allows a component DBMS to control the operations requested by local and external operations.
- Association autonomy gives a power to component DBS to disassociate itself from a federation which means FDBS can operate independently of any single DBS.

The ANSI/X3/SPARC Study Group outlined a three level data description architecture, the components of which are the conceptual schema, internal schema and external schema of databases. The three level architecture is however inadequate to describing the architectures of an FDBS. It was therefore extended to support the three dimensions of the FDBS namely Distribution, Autonomy and Heterogeneity. The five level schema architecture is explained below.

Concurrency control

The *Heterogeneity* and *Autonomy* requirements pose special challenges concerning concurrency control in an FDBS, which is crucial for the correct execution of its concurrent transactions (see also Global concurrency control). Achieving global serializability, the major correctness criterion, under these requirements has been characterized as very difficult and unsolved. Commitment ordering, introduced in 1991, has provided a general solution for this issue (See Global serializability; See Commitment ordering also for the architectural aspects of the solution).

Five Level Schema Architecture for FDBSs

The five level schema architecture includes the following:

- Local Schema is the conceptual concept *[unclear]* expressed in primary data model of component DBMS.
- Component Schema is derived by translating local schema into a model called the canonical data model or common data model. They are useful when semantics missed in local schema are incorporated in the component. They help in integration of data for tightly coupled FDBS.
- Export Schema represents a subset of a component schema that is available to the FDBS. It may include access control information regarding its use by specific federation user. The export schema help in managing flow of control of data.
- Federated Schema is an integration of multiple export schema. It includes information on data distribution that is generated when integrating export schemas.
- External Schema defines a schema for a user/applications or a class of users/applications.

While accurately representing the state of the art in data integration, the Five Level Schema Architecture above does suffer from a major drawback, namely IT imposed look and feel. Modern data users demand control over how data is presented; their needs are somewhat in conflict with such bottom-up approaches to data integration.

References

External links

- Schema coordination in federated database management: a comparison with schema integration (http://citeseer.ist.psu.edu/cache/papers/cs/9149/http:zSzzSzwww.bm.ust.hkzSz~zhaozSzDSS96.pdf/schema-coordination-in-federated.pdf)
- Storage of Behaviour of Object Database (http://www.computing.dcu.ie/~dalenk/publications/PhD Transfer talk.ppt)
- DB2 and Federated Databases (http://www.ibm.com/developerworks/db2/library/techarticle/dm-0504zikopoulos/)
- Tutorial on Federated Database (http://www.vldb.org/conf/1991/P489.PDF)
- GaV and LaV explained (http://www.dcs.bbk.ac.uk/~lucas/talks/SCSIS_RD_200507.pps)
- Issues of where to perform the join aka "pushdown" and other performance characteristics (http://www.ibm.com/developerworks/db2/library/techarticle/0304lurie/0304lurie.html)
- Worked example federating Oracle, Informix, DB2, and Excel (http://www.ibm.com/developerworks/db2/library/techarticle/0307lurie/0307lurie.html)
- Composite Information Server - a commercial federated database product (http://www.compositesw.com/products/cis.shtml)
- Freitas, André, Edward Curry, João Gabriel Oliveira, and Sean O'Riain. 2012. "Querying Heterogeneous Datasets on the Linked Data Web: Challenges, Approaches, and Trends." (http://www.edwardcurry.org/publications/freitas_IC_12.pdf) IEEE Internet Computing 16 (1): 24–33.
- IBM Gaian Database: A dynamic Distributed Federated Database (https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=f6ce657b-f385-43b2-8350-458e6e4a344f)
- Federated system and methods and mechanisms of implementing and using such a system (http://www.google.com/patents/US7392255)

Single Source of Truth

In Information Systems design and theory, as instantiated at the *Enterprise Level*, **Single Source Of Truth** (SSOT) refers to the practice of structuring information models and associated schemata such that every data element is stored exactly once (e.g., in no more than a single row of a single table). Any possible linkages to this data element (possibly in other areas of the relational schema or even in distant federated databases) are by reference only. Thus, when any such data element is updated, this update propagates to the enterprise at large, without the possibility of a duplicate value somewhere in the distant enterprise not being updated (because there would be no duplicate values that *needed* updating).[*citation needed*]

Deployment of an SSOT architecture is becoming increasingly important in enterprise settings where incorrectly linked duplicate or de-normalized data elements (a direct consequence of intentional or unintentional denormalization of any explicit data model) poses a risk for retrieval of outdated, and therefore incorrect, information. A common example would be the electronic health record, where it is imperative to accurately validate patient identity against a single referential repository, which serves as the SSOT. Duplicate representations of data within the enterprise would be implemented by the use of pointers rather than duplicate database tables, rows, or cells. This ensures that data updates to elements in the authoritative location are comprehensively distributed to all federated database constituencies in the larger overall enterprise architecture.[*citation needed*]

SSOT systems provide data that is authentic, relevant, and referable.Wikipedia:Please clarify[*citation needed*]

Implementing a Single Source of Truth

The "ideal" implementation of SSOT as described above is rarely possible in most enterprises. This is because many organisations have multiple information systems, each of which needs access to data relating to the same entities (e.g., customer). Often these systems are purchased "off-the-shelf" from vendors and cannot be modified in non-trivial ways. Each of these various systems therefore needs to store its own version of common data or entities, and therefore each system must retain its own copy of a record (hence immediately violating the SSOT approach defined above). For example, an ERP (Enterprise Resource Planning) system (such as SAP or Oracle e-Business Suite) may store a customer record; the CRM (Customer Relationship Management) system also needs a copy of the customer record (or part of it) and the warehouse despatch system might also need a copy of some or all of the customer data (e.g., shipping address). In cases where vendors do not support such modifications, it is not always possible to replace these records with pointers to the SSOT.

For organisations (with more than one information system) wishing to implement a Single Source of Truth (without modifying all but one master system to store pointers to other systems for all entities), three supporting technologies are commonly used:[*citation needed*]

- Enterprise Service Bus (ESB)
- Master Data Management (MDM)
- Data Warehouse (DW)

Enterprise Service Bus (ESB)

An Enterprise Service Bus (ESB) allows any number of systems in an organisation to receive updates of data that has changed in another system. To implement a Single Source of Truth, a single source system of correct data for any entity must be identified. Changes to this entity (creates, updates, and deletes) are then published via the ESB; other systems which need to retain a copy of that data subscribe to this update, and update their own records accordingly. For any given entity, the master source must be identified (sometimes called the Golden Record). It should be noted that any given system could publish (be the source of truth for) information on a particular entity (e.g., customer) and also subscribe to updates from another system for information on some other entity (e.g., product).[*citation needed*]

An alternative approach is point-to-point data updates, but these become exponentially more expensive to maintain as the number of systems increases, and this approach is increasingly out of favour as an IT architecture.[*citation needed*]

Master Data Management (MDM)

An MDM system can act as the source of truth for any given entity that might not necessarily have an alternative "source of truth" in another system. Typically the MDM acts as a hub for multiple systems, many of which could allow (be the source of truth for) updates to different aspects of information on a given entity. For example, the CRM system may be the "source of truth" for most aspects of the customer, and is updated by a call centre operator. However, a customer may (for example) also update their address via a customer service web site, with a different back-end database from the CRM system. The MDM application receives updates from multiple sources, acts as a broker to determine which updates are to be regarded as authoritative (the Golden Record) and then syndicates this updated data to all subscribing systems. The MDM application normally requires an ESB to syndicate its data to multiple subscribing systems.[*citation needed*]

Customer Data Integration (CDI), as a common application of Master Data Management, is sometimes abbreviated CDI-MDM.[*citation needed*]

Data Warehouse (DW)

While the primary purpose of a data warehouse is to support reporting and analysis of data that has been combined from multiple sources, the fact that such data has been combined (according to business logic embedded in the data transformation and integration processes) means that the data warehouse is often used as a *de facto* SSOT. Generally, however, the data available from the data warehouse is not used to update other systems; rather the DW becomes the "single source of truth" for reporting to multiple stakeholders. In this context, the Data Warehouse is more correctly referred to as a "Single Version of the Truth" since other versions of the truth exist in its operational data sources (no data originates in the DW; it is simply a reporting mechanism for data loaded from operational systems).[*citation needed*]

References

XLeratorDB

XLeratorDB is a suite of database function libraries that enable Microsoft SQL Server to perform a wide range of additional (non-native) business intelligence and ad hoc analytics. The libraries, which are embedded and run centrally on the database, include more than 450 individual functions similar to those found in Microsoft Excel spreadsheets. The individual functions are grouped and sold as six separate libraries based on usage: finance, statistics, math, engineering, unit conversions and strings. WestClinTech, the company that developed **XLeratorDB**, claims it is "the first commercial function package add-in for Microsoft SQL Server."[1]

Company history

WestClinTech (LLC), founded by software industry veterans Charles Flock and Joe Stampf in 2008, is located in Irvington, New York, USA. Flock was a co-founder of The Frustum Group, developer of the OPICS enterprise banking and trading platform, which was acquired by London-based Misys, PLC in 1996.[2] Stampf joined Frustum in 1994 and with Flock remained active with the company after acquisition, helping to develop successive generations of OPICS now employed by over 150 leading financial institutions worldwide.[3]

Following a full year of research, development and testing, WestClinTech introduced and recorded its first commercial sale of **XLeratorDB** in April 2009.[4][5] In September 2009, **XLeratorDB** became available to all Federal agencies through NASA's Strategic Enterprise-Wide Procurement (SEWP-IV) program, a government-wide acquisition contract.[6]

Technology

XLeratorDB uses Microsoft SQL CLR(Common Language Runtime) technology.[7] SQL CLR allows managed code to be hosted by, and run in, the Microsoft SQL Server environment. SQL CLR relies on the creation, deployment and registration of .NET Framework assemblies that are physically stored in managed code dynamic-link libraries (DLL). The assemblies may contain .NET namespaces, classes, functions, and properties. Because managed code compiles to native code prior to execution, functions using SQL CLR can achieve significant performance increases versus the equivalent functions written in T-SQL in some scenarios.[8]

XLeratorDB requires Microsoft SQL Server 2005 or SQL Server 2005 Express editions, or later (compatibility mode 90 or higher).[9] The product installs with PERMISSION_SET=SAFE. SAFE mode, the most restrictive permission set, is accessible by all users. Code executed by an assembly with SAFE permissions cannot access external system resources such as files, the network, the internet, environment variables, or the registry.[10]

Functions

In computer science, a function is a portion of code within a larger program which performs a specific task and is relatively independent of the remaining code. As used in database and spreadsheet applications these functions generally represent mathematical formulas widely used across a variety of fields. While this code may be user-generated, it is also embedded as a pre-written sub-routine in applications. These functions are typically identified by common nomenclature which corresponds to their underlying operations: e.g. **IRR** identifies the function which calculates Internal Rate of Return on a series of periodic cash flows.

Function uses

As sub-routines functions can be integrated and used in a variety of ways, and in a wide variety of larger, more complicated applications. Within large enterprise applications they may, for example, play an important role in defining business rules or risk management parameters, while remaining virtually undetected by end users. Within database management systems and spreadsheets, however, these kinds of functions also represent discrete sets of tools; they can be accessed directly and utilized on a stand-alone basis, or in more complex, user-defined configurations. In this context, functions can be used for business intelligence and ad hoc analysis of data in fields such as finance, statistics, engineering, math, etc.

Function types

XLeratorDB uses three kinds of functions to perform analytic operations: scalar, aggregate, and a hybrid form which WestClinTech calls ***Range Queries***. Scalar functions take a single value, perform an operation and return a single value.[11] An example of this type of function is **LOG**, which returns the logarithm of a number to a specified base.[12] Aggregate functions operate on a series of values but return a single, summarizing value. An example of this type of function is **AVG**, which returns the average of values in a specified group.[13]

In **XLeratorDB** there are some functions which have characteristics of aggregate functions (operating on multiple series of values) but cannot be processed in SQL CLR using single column inputs, such as **AVG** does. For example, irregular internal rate of return (**XIRR**), a financial function, operates on a collection of cash flow values from one column, but must also apply variable period lengths from another column and an initial iterative assumption from a third, in order to return a single, summarizing value. WestClinTech documentation notes that ***Range Queries*** specify the data to be included in the result set of the function independently of the WHERE clause associated with the T-SQL statement, by incorporating a SELECT statement into the function as a string argument; the function then traps that SELECT statement, executes it internally and processes the result.[14]

Some **XLeratorDB** functions that employ ***Range Queries*** are: **NPV**, **XNPV**, **IRR**, **XIRR**, **MIRR**, **MULTINOMIAL**, and **SERIESSUM**. Within the application these functions are identified by a "**_q**" naming convention: e.g. **NPV_q**, **IRR_q**, etc.[15]

Analytic functions

SQL Server functions

Microsoft SQL Server is the #3 selling database management system (DBMS), behind Oracle and IBM.[16] (While versions of SQL Server have been on the market since 1987,[17] **XLeratorDB** is compatible with only the 2005 edition and later.) Like all major DBMS, SQL Server performs a variety of data mining operations by returning or arraying data in different views (also known as drill-down). In addition, SQL Server uses Transact-SQL (T-SQL) [18] to execute four major classes of pre-defined functions in native mode.[19] Functions operating on the DBMS offer several advantages over client layer applications like Excel: they utilize the most up-to-date data available; they can process far larger quantities of data; and, the data is not subject to exporting and transcription errors.[20]

SQL Server 2008 includes a total of 58 functions that perform relatively basic aggregation (12), math (23) and string manipulation (23) operations useful for analytics; it includes no native functions that perform more complex operations directly related to finance, statistics or engineering.[21]

Excel functions

Microsoft Excel, a component of Microsoft Office suite, is one of the most widely used spreadsheet applications on the market today.[22] In addition to its inherent utility as a stand-alone desktop application, Excel overlaps and complements the functionality of DBMS in several ways: storing and arraying data in rows and columns; performing certain basic tasks such as pivot table[23] and aggregating values; and facilitating sharing, importing and exporting of database data. Excel's chief limitation relative to a true database is capacity; Excel 2003 is limited to some 65k rows and 256 columns; Excel 2007 extends this capacity to roughly 1million rows and 16k columns.[24] By comparison, SQL Server is able to manage over 500k terabytes of memory.[25]

Excel offers, however, an extensive library of specialized pre-written functions which are useful for performing ad hoc analysis on database data. Excel 2007 includes over 300 of these pre-defined functions, although customized functions can also be created by users, or imported from third party developers as add-ons. Excel functions are grouped by type:[26]

Excel Functions

Financial	Statistical	Engineering	Math and trig
Information	Date and time	Text and data	Logical
Add-ins and automation	Lookup and reference	Cube	Database and list management

Excel business intelligence functions

Operating on the client computing layer Excel plays an important role as a business intelligence tool [27] because it:

- performs a wide array of complex analytic functions not native to most DBMS software
- offers far greater ad hoc reporting and analytic flexibility than most enterprise software
- provides a medium for sharing and collaborating because of its ubiquity throughout the enterprise

Microsoft reinforces this positioning with Business Intelligence documentation that positions Excel in a clearly pivotal role.[28]

XLeratorDB vs. Excel functions

While operating within the database environment, **XLeratorDB** functions utilize the same naming conventions and input formats, and in most cases, return the same calculation results as Excel functions.[29] **XLeratorDB**, coupled with SQL Server's native capabilities, compares to Excel's function sets as follows:

Excel 2007		**XLeratorDB + SQL Server**			
Function Type	**Total**	**Total**	**Match**	**New**	**Native**
Financial	52	93	50	43	0
Statistics	83	171	65	94	12
Math	59	76	34	19	23
Engineering	39	44	38	6	0
Conversions*	49	78	0	78	0
Strings	26	63	11	29	23
**Microsoft includes these functions within Engineering using variable input configurations*					

References

[1] WestClinTech - SQL Server Functions (http://WestClinTech.com/Home/tabid/36/Default.aspx)

[2] History of Misys PLC – FundingUniverse (http://www.fundinguniverse.com/company-histories/Misys-PLC-Company-History.html)

[3] http://www.misys.com/tcm/markets/otc_derivative_trading_solutions-opics_plus.html

[4] Latest ArticleID Articles | SQL Server Pro (http://www.sqlmag.com/Articles/ArticleID/102126/102126.html?Ad=1)

[5] WestClinTech - Case Study - Newland Communities (http://WestClinTech.com/CaseStudyNewlandCommunities/tabid/178/Default.aspx)

[6] Nasa Sewp (http://www.sewp.nasa.gov/index.shtml)

[7] About XLeratorDB > Technology (http://WestClinTech.com/wct_local/Documentation/AboutXLeratorDB/tabid/148/topic/Technology/Default.aspx)

[8] This needs a reference

[9] FAQ (http://WestClinTech.com/Support/FAQ/tabid/143/Default.aspx#PL_A1_RequiredToInstall)

[10] See SQL Server DBMS documentation at: http://msdn.microsoft.com/en-us/library/ms130214.aspx

[11] Also see: http://msdn.microsoft.com/en-us/library/ms174318.aspx

[12] Excel definition at: http://office.microsoft.com/en-us/excel/HP100624401033.aspx

[13] SQL Server definition at: http://msdn.microsoft.com/en-us/library/ms177677.aspx

[14] About XLeratorDB > Range Queries (http://WestClinTech.com/wct_local/Documentation/AboutXLeratorDB/tabid/148/topic/Range Queries/Default.aspx)

[15] See XLeratorDB Function Packages information box, above

[16] IDC COMPETITIVE ANALYSIS: Worldwide Relational Database Management Systems 2007 Vendor Shares, Carl W. Olofson, June 2008, IDC #212840, Volume: 1, Tab: Markets

[17] http://e-articles.info/e/a/title/A-Brief-History-of-Microsoft-SQL-Server/

[18] SQL Server Language Reference (http://msdn.microsoft.com/en-us/library/ms166026(SQL.90).aspx)

[19] Built-in Functions (Transact-SQL) (http://msdn.microsoft.com/en-us/library/ms174318.aspx)

[20] IBM refers to this as "no-paste analytics. See Data Warehousing documentation.

[21] Functions (Transact-SQL) (http://msdn.microsoft.com/en-us/library/ms174318(v=SQL.100).aspx)

[22] Various sources suggest Office/Excel market share exceeds 90%, but this needs a specific source/citation

[23] What is pivot table? - Definition from WhatIs.com (http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci875976,00.html)

[24] See Excel documentation

[25] Maximum Capacity Specifications for SQL Server (http://msdn.microsoft.com/en-us/library/ms143432.aspx)

[26] Excel functions (alphabetical list) - Excel - Office.com (http://office.microsoft.com/en-us/excel/HA102775241033.aspx)

[27] See general Business Intelligence documentation Microsoft and IBM, for example: http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/cognos/pdfs/factsheets/fs_cognos8bi_analysis_for_microsoft_excel.pdf

[28] 2007 Microsoft Office System for Business Intelligence Fact Sheet: Fact Sheet October 2005 (http://www.microsoft.com/presspass/newsroom/office/BusinessIntelligenceFS.mspx)

[29] WestClinTech - SQL Server Functions - Blog - XLeratorDB/statistics is now available (http://www.WestClinTech.com/Blog/tabid/132/EntryID/19/Default.aspx)

External links

- XLeratorDB website (http://www.WestClinTech.com/)
- Microsoft SQL Server Documentation (http://www.microsoft.com/sqlserver/2008/en/us/default.aspx)
- Microsoft Excel Documentation (http://office.microsoft.com/en-us/excel/default.aspx)

Design

Dimension (data warehouse)

In a data warehouse, **Dimensions** provide structured labeling information to otherwise unordered numeric measures. The dimension is a data set composed of individual, non-overlapping data elements. The primary functions of dimensions are threefold: to provide filtering, grouping and labelling.

These functions are often described as "slice and dice". Slicing refers to filtering data. Dicing refers to grouping data. A common data warehouse example involves sales as the measure, with customer and product as dimensions. In each sale a customer buys a product. The data can be sliced by removing all customers except for a group under study, and then diced by grouping by product.

A dimensional data element is similar to a categorical variable in statistics.

Typically dimensions in a data warehouse are organized internally into one or more hierarchies. "Date" is a common dimension, with several possible hierarchies:

- "Days (are grouped into) Months (which are grouped into) Years",
- "Days (are grouped into) Weeks (which are grouped into) Years"
- "Days (are grouped into) Months (which are grouped into) Quarters (which are grouped into) Years"
- etc.

Types

Conformed dimension

A conformed dimension is a set of data attributes that have been physically referenced in multiple database tables using the same key value to refer to the same structure, attributes, domain values, definitions and concepts. A conformed dimension cuts across many facts.

Dimensions are conformed when they are either exactly the same (including keys) or one is a perfect subset of the other. Most important, the row headers produced in two different answer sets from the same conformed dimension(s) must be able to match perfectly.

Conformed dimensions are either identical or strict mathematical subsets of the most granular, detailed dimension. Dimension tables are not conformed if the attributes are labeled differently or contain different values. Conformed dimensions come in several different flavors. At the most basic level, conformed dimensions mean exactly the same thing with every possible fact table to which they are joined. The date dimension table connected to the sales facts is identical to the date dimension connected to the inventory facts.[1]

Junk dimension

A junk dimension is a convenient grouping of typically low-cardinality flags and indicators. By creating an abstract dimension, these flags and indicators are removed from the fact table while placing them into a useful dimensional framework.[2] A Junk Dimension is a dimension table consisting of attributes that do not belong in the fact table or in any of the existing dimension tables. The nature of these attributes is usually text or various flags, e.g. non-generic comments or just simple yes/no or true/false indicators. These kinds of attributes are typically remaining when all the obvious dimensions in the business process have been identified and thus the designer is faced with the challenge of where to put these attributes that do not belong in the other dimensions.

One solution is to create a new dimension for each of the remaining attributes, but due to their nature, it could be necessary to create a vast number of new dimensions resulting in a fact table with a very large number of foreign keys. The designer could also decide to leave the remaining attributes in the fact table but this could make the row length of the table unnecessarily large if, for example, the attributes is a long text string.

The solution to this challenge is to identify all the attributes and then put them into one or several Junk Dimensions. One Junk Dimension can hold several true/false or yes/no indicators that have no correlation with each other, so it would be convenient to convert the indicators into a more describing attribute. An example would be an indicator about whether a package had arrived, instead of indicating this as “yes” or “no”, it would be converted into “arrived” or “pending” in the junk dimension. The designer can choose to build the dimension table so it ends up holding all the indicators occurring with every other indicator so that all combinations are covered. This sets up a fixed size for the table itself which would be 2^x rows, where x is the number of indicators. This solution is appropriate in situations where the designer would expect to encounter a lot of different combinations and where the possible combinations are limited to an acceptable level. In a situation where the number of indicators are large, thus creating a very big table or where the designer only expect to encounter a few of the possible combinations, it would be more appropriate to build each row in the junk dimension as new combinations are encountered. To limit the size of the tables, multiple junk dimensions might be appropriate in other situations depending on the correlation between various indicators.

Junk dimensions are also appropriate for placing attributes like non-generic comments from the fact table. Such attributes might consist of data from an optional comment field when a customer places an order and as a result will probably be blank in many cases. Therefore the junk dimension should contain a single row representing the blanks as a surrogate key that will be used in the fact table for every row returned with a blank comment field[3]

Degenerate dimension

A degenerate dimension is a key, such as a transaction number, invoice number, ticket number, or bill-of-lading number, that has no attributes and hence does not join to an actual dimension table. Degenerate dimensions are very common when the grain of a fact table represents a single transaction item or line item because the degenerate dimension represents the unique identifier of the parent. Degenerate dimensions often play an integral role in the fact table's primary key.[4]

Role-playing dimension

Dimensions are often recycled for multiple applications within the same database. For instance, a "Date" dimension can be used for "Date of Sale", as well as "Date of Delivery", or "Date of Hire". This is often referred to as a "role-playing dimension".

Use of ISO representation terms

When referencing data from a metadata registry such as ISO/IEC 11179, representation terms such as **Indicator** (a boolean true/false value), **Code** (a set of non-overlapping enumerated values) are typically used as dimensions. For example using the National Information Exchange Model (NIEM) the data element name would be **PersonGenderCode** and the enumerated values would be **male**, **female** and **unknown**.

Common patterns

Date and time[5]

Since many fact tables in a data warehouse are time series of observations, one or more date dimensions are often needed. One of the reasons to have date dimensions is to place calendar knowledge in the data warehouse instead of hard coded in an application. While a simple SQL date/timestamp is useful for providing accurate information about the time a fact was recorded, it can not give information about holidays, fiscal periods, etc. An SQL date/timestamp can still be useful to store in the fact table, as it allows for precise calculations.

Having both the date and time of day in the same dimension, may easily result in a huge dimension with millions of rows. If a high amount of detail is needed it is usually a good idea to split date and time into two or more separate dimensions. A time dimension with a grain of seconds in a day will only have 86400 rows. A more or less detailed grain for date/time dimensions can be chosen depending on needs. As examples, date dimensions can be accurate to year, quarter, month or day and time dimensions can be accurate to hours, minutes or seconds.

As a rule of thumb, time of day dimension should only be created if hierarchical groupings are needed or if there are meaningful textual descriptions for periods of time within the day (ex. “evening rush” or “first shift”).

If the rows in a fact table are coming from several timezones, it might be useful to store date and time in both local time and a standard time. This can be done by having two dimensions for each date/time dimension needed – one for local time, and one for standard time. Storing date/time in both local and standard time, will allow for analysis on when facts are created in a local setting and in a global setting as well. The standard time chosen can be a global standard time (ex. UTC), it can be the local time of the business' headquarter, or any other time zone that would make sense to use.

References

- Kimball, Ralph et al. (1998); *The Data Warehouse Lifecycle Toolkit*, p17. Pub. Wiley. ISBN 0-471-25547-5.
- Kimball, Ralph (1996); *The Data Warehouse Toolkit*, p. 100. Pub. Wiley. ISBN 0-471-15337-0.

Notes

[1] Ralph Kimball, Margy Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Second Edition, Wiley Computer Publishing, 2002. ISDN 0471-20024-7, Pages 82-87, 394

[2] Ralph Kimball, Margy Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Second Edition, Wiley Computer Publishing, 2002. ISDN 0471-20024-7, Pages 202, 405

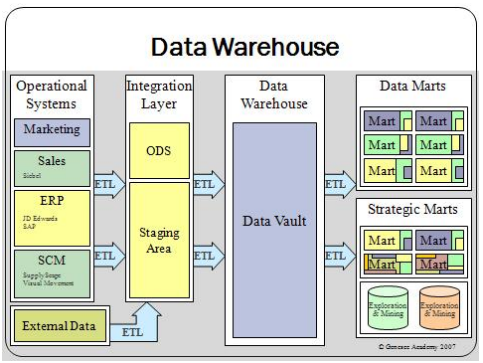
[3] Kimball, Ralph, et al. (2008): The Data Warehouse Lifecycle Toolkit, Second Edition, Wiley Publishing Inc., Indianapolis, IN. Pages 263-265

[4] Ralph Kimball, Margy Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Second Edition, Wiley Computer Publishing, 2002. ISDN 0471-20024-7, Pages 50, 398

[5] Ralph Kimball, The Data Warehouse Toolkit, Second Edition, Wiley Publishing, Inc., 2008. ISBN 978-0-470-14977-5, Pages 253-256

Fact (data warehouse)

In computing, a **data warehouse** (**DW**, **DWH**), or an **enterprise data warehouse** (**EDW**), is a database used for reporting (1) and data analysis (2). Integrating data from one or more disparate sources creates a central repository of data, a data warehouse (DW). Data warehouses store current and historical data and are used for creating trending reports for senior management reporting such as annual and quarterly comparisons.



Data Warehouse Overview

The data stored in the warehouse is uploaded from the operational systems (such as marketing, sales, etc., shown in the figure to the right). The data may pass through an operational data store for additional operations before it is used in the DW for reporting.

The typical extract transform load (ETL)-based data warehouse uses staging, data integration, and access layers to house its key functions. The staging layer or staging database stores raw data extracted from each of the disparate source data systems. The integration layer integrates the disparate data sets by transforming the data from the staging layer often storing this transformed data in an operational data store (ODS) database. The integrated data are then moved to yet another database, often called the data warehouse database, where the data is arranged into hierarchical groups often called dimensions and into facts and aggregate facts. The combination of facts and dimensions is sometimes called a star schema. The access layer helps users retrieve data.

A data warehouse constructed from an integrated data source systems does not require ETL, staging databases, or operational data store databases. The integrated data source systems may be considered to be a part of a distributed operational data store layer. Data federation methods or data virtualization methods may be used to access the distributed integrated source data systems to consolidate and aggregate data directly into the data warehouse database tables. Unlike the ETL-based data warehouse, the integrated source data systems and the data warehouse are all integrated since there is no transformation of dimensional or reference data. This integrated data warehouse architecture supports the drill down from the aggregate data of the data warehouse to the transactional data of the integrated source data systems.

A data mart is a small data warehouse focused on a specific area of interest. Data warehouses can be subdivided into data marts for improved performance and ease of use within that area. Alternatively, an organization can create one or more data marts as first steps towards a larger and more complex enterprise data warehouse.

This definition of the data warehouse focuses on data storage. The main source of the data is cleaned, transformed, cataloged and made available for use by managers and other business professionals for data mining, online analytical processing, market research and decision support (Marakas & O'Brien 2009). However, the means to retrieve and analyze data, to extract, transform and load data, and to manage the data dictionary are also considered essential components of a data warehousing system. Many references to data warehousing use this broader context. Thus, an expanded definition for data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.

Benefits of a data warehouse

A data warehouse maintains a copy of information from the source transaction systems. This architectural complexity provides the opportunity to :

- Congregate data from multiple sources into a single database so a single query engine can be used to present data.
- Mitigate the problem of database isolation level lock contention in transaction processing systems caused by attempts to run large, long running, analysis queries in transaction processing databases.
- Maintain data history, even if the source transaction systems do not.
- Integrate data from multiple source systems, enabling a central view across the enterprise. This benefit is always valuable, but particularly so when the organization has grown by merger.
- Improve data quality, by providing consistent codes and descriptions, flagging or even fixing bad data.
- Present the organization's information consistently.
- Provide a single common data model for all data of interest regardless of the data's source.
- Restructure the data so that it makes sense to the business users.
- Restructure the data so that it delivers excellent query performance, even for complex analytic queries, without impacting the operational systems.
- Add value to operational business applications, notably customer relationship management (CRM) systems.
- Making decision–support queries easier to write.

Generic data warehouse environment

The environment for data warehouses and marts includes the following:

- Source systems that provide data to the warehouse or mart;
- Data integration technology and processes that are needed to prepare the data for use;
- Different architectures for storing data in an organization's data warehouse or data marts;
- Different tools and applications for the variety of users;
- Metadata, data quality, and governance processes must be in place to ensure that the warehouse or mart meets its purposes.

In regards to source systems listed above, Rainer states, “A common source for the data in data warehouses is the company’s operational databases, which can be relational databases”.

Regarding data integration, Rainer states, “It is necessary to extract data from source systems, transform them, and load them into a data mart or warehouse”.

Rainer discusses storing data in an organization’s data warehouse or data marts. “There are a variety of possible architectures to store decision-support data”.

Metadata are data about data. “IT personnel need information about data sources; database, table, and column names; refresh schedules; and data usage measures“.

Today, the most successful companies are those that can respond quickly and flexibly to market changes and opportunities. A key to this response is the effective and efficient use of data and information by analysts and managers. A “data warehouse” is a repository of historical data that are organized by subject to support decision makers in the organization. Once data are stored in a data mart or warehouse, they can be accessed.

History

The concept of data warehousing dates back to the late 1980s when IBM researchers Barry Devlin and Paul Murphy developed the "business data warehouse". In essence, the data warehousing concept was intended to provide an architectural model for the flow of data from operational systems to decision support environments. The concept attempted to address the various problems associated with this flow, mainly the high costs associated with it. In the absence of a data warehousing architecture, an enormous amount of redundancy was required to support multiple decision support environments. In larger corporations it was typical for multiple decision support environments to operate independently. Though each environment served different users, they often required much of the same stored data. The process of gathering, cleaning and integrating data from various sources, usually from long-term existing operational systems (usually referred to as legacy systems), was typically in part replicated for each environment. Moreover, the operational systems were frequently reexamined as new decision support requirements emerged. Often new requirements necessitated gathering, cleaning and integrating new data from "data marts" that were tailored for ready access by users.

Key developments in early years of data warehousing were:

- 1960s — General Mills and Dartmouth College, in a joint research project, develop the terms *dimensions* and *facts*.[1]
- 1970s — ACNielsen and IRI provide dimensional data marts for retail sales.
- 1970s — Bill Inmon begins to define and discuss the term: Data Warehouse
- 1975 — Sperry Univac Introduce MAPPER (MAintain, Prepare, and Produce Executive Reports) is a database management and reporting system that includes the world's first 4GL. It was the first platform designed for building Information Centers (a forerunner of contemporary Enterprise Data Warehousing platforms)
- 1983 — Teradata introduces a database management system specifically designed for decision support.
- 1983 — Sperry Corporation Martyn Richard Jones defines the Sperry Information Center approach, which while not being a true DW in the Inmon sense, did contain many of the characteristics of DW structures and process as defined previously by Inmon, and later by Devlin. First used at the TSB England & Wales
- 1984 — Metaphor Computer Systems, founded by David Liddle and Don Massaro, releases Data Interpretation System (DIS). DIS was a hardware/software package and GUI for business users to create a database management and analytic system.
- 1988 — Barry Devlin and Paul Murphy publish the article *An architecture for a business and information system* [2] in *IBM Systems Journal* where they introduce the term "business data warehouse".
- 1990 — Red Brick Systems, founded by Ralph Kimball, introduces Red Brick Warehouse, a database management system specifically for data warehousing.
- 1991 — Prism Solutions, founded by Bill Inmon, introduces Prism Warehouse Manager, software for developing a data warehouse.
- 1992 — Bill Inmon publishes the book *Building the Data Warehouse*.
- 1995 — The Data Warehousing Institute, a for-profit organization that promotes data warehousing, is founded.
- 1996 — Ralph Kimball publishes the book *The Data Warehouse Toolkit*.
- 2000 — Daniel Linstedt releases the *Data Vault*, enabling real time auditable Data Warehouses warehouse.

Information storage

Facts

A fact is a value or measurement, which represents a fact about the managed entity or system.

Facts as reported by the reporting entity are said to be at raw level.

E.g. if a BTS received 1,000 requests for traffic channel allocation, it allocates for 820 and rejects the remaining then it would report 3 **facts** or measurements to a management system:

- tch_req_total = 1000
- tch_req_success = 820
- tch_req_fail = 180

Facts at raw level are further aggregated to higher levels in various dimensions to extract more service or business-relevant information out of it. These are called aggregates or summaries or aggregated facts.

E.g. if there are 3 BTSs in a city, then facts above can be aggregated from BTS to city level in network dimension. E.g.

- $tch\_req\_success\_city = tch\_req\_success\_bts1 + tch\_req\_success\_bts2 + tch\_req\_success\_bts3$
- $avg\_tch\_req\_success\_city = (tch\_req\_success\_bts1 + tch\_req\_success\_bts2 + tch\_req\_success\_bts3)/3$

Dimensional vs. normalized approach for storage of data

There are three or more leading approaches to storing data in a data warehouse — the most important approaches are the dimensional approach and the normalized approach.

The dimensional approach, whose supporters are referred to as “Kimballites”, believe in Ralph Kimball's approach in which it is stated that the data warehouse should be modeled using a Dimensional Model/star schema. The normalized approach, also called the 3NF model (Third Normal Form), whose supporters are referred to as “Inmonites”, believe in Bill Inmon's approach in which it is stated that the data warehouse should be modeled using an E-R model/normalized model.

In a dimensional approach, transaction data are partitioned into "facts", which are generally numeric transaction data, and "dimensions", which are the reference information that gives context to the facts. For example, a sales transaction can be broken up into facts such as the number of products ordered and the price paid for the products, and into dimensions such as order date, customer name, product number, order ship-to and bill-to locations, and salesperson responsible for receiving the order.

A key advantage of a dimensional approach is that the data warehouse is easier for the user to understand and to use. Also, the retrieval of data from the data warehouse tends to operate very quickly.[*citation needed*] Dimensional structures are easy to understand for business users, because the structure is divided into measurements/facts and context/dimensions. Facts are related to the organization’s business processes and operational system whereas the dimensions surrounding them contain context about the measurement (Kimball, Ralph 2008).

The main disadvantages of the dimensional approach are the following:

1. In order to maintain the integrity of facts and dimensions, loading the data warehouse with data from different operational systems is complicated.
2. It is difficult to modify the data warehouse structure if the organization adopting the dimensional approach changes the way in which it does business.

In the normalized approach, the data in the data warehouse are stored following, to a degree, database normalization rules. Tables are grouped together by *subject areas* that reflect general data categories (e.g., data on customers, products, finance, etc.). The normalized structure divides data into entities, which creates several tables in a relational database. When applied in large enterprises the result is dozens of tables that are linked together by a web

of joins. Furthermore, each of the created entities is converted into separate physical tables when the database is implemented (Kimball, Ralph 2008)[*citation needed*]. The main advantage of this approach is that it is straightforward to add information into the database. Some disadvantages of this approach are that, because of the number of tables involved, it can be difficult for users to join data from different sources into meaningful information and to access the information without a precise understanding of the sources of data and of the data structure of the data warehouse.

It should be noted that both normalized and dimensional models can be represented in entity-relationship diagrams as both contain joined relational tables. The difference between the two models is the degree of normalization (also known as Normal Forms).

These approaches are not mutually exclusive, and there are other approaches. Dimensional approaches can involve normalizing data to a degree (Kimball, Ralph 2008).

In Information-Driven Business, Robert Hillard proposes an approach to comparing the two approaches based on the information needs of the business problem. The technique shows that normalized models hold far more information than their dimensional equivalents (even when the same fields are used in both models) but this extra information comes at the cost of usability. The technique measures information quantity in terms of Information Entropy and usability in terms of the Small Worlds data transformation measure.

Top-down versus bottom-up design methodologies

Bottom-up design

Ralph Kimball,[2] designed an approach to data warehouse design known as *bottom-up*.

In the *bottom-up* approach, data marts are first created to provide reporting and analytical capabilities for specific business processes.

Data marts contain, primarily, dimensions and facts. Facts can contain either atomic data and, if necessary, summarized data. The single data mart often models a specific business area such as "Sales" or "Production." These data marts can eventually be integrated to create a comprehensive data warehouse. The data warehouse bus architecture is primarily an implementation of "the bus", a collection of conformed dimensions and conformed facts, which are dimensions that are shared (in a specific way) between facts in two or more data marts.

The integration of the data marts in the data warehouse is centered on the conformed dimensions (residing in "the bus") that define the possible integration "points" between data marts. The actual integration of two or more data marts is then done by a process known as "Drill across". A drill-across works by grouping (summarizing) the data along the keys of the (shared) conformed dimensions of each fact participating in the "drill across" followed by a join on the keys of these grouped (summarized) facts.

Maintaining tight management over the data warehouse bus architecture is fundamental to maintaining the integrity of the data warehouse. The most important management task is making sure dimensions among data marts are consistent.

Business value can be returned as quickly as the first data marts can be created, and the method lends itself well to an exploratory and iterative approach to building data warehouses. For example, the data warehousing effort might start in the "Sales" department, by building a Sales-data mart. Upon completion of the Sales-data mart, the business might then decide to expand the warehousing activities into the, say, "Production department" resulting in a Production data mart. The requirement for the Sales data mart and the Production data mart to be integrable, is that they share the same "Bus", that will be, that the data warehousing team has made the effort to identify and implement the conformed dimensions in the bus, and that the individual data marts links that information from the bus. The Sales-data mart is good as it is (assuming that the bus is complete) and the Production-data mart can be constructed virtually independent of the Sales-data mart (but not independent of the Bus).

If integration via the bus is achieved, the data warehouse, through its two data marts, will not only be able to deliver the specific information that the individual data marts are designed to do, in this example either "Sales" or "Production" information, but can deliver integrated Sales-Production information, which, often, is of critical business value.

Top-down design

Bill Inmon, has defined a data warehouse as a centralized repository for the entire enterprise.[3] The *top-down* approach is designed using a normalized enterprise data model. "Atomic" data, that is, data at the lowest level of detail, are stored in the data warehouse. Dimensional data marts containing data needed for specific business processes or specific departments are created from the data warehouse. In the Inmon vision, the data warehouse is at the center of the "Corporate Information Factory" (CIF), which provides a logical framework for delivering business intelligence (BI) and business management capabilities. Gartner released a research note confirming Inmon's definition in 2005[4] with additional clarity plus they added one additional attribute.

The data warehouse is:

Subject-oriented

> The data in the data warehouse is organized so that all the data elements relating to the same real-world event or object are linked together.

Non-volatile

> Data in the data warehouse are never over-written or deleted — once committed, the data are static, read-only, and retained for future reporting.

Integrated

> The data warehouse contains data from most or all of an organization's operational systems and these data are made consistent.

Time-variant

> For an **operational system**, the stored data contains the current value. The data warehouse, however, contains the history of data values.

No virtualization

> A data warehouse is a physical repository.

The top-down design methodology generates highly consistent dimensional views of data across data marts since all data marts are loaded from the centralized repository. Top-down design has also proven to be robust against business changes. Generating new dimensional data marts against the data stored in the data warehouse is a relatively simple task. The main disadvantage to the top-down methodology is that it represents a very large project with a very broad scope. The up-front cost for implementing a data warehouse using the top-down methodology is significant, and the duration of time from the start of project to the point that end users experience initial benefits can be substantial. In addition, the top-down methodology can be inflexible and unresponsive to changing departmental needs during the implementation phases.

Hybrid design

Data warehouse (DW) solutions often resemble the hub and spokes architecture. Legacy systems feeding the DW/BI solution often include customer relationship management (CRM) and enterprise resource planning solutions (ERP), generating large amounts of data. To consolidate these various data models, and facilitate the extract transform load (ETL) process, DW solutions often make use of an operational data store (ODS). The information from the ODS is then parsed into the actual DW. To reduce data redundancy, larger systems will often store the data in a normalized way. Data marts for specific reports can then be built on top of the DW solution.

It is important to note that the DW database in a hybrid solution is kept on third normal form to eliminate data redundancy. A normal relational database however, is not efficient for business intelligence reports where dimensional modelling is prevalent. Small data marts can shop for data from the consolidated warehouse and use the filtered, specific data for the fact tables and dimensions required. The DW effectively provides a single source of information from which the data marts can read, creating a highly flexible solution from a BI point of view. The hybrid architecture allows a DW to be replaced with a master data management solution where operational, not static information could reside.

The Data Vault Modeling components follow hub and spokes architecture. This modeling style is a hybrid design, consisting of the best practices from both 3rd normal form and star schema. The Data Vault model is not a true 3rd normal form, and breaks some of the rules that 3NF dictates be followed. It is however, a top-down architecture with a bottom up design. The Data Vault model is geared to be strictly a data warehouse. It is not geared to be end-user accessible, which when built, still requires the use of a data mart or star schema based release area for business purposes.

Data warehouses versus operational systems

Operational systems are optimized for preservation of data integrity and speed of recording of business transactions through use of database normalization and an entity-relationship model. Operational system designers generally follow the Codd rules of database normalization in order to ensure data integrity. Codd defined five increasingly stringent rules of normalization. Fully normalized database designs (that is, those satisfying all five Codd rules) often result in information from a business transaction being stored in dozens to hundreds of tables. Relational databases are efficient at managing the relationships between these tables. The databases have very fast insert/update performance because only a small amount of data in those tables is affected each time a transaction is processed. Finally, in order to improve performance, older data are usually periodically purged from operational systems.

Evolution in organization use

These terms refer to the level of sophistication of a data warehouse:

Offline operational data warehouse

> Data warehouses in this stage of evolution are updated on a regular time cycle (usually daily, weekly or monthly) from the operational systems and the data is stored in an integrated reporting-oriented data

Offline data warehouse

> Data warehouses at this stage are updated from data in the operational systems on a regular basis and the data warehouse data are stored in a data structure designed to facilitate reporting.

On time data warehouse

> Online Integrated Data Warehousing represent the real time Data warehouses stage data in the warehouse is updated for every transaction performed on the source data

Integrated data warehouse

> These data warehouses assemble data from different areas of business, so users can look up the information they need across other systems.

References

[1] Kimball 2002, pg. 16
[2] Kimball 2002, pg. 310
[3] Ericsson 2004, pp. 28–29
[4] Gartner, Of Data Warehouses, Operational Data Stores, Data Marts and Data Outhouses, Dec 2005

Further reading

- Davenport, Thomas H. and Harris, Jeanne G. *Competing on Analytics: The New Science of Winning* (2007) Harvard Business School Press. ISBN 978-1-4221-0332-6
- Ganczarski, Joe. *Data Warehouse Implementations: Critical Implementation Factors Study* (2009) VDM Verlag ISBN 3-639-18589-7 ISBN 978-3-639-18589-8
- Kimball, Ralph and Ross, Margy. *The Data Warehouse Toolkit* Second Edition (2002) John Wiley and Sons, Inc. ISBN 0-471-20024-7
- Linstedt, Graziano, Hultgren. *The Business of Data Vault Modeling* Second Edition (2010) Dan linstedt, ISBN 978-1-4357-1914-9
- William Inmon. *Building the Data Warehouse* 2005) John Wiley and Sons, ISBN 978-8-1265-0645-3

External links

- Ralph Kimball articles (http://www.kimballgroup.com/html/articles.html)
- International Journal of Computer Applications (http://www.ijcaonline.org/archives/number3/77-172)
- Data Warehouse Introduction (http://dwreview.com/DW_Overview.html)
- Time to Reconsider the Data Warehouse (Global Association of Risk Professionals) (http://www.garp.org/risk-news-and-resources/2013/june/time-to-reconsider-the-data-warehouse.aspx?)

Measure (data warehouse)

In a data warehouse, a **measure** is a property on which calculations (e.g., sum, count, average, minimum, maximum) can be made.

Example

For example if a retail store sold a specific product, the quantity and prices of each item sold could be added or averaged to find the total number of items sold and total or average price of the goods.

Use of ISO representation terms

When entering data into a metadata registry such as ISO/IEC 11179, representation terms such as **Number**, **Value** and **Measure** are typically used as measures.

References

- Kimball, Ralph et al. (1998); *The Data Warehouse Lifecycle Toolkit*, p17. Pub. Wiley. ISBN 0-471-25547-5.
- Kimball, Ralph (1996); *The Data Warehouse Toolkit*, p100. Pub. Wiley. ISBN 0-471-15337-0.

Fact table

In data warehousing, a **fact table** consists of the measurements, metrics or facts of a business process. It is located at the center of a star schema or a snowflake schema surrounded by dimension tables. Where multiple fact tables are used, these are arranged as a fact constellation schema. A fact table typically has two types of columns: those that contain facts and those that are foreign keys to dimension tables. The primary key of a fact table is usually a composite key that is made up of all of its foreign keys. Fact tables contain the content of the data warehouse and store different types of measures like additive, non additive, and semi additive measures.

Fact tables provide the (usually) additive values that act as independent variables by which dimensional attributes are analyzed. Fact tables are often defined by their *grain*. The grain of a fact table represents the most atomic level by which the facts may be defined. The grain of a SALES fact table might be stated as "Sales volume by Day by Product by Store". Each record in this fact table is therefore uniquely defined by a day, product and store. Other dimensions might be members of this fact table (such as location/region) but these add nothing to the uniqueness of the fact records. These "affiliate dimensions" allow for additional slices of the independent facts but generally provide insights at a higher level of aggregation (a region contains many stores).

Example

If the business process is SALES, then the corresponding fact table will typically contain columns representing both raw facts and aggregations in rows such as:

- *$12,000*, being "sales for New York store for 15-Jan-2005"
- *$34,000*, being "sales for Los Angeles store for 15-Jan-2005"
- *$22,000*, being "sales for New York store for 16-Jan-2005"
- *$50,000*, being "sales for Los Angeles store for 16-Jan-2005"
- *$21,000*, being "average daily sales for Los Angeles Store for Jan-2005"
- *$65,000*, being "average daily sales for Los Angeles Store for Feb-2005"
- *$33,000*, being "average daily sales for Los Angeles Store for year 2005"

"average daily sales" is a measurement which is stored in the fact table. The fact table also contains foreign keys from the dimension tables, where time series (e.g. dates) and other dimensions (e.g. store location, salesperson, product) are stored.

All foreign keys between fact and dimension tables should be surrogate keys, not reused keys from operational data.

Measure types

- Additive - Measures that can be added across any dimension.
- Non Additive - Measures that cannot be added across any dimension.
- Semi Additive - Measures that can be added across some dimensions.

A fact table might contain either detail level facts or facts that have been aggregated (fact tables that contain aggregated facts are often instead called summary tables).

Special care must be taken when handling ratios and percentage. One good design rule[1] is to never store percentages or ratios in fact tables but only calculate these in the data access tool. Thus only store the numerator and denominator in the fact table, which then can be aggregated and the aggregated stored values can then be used for calculating the ratio or percentage in the data access tool.

In the real world, it is possible to have a fact table that contains no measures or facts. These tables are called "factless fact tables", or "junction tables".

The "Factless fact tables" can for example be used for modeling many-to-many relationships or capture events.

Types of fact tables

There are basically three fundamental measurement events, which characterizes all fact tables.

- Transactional

  A transactional table is the most basic and fundamental. The grain associated with a transactional fact table is usually specified as "one row per line in a transaction", e.g., every line on a receipt. Typically a transactional fact table holds data of the most detailed level, causing it to have a great number of dimensions associated with it.

- Periodic snapshots

  The periodic snapshot, as the name implies, takes a "picture of the moment", where the moment could be any defined period of time, e.g. a performance summary of a salesman over the previous month. A periodic snapshot table is dependent on the transactional table, as it needs the detailed data held in the transactional fact table in order to deliver the chosen performance output.

- Accumulating snapshots

  This type of fact table is used to show the activity of a process that has a well-defined beginning and end, e.g., the processing of an order. An order moves through specific steps until it is fully processed. As steps towards fulfilling the order are completed, the associated row in the fact table is updated. An accumulating snapshot table often has multiple date columns, each representing a milestone in the process. Therefore, it's important to have an entry in the associated date dimension that represents an unknown date, as many of the milestone dates are unknown at the time of the creation of the row.

- Temporal snapshots

  By applying Temporal Database theory and modelling techniques the Temporal Snapshot Fact Table allows to have the equivalent of daily snapshots without really having daily snapshots. It introduces the concept of Time Intervals into a fact table, allowing to save a lot of space, optimizing performances while allowing the end user to have the logical equivalent of the "picture of the moment" he is interested in.

Steps in designing a fact table

- Identify a business process for analysis (like sales).
- Identify measures of facts (sales dollar), by asking questions like 'What number of XX are relevant for the business process?', replacing the XX with various options that make sense within the context of the business.
- Identify dimensions for facts (product dimension, location dimension, time dimension, organization dimension), by asking questions that make sense within the context of the business, like 'Analyse by XX', where XX is replaced with the subject to test.
- List the columns that describe each dimension (region name, branch name, business unit name).
- Determine the lowest level (granularity) of summary in a fact table (e.g. sales dollars).

An alternative approach is the four step design process described in Kimball.

References

[1] Kimball & Ross - The Data Warehouse Toolkit, 2nd Ed [Wiley 2002]

Degenerate dimension

The Kimball definition

According to Ralph Kimball, in a data warehouse, a **degenerate dimension** is a dimension key in the fact table that does not have its own dimension table, because all the interesting attributes have been placed in analytic dimensions. The term "degenerate dimension" was originated by Ralph Kimball.

As Bob Becker says,

> Degenerate dimensions commonly occur when the fact table's grain is a single transaction (or transaction line). Transaction control header numbers assigned by the operational business process are typically degenerate dimensions, such as order, ticket, credit card transaction, or check numbers. These degenerate dimensions are natural keys of the "parents" of the line items.
>
> Even though there is no corresponding dimension table of attributes, degenerate dimensions can be quite useful for grouping together related fact tables rows. For example, retail point-of-sale transaction numbers tie all the individual items purchased together into a single market basket. In health care, degenerate dimensions can group the claims items related to a single hospital stay or episode of care.

Other uses of the term

Although most writers and practitioners use the term **degenerate dimension** correctly, it is very easy to find misleading definitions in online and printed sources. For example, the Oracle FAQ defines a degenerate dimension as a "data dimension that is stored in the fact table rather than a separate dimension table. This eliminates the need to join to a dimension table. You can use the data in the degenerate dimension to limit or 'slice and dice' your fact table measures."

This common interpretation implies that it is good dimensional modeling practice to place dimension attributes in the fact table, as long as you call them a degenerate dimension. This is not the case; the concept of **degenerate dimension** was developed by Kimball to support a specific, well-defined exception to the otherwise ironclad rule that dimension attributes are always pulled out into dimension tables.

External reference

Becker, Bob (3 June 2003). "Design Tip #46: Another Look At Degenerate Dimensions" [1]. *Fact Table Core Concepts*. Kimball Group. Retrieved 25 January 2013.

Notes

[1] http://www.kimballgroup.com/2003/06/03/design-tip-46-another-look-at-degenerate-dimensions/

References

- Kimball, Ralph et al. (1998); *The Data Warehouse Lifecycle Toolkit*, p17. Pub. Wiley. ISBN 0-471-25547-5.
- Kimball, Ralph (1996); *The Data Warehouse Toolkit*, p. 100. Pub. Wiley. ISBN 0-471-15337-0.

Slowly changing dimension

Dimension is a term in data management and data warehousing. It's the logical groupings of data such as geographical location, customer or product information. With **Slowly Changing Dimensions (SCDs)** data changes slowly, rather than changing on a time-based, regular schedule.

For example, you may have a dimension in your database that tracks the sales records of your company's salespeople. Creating sales reports seems simple enough, until a salesperson is transferred from one regional office to another. How do you record such a change in your sales dimension?

You could calculate the sum or average of each salesperson's sales, but if you use that to compare the performance of salespeople, that might give misleading information. If the salesperson was transferred and used to work in a hot market where sales were easy, and now works in a market where sales are infrequent, his/her totals will look much stronger than the other salespeople in their new region. Or you could create a second salesperson record and treat the transferred person as a new sales person, but that creates problems.

Dealing with these issues involves SCD management methodologies referred to as Type 0 through 6. Type 6 SCDs are also sometimes called Hybrid SCDs.

Type 0

The **Type 0** method is passive. It manages dimensional changes and no action is performed. Values remain as they were at the time the dimension record was first inserted. In certain circumstances history is preserved with a Type 0. High order types are employed to guarantee the preservation of history whereas Type 0 provides the least or no control.

The most common types are I, II, and III.

Type 1

This methodology overwrites old with new data, and therefore does not track historical data.

Example of a supplier table:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	CA

In the above example, Supplier_Code is the natural key and Supplier_Key is a surrogate key. Technically, the surrogate key is not necessary, since the row will be unique by the natural key (Supplier_Code). However, to

optimize performance on joins use integer rather than character keys.

If the supplier relocates the headquarters to Illinois the record would be overwritten:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	IL

The disadvantage of the Type I method is that there is no history in the data warehouse. It has the advantage however that it's easy to maintain.

If you have calculated an aggregate table summarizing facts by state, it will need to be recalculated when the Supplier_State is changed.

Type 2

This method tracks historical data by creating multiple records for a given natural key in the dimensional tables with separate surrogate keys and/or different version numbers. Unlimited history is preserved for each insert.

For example, if the supplier relocates to Illinois the version numbers will be incremented sequentially:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Version.
123	ABC	Acme Supply Co	CA	0
124	ABC	Acme Supply Co	IL	1

Another method is to add 'effective date' columns.

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Start_Date	End_Date
123	ABC	Acme Supply Co	CA	01-Jan-2000	21-Dec-2004
124	ABC	Acme Supply Co	IL	22-Dec-2004	

The null End_Date in row two indicates the current tuple version. In some cases, a standardized surrogate high date (e.g. 9999-12-31) may be used as an end date, so that the field can be included in an index, and so that null-value substitution is not required when querying.

Transactions that reference a particular surrogate key (Supplier_Key) are then permanently bound to the time slices defined by that row of the slowly changing dimension table. An aggregate table summarizing facts by state continues to reflect the historical state, i.e. the state the supplier was in at the time of the transaction; no update is needed.

If there are retrospective changes made to the contents of the dimension, or if new attributes are added to the dimension (for example a Sales_Rep column) which have different effective dates from those already defined, then this can result in the existing transactions needing to be updated to reflect the new situation. This can be an expensive database operation, so Type 2 SCDs are not a good choice if the dimensional model is subject to change.

Type 3

This method tracks changes using separate columns and preserves limited history. The Type 3 preserves limited history as it's limited to the number of columns designated for storing historical data. The original table structure in Type 1 and Type 2 is the same but Type III adds additional columns. In the following example, an additional column has been added to the table to record the supplier's original state - only the previous history is stored.

Supplier_Key	Supplier_Code	Supplier_Name	Original_Supplier_State	Effective_Date	Current_Supplier_State
123	ABC	Acme Supply Co	CA	22-Dec-2004	IL

This record contains a column for the original state and current state—cannot track the changes if the supplier relocates a second time.

One variation of this is to create the field Previous_Supplier_State instead of Original_Supplier_State which would track only the most recent historical change.

Type 4

The **Type 4** method is usually referred to as using "history tables", where one table keeps the current data, and an additional table is used to keep a record of some or all changes. Both the surrogate keys are referenced in the Fact table to enhance query performance.For the above example the original table name is **Supplier** and the history table is **Supplier_History.**

Supplier

Supplier_key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	IL

Supplier_History

Supplier_key	Supplier_Code	Supplier_Name	Supplier_State	Create_Date
123	ABC	Acme Supply Co	CA	22-Dec-2004

This method resembles how database audit tables and change data capture techniques function.

Type 6 / hybrid

The **Type 6** method combines the approaches of types 1, 2 and 3 (1 + 2 + 3 = 6). One possible explanation of the origin of the term was that it was coined by Ralph Kimball during a conversation with Stephen Pace from Kalido[*citation needed*]. Ralph Kimball calls this method "Unpredictable Changes with Single-Version Overlay" in *The Data Warehouse Toolkit*.

The Supplier table starts out with one record for our example supplier:

Supplier_Key	Supplier_Code	Supplier_Name	Current_State	Historical_State	Start_Date	End_Date	Current_Flag
123	ABC	Acme Supply Co	CA	CA	01-Jan-2000	31-Dec-9999	Y

The Current_State and the Historical_State are the same. The Current_Flag attribute indicates that this is the current or most recent record for this supplier.

When Acme Supply Company moves to Illinois, we add a new record, as in Type 2 processing:

Supplier_Key	Supplier_Code	Supplier_Name	Current_State	Historical_State	Start_Date	End_Date	Current_Flag
123	ABC	Acme Supply Co	IL	CA	01-Jan-2000	21-Dec-2004	N
124	ABC	Acme Supply Co	IL	IL	22-Dec-2004	31-Dec-9999	Y

We overwrite the Current_State information in the first record (Supplier_Key = 123) with the new information, as in Type 1 processing. We create a new record to track the changes, as in Type 2 processing. And we store the history in a second State column (Historical_State), which incorporates Type 3 processing.

For example if the supplier were to relocate again, we would add another record to the Supplier dimension, and we would overwrite the contents of the Current_State column:

Supplier_Key	Supplier_Code	Supplier_Name	Current_State	Historical_State	Start_Date	End_Date	Current_Flag
123	ABC	Acme Supply Co	NY	CA	01-Jan-2000	21-Dec-2004	N
124	ABC	Acme Supply Co	NY	IL	22-Dec-2004	03-Feb-2008	N
125	ABC	Acme Supply Co	NY	NY	04-Feb-2008	31-Dec-9999	Y

Note that, for the current record (Current_Flag = 'Y'), the Current_State and the Historical_State are always the same.

Type 2 / Type 6 fact implementation

Type 2 surrogate key with Type 3 attribute

In many Type 2 and Type 6 SCD implementations, the surrogate key from the dimension is put into the fact table in place of the natural key when the fact data is loaded into the data repository. The surrogate key is selected for a given fact record based on its effective date and the Start_Date and End_Date from the dimension table. This allows the fact data to be easily joined to the correct dimension data for the corresponding effective date.

Here is the Supplier table as we created it above using Type 6 Hybrid methodology:

Supplier_Key	Supplier_Code	Supplier_Name	Current_State	Historical_State	Start_Date	End_Date	Current_Flag
123	ABC	Acme Supply Co	NY	CA	01-Jan-2000	21-Dec-2004	N
124	ABC	Acme Supply Co	NY	IL	22-Dec-2004	03-Feb-2008	N
125	ABC	Acme Supply Co	NY	NY	04-Feb-2008	31-Dec-9999	Y

Once the Delivery table contains the correct Supplier_Key, it can easily be joined to the Supplier table using that key. The following SQL retrieves, for each fact record, the current supplier state and the state the supplier was located in at the time of the delivery:

```
SELECT
  delivery.delivery_cost,
  supplier.supplier_name,
  supplier.historical_state,
  supplier.current_state
FROM delivery
INNER JOIN supplier
  ON delivery.supplier_key = supplier.supplier_key
```

Pure Type 6 implementation

Having a Type 2 surrogate key for each time slice can cause problems if the dimension is subject to change.

A pure Type 6 implementation does not use this, but uses a Surrogate Key for each master data item (e.g. each unique supplier has a single surrogate key).

This avoids any changes in the master data having an impact on the existing transaction data.

It also allows more options when querying the transactions.

Here is the Supplier table using the pure Type 6 methodology:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Start_Date	End_Date
456	ABC	Acme Supply Co	CA	01-Jan-2000	21-Dec-2004
456	ABC	Acme Supply Co	IL	22-Dec-2004	03-Feb-2008
456	ABC	Acme Supply Co	NY	04-Feb-2008	31-Dec-9999

The following example shows how the query must be extended to ensure a single supplier record is retrieved for each transaction.

```
SELECT
  supplier.supplier_code,
  supplier.supplier_state
FROM supplier
INNER JOIN delivery
  ON supplier.supplier_key = delivery.supplier_key
 AND delivery.delivery_date >= supplier.start_date
 AND delivery.delivery_date <= supplier.end_date
```

A fact record with an effective date (Delivery_Date) of August 9, 2001 will be linked to Supplier_Code of ABC, with a Supplier_State of 'CA'. A fact record with an effective date of October 11, 2007 will also be linked to the same Supplier_Code ABC, but with a Supplier_State of 'IL'.

Whilst more complex, there are a number of advantages of this approach, including:

1. If there is more than one date on the fact (e.g. Order Date, Delivery Date, Invoice Payment Date) you can choose which date to use for a query.
2. You can do "as at now", "as at transaction time" or "as at a point in time" queries by changing the date filter logic.
3. You don't need to reprocess the Fact table if there is a change in the dimension table (e.g. adding additional fields retrospectively which change the time slices, or if you make a mistake in the dates on the dimension table you can correct them easily).
4. You can introduce bi-temporal dates in the dimension table.
5. You can join the fact to the multiple versions of the dimension table to allow reporting of the same information with different effective dates, in the same query.

The following example shows how a specific date such as '2012-01-01 00:00:00' (which could be the current datetime) can be used.

```
SELECT
  supplier.supplier_code,
  supplier.supplier_state
FROM supplier
INNER JOIN delivery
  ON supplier.supplier_key = delivery.supplier_key
```

```
 AND '2012-01-01 00:00:00' >= supplier.start_date
 AND '2012-01-01 00:00:00' <= supplier.end_date
```

Both surrogate and natural key

An alternative implementation is to place *both* the surrogate key and the natural key into the fact table. This allows the user to select the appropriate dimension records based on:

- the primary effective date on the fact record (above),
- the most recent or current information,
- any other date associated with the fact record.

This method allows more flexible links to the dimension, even if you have used the Type 2 approach instead of Type 6.

Here is the Supplier table as we might have created it using Type 2 methodology:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Start_Date	End_Date	Current_Flag
123	ABC	Acme Supply Co	CA	01-Jan-2000	21-Dec-2004	N
124	ABC	Acme Supply Co	IL	22-Dec-2004	03-Feb-2008	N
125	ABC	Acme Supply Co	NY	04-Feb-2008	31-Dec-9999	Y

The following SQL retrieves the most current Supplier_Name and Supplier_State for each fact record:

```
SELECT
  delivery.delivery_cost,
  supplier.supplier_name,
  supplier.supplier_state
FROM delivery
INNER JOIN supplier
  ON delivery.supplier_code = supplier.supplier_code
WHERE supplier.current_flag = 'Y'
```

If there are multiple dates on the fact record, the fact can be joined to the dimension using another date instead of the primary effective date. For instance, the Delivery table might have a primary effective date of Delivery_Date, but might also have an Order_Date associated with each record.

The following SQL retrieves the correct Supplier_Name and Supplier_State for each fact record based on the Order_Date:

```
SELECT
  delivery.delivery_cost,
  supplier.supplier_name,
  supplier.supplier_state
FROM delivery
INNER JOIN supplier
  ON delivery.supplier_code = supplier.supplier_code
 AND delivery.order_date >= supplier.start_date
 AND delivery.order_date <= supplier.end_date
```

Some cautions:

- If the join query is not written correctly, it may return duplicate rows and/or give incorrect answers.
- The date comparison might not perform well.

- Some Business Intelligence tools do not handle generating complex joins well.
- The ETL processes needed to create the dimension table needs to be carefully designed to ensure that there are no overlaps in the time periods for each distinct item of reference data.

Combining types

Different SCD Types can be applied to different columns of a table. For example, we can apply Type 1 to the Supplier_Name column and Type 2 to the Supplier_State column of the same table.

Notes

References

- Bruce Ottmann, Chris Angus: *Data processing system*, US Patent Office, Patent Number 7,003,504 (http://patft.uspto.gov/netacgi/nph-Parser?u=/netahtml/srchnum.htm&Sect1=PTO1&Sect2=HITOFF&p=1&r=1&l=50&f=G&d=PALL&s1=7003504.PN.&OS=PN/7003504&RS=PN/7003504). February 21, 2006
- Ralph Kimball:*Kimball University: Handling Arbitrary Restatements of History* (http://www.informationweek.com/showArticle.jhtml?articleID=204800027&pgno=1&queryText=). December 9, 2007

Star schema

In computing, the **Star Schema** (also called **star-join schema**) is the simplest style of data mart schema. The star schema consists of one or more fact tables referencing any number of dimension tables. The star schema is an important special case of the snowflake schema, and is more effective for handling simpler queries.

The star schema gets its name from the physical model's[1] resemblance to a star with a fact table at its center and the dimension tables surrounding it representing the star's points.

Model

The star schema separates business process data into facts, which hold the measurable, quantitative data about a business, and dimensions which are descriptive attributes related to fact data. Examples of fact data include sales price, sale quantity, and time, distance, speed, and weight measurements. Related dimension attribute examples include product models, product colors, product sizes, geographic locations, and salesperson names.

A star schema that has many dimensions is sometimes called a *centipede schema*.[2] Having dimensions of only a few attributes, while simpler to maintain, results in queries with many table joins and makes the star schema less easy to use.

Fact tables

Fact tables record measurements or metrics for a specific event. Fact tables generally consist of numeric values, and foreign keys to dimensional data where descriptive information is kept. Fact tables are designed to a low level of uniform detail (referred to as "granularity" or "grain"), meaning facts can record events at a very atomic level. This can result in the accumulation of a large number of records in a fact table over time. Fact tables are defined as one of three types:

- Transaction fact tables record facts about a specific event (e.g., sales events)
- Snapshot fact tables record facts at a given point in time (e.g., account details at month end)

- Accumulating snapshot tables record aggregate facts at a given point in time (e.g., total month-to-date sales for a product)

Fact tables are generally assigned a surrogate key to ensure each row can be uniquely identified.

Dimension tables

Dimension tables usually have a relatively small number of records compared to fact tables, but each record may have a very large number of attributes to describe the fact data. Dimensions can define a wide variety of characteristics, but some of the most common attributes defined by dimension tables include:

- Time dimension tables describe time at the lowest level of time granularity for which events are recorded in the star schema
- Geography dimension tables describe location data, such as country, state, or city
- Product dimension tables describe products
- Employee dimension tables describe employees, such as sales people
- Range dimension tables describe ranges of time, dollar values, or other measurable quantities to simplify reporting

Dimension tables are generally assigned a surrogate primary key, usually a single-column integer data type, mapped to the combination of dimension attributes that form the natural key.

Benefits

Star schemas are denormalized, meaning the normal rules of normalization applied to transactional relational databases are relaxed during star schema design and implementation. The benefits of star schema denormalization are:

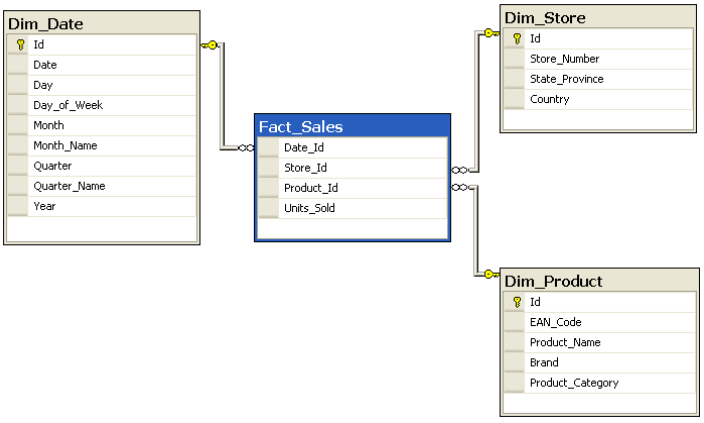
- Simpler queries - star schema join logic is generally simpler than the join logic required to retrieve data from a highly normalized transactional schemas.
- Simplified business reporting logic - when compared to highly normalized schemas, the star schema simplifies common business reporting logic, such as period-over-period and as-of reporting.
- Query performance gains - star schemas can provide performance enhancements for read-only reporting applications when compared to highly normalized schemas.
- Fast aggregations - the simpler queries against a star schema can result in improved performance for aggregation operations.
- Feeding cubes - star schemas are used by all OLAP systems to build proprietary OLAP cubes efficiently; in fact, most major OLAP systems provide a ROLAP mode of operation which can use a star schema directly as a source without building a proprietary cube structure.

Disadvantages

The main disadvantage of the star schema is that data integrity is not enforced as well as it is in a highly normalized database. One-off inserts and updates can result in data anomalies which normalized schemas are designed to avoid. Generally speaking, star schemas are loaded in a highly controlled fashion via batch processing or near-real time "trickle feeds", to compensate for the lack of protection afforded by normalization.

Example

Consider a database of sales, perhaps from a store chain, classified by date, store and product. The image of the schema to the right is a star schema version of the sample schema provided in the snowflake schema article.



Star schema used by example query.

`Fact_Sales` is the fact table and there are three dimension tables `Dim_Date`, `Dim_Store` and `Dim_Product`.

Each dimension table has a primary key on its `Id` column, relating to one of the columns (viewed as rows in the example schema) of the `Fact_Sales` table's three-column (compound) primary key (`Date_Id`, `Store_Id`, `Product_Id`). The non-primary key `Units_Sold` column of the fact table in this example represents a measure or metric that can be used in calculations and analysis. The non-primary key columns of the dimension tables represent additional attributes of the dimensions (such as the `Year` of the `Dim_Date` dimension).

For example, the following query answers how many TV sets have been sold, for each brand and country, in 1997:

```
SELECT
      P.Brand,
      S.Country as Countries,
      SUM(F.Units_Sold)
FROM Fact_Sales F
INNER JOIN Dim_Date D    ON F.Date_Id = D.Id
INNER JOIN Dim_Store S   ON F.Store_Id = S.Id
INNER JOIN Dim_Product P ON F.Product_Id = P.Id
WHERE
      D.Year = 1997
AND P.Product_Category = 'tv'
GROUP BY
      P.Brand,
      S.Country
```

References

[1] C J Date, "An Introduction to Database Systems (Eighth Edition)", p. 708

[2] Ralph Kimball and Margy Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)*, p. 393

External links

- Designing the Star Schema Database by Craig Utley (http://ciobriefings.com/Publications/WhitePapers/DesigningtheStarSchemaDatabase/tabid/101/Default.aspx)
- Stars: A Pattern Language for Query Optimized Schema (http://c2.com/ppr/stars.html)
- Star schema optimizations (http://www.dwoptimize.com/2007/06/aiming-for-stars.html)
- Fact constellation schema (http://datawarehouse4u.info/Data-warehouse-schema-architecture-fact-constellation-schema.html)
- Data Warehouses, Schemas and Decision Support Basics by Dan Power (http://www.b-eye-network.com/view/8451)

Dimension table

In data warehousing, a **dimension table** is one of the set of companion tables to a fact table.

The fact table contains business facts (or *measures*), and foreign keys which refer to candidate keys (normally primary keys) in the dimension tables.

Contrary to *fact* tables, *dimension* tables contain descriptive attributes (or fields) that are typically textual fields (or discrete numbers that behave like text). These attributes are designed to serve two critical purposes: query constraining and/or filtering, and query result set labeling.

Dimension attributes should be:

- Verbose (labels consisting of full words)
- Descriptive
- Complete (having no missing values)
- Discretely valued (having only one value per dimension table row)
- Quality assured (having no misspellings or impossible values)

Dimension table rows are uniquely identified by a single key field. It is recommended that the key field be a simple integer because a key value is meaningless, used only for joining fields between the fact and dimension tables.

The use of surrogate dimension keys brings several advantages, including:

- Performance. Join processing is made much more efficient by using a single field (the surrogate key)
- Buffering from operational key management practices. This prevents situations where removed data rows might reappear when their natural keys get reused or reassigned after a long period of dormancy
- Mapping to integrate disparate sources
- Handling unknown or not-applicable connections
- Tracking changes in dimension attribute values

Although surrogate key use places a burden put on the ETL system, pipeline processing can be improved, and ETL tools have built-in improved surrogate key processing.

The goal of a dimension table is to create standardized, conformed dimensions that can be shared across the enterprise's data warehouse environment, and enable joining to multiple fact tables representing various business processes.

Conformed dimensions are important to the enterprise nature of DW/BI systems because they promote:

- Consistency. Every fact table is filtered consistently, so that query answers are labeled consistently.

- Integration. Queries can drill into different process fact tables separately for each individual fact table, then join the results on common dimension attributes.
- Reduced development time to market. The common dimensions are available without recreating them.

Over time, the attributes of a given row in a dimension table may change. For example, the shipping address for a company may change. Kimball refers to this phenomenon as Slowly Changing Dimensions. Strategies for dealing with this kind of change are divided into three categories:

- Type One. Simply overwrite the old value(s).
- Type Two. Add a new row containing the new value(s), and distinguish between the rows using Tuple-versioning techniques.
- Type Three. Add a new attribute to the existing row.

References

- Kimball, Ralph. *The Data Warehouse Lifecycle Toolkit* Second Edition. Winely Publishing Inc., 2008, p.241-246.
- Kimball, Ralph et al. (1998); *The Data Warehouse Lifecycle Toolkit*, p17. Pub. Wiley. ISBN 0-471-25547-5.
- Kimball, Ralph (1996); *The Data Warehouse Toolkit*, p100. Pub. Wiley. ISBN 0-471-15337-0.

Dimensional Fact Model

The *Dimensional Fact Model* (DFM) is an ad hoc formalism specifically devised to support the conceptual modeling phase in a DW project.

Data Warehouses (DWs) are databases used by decision makers to analyze the status and the development of an organization. DWs are based on large amounts of data integrated from heterogeneous sources into multidimensional databases, and they are optimized for accessing data in a way that comes natural to human analysts (e.g., OLAP applications).

Data in a DW are organized according to the multidimensional model, that hinges on the concepts of fact (a focus of interest for the decision-making process, such as sales and orders) and dimension (a coordinate for analyzing a fact, such as time, customer, and product). Each fact is quantified through a set of numerical measures, such as the quantity of product sold, the price of products, etc.

DW design and development require ad hoc methodologies and an appropriate life-cycle.[][1][2]

Overview

The DFM is a graphical conceptual model, specifically devised for multidimensional design, in order to:

- lend effective support to conceptual design
- create an environment in which user queries may be formulated intuitively
- make communication possible between designers and end users with the goal of formalizing requirement specifications
- build a stable platform for logical design
- provide clear and expressive design documentation.

The conceptual representation generated by the DFM consists of a set of fact schemata. Fact schemata model facts, measures, dimensions, and hierarchies (see Figure 1). Besides these basic elements, the DFM includes a large set of constructs for expressing the multitude of conceptual nuances that characterize actual modeling scenarios in projects of small to large complexity. A multidimensional schema modeled with the DFM can easily (i.e., semi-automatically) be implemented on both ROLAP and MOLAP platforms.

Basic concepts

A *fact* is a concept relevant to decision-making processes. It typically models a set of events taking place within a company. Examples of facts in the commercial domain are sales, shipments, purchases, and complaints.

A *measure* is a numerical property of a fact and describes a quantitative attribute that is relevant to analysis. For example, each sale is measured by the number of units sold, the unit price, and the total receipts.

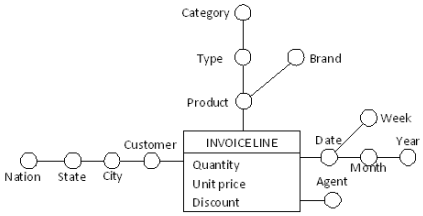


Figure 1: a simple fact schema for the invoice fact

A *dimension* is a property, with a finite domain, that describes an analysis coordinate of the fact. A fact generally has multiple dimensions that define its minimum representation granularity. Typical dimensions for the sales fact are products, stores, and dates; in which case, the basic information that can be represented is product sales in one store in one day.

A *fact* is represented by a box that displays the fact name along with the measure names. Small circles represent the dimensions, which are linked to the fact by straight lines (see Figure 1).

A *dimensional attribute* is a property, with a finite domain, of a dimension. Like dimensions, a dimensional attribute is represented by a circle. For instance, a product may be described by its type, category, and brand; a customer may be represented by city and nation. The relationships among the dimensional attributes are expressed by hierarchies.

A *hierarchy* is a directed tree whose nodes are dimensional attributes and whose arcs model many-to-one associations between dimensional attribute pairs. A hierarchy includes a dimension, positioned at the tree's root, and all of the dimensional attributes that describe it. Arcs are graphically represented by straight lines that connect dimensional attributes. Hierarchies define the way elemental business events can be selected and aggregated for decision-making processes.

Advanced concepts

A *descriptive attribute* specifies a property of a dimension attribute, to which it is related by a x-to-one association. Descriptive attributes cannot be used for aggregation; they are always leaves of a hierarchy and are graphically represented by horizontal lines, like Telephone number in Figure 2.

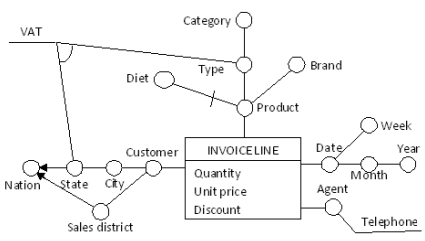


Figure 2: a more complex fact schema for the invoice fact

A *cross-dimensional attribute* is a dimensional or descriptive attribute whose value is defined by the combination of two or more dimensional attributes, possibly belonging to different hierarchies. For example, if a product value added tax (VAT) depends both on the product category and on the country where the product is sold, you can use a cross-dimensional attribute to represent it. Figure 2 shows this example by joining the arcs that define a product VAT with a circular arc.

A *convergence* takes place when two dimensional attributes within a hierarchy are connected by two or more alternative paths of many-to-one associations. Convergences are represented by letting two or more arcs reach the same dimensional attribute. For instance, in Figure 2 the geographic hierarchy on the customer dimension contains a convergence if we assume that, though no inclusion relationships exists between districts and cities/states, sales districts never cross the nation boundaries. In this case, each customer belongs to exactly one nation whichever of the two paths is followed.

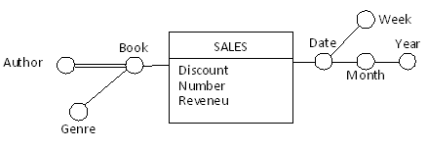


Figure 3: a fact schema for the book sales fact

Optional arcs are used to model scenarios for which an association represented in a fact schema is not defined for a subset of events. Optional arcs are marked with a dash. For instance, attribute *diet* in Figure 2 takes a value (such as cholesterol-free, gluten-free, or sugar-free) only for food products; for the other products, it is undefined.

A *multiple arc* models a many-to-many association between the two dimensional attributes it connects. Graphically, it is denoted by doubling the line that represents the arc. Consider the fact schema modeling the sales of books, represented in Figure 3, whose dimensions are date and book. It would certainly be interesting to aggregate and select sales on the basis of book authors. However, it would not be accurate to model author as a dimensional child attribute of book because a book may have more than one author, and authors can write more than one book. Hence, the relationship between books and authors is modeled as a multiple arc.

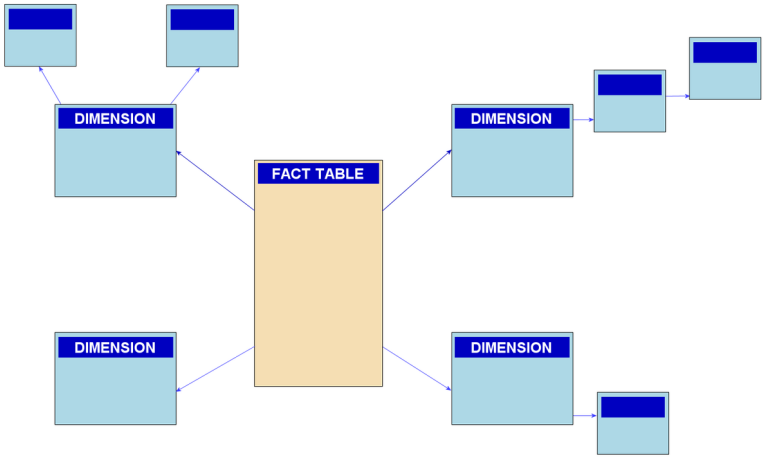
References

[1] (http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470479574.html) Ralph Kimball, et al.. The Data Warehouse Lifecycle Toolkit. Wiley, 2008.

[2] Oscar Romero, Alberto Abelló. A Survey of Multidimensional Modeling Methodologies. In Int. Journal on Data Warehousing and Mining (IJDWM), volume 5, number 2. Idea Group 2009. Pages 1-23. ISSN: 1548-3924

Snowflake schema

In computing, a **snowflake schema** is a logical arrangement of tables in a multidimensional database such that the entity relationship diagram resembles a snowflake shape. The snowflake schema is represented by centralized fact tables which are connected to multiple dimensions.[*citation needed*]. "Snowflaking" is a method of normalising the dimension tables in a star schema. When it is completely normalised along all the dimension tables, the resultant structure resembles a snowflake with the fact table in the middle. The principle behind snowflaking is normalisation of the dimension tables by removing low cardinality attributes and forming separate tables.[1]



The snowflake schema is a variation of the star schema, featuring normalization of dimension tables.

The snowflake schema is similar to the star schema. However, in the snowflake schema, dimensions are normalized into multiple related tables, whereas the star schema's dimensions are denormalized with each dimension represented by a single table. A complex snowflake shape emerges when the dimensions of a snowflake schema are elaborate, having multiple levels of relationships, and the child tables have multiple parent tables ("forks in the road").

Common uses

Star and snowflake schemas are most commonly found in dimensional data warehouses and data marts where speed of data retrieval is more important than the efficiency of data manipulations. As such, the tables in these schemas are not normalized much, and are frequently designed at a level of normalization short of third normal form.[*citation needed*]

Deciding whether to employ a star schema or a snowflake schema should involve considering the relative strengths of the database platform in question and the query tool to be employed. Star schemas should be favored with query tools that largely expose users to the underlying table structures, and in environments where most queries are simpler in nature. Snowflake schemas are often better with more sophisticated query tools that create a layer of abstraction between the users and raw table structures for environments having numerous queries with complex criteria.[*citation needed*]

Data normalization and storage

Normalization splits up data to avoid redundancy (duplication) by moving commonly repeating groups of data into new tables. Normalization therefore tends to increase the number of tables that need to be joined in order to perform a given query, but reduces the space required to hold the data and the number of places where it needs to be updated if the data changes.[*citation needed*]

From a space storage point of view, the dimensional tables are typically small compared to the fact tables. This often removes the storage space benefit of snowflaking the dimension tables, as compared with a star schema.[*citation needed*]

Some database developers compromise by creating an underlying snowflake schema with views built on top of it that perform many of the necessary joins to simulate a star schema. This provides the storage benefits achieved through the normalization of dimensions with the ease of querying that the star schema provides. The tradeoff is that requiring the server to perform the underlying joins automatically can result in a performance hit when querying as well as extra joins to tables that may not be necessary to fulfill certain queries.[*citation needed*]

Benefits

The snowflake schema is in the same family as the star schema logical model. In fact, the star schema is considered a special case of the snowflake schema. The snowflake schema provides some advantages over the star schema in certain situations, including:

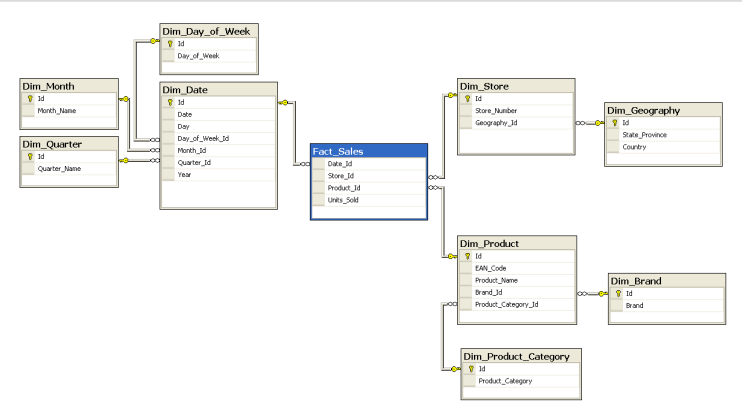
- Some OLAP multidimensional database modeling tools are optimized for snowflake schemas.
- Normalizing attributes results in storage savings, the tradeoff being additional complexity in source query joins.

Disadvantages

The primary disadvantage of the snowflake schema is that the additional levels of attribute normalization adds complexity to source query joins, when compared to the star schema. When compared to a highly normalized transactional schema, the snowflake schema's denormalization removes the data integrity assurances provided by normalized schemas. Data loads into the snowflake schema must be highly controlled and managed to avoid update and insert anomalies.

Examples

The example schema shown to the right is a snowflaked version of the star schema example provided in the star schema article.[*citation needed*]



Snowflake schema used by example query.

The following example query is the snowflake schema equivalent of the star schema example code which returns the total number of units sold by brand and by country for 1997. Notice that the snowflake schema query requires many more joins than the star schema version in order to fulfill even a simple query. The benefit of using the snowflake schema in this example is that the storage requirements are lower since the snowflake schema eliminates many duplicate values from the dimensions themselves.[*citation needed*]

```
SELECT
      B.Brand,
      G.Country,
      SUM(F.Units_Sold)
FROM Fact_Sales F
INNER JOIN Dim_Date D             ON F.Date_Id = D.Id
INNER JOIN Dim_Store S            ON F.Store_Id = S.Id
INNER JOIN Dim_Geography G        ON S.Geography_Id = G.Id
INNER JOIN Dim_Product P          ON F.Product_Id = P.Id
INNER JOIN Dim_Brand B            ON P.Brand_Id = B.Id
INNER JOIN Dim_Product_Category C ON P.Product_Category_Id = C.Id
WHERE
      D.Year = 1997 AND
      C.Product_Category = 'tv'
GROUP BY
      B.Brand,
      G.Country
```

References

[1] Paulraj Ponniah. *Data Warehousing Fundamentals for IT Professionals.* Wiley, 2010, pp. 29–32. ISBN 0470462078.

Paulraj Ponniah. *Data Warehousing Fundamentals for IT Professionals.* Wiley, 2010, pp. 29–32. ISBN 0470462078.

Bibliography

- Anahory, S.; D. Murray. *Data Warehousing in the Real World: A Practical Guide for Building Decision Support Systems*. Addison Wesley Professional.
- Kimball, Ralph (1996). *The Data Warehousing Toolkit*. John Wiley.

External links

- " Why is the Snowflake Schema a Good Data Warehouse Design? (http://www.dcs.bbk.ac.uk/~mark/download/star.pdf)" by Mark Levene and George Loizou
- Reverse Snowflake Joins (http://sourceforge.net/projects/revj/)

Denormalization

In computing, **denormalization** is the process of attempting to optimize the read performance of a database by adding redundant data or by grouping data.[1][2] In some cases, denormalization is a means of addressing performance or scalability in relational database software.

A normalized design will often store different but related pieces of information in separate logical tables (called relations). If these relations are stored physically as separate disk files, completing a database query that draws information from several relations (a *join operation*) can be slow. If many relations are joined, it may be prohibitively slow. There are two strategies for dealing with this. The preferred method is to keep the logical design normalized, but allow the database management system (DBMS) to store additional redundant information on disk to optimise query response. In this case it is the DBMS software's responsibility to ensure that any redundant copies are kept consistent. This method is often implemented in SQL as indexed views (Microsoft SQL Server) or materialised views (Oracle). A view represents information in a format convenient for querying, and the index ensures that queries against the view are optimised.

The more usual approach is to denormalize the logical data design. With care this can achieve a similar improvement in query response, but at a cost—it is now the database designer's responsibility to ensure that the denormalized database does not become inconsistent. This is done by creating rules in the database called *constraints*, that specify how the redundant copies of information must be kept synchronised. It is the increase in logical complexity of the database design and the added complexity of the additional constraints that make this approach hazardous. Moreover, constraints introduce a trade-off, speeding up reads (`SELECT` in SQL) while slowing down writes (`INSERT`, `UPDATE`, and `DELETE`). This means a denormalized database under heavy write load may actually offer *worse* performance than its functionally equivalent normalized counterpart.

A denormalized data model is not the same as a data model that has not been normalized, and denormalization should only take place after a satisfactory level of normalization has taken place and that any required constraints and/or rules have been created to deal with the inherent anomalies in the design. For example, all the relations are in third normal form and any relations with join and multi-valued dependencies are handled appropriately.

Examples of denormalization techniques include:

- Materialised views, which may implement the following:
  - Storing the count of the "many" objects in a one-to-many relationship as an attribute of the "one" relation
  - Adding attributes to a relation from another relation with which it will be joined

- Star schemas, which are also known as fact-dimension models and have been extended to snowflake schemas
- Prebuilt summarisation or OLAP cubes

Denormalization techniques are often used to improve the scalability of Web applications.[3]

References

[1] G. L. Sanders and S. K. Shin. Denormalization effects on performance of RDBMS (http://www.hicss.hawaii.edu/HICSS_34/PDFs/DTDMK04.pdf). In Proceedings of the HICSS Conference, January 2001.

[2] S. K. Shin and G. L. Sanders. Denormalization strategies for data retrieval from data warehouses (http://portal.acm.org/citation.cfm?id=1217757). Decision Support Systems, 42(1):267-282, October 2006.

[3] Z. Wei, J. Dejun, G. Pierre, C.-H. Chi and M. van Steen. Service-Oriented Data Denormalization for Scalable Web Applications (http://www.globule.org/publi/SODDSWA_www2008.html). In Proceedings of the International World-Wide Web conference, April 2008.

Single version of the truth

In computerized business management, **SVOT**, or **Single Version of the Truth**, is a technical concept describing the data warehousing ideal of having either a single centralised database, or at least a distributed synchronised database, which stores all of an organisation's data in a consistent and non-redundant form. This contrasts with the related concept of Single Source of Truth (SSOT), which refers to is a data storage principle to always source a particular piece of information from one place.[*citation needed*]

SVOT applied to message sequencing

In some systems and in the context of message processing systems (often realtime systems), this term also refers to the goal of establishing a single agreed sequence of messages within a database formed by a particular but arbitrary sequencing of records. The key concept is that data combined in a certain sequence is a "truth" which may be analyzed and processed giving particular results, and that although the sequence is arbitrary (and thus another correct but equally arbitrary sequencing would ultimately provide different results in any analysis), it is desirable to agree that the sequence enshrined in the "single version of the truth" is the version that will be considered "the truth", and that any conclusions drawn from analysis of the database are valid and unarguable, and (in a technical context) the database may be duplicated to a backup environment to ensure a persistent record is kept of the "single version of the truth".

The key point is when the database is created using an external data source (such as a sequence of trading messages from a stock exchange) an arbitrary selection is made of one possibility from two or more equally valid representations of the input data, but henceforth the decision sets "in stone" one and only one version of the truth.

Critique of SVOT as applied to message sequencing

Critics of "SVOT as applied to message sequencing" argue that this concept is not scalable. As the world moves towards systems spread over many processing nodes, the effort involved in negotiating a single agreed-upon sequence becomes prohibitive.

References

Bibliography

- Julia King (2003-12-22). "Business Intelligence: One Version of the Truth" (http://www.computerworld.com/databasetopics/data/story/0,10801,88349,00.html). *ComputerWorld*.

- Leslie Lamport (July 1978). "Time, Clocks, and the Ordering of Events in a Distributed System" (http://research.microsoft.com/en-us/um/people/lamport/pubs/time-clocks.pdf). *CACM: Operating Systems*.
- Bill Inmon (2004-09-09). "The Single Version Of The Truth" (http://www.b-eye-network.com/view/282). *Business Intelligence Network* (Powell Media LLC).
- http://www.industryweek.com/EventDetail.aspx?EventID=179
- Chisholm, Malcolm (December 2006), "There is No Single Version of the Truth" (http://www.information-management.com/issues/20061201/1069851-1.html), *Information Management Magazine*, retrieved 2010-03-15

Transaction data

Transaction data are data describing an event (the change as a result of a transaction) and is usually described with verbs. Transaction data always has a time dimension, a numerical value and refers to one or more objects (i.e. the reference data).

Typical transactions are:

- Financial: orders, invoices, payments
- Work: Plans, activity records
- Logistics: Deliveries, storage records, travel records, etc.

Typical transaction processing systems (systems generating transactions) are SAP and Oracle Financials.

Records management

Recording and retaining transactions is called records management. The record of the transaction is stored in a place where the retention can be guaranteed and where data are archived/removed following a retention period. The format of the transaction can be data (to be stored in a database), but it can also be a document.

Data Warehousing

Transaction data can be summarised in a Data warehouse, which helps accessibility and analysis of the data.

Enterprise bus matrix

The *'Enterprise Bus Matrix* is a Data Warehouse planning tool and model created by Ralph Kimball, and is part of the Data Warehouse Bus Architecture. The Matrix is the logical definition of one of the core concepts of Kimball's approach to Dimensional Modeling – Conformed dimensions.[1]

The Bus Matrix defines part of the Data Warehouse Bus Architecture and is an output of the Business Requirements phase in The Kimball Lifecycle. It is applied in the following phases of dimensional modeling and development of the Data Warehouse . The matrix can be categorized as a hybrid model, being part technical design tool, part project management tool and part communication tool [2]

Background

The Enterprise Bus Matrix stems from the issue of how one goes about creating the overall Data Warehouse environment. Historically there has been the structure of the centralized and planned approach and the more loosely defined, department specific, solutions developed in a more independent matter. Autonomous projects can result in a range of isolated stove pipe data marts. Naturally each approach has its issues; the overall visionary approach often struggles with long delivery cycles and lack of reaction time as the formalities and scope issues is evident. On the other hand the development of isolated data marts, leading to Stovepipe systems that lacks synergy in development. Over time this approach will lead to a so-called data-mart-in-a-box architecture [3] where interoperability and lack of cohesion is apparent, and can hinder the realization of an overall enterprise Data Warehouse. As an attempt to handle this matter Ralph Kimball introduced the enterprise bus.

The Bus Matrix

In short words the bus matrix purpose is one of high abstraction and visionary planning on the Data Warehouse architectural level. By dictating coherency in the development and implementation of an overall Data Warehouse the Bus Architecture approach enables an overall vision of the broader enterprise integration and consistency while at the same time dividing the problem into more manageable parts – all in a technology and software independent manner .[4]

The bus matrix and architecture builds upon the concept of conformed dimensions - creating a structure of common dimensions that ideally can be used across the enterprise by all business processes related to the DW and the corresponding fact tables from which they derive their context. According to Kimball and Marg Rosses article "Differences of Opinion" [5] "*The Enterprise Data warehouse built on the bus architecture "identifies and enforces the relationship between business process metrics (facts) and descriptive attributes (dimensions)*".

The concept of a bus[6] is well known in the language of Information Technology, and is what reflects the conformed dimension concept in the Data Warehouse, creating the skeletal structure where all parts of a system connect, ensuring interoperability and consistency of data, and at the same time considers future expansion. This makes the conformed dimensions act as the integration 'glue', creating a robust backbone of the enterprise Data Warehouse.[7]

Establishment and applicability

Figure 1[8] shows the base for a single document planning tool for the whole of the DW implementation - a graphical overview of the enterprises core business processes or events each correspond to a measurement table of facts, that typically is complemented by a major source system in the horizontal rows. In the vertical columns the groups of contextual data is found as the common, conformed dimensions.

In this way the shared dimensions are defined, as each process indicates what dimensions it applies to through the cells figure 2. By this definition and coordination of conformed dimensions and processes the development of the overall data DW bus architecture is realized. The matrix identifies the shared dimensions related to processes and fact tables, and can be a tool for planning, prioritizing what needs to be approached, coordinating implementation and communicating the importance for conformed dimensions .

Kimball extends the matrix bus in detail as seen in figure 3 by introducing the other steps of the Datawarehouse Methodology; The Fact tables, Granularity, and at last the description of the needed facts. description of the fact tables, granularity and fact instances of each process, structuring and specifying what is needed across the enterprise in a more specific matter, further exemplifying how the matrix can be used as a planning too.

References

[1] http://www.kimballgroup.com/2003/09/15/design-tip-49-off-the-bench/
[2] Kimball, Ralph & Ross, Margy; The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition John Wiley & Sons, 2002
[3] http://www.mimno.com/avoiding-mistakes3.html#6
[4] http://www.b-eye-network.com/view/713
[5] http://intelligent-enterprise.informationweek.com/showArticle.jhtml;jsessionid=0OVJNEHMPRXGRQE1GHRSKH4ATMY32JVN?articleID=17800088
[6] http://en.wikipedia.org/wiki/Bus_(computing)
[7] http://intelligent-enterprise.informationweek.com/showArticle.jhtml;jsessionid=GMS3H4SOBFQBBQE1GHOSKH4ATMY32JVN?articleID=17800088&pgno=2
[8] http://www.widama.us/Documents/Kimball-DimensionalModeling.PDF

OLAP

Statistical database

A **statistical database** is a database used for statistical analysis purposes. It is an OLAP (online analytical processing), instead of OLTP (online transaction processing) system. Modern decision, and classical statistical databases are often closer to the relational model than the multidimensional model commonly used in OLAP systems today.

Statistical databases typically contain parameter data and the measured data for these parameters. For example, parameter data consists of the different values for varying conditions in an experiment (e.g., temperature, time). The measured data (or variables) are the measurements taken in the experiment under these varying conditions.

Many statistical databases are sparse with many null or zero values. It is not uncommon for a statistical database to be 40% to 50% sparse. There are two options for dealing with the sparseness: (1) leave the null values in there and use compression techniques to squeeze them out or (2) remove the entries that only have null values.

Statistical databases often incorporate support for advanced statistical analysis techniques, such as correlations, which go beyond SQL. They also pose unique security concerns, which were the focus of much research, particularly in the late 1970s and early to mid-1980s.

Security in statistical databases

In a statistical database, it is often desired to allow query access only to aggregate data, not individual records. Securing such a database is a difficult problem, since intelligent users can use a combination of aggregate queries to derive information about a single individual.

Some common approaches are:

- only allowing aggregate queries (SUM, COUNT, AVG, STDEV, etc.)
- rather than returning exact values for sensitive data like income, only return which partition it belongs to (e.g. 35k-40k)
- return imprecise counts (e.g. rather than 141 records met query, only indicate 130-150 records met it.)
- don't allow overly selective WHERE clauses
- audit all users queries, so users using system incorrectly can be investigated
- use intelligent agents to detect automatically inappropriate system use

Research in this area has largely stalled; reference 3 below showed that, in general, securing statistical databases was an impossible aim: if they were open to legitimate use, they were also open to abuse; and if they were restricted so tightly as to be incapable of abuse, they would then be useless for practical statistical purposes. To quote:

> The conclusion is that statistical databases are almost always subject to compromise. Severe restrictions on allowable query set sizes will render the database useless as a source of statistical information but will not secure the confidential records.[1]

Some further reading

Statistical and Scientific Database Management (SSDBM) [2] An important series of conferences in this field

Some key papers in this field:

1. doi:10.1145/320613.320616 [3] - Dorothy E. Denning, Secure statistical databases with random sample queries, ACM Transactions on Database Systems (TODS), Volume 5, Issue 3 (September 1980), Pages: 291 - 315
2. doi:10.1145/319830.319834 [4] - Wiebren de Jonge, Compromising statistical databases responding to queries about means, ACM Transactions on Database Systems, Volume 8, Issue 1 (March 1983), Pages: 60 - 80
3. doi:10.1145/320128.320138 [5] - Dorothy E. Denning, Jan Schlörer, A fast procedure for finding a tracker in a statistical database, ACM Transactions on Database Systems, Volume 5, Issue 1 (March 1980) . Pages: 88 - 102
4. A. Shoshani, “Statistical Databases: Characteristics, Problems, and some Solutions,” in Proceedings of the 8th International Conference on Very Large Data Bases, San Francisco, CA, USA, 1982, pp. 208–222.

References

[1] Dorothy E. Denning, Peter J. Denning, and Mayer D. Schwartz, "The Tracker: A Threat to Statistical Database Security," *ACM Transactions on Database Systems (TODS),* Volume 4, Issue 1 (March 1979), Pages: 76 - 96, .

[2] http://www.informatik.uni-trier.de/~ley/db/conf/ssdbm/

[3] http://dx.doi.org/10.1145%2F320613.320616

[4] http://dx.doi.org/10.1145%2F319830.319834

[5] http://dx.doi.org/10.1145%2F320128.320138

Online transaction processing

Online transaction processing, or **OLTP**, is a class of information systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing. The term is somewhat ambiguous; some understand a "transaction" in the context of computer or database transactions, while others (such as the Transaction Processing Performance Council) define it in terms of business or commercial transactions.[1] OLTP has also been used to refer to processing in which the system responds immediately to user requests. An automated teller machine (ATM) for a bank is an example of a commercial transaction processing application.

Requirements

Online transaction processing increasingly requires support for transactions that span a network and may include more than one company. For this reason, new online transaction processing software uses client or server processing and brokering software that allows transactions to run on different computer platforms in a network.

In large applications, efficient OLTP may depend on sophisticated transaction management software (such as CICS) and/or database optimization tactics to facilitate the processing of large numbers of concurrent updates to an OLTP-oriented database.

For even more demanding decentralized database systems, OLTP brokering programs can distribute transaction processing among multiple computers on a network. OLTP is often integrated into service-oriented architecture (SOA) and Web services.

Online Transaction Processing (OLTP) involves gathering input information, processing the information and updating existing information to reflect the gathered and processed information. As of today, most organizations use a database management system to support OLTP. OLTP is carried in a client server system.

Benefits

Online Transaction Processing has two key benefits: simplicity and efficiency. Reduced paper trails and the faster, more accurate forecasts for revenues and expenses are both examples of how OLTP makes things simpler for businesses.

Disadvantages

Additionally, like many modern online information technology solutions, some systems require offline maintenance which further affects the cost-benefit analysis.

Contrasted to

- Batch Processing
- Grid Computing

References

[1] Transaction Processing Performance Council website (http://www.tpc.org/)

External links

- H-Store Project (http://hstore.cs.brown.edu) (architectural and application shifts affecting OLTP performance)
- IBM CICS official website (http://www.ibm.com/cics)
- Transaction Processing Performance Council (http://www.tpc.org/)
- OLTP Schema (http://dbms.knowledgehills.com/What-is-Online-Transaction-Processing-(OLTP)-Schema/a32p2)
- Transaction Processing: Concepts & Techniques Management (http://www.amazon.com/dp/1558601902)

Operational system

An **operational system** is a term used in data warehousing to refer to a system that is used to process the day-to-day transactions of an organization. These systems are designed in a manner that processing of day-to-day transactions is performed efficiently and the integrity of the transactional data is preserved.

Synonyms

Sometimes operational systems are referred to as operational databases, transaction processing systems, or online transaction processing systems (OLTP). However, the use of the last two terms as synonyms may be confusing, because operational systems can be batch processing systems as well.

Any Enterprise must necessarily maintain a lot of data about its operation. This is its "Operational Data".

Organization	Probably
Manufacturing Company	Product data
Bank	Account Data
Hospital	Patient Data
University	Student Data
Government Department	Planning data

Operational data store

An **operational data store** (or "**ODS**") is a database designed to integrate data from multiple sources for additional operations on the data. The data is then passed back to operational systems for further operations and to the data warehouse for reporting.

Because the data originates from multiple sources, the integration often involves cleaning, resolving redundancy and checking against business rules for integrity. An ODS is usually designed to contain low-level or atomic (indivisible) data (such as transactions and prices) with limited history that is captured "real time" or "near real time" as opposed to the much greater volumes of data stored in the data warehouse generally on a less-frequent basis.

General Use

The general purpose of an ODS is to integrate data from disparate source systems in a single structure, using data integration technologies like data virtualization, data federation, or extract, transform, and load. This will allow operational access to the data for operational reporting, master data or reference data management.

An ODS is not a replacement or substitute for a data warehouse but in turn could become a source.

Publications

- Inmon, William (1999). *Building the Operational Data Store* (2nd ed.). New York: John Wiley & Sons. ISBN 0-471-32888-X.

External links

- ODS Architecture Patterns (EA Reference Architecture)
- Bill Inmon Information Management article on ODS [1]
- Bill Inmon Information Management article on the five classes of ODS [2]
- Claudia Imhoff Information Management article on ODS [3]

References

[1] http://www.dmreview.com/issues/19980701/469-1.html
[2] http://www.information-management.com/issues/20000101/1749-1.html
[3] http://www.dmreview.com/issues/20000701/2361-1.html

Operational database

An **operational database** stores information about the activities of an organization, for example customer relationship management transactions, in a computer database. Operational databases can be known as **production database**, especially when distinguishing from a "test database" or "development database" used by programmers.

Use in business

Operational databases allow a business to enter, gather, and retrieve large quantities of specific information, such as training status, personal employee information, sales, customer complaints, and previous proposal information. Storing information in a centralized area can increase retrieval time for users. An important feature of storing information in an operational database is the ability to share information across the company. Operational databases can be used to monitor activities, to audit suspicious transactions, or to review the history of dealings with a particular customer. They can also be part of the actual process of making and fulfilling a purchase, for example in e-commerce.

Data warehouse terminology

In data warehousing, the term is even more specific: the operational database is the one which is accessed by an operational system (for example a customer-facing website or the application used by the customer service department) to carry out regular operations of an organization. Operational databases usually use an online transaction processing database which is optimized for faster transaction processing (create, read, update and delete operations).

The contents of the data warehouse are deleted from the operational database, but not necessarily updated in real time (if the machines are separate). Data warehouses tend to be optimized for faster read-only queries as an online analytical processing database to be used for back-office applications like business intelligence.

References

- O'Brien, J., and Marakas, G., (2008). Management Information Systems. Computer Software (pp. 185). New York, New York: McGraw-Hill

Online analytical processing

In computing, **online analytical processing**, or **OLAP** /ˈoʊlæp/, is an approach to answering multi-dimensional analytical (MDA) queries swiftly. OLAP is part of the broader category of business intelligence, which also encompasses relational database, report writing and data mining. Typical applications of OLAP include business reporting for sales, marketing, management reporting, business process management (BPM), budgeting and forecasting, financial reporting and similar areas, with new applications coming up, such as agriculture. The term *OLAP* was created as a slight modification of the traditional database term OLTP (Online Transaction Processing).

OLAP tools enable users to analyze multidimensional data interactively from multiple perspectives. OLAP consists of three basic analytical operations: consolidation (roll-up), drill-down, and slicing and dicing.[1] Consolidation involves the aggregation of data that can be accumulated and computed in one or more dimensions. For example, all sales offices are rolled up to the sales department or sales division to anticipate sales trends. By contrast, the drill-down is a technique that allows users to navigate through the details. For instance, users can view the sales by individual products that make up a region's sales. Slicing and dicing is a feature whereby users can take out (slicing) a specific set of data of the OLAP cube and view (dicing) the slices from different viewpoints.

Databases configured for OLAP use a multidimensional data model, allowing for complex analytical and ad hoc queries with a rapid execution time. They borrow aspects of navigational databases, hierarchical databases and relational databases.

Overview of OLAP systems

The core of any OLAP system is an OLAP cube (also called a 'multidimensional cube' or a hypercube). It consists of numeric facts called *measures* which are categorized by *dimensions*. The measures are placed at the intersections of the hypercube, which is spanned by the dimensions as a Vector space. The usual interface to manipulate an OLAP cube is a matrix interface like Pivot tables in a spreadsheet program, which performs projection operations along the dimensions, such as aggregation or averaging.

The cube metadata is typically created from a star schema or snowflake schema or fact constellation of tables in a relational database. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables.

Each *measure* can be thought of as having a set of *labels*, or meta-data associated with it. A *dimension* is what describes these *labels*; it provides information about the *measure*.

A simple example would be a cube that contains a store's sales as a *measure*, and Date/Time as a *dimension*. Each Sale has a Date/Time *label* that describes more about that sale.

Any number of *dimensions* can be added to the structure such as Store, Cashier, or Customer by adding a foreign key column to the fact table. This allows an analyst to view the *measures* along any combination of the *dimensions*.

For example:

```
 Sales Fact Table
+-------------+----------+
| sale_amount | time_id  |
+-------------+----------+            Time Dimension
|      2008.10|     1234 |---+     +---------+-------------------+
+-------------+----------+   |     | time_id | timestamp         |
                             |     +---------+-------------------+
                             +---->|   1234  | 20080902 12:35:43 |
                                   +---------+-------------------+
```

Multidimensional databases

Multidimensional structure is defined as “a variation of the relational model that uses multidimensional structures to organize data and express the relationships between data”.[2] The structure is broken into cubes and the cubes are able to store and access data within the confines of each cube. “Each cell within a multidimensional structure contains aggregated data related to elements along each of its dimensions”.[3] Even when data is manipulated it remains easy to access and continues to constitute a compact database format. The data still remains interrelated. Multidimensional structure is quite popular for analytical databases that use online analytical processing (OLAP) applications.[4] Analytical databases use these databases because of their ability to deliver answers to complex business queries swiftly. Data can be viewed from different angles, which gives a broader perspective of a problem unlike other models.[5]

Aggregations

It has been claimed that for complex queries OLAP cubes can produce an answer in around 0.1% of the time required for the same query on OLTP relational data. The most important mechanism in OLAP which allows it to achieve such performance is the use of *aggregations*. Aggregations are built from the fact table by changing the granularity on specific dimensions and aggregating up data along these dimensions. The number of possible aggregations is determined by every possible combination of dimension granularities.

The combination of all possible aggregations and the base data contains the answers to every query which can be answered from the data.

Because usually there are many aggregations that can be calculated, often only a predetermined number are fully calculated; the remainder are solved on demand. The problem of deciding which aggregations (views) to calculate is known as the view selection problem. View selection can be constrained by the total size of the selected set of aggregations, the time to update them from changes in the base data, or both. The objective of view selection is typically to minimize the average time to answer OLAP queries, although some studies also minimize the update time. View selection is NP-Complete. Many approaches to the problem have been explored, including greedy algorithms, randomized search, genetic algorithms and A* search algorithm.

Types

OLAP systems have been traditionally categorized using the following taxonomy.

Multidimensional

MOLAP is a "multi-dimensional online analytical processing". 'MOLAP' is the 'classic' form of OLAP and is sometimes referred to as just OLAP. MOLAP stores this data in an optimized multi-dimensional array storage, rather than in a relational database. Therefore it requires the pre-computation and storage of information in the cube - the operation known as processing. MOLAP tools generally utilize a pre-calculated data set referred to as a data cube. The data cube contains all the possible answers to a given range of questions. MOLAP tools have a very fast response time and the ability to quickly write back data into the data set.

Advantages of MOLAP

- Fast query performance due to optimized storage, multidimensional indexing and caching.
- Smaller on-disk size of data compared to data stored in relational database due to compression techniques.
- Automated computation of higher level aggregates of the data.
- It is very compact for low dimension data sets.
- Array models provide natural indexing.
- Effective data extraction achieved through the pre-structuring of aggregated data.

Disadvantages of MOLAP

- Within some MOLAP Solutions the processing step (data load) can be quite lengthy, especially on large data volumes. This is usually remedied by doing only incremental processing, i.e., processing only the data which have changed (usually new data) instead of reprocessing the entire data set.
- MOLAP tools traditionally have difficulty querying models with dimensions with very high cardinality (i.e., millions of members).
- Some MOLAP products have difficulty updating and querying models with more than ten dimensions. This limit differs depending on the complexity and cardinality of the dimensions in question. It also depends on the number of facts or measures stored. Other MOLAP products can handle hundreds of dimensions.
- Some MOLAP methodologies introduce data redundancy.

Relational

ROLAP works directly with relational databases. The base data and the dimension tables are stored as relational tables and new tables are created to hold the aggregated information. Depends on a specialized schema design. This methodology relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP's slicing and dicing functionality. In essence, each action of slicing and dicing is equivalent to adding a "WHERE" clause in the SQL statement. ROLAP tools do not use pre-calculated data cubes but instead pose the query to the standard relational database and its tables in order to bring back the data required to answer the question. ROLAP tools feature the ability to ask any question because the methodology does not limit to the contents of a cube. ROLAP also has the ability to drill down to the lowest level of detail in the database.

Hybrid

There is no clear agreement across the industry as to what constitutes "Hybrid OLAP", except that a database will divide data between relational and specialized storage. For example, for some vendors, a HOLAP database will use relational tables to hold the larger quantities of detailed data, and use specialized storage for at least some aspects of the smaller quantities of more-aggregate or less-detailed data. HOLAP addresses the shortcomings of MOLAP and ROLAP by combining the capabilities of both approaches. HOLAP tools can utilize both pre-calculated cubes and relational data sources.

Comparison

Each type has certain benefits, although there is disagreement about the specifics of the benefits between providers.

- Some MOLAP implementations are prone to database explosion, a phenomenon causing vast amounts of storage space to be used by MOLAP databases when certain common conditions are met: high number of dimensions, pre-calculated results and sparse multidimensional data.
- MOLAP generally delivers better performance due to specialized indexing and storage optimizations. MOLAP also needs less storage space compared to ROLAP because the specialized storage typically includes compression techniques.
- ROLAP is generally more scalable. However, large volume pre-processing is difficult to implement efficiently so it is frequently skipped. ROLAP query performance can therefore suffer tremendously.
- Since ROLAP relies more on the database to perform calculations, it has more limitations in the specialized functions it can use.
- HOLAP encompasses a range of solutions that attempt to mix the best of ROLAP and MOLAP. It can generally pre-process swiftly, scale well, and offer good function support.

Other types

The following acronyms are also sometimes used, although they are not as widespread as the ones above:

- **WOLAP** - Web-based OLAP
- **DOLAP** - **Desktop** OLAP
- **RTOLAP** - Real-Time OLAP

APIs and query languages

Unlike relational databases, which had SQL as the standard query language, and widespread APIs such as ODBC, JDBC and OLEDB, there was no such unification in the OLAP world for a long time. The first real standard API was OLE DB for OLAP specification from Microsoft which appeared in 1997 and introduced the MDX query language. Several OLAP vendors - both server and client - adopted it. In 2001 Microsoft and Hyperion announced the XML for Analysis specification, which was endorsed by most of the OLAP vendors. Since this also used MDX as a query language, MDX became the de facto standard. Since September-2011 LINQ can be used to query SSAS OLAP cubes from Microsoft .NET.

Products

History

The first product that performed OLAP queries was *Express,* which was released in 1970 (and acquired by Oracle in 1995 from Information Resources). However, the term did not appear until 1993 when it was coined by Edgar F. Codd, who has been described as "the father of the relational database". Codd's paper resulted from a short consulting assignment which Codd undertook for former Arbor Software (later Hyperion Solutions, and in 2007 acquired by Oracle), as a sort of marketing coup. The company had released its own OLAP product, *Essbase*, a year earlier. As a result Codd's "twelve laws of online analytical processing" were explicit in their reference to Essbase. There was some ensuing controversy and when Computerworld learned that Codd was paid by Arbor, it retracted the article. OLAP market experienced strong growth in late 90s with dozens of commercial products going into market. In 1998, Microsoft released its first OLAP Server - Microsoft Analysis Services, which drove wide adoption of OLAP technology and moved it into mainstream.

Market structure

Below is a list of top OLAP vendors in 2006, with figures in millions of US Dollars.

Vendor	Global Revenue	Consolidated company
Microsoft Corporation	1,806	Microsoft
Hyperion Solutions Corporation	1,077	Oracle
Cognos	735	IBM
Business Objects	416	SAP
MicroStrategy	416	MicroStrategy
SAP AG	330	SAP
Cartesis (SAP)	210	SAP
Applix	205	IBM
Infor	199	Infor
Oracle Corporation	159	Oracle
Others	152	Others

Total	**5,700**	

Bibliography

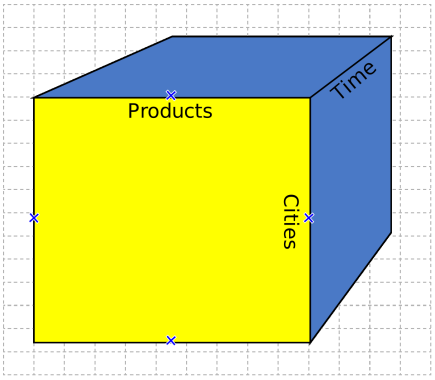
- Daniel Lemire (December 2007). "Data Warehousing and OLAP-A Research-Oriented Bibliography" [6].
- Erik Thomsen. (1997). *OLAP Solutions: Building Multidimensional Information Systems, 2nd Edition*. John Wiley & Sons. ISBN 978-0-471-14931-6.
- Ling Liu and Tamer M. Özsu (Eds.) (2009). "Encyclopedia of Database Systems [7], 4100 p. 60 illus. ISBN 978-0-387-49616-0.
- O'Brien, J. A., & Marakas, G. M. (2009). Management information systems (9th ed.). Boston, MA: McGraw-Hill/Irwin.

References

[1] O'Brien & Marakas, 2011, p. 402-403
[2] O'Brien & Marakas, 2009, pg 177
[3] O'Brien & Marakas, 2009, pg 178
[4] (O'Brien & Marakas, 2009)
[5] Williams, C., Garza, V.R., Tucker, S, Marcus, A.M. (1994, January 24). Multidimensional models boost viewing options. InfoWorld, 16(4)
[6] http://www.daniel-lemire.com/OLAP/
[7] http://www.springer.com/computer/database+management+&+information+retrieval/book/978-0-387-49616-0

OLAP cube

An **OLAP cube** is an array of data understood in terms of its 0 or more dimensions. *OLAP* is an acronym for online analytical processing. OLAP is a computer-based technique for analyzing business data in the search for business intelligence.



An example of an OLAP cube

Terminology

A cube can be considered a generalization of a three-dimensional spreadsheet. For example, a company might wish to summarize financial data by product, by time-period, and by city to compare actual and budget expenses. Product, time, city and scenario (actual and budget) are the data's dimensions.

Cube is a shortcut for *multidimensional dataset*, given that data can have an arbitrary number of *dimensions*. The term hypercube is sometimes used, especially for data with more than three dimensions.

Each cell of the cube holds a number that represents some *measure* of the business, such as sales, profits, expenses, budget and forecast.

OLAP data is typically stored in a star schema or snowflake schema in a relational data warehouse or in a special-purpose data management system. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables.

Hierarchy

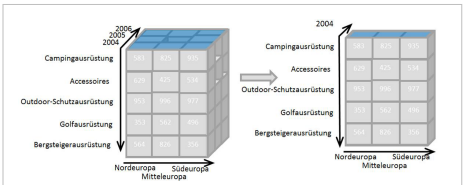
The elements of a dimension can be organized as a hierarchy, a set of parent-child relationships, typically where a parent member summarizes its children. Parent elements can further be aggregated as the children of another parent.

For example May 2005's parent is Second Quarter 2005 which is in turn the child of Year 2005. Similarly cities are the children of regions; products roll into product groups and individual expense items into types of expenditure.

Operations

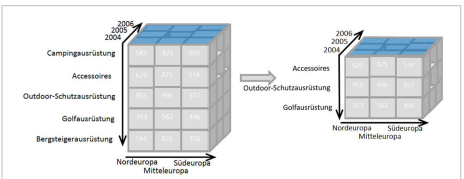
Conceiving data as a cube with hierarchical dimensions leads to conceptually straightforward operations to facilitate analysis. Aligning the data content with a familiar visualization enhances analyst learning and productivity. The user-initiated process of navigating by calling for page displays interactively, through the specification of slices via rotations and drill down/up is sometimes called "slice and dice". Common operations include slice and dice, drill down, roll up, and pivot.

Slice is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with one fewer dimension. The picture shows a slicing operation: The sales figures of all sales regions and all product categories of the company in the year 2004 are "sliced" out of the data cube.



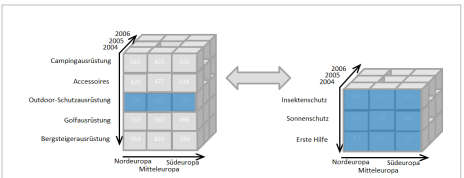
OLAP slicing

Dice: The dice operation produces a subcube by allowing the analyst to pick specific values of multiple dimensions. The picture shows a dicing operation: The new cube shows the sales figures of a limited number of product categories, the time and region dimensions cover the same range as before.



OLAP dicing

Drill Down/Up allows the user to navigate among levels of data ranging from the most summarized (up) to the most detailed (down). The picture shows a drill-down operation: The analyst moves from the summary category "Outdoor-Schutzausrüstung" to see the sales figures for the individual products.

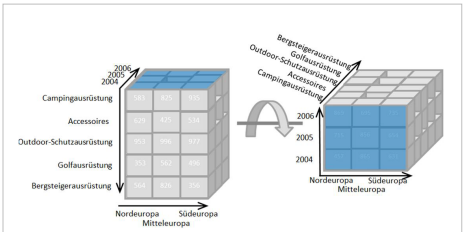


OLAP Drill-up and drill-down

Roll-up: A roll-up involves summarizing the data along a dimension. The summarization rule might be computing totals along a hierarchy or applying a set of formulas such as "profit = sales - expenses".

Pivot allows an analyst to rotate the cube in space to see its various faces. For example, cities could be arranged vertically and products horizontally while viewing data for a particular quarter. Pivoting could replace products with time periods to see data across time for a single product.

The picture shows a pivoting operation: The whole cube is rotated, giving another perspective on the data.



OLAP pivoting

Mathematical definition

In database theory, an OLAP cube is an abstract representation of a projection of an RDBMS relation. Given a relation of order N, consider a projection that subtends X, Y, and Z as the key and W as the residual attribute. Characterizing this as a function,

$$f : (X,Y,Z) \rightarrow W,$$

the attributes X, Y, and Z correspond to the axes of the cube, while the W value into which each (X, Y, Z) triple maps corresponds to the data element that populates each cell of the cube.

Insofar as two-dimensional output devices cannot readily characterize four dimensions, it is more practical to project "slices" of the data cube (we say *project* in the classic vector analytic sense of dimensional reduction, not in the SQL sense, although the two are conceptually similar),

$$g : (X,Y) \rightarrow W$$

which may suppress a primary key, but still have some semantic significance, perhaps a slice of the triadic functional representation for a given Z value of interest.

The motivation behind OLAP displays harks back to the *cross-tabbed report* paradigm of 1980s DBMS. The resulting spreadsheet-style display, where values of X populate row $1; values of Y populate column $A; and values of $g : (X, Y) \rightarrow W$ populate the individual cells "southeast of" $B2, so to speak, $B2 itself included.

References

External links

- Daniel Lemire (December 2007). "Data Warehousing and OLAP - A Research-Oriented Bibliography" (http://www.daniel-lemire.com/OLAP/). Retrieved 2008-03-05.

Aggregate (data warehouse)

Aggregates are used in dimensional models of the data warehouse to produce dramatic positive effects on the time it takes to query large sets of data. At the simplest form an **aggregate** is a simple summary table that can be derived by performing a *Group by* SQL query. A more common use of aggregates is to take a dimension and change the granularity of this dimension. When changing the granularity of the dimension the fact table has to be partially summarized to fit the new grain of the new dimension, thus creating new dimensional and fact tables, fitting this new level of grain. Aggregates are sometimes referred to as pre-calculated summary data, since aggregations are usually precomputed, partially summarized data, that are stored in new aggregated tables. When facts are aggregated, it is either done by eliminating dimensionality or by associating the facts with a rolled up dimension. Rolled up dimensions should be shrunken versions of the dimensions associated with the granular base facts. This way, the aggregated dimension tables should conform to the base dimension tables.[1] So the reason why aggregates can make such a dramatic increase in the performance of the data warehouse is the reduction of the number of rows to be accessed when responding to a query.[2]

Ralph Kimball, who is widely regarded as one of the original architects of data warehousing, says:

> *The single most dramatic way to affect performance in a large data warehouse is to provide a proper set of aggregate (summary) records that coexist with the primary base records. Aggregates can have a very significant effect on performance, in some cases speeding queries by a factor of one hundred or even one thousand. No other means exist to harvest such spectacular gains.*

Having aggregates and atomic data increases the complexity of the dimensional model. This complexity should be transparent to the users of the data warehouse, thus when a request is made, the data warehouse should return data from the table with the correct grain. So when requests to the data warehouse are made, aggregate navigator functionality should be implemented, to help determine the correct table with the correct grain. The number of possible aggregations is determined by every possible combination of dimension granularities. Since it would produce a lot of overhead to build all possible aggregations, it is a good idea to choose a subset of tables on which to make aggregations. The best way to choose this subset and decide which aggregations to build is to monitor queries and design aggregations to match query patterns.[3]

Aggregate navigator

Having aggregate data in the dimensional model makes the environment more complex. To make this extra complexity transparent to the user, functionality known as aggregate navigation is used to query the dimensional and fact tables with the correct grain level. The aggregate navigation essentially examines the query to see if it can be answered using a smaller, aggregate table.[4]

Implementations of aggregate navigators can be found in a range of technologies:

- OLAP engines
- Materialized views
- Relational OLAP (ROLAP) services
- BI application servers or query tools

It is generally recommended to use either of the first three technologies, since the benefits in the latter case is restricted to a single front end BI tool[5]

Problems/challenges

- Since dimensional models only gains from aggregates on large data sets, at what size of the data sets should one start considering using aggregates?
- Similarly, is a data warehouses always handling data sets that are too large for direct queries, or is it sometimes a good idea to omit the aggregate tables, when starting a new data warehouse project. Thus will, omitting aggregates in the first iteration of building a new data warehouse, make the structure of the dimensional model simpler?

References

[1] Ralph Kimball, Margy Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, Second Edition, Wiley Computer Publishing, 2002. ISBN 0-471-20024-7, Page 356

[2] Christopher Adamson, *Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance*, Wiley Publishing, Inc., 2006 ISBN 978-0-471-77709-0, Page 23

[3] Ralph Kimball et al., *The Data Warehouse Toolkit*, Second Edition, Wiley Publishing, Inc., 2008. ISBN 978-0-470-14977-5, Page 355

[4] Ralph Kimball et al., *The Data Warehouse Toolkit*, Second Edition, Wiley Publishing, Inc., 2008. ISBN 978-0-470-14977-5, Page 137

[5] Ralph Kimball et al., *The Data Warehouse Toolkit*, Second Edition, Wiley Publishing, Inc., 2008. ISBN 978-0-470-14977-5, Page 354

MOLAP

MOLAP stands for **M**ultidimensional **O**n**l**ine **A**nalytical **P**rocessing.

MOLAP is an alternative to the ROLAP (Relational OLAP) technology. While both ROLAP and MOLAP analytic tools are designed to allow analysis of data through the use of a multidimensional data model, MOLAP differs significantly in that (in some software) it requires the pre-computation and storage of information in the cube — the operation known as *processing*. Most MOLAP solutions store these data in an optimized multidimensional array storage, rather than in a relational database (i.e. in ROLAP).

There are many methodologies and algorithms for efficient data storage, aggregation and *implementation specific* business logic with a MOLAP Solution. As a result there are many misconceptions to what the term specifically implies.

MOLAP vs. ROLAP

Advantages of MOLAP

- Fast query performance due to optimized storage, multidimensional indexing and caching.
- Smaller on-disk size of data compared to data stored in relational database due to compression techniques.
- Automated computation of higher level aggregates of the data.
- It is very compact for low dimension data sets.
- Array models provide natural indexing.
- Effective data extraction achieved through the pre-structuring of aggregated data.

Disadvantages of MOLAP

- Within some MOLAP Solutions the processing step (data load) can be quite lengthy, especially on large data volumes. This is usually remedied by doing only incremental processing, i.e., processing only the data which have changed (usually new data) instead of reprocessing the entire data set.
- MOLAP tools traditionally have difficulty querying models with dimensions with very high cardinality (i.e., millions of members).
- Some MOLAP products have difficulty updating and querying models with more than ten dimensions. This limit differs depending on the complexity and cardinality of the dimensions in question. It also depends on the number of facts or measures stored. Other MOLAP products can handle hundreds of dimensions.
- Some MOLAP methodologies introduce data redundancy.

Trends

Many commercial OLAP tools now use a "Hybrid OLAP" (HOLAP) approach, which allows the model designer to decide which portion of the data will be stored in MOLAP .

Products

Examples of commercial products that use MOLAP are Cognos Powerplay, Oracle Database OLAP Option, MicroStrategy, Microsoft Analysis Services, Essbase, TM1, Board Toolkit, Lilith Hicare and Daptech Keystone. There is also an open source MOLAP server Palo.

Not all MOLAP are generic databases. Some commercial products focus on a specific business function or type, such as Oracle© RPAS (Retail Predictive Application Server)[1]

References

[1] " Oracle© Retail Predictive Analysis Server User Guide (http://docs.oracle.com/cd/E12478_01/rpas/pdf/134/rpas-134-ug.pdf)", December 2012, Oracle© Corporation.

ROLAP

ROLAP stands for **R**elational **Onl**ine **A**nalytical **P**rocessing.

ROLAP is an alternative to the MOLAP (Multidimensional OLAP) technology. While both ROLAP and MOLAP analytic tools are designed to allow analysis of data through the use of a multidimensional data model, ROLAP differs significantly in that it does not require the pre-computation and storage of information. Instead, ROLAP tools access the data in a relational database and generate SQL queries to calculate information at the appropriate level when an end user requests it. With ROLAP, it is possible to create additional database tables (*summary tables* or *aggregations*) which summarize the data at any desired combination of dimensions.

While ROLAP uses a relational database source, generally the database must be carefully designed for ROLAP use. A database which was designed for OLTP will not function well as a ROLAP database. Therefore, ROLAP still involves creating an additional copy of the data. However, since it is a database, a variety of technologies can be used to populate the database.

ROLAP vs. MOLAP

The discussion of the advantages and disadvantages of ROLAP below, focus on those things that are true of the most widely used ROLAP and MOLAP tools available today. In some cases there will be tools which are exceptions to any generalization made.

Advantages of ROLAP

- ROLAP is considered to be more scalable in handling large data volumes, especially models with dimensions with very high cardinality (i.e., millions of members).
- With a variety of data loading tools available, and the ability to fine tune the ETL code to the particular data model, load times are generally much shorter than with the automated MOLAP loads.
- The data are stored in a standard relational database and can be accessed by any SQL reporting tool (the tool does not have to be an OLAP tool).
- ROLAP tools are better at handling *non-aggregatable facts* (e.g., textual descriptions). MOLAP tools tend to suffer from slow performance when querying these elements.
- By decoupling the data storage from the multi-dimensional model, it is possible to successfully model data that would not otherwise fit into a strict dimensional model.
- The ROLAP approach can leverage database authorization controls such as row-level security, whereby the query results are filtered depending on preset criteria applied, for example, to a given user or group of users (SQL WHERE clause).

Disadvantages of ROLAP

- There is a consensus in the industry that ROLAP tools have slower performance than MOLAP tools. However, see the discussion below about ROLAP performance.
- The loading of *aggregate tables* must be managed by custom ETL code. The ROLAP tools do not help with this task. This means additional development time and more code to support.
- When the step of creating aggregate tables is skipped, the query performance then suffers because the larger detailed tables must be queried. This can be partially remedied by adding additional aggregate tables, however it is still not practical to create aggregate tables for all combinations of dimensions/attributes.
- ROLAP relies on the general purpose database for querying and caching, and therefore several special techniques employed by MOLAP tools are not available (such as special hierarchical indexing). However, modern ROLAP

tools take advantage of latest improvements in SQL language such as CUBE and ROLLUP operators, DB2 Cube Views, as well as other SQL OLAP extensions. These SQL improvements can mitigate the benefits of the MOLAP tools.

- Since ROLAP tools rely on SQL for all of the computations, they are not suitable when the model is heavy on calculations which don't translate well into SQL. Examples of such models include budgeting, allocations, financial reporting and other scenarios.

Performance of ROLAP

OLAP Survey

In the OLAP industry ROLAP is usually perceived as being able to scale for large data volumes, but suffering from slower query performance as opposed to MOLAP. The OLAP Survey [1], the largest independent survey across all major OLAP products, being conducted for 6 years (2001 to 2006) have consistently found that companies using ROLAP report slower performance than those using MOLAP even when data volumes were taken into consideration.

However, as with any survey there are a number of subtle issues that must be taken into account when interpreting the results.

- The survey shows that ROLAP tools have 7 times more users than MOLAP tools within each company. Systems with more users will tend to suffer more performance problems at peak usage times.
- There is also a question about complexity of the model, measured both in number of dimensions and richness of calculations. The survey does not offer a good way to control for these variations in the data being analyzed.

Downside of flexibility

Some companies select ROLAP because they intend to re-use existing relational database tables—these tables will frequently not be optimally designed for OLAP use. The superior flexibility of ROLAP tools allows this less than optimal design to work, but performance suffers. MOLAP tools in contrast would force the data to be re-loaded into an optimal OLAP design.

Trends

The undesirable trade-off between additional ETL cost and slow query performance has ensured that most commercial OLAP tools now use a "Hybrid OLAP" (HOLAP) approach, which allows the model designer to decide which portion of the data will be stored in MOLAP and which portion in ROLAP.

Products

Examples of commercial products using ROLAP include Microsoft Analysis Services, MicroStrategy, SAP Business Objects [2], Oracle Business Intelligence Suite Enterprise Edition (the former Siebel Analytics) and Tableau Software [3]. There is also an open source ROLAP server, Mondrian. ROLAP is also a company that operates in the detection of anatocism and usury in the current accounts. Rolap website [4]

References

[1] http://www.olapreport.com/survey.htm
[2] http://www.sap.com/solutions/analytics/business-intelligence/index.epx
[3] http://www.tableausoftware.com
[4] http://www.rolap.it

HOLAP

HOLAP (**H**ybrid **Onl**ine **A**nalytical **P**rocessing) is a combination of ROLAP (Relational OLAP) and MOLAP (Multidimensional OLAP) which are other possible implementations of OLAP. HOLAP allows storing part of the data in a MOLAP store and another part of the data in a ROLAP store, allowing a tradeoff of the advantages of each. The degree of control that the cube designer has over this partitioning varies from product to product.

Vertical partitioning

In this mode HOLAP stores *aggregations* in MOLAP for fast query performance, and detailed data in ROLAP to optimize time of cube *processing*.

Horizontal partitioning

In this mode HOLAP stores some slice of data, usually the more recent one (i.e. sliced by Time dimension) in MOLAP for fast query performance, and older data in ROLAP. Moreover, we can store some dices in MOLAP and others in ROLAP, leveraging the fact that in a large cuboid, there will be dense and sparse subregions.[1]

Products

The first product to provide HOLAP storage was Holos, but the technology also became available in other commercial products such as Microsoft Analysis Services, Oracle Database OLAP Option, MicroStrategy and SAP AG BI Accelerator. The hybrid OLAP approach combines ROLAP and MOLAP technology, benefiting from the greater scalability of ROLAP and the faster computation of MOLAP. For example, a HOLAP server may allow large volumes of detail data to be stored in a relational database, while aggregations are kept in a separate MOLAP store. The Microsoft SQL Server 7.0 OLAP Services supports a hybrid OLAP server

Footnotes

[1] Owen Kaser and Daniel Lemire, Attribute Value Reordering for Efficient Hybrid OLAP (http://arxiv.org/abs/cs.DB/0702143), Information Sciences, Volume 176, Issue 16, pages 2279-2438, 2006.

Thomsen Diagrams

Thomsen Diagrams are the diagrammatic methodology developed by Erik Thomsen in 1997[1] is essentially a metaphor for describing multi-dimensional data spaces in the OLAP system. In effect, it may be thought of as a multi-dimensional domain structure. In the structure, each dimension is represented by a vertical line, and hence each dimension is described independently.

Every member of a dimension is represented by a unit interval on the line. A multi-dimensional model is built by combining the resultant lines for the particular dimensions.

The Thomsen diagrammatical technique is not based on angular defined dimensions, and is thus able to represent any number of dimensions. It may be referred to as a multi-dimensional type structure[2] (MTS). The MTS permits the viewing of information about hierarchies and data flows, both within and between structures, hence enhancing the capabilities of the OLAP system.

References

[1] Erik Thomsen, 1997, OLAP Solutions: Building Multidimensional Information Systems (1st ed.), John Wiley
[2] Erik Thomsen, 1997, OLAP Solutions: Building Multidimensional Information Systems (2nd ed.), John Wiley

External links

- Peter O'Donnell and Nick Draper 2004, An Experimental Evaluation of an Alternative to the Pivot Table for *Ad Hoc* Access to OLAP Data (PDF (http://vishnu.infotech.monash.edu.au/dss2004/proceedings/pdf/60_ODonnell_Draper.pdf), PDF (alternative link) (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.1535)) - Wherein Mr. Thomsen's diagrams are called "Thomsen Diagrams".

Spreadmart

A **spreadmart** is a concept describing the tendency of spreadsheets to "run amok" in organizations. The definition of a spreadmart as used by The Data Warehousing Institute (TDWI) in a 2008 survey is:

> A spreadmart is a reporting or analysis system running on a desktop database (e.g., spreadsheet, Access database, or dashboard) that is created and maintained by an individual or group that performs all the tasks normally done by a data mart or data warehouse, such as extracting, transforming, and formatting data as well as defining metrics, submitting queries, and formatting and publishing reports to others. Also known as data shadow systems, human data warehouses, or IT shadow systems.

Note that this definition also includes situations where Business Intelligence tools are used in the manner as described above.

Critics like Stephen Samild argue that the definition stems from a biased view that sees a Data Warehouse as desirable end-result, whereas *One might more accurately define data marts and data warehouses as "scaled-up systems which perform some of the tasks normally done by a spreadmart"*.[1] In the rest of the article Stephen Samild argues that a spreadmart fulfills a number of roles that a data warehouse cannot fulfill as easily or as cheaply due to the lack of integration with unstructured data, the lack of read-write capabilities, the long time span needed for integration of new sources in the data warehouse and the inherent 'free form' of many analytical presentations done in Word, PowerPoint or Excel.

Typically a spreadmart is created by individuals at different times using different data sources and rules for defining metrics in an organization, creating a fractured view of the enterprise. The concept was coined in 2002 by Wayne Eckerson at TDWI in his article *Taming Spreadsheet Jockeys*.[2]

Usually, spreadmarts grow where standard Business Intelligence (BI) reporting is too inflexible and too slow. A Business analyst uses the "export to Microsoft Excel" button in his BI software and creates his own report with the exported data table. By this, the number of independently generated spreadsheets dealing with a particular group of analyses grows inside the company, and the data inside each spreadsheet is uncoupled from its source. When this happens, the data reflected in the spreadsheets is no longer verifiable and is not automatically kept up to date. Usually these spreadsheet files are distributed via email to colleagues resulting in even more copies of the data roaming through the enterprise. With Microsoft PowerPivot for Microsoft SharePoint, Excel spreadsheets can be distributed as dashboards throughout the entire company, giving even more users the tools to create spreadmarts.

The growth of spreadmarts poses a real risk for a company, since undefined and uncoupled data is floating from spreadsheet to spreadsheet, and can be used to draw false conclusions that lead to wrong decisions, which will cost time and money to discover and correct. Although Business Intelligence 2.0 software vendors claim to have overcome this issue, locally installed spreadsheet and graphing software continues to be easier to access and use, giving the business analyst the freedom to create the needed analysis quickly, and choose to live with the risk of data inconsistency that goes with it.

Related technologies

- Microsoft Excel with PowerPivot as the standard
- OpenOffice.org Calc, the open source alternative to Excel
- A whole host of Business Intelligence software vendors, for example Cognos, provide software that couples spreadsheets to the data in a more persistent manner.

Notes

[1] Analysis is Read-Write
[2] Taming Spreadsheet Jockeys

References

- Eckerson, Wayne (July 2002). "Taming Spreadsheet Jockeys" (http://www.webcitation.org/5YYQIPcDF). *TDWI Case Studies and Solutions*. TDWI. Archived from the original (http://www.tdwi.org/display.aspx?ID=7167) on 2008-06-13. Retrieved 2008-06-13.
- Samild, Stephen (September 2011). "Analysis is Read-Write" (http://www.webcitation.org/6Csvn5tP2). *Analyst First*. Archived from the original (http://analystfirst.com/tag/spreadmarts/) on 2012-12-13. Retrieved 2012-12-13.

External links

- Strategies for Managing Spreadmarts (http://tdwi.org/articles/2008/05/27/strategies-for-managing-spreadmarts#1) By Wayne Eckerson, Director of Research, TDWI
- Can the Spreadmart Beast Be Tamed (http://www.sqlmag.com/Articles/Index.cfm?ArticleID=50661&DisplayTab=Article) By Brian Moran at SQL Server Magazine (Archived at 2008-06-13 (http://www.webcitation.org/5YYQZ1kzG))
- Excel is the most popular decision support tool for business users worldwide (http://www.doubletongued.org/index.php/citations/spreadmart_1/) from Double-Tongued Dictionary
- Business Intelligence Can Stop The Spread of Spread-Mart (http://spotfireblog.tibco.com/?p=1131) By David Wallace from TIBCO Spotfire Blogging Team
- The Rise and Fall of Spreadmarts (http://www.information-management.com/issues/20030901/7274-1.html) By Wayne Eckerson from Information Management Magazine

MultiDimensional eXpressions

Multidimensional Expressions (MDX) is a query language for OLAP databases, much like SQL is a query language for relational databases. It is also a calculation language, with syntax similar to spreadsheet formulas.

Background

The *MultiDimensional eXpressions (MDX) language* provides a specialized syntax for querying and manipulating the multidimensional data stored in OLAP cubes. While it is possible to translate some of these into traditional SQL, it would frequently require the synthesis of clumsy SQL expressions even for very simple MDX expressions. MDX has been embraced by a wide majority of OLAP vendors and has become the standard for OLAP systems.

History

MDX was first introduced as part of the OLE DB for OLAP specification in 1997 from Microsoft. It was invented by the group of SQL Server engineers including Mosha Pasumansky. The specification was quickly followed by commercial release of Microsoft OLAP Services 7.0 in 1998 and later by Microsoft Analysis Services. The latest version of the OLE DB for OLAP specification was issued by Microsoft in 1999.

While it was not an open standard, but rather a Microsoft-owned specification, it was adopted by the wide range of OLAP vendors. This included both vendors on the server side such as Applix, icCube, MicroStrategy, NCR, Oracle Corporation, SAS, SAP, Teradata, Symphony Teleca, and vendors on the client side such as Panorama Software, PowerOLAP, XLCubed, Proclarity, AppSource, Jaspersoft, Cognos, Business Objects, Brio Technology, Crystal Reports, Microsoft Excel, Tagetik, and Microsoft Reporting Services.

With the invention of XML for Analysis, which standardized MDX as a query language, even more companies - such as Hyperion Solutions - began supporting MDX.

The XML for Analysis specification referred back to the OLE DB for OLAP specification for details on the MDX Query Language. In Analysis Services 2005, Microsoft has added some MDX Query Language extensions like subselects. Products like Microsoft Excel 2007 have started to use these new MDX Query Language extensions. Some refer to this newer variant of MDX as MDX 2005.

mdXML

In 2001 the XMLA Council released the XML for Analysis standard, which included mdXML as a query language. In the current XMLA 1.1 specification, mdXML is essentially MDX wrapped in the XML `<Statement>` tag.

MDX data types

There are six primary data types in MDX

- **Scalar**. Scalar is either a number or a string. It can be specified as a literal, e.g. number 5 or string "OLAP" or it can be returned by an MDX function, e.g. `Aggregate` (number), `UniqueName` (string), `.Value` (number or string) etc.
- **Dimension/Hierarchy**. Dimension is a dimension of a cube. A dimension is a primary organizer of measure and attribute information in a cube. MDX does not know of, nor does it assume any, dependencies between dimensions- they are assumed to be mutually independent. A dimension will contain some members (see below) organized in some hierarchy or hierarchies containing levels. It can be specified by its unique name, e.g. `[Time]` or it can be returned by an MDX function, e.g. `.Dimension`. Hierarchy is a dimension hierarchy of a cube. It can be specified by its unique name, e.g. `[Time].[Fiscal]` or it can be returned by an MDX function, e.g. `.Hierarchy`. Hierarchies are contained within dimensions. (*OLEDB for OLAP MDX*

specification does not distinguish between dimension and hierarchy data types. Some implementations, such as Microsoft Analysis Services, treat them differently.)

- **Level**. Level is a level in a dimension hierarchy. It can be specified by its unique name, e.g. `[Time].[Fiscal].[Month]` or it can be returned by an MDX function, e.g. `.Level`.
- **Member**. Member is a member in a dimension hierarchy. It can be specified by its unique name, e.g. `[Time].[Fiscal].[Month].[August 2006]`, by qualified name, e.g. `[Time].[Fiscal].[2006].[Q2].[August 2006]` or returned by an MDX function, e.g. `.PrevMember`, `.Parent`, `.FirstChild` etc. Note that all members are specific to a hierarchy. If the self-same product is a member of two different hierarchies (`[Product].[ByManufacturer]` and `[Product].[ByCategory]`), there will be two different members visible that may need to be coordinated in sets and tuples (see below).
- **Tuple**. Tuple is an ordered collection of one or more members from different dimensions. Tuples can be specified by enumerating the members, e.g. `([Time].[Fiscal].[Month].[August], [Customer].[By Geography].[All Customers].[USA], [Measures].[Sales])` or returned by an MDX function, e.g. `.Item`.
- **Set**. Set is an ordered collection of tuples with the same dimensionality, or hierarchality in the case of Microsoft's implementation. It can be specified enumerating the tuples, e.g. `{([Measures].[Sales], [Time].[Fiscal].[2006]), ([Measures].[Sales], [Time].[Fiscal].[2007])}` or returned by MDX function or operator, e.g. `Crossjoin`, `Filter`, `Order`, `Descendants` etc.
- Other data types. Member properties are equivalent to *attributes* in the data warehouse sense. They can be retrieved by name in a query through an axis PROPERTIES clause of a query. The scalar data value of a member property for some member can be accessed in an expression through MDX, either by naming the property (for example, `[Product].CurrentMember.[Sales Price]`) or by using a special access function (for example, `[Product].CurrentMember.Properties("Sales Price")`). In limited contexts, MDX allows other data types as well - for example Array can be used inside the `SetToArray` function to specify an array that is not processed by MDX but passed to a user-defined function in an ActiveX library. Objects of other data types are represented as scalar strings indicating the object names, such as measure group name in Microsoft's `MeasureGroupMeasures` function or KPI name in for example Microsoft's `KPIValue` or `KPIGoal` functions.

Example query

The following example, adapted from the SQL Server 2000 Books Online, shows a basic MDX query that uses the SELECT statement. This query returns a result set that contains the 2002 and 2003 store sales amounts for stores in the state of California.

```
SELECT
   { [Measures].[Store Sales] } ON COLUMNS,
   { [Date].[2002], [Date].[2003] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

In this example, the query defines the following result set information:

- The SELECT clause sets the query axes as the Store Sales member of the Measures dimension, and the 2002 and 2003 members of the Date dimension.
- The FROM clause indicates that the data source is the Sales cube.
- The WHERE clause defines the "slicer axis" as the California member of the Store dimension.

Note: You can specify up to 128 query axes in an MDX query.

If you create two axes, one must be the column axis and one must be the row axis, although it doesn't matter in which order they appear within the query. if you create a query that has only one axis, it must be the column axis. The square brackets around the particular object identifier are optional as long as the object identifier: is not one of reserved words, does not otherwise contain any characters other than letters, numbers or underscores.

```
SELECT
    [Measures].[Store Sales] ON COLUMNS,
    [Date].Members ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

The Members() function returns the set of members in a dimension, level or hierarchy.

References

External reference

- George Spofford, Sivakumar Harinath, Chris Webb, Dylan Hai Huang, Francesco Civardi: *MDX-Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase*. Wiley, 2006, ISBN 0-471-74808-0
- Mosha Pasumansky, Mark Whitehorn, Rob Zare: *Fast Track to MDX*. ISBN 1-84628-174-1
- Larry Sackett: *MDX Reporting and Analytics with SAP NetWeaver BW*. SAP Press, 2008, 978-1-59229-249-3
- MDX Reference from SQL Server 2008 Books Online (http://msdn2.microsoft.com/en-us/library/ms145506.aspx)
- Links to MDX resources (http://www.mosha.com/msolap/mdx.htm)
- MDX Gentle Tutorial (http://www.iccube.com/support/documentation/mdx_tutorial/gentle_introduction.html)
- MDX Essentials Series (http://www.databasejournal.com/article.php/1459531/) by William Pearson in the Database Journal
- MDX Video Tutorial (http://www.learn-with-video-tutorials.com/mdx-video-tutorial-free)

OLAP Servers and Tools

Comparison of OLAP Servers

The following tables compare general and technical information for a number of online analytical processing (OLAP) servers. Please see the individual products articles for further information.

General information

OLAP Server	Company	Website	Latest stable version	Software license	License Pricing
Essbase	Oracle		11.1.2.3	Proprietary	[1]
icCube	MISConsulting SA		4.1.1	Proprietary	community/[2]
Infor BI OLAP Server	Infor		10.4	Proprietary	-
Microsoft Analysis Services	Microsoft		2012	Proprietary	[3]
MicroStrategy Intelligence Server	MicroStrategy		9	Proprietary	-
Mondrian OLAP server	Pentaho		3.5.0	EPL	free
Oracle Database OLAP Option	Oracle		11g R2	Proprietary	[1]
Palo	Jedox		5.0	GPL v2 or EULA	-
SAS OLAP Server	SAS Institute		9.3	Proprietary	-
SAP NetWeaver BW	SAP		7.20	Proprietary	-
Cognos TM1	IBM		10.2 FP1	Proprietary	-

Data storage modes

OLAP Server	MOLAP	ROLAP	HOLAP	Offline
Essbase	Yes	Yes	Yes	
icCube	Yes	No	No	Offline Cubes [4]
Infor BI OLAP Server	Yes	No	No	Local cubes
Microsoft Analysis Services	Yes	Yes	Yes	Local cubes, PowerPivot for Excel
MicroStrategy Intelligence Server	Yes	Yes	Yes	MicroStrategy Office [5], Dynamic Dashboards [6]
Mondrian OLAP server	No	Yes	No	
Oracle Database OLAP Option	Yes	Yes	Yes	

Palo	Yes	No	No	
SAS OLAP Server	Yes	Yes	Yes	
IBM TM1	Yes	No	No	
IBM Cognos	Yes	Yes	Yes	
SAP NetWeaver BW	Yes	Yes	No	

APIs and query languages

APIs and query languages OLAP servers support.

OLAP Server	XML for Analysis	OLE DB for OLAP	MDX	Stored procedures	Custom functions	SQL	LINQ	Visualization
Essbase	Yes	Yes	Yes	*?*	Yes	No	Yes	*?*
icCube	Yes	Yes	Yes	Java, R	Yes	No	Yes	Java, Javascript
Infor BI OLAP Server	Yes	Yes	Yes	OLAP Rules, Push Rules, Application Engine	Yes	No	No	Application Studio
Microsoft Analysis Services	Yes	Yes	Yes	.NET	Yes	Yes	Yes	Microsoft Excel, Microsoft PowerPivot, Microsoft Power View
MicroStrategy Intelligence Server	Yes	No	Yes	Yes	Yes	Yes	Yes	*?*
Mondrian OLAP server	Yes	Yes	Yes	No	Yes	No	Yes	*?*
Palo	Yes	Yes	Yes	Cube Rules, SVS Triggers	Yes	No	Yes	*?*
Oracle Database OLAP Option	No	Yes	Yes	Java, PL/SQL, OLAP DML	Yes	Yes	No	*?*
SAS OLAP Server	Yes	Yes	Yes	No	No	No	Yes	Web Report Studio
SAP NetWeaver BW	Yes	Yes	Yes	No	Yes	No	Yes	*?*
Cognos TM1	Yes	Yes	Yes	Yes	Yes	No	Yes	*?*

OLAP distinctive features

A list of OLAP features that are not supported by all vendors. All vendors support features such as parent-child, multilevel hierarchy, drilldown.

OLAP Server	Real Time	Semi-additive measures	Write-back	Many-to-Many	Partitioning
Essbase	*?*	Yes	Yes	*?*	Yes
icCube	Yes	Yes	Yes	Yes	Yes
Infor BI OLAP Server	*?*	Yes	Yes	*?*	Yes
Microsoft Analysis Services	*?*	Yes	Yes	Yes	Yes
MicroStrategy Intelligence Server	*?*	Yes	Yes	*?*	Yes
Mondrian OLAP server	*?*	No	Planned	*?*	No
Oracle Database OLAP Option	*?*	Yes	Yes	*?*	Yes
Palo	*?*	*?*	Yes	*?*	*?*
TM1	*?*	Yes	Yes	*?*	No
IBM Cognos	Yes	Yes	No	Yes	Yes
SAS OLAP Server	?	Yes	Yes	*?*	Yes
SAP NetWeaver BW	?	Yes	Yes	*?*	Yes

System limits

OLAP Server	# cubes	# measures	# dimensions	# hierarchies in dimension	# levels in hierarchy	# dimension members
Essbase	?	?	?	256	?	20,000,000 (ASO), 1,000,000 (BSO)
icCube	2,147,483,647	2,147,483,647	2,147,483,647	2,147,483,647	2,147,483,647	2,147,483,647
Infor BI OLAP Server	?	?	?	?	?	1,000,000
Microsoft Analysis Services	2,147,483,647	2,147,483,647	2,147,483,647	2,147,483,647	2,147,483,647	2,147,483,647
MicroStrategy Intelligence Server	Unrestricted[1]	Unrestricted[1]	Unrestricted[1]	Unrestricted[1]	Unrestricted[1]	Unrestricted[1]
Palo	?	?	?	?	?	?
SAS OLAP Server	Unrestricted[1]	1024	128	128	19	4,294,967,296

[1] Please update as 'unrestricted', is just not possible

Security

OLAP Server	Authentication	Network encryption	On-the-Fly[1]	Data access		
				Cell security	Dimension security	Visual totals
Essbase	Essbase authentication, LDAP authentication	SSL	?	Yes	Yes	Yes
icCube	HTTP Basic/Form Authentication, Windows SSO (NTLM,Kerberos)	SSL	Yes	Yes	Yes	Yes
Infor BI OLAP Server	OLAP authentication, Infor Federation Services, LDAP, Microsoft Active Directory	Yes	Yes	Yes	Yes	*?*
Microsoft Analysis Services	NTLM, Kerberos	SSL and SSPI	?	Yes	Yes	Yes
MicroStrategy Intelligence Server	Host authentication, database authentication, LDAP, Microsoft Active Directory, NTLM, SiteMinder, Tivoli, SAP, Kerberos	SSL, AES[7]	?	Yes	Yes	Yes
Oracle Database OLAP Option	Oracle Database authentication	SSL	?	Yes	Yes	*?*
Palo	Palo authentication, LDAP, Microsoft Active Directory	SSL	Yes	Yes	Yes	*?*
SAS OLAP Server	Host authentication,SAS token authentication, LDAP, Microsoft Active Directory	Yes	?	Yes	Yes	Yes

[1] On-the-Fly : The ability to define authentication dynamically via programmatic interfaces. New users do not require restarting the server or redefining the security.

Operating systems

The OLAP servers can run on the following operating systems:

OLAP Server	Windows	Linux	UNIX	z/OS
Essbase	Yes	Yes	Yes	No
icCube	Yes	Yes	Yes	Yes
Infor BI OLAP Server	Yes	No	No	No
Microsoft Analysis Services	Yes	No	No	No
MicroStrategy Intelligence Server	Yes	Yes	Yes	No
Mondrian OLAP server	Yes	Yes	Yes	Yes
Oracle Database OLAP Option	Yes	Yes	Yes	Yes
Palo	Yes	Yes	Yes	No
SAS OLAP Server	Yes	Yes	Yes	Yes
SAP NetWeaver BW	Yes	Yes	Yes	Yes
TM1	Yes	Yes	Yes	No

Note (1):The server availability depends on Java Virtual Machine not on the operating system

Support information

OLAP Server	Issue Tracking System	Forum/Blog	Roadmap	Source code
Essbase	myOracle Support [8]	[9]	[10]	Closed
icCube	YouTrack [11]	[12]	[13]	Open
Infor BI OLAP Server	Infor Xtreme		Available upon request	Closed
Microsoft Analysis Services	Connect [14]	[15]	-	Closed
MicroStrategy Intelligence Server	MicroStrategy Resource Center [16]	[17]	-	Closed
Mondrian OLAP server	Jira [18]	[19]	[20]	Open
Oracle Database OLAP Option	myOracle Support [8]	[9]		Closed
Palo	Mantis [21]	[22]		Open
SAS OLAP Server	Support [23]			Closed
SAP NetWeaver BW	OSS [24]	[25]	[26]	Closed
TM1	IBM Service Request [27]	[28]		Closed

References

[1] http://www.oracle.com/us/corporate/pricing/index.htm
[2] http://www.iccube.com/purchase/prices
[3] http://www.microsoft.com/sqlserver/2008/en/us/pricing.aspx
[4] http://www.iccube.com/support/documentation/user_guide/using/offline_cubes.html
[5] http://www.microstrategy.com/Software/Products/User_Interfaces/Office/
[6] http://www.microstrategy.com/Software/Products/Service_Modules/Report_Services/
[7] MicroStrategy Intelligence Server Features (http://latam.microstrategy.com/Software/Products/Intelligence_Server/features.asp)
[8] http://support.oracle.com
[9] http://forums.oracle.com/forums/main.jspa?categoryID=84
[10] http://communities.ioug.org/Portals/2/Oracle_Essbase_Roadmap_Sep_09.pdf
[11] http://issues.iccube.com/
[12] http://www.iccube.com/support/forum
[13] http://www.iccube.com/support/qa/roadmap
[14] https://connect.microsoft.com/SQLServer
[15] http://social.msdn.microsoft.com/Forums/en-US/sqlanalysisservices/threads
[16] https://resource.microstrategy.com/Support/MainSearch.aspx
[17] https://resource.microstrategy.com/Forum/
[18] http://jira.pentaho.com/browse/MONDRIAN
[19] http://forums.pentaho.org/forumdisplay.php?f=79
[20] http://mondrian.pentaho.org/documentation/roadmap.php
[21] http://bugs.palo.net/mantis/main_page.php
[22] http://www.jedox.com/community/palo-forum/board.php?boardid=9
[23] http://support.sas.com/forums/index.jspa
[24] http://service.sap.com/
[25] http://forums.sdn.sap.com/index.jspa

[26] http://esworkplace.sap.com/socoview(bD1lbiZjPTAwMSZkPW1pbg==)/render.asp?id=2270EAD629814D05A7ECECECECC8D002&fragID=&packageid=DEE98D07DF9FA9F1B3C7001A64D3F462
[27] http://ibm.com/support/servicerequest/
[28] http://forums.olapforums.com/

Applix

Applix Inc.

Industry	Computer Software
Fate	Acquired by Cognos, October 25, 2007; Acquired by IBM, January 31, 2008
Founded	1983
Headquarters	Westborough, Massachusetts, USA
Area served	Worldwide
Products	• Applix TM1 • Applix Web • Applix Executive Viewer • Applix Business Analytics Platform
Owner(s)	IBM
Website	www-01.ibm.com/software/analytics/cognos/ [1]

Applix Inc. was a computer software company founded in 1983 based in Westborough, Massachusetts that published Applix TM1, a MOLAP database server, and related presentation tools, including Applix Web and Applix Executive Viewer. Together, Applix TM1, Applix Web and Applix Executive Viewer were the three core components of the *Applix Business Analytics Platform*. Applix did business in over 50 countries with over 3,000 customers worldwide.

On October 25, 2007, Applix was acquired by Cognos. Cognos rebranded all Applix products under its name following the acquisition. On January 31, 2008, Cognos was itself acquired by IBM.

Prior to OLAP industry consolidation in 2007, Applix was the purest OLAP vendor among publicly traded independent BI vendors, and had the greatest growth rate. TM1 is now marketed as IBM Cognos TM1 and the latest version, IBM Cognos TM1 10.1, became publicly available on February 7th, 2012.

Products and technology

IBM Cognos TM1 (formerly Applix TM1) is enterprise planning software used to implement collaborative planning, budgeting and forecasting solutions, as well as analytical and reporting applications. Data in IBM Cognos TM1 is stored and represented as multidimensional cubes, with data being stored at the "leaf" level. Computations on the leaf data are performed in real-time (for example, to aggregate numbers up a dimensional hierarchy). IBM Cognos TM1 also includes a data orchestration environment for accessing external data and systems, as well as capabilities designed for common business planning and budgeting requirements (e.g. workflow, top-down adjustments).

New features added to IBM Cognos TM1 9.5 include *TM1 Contributor*, which combines an OLAP engine with the enterprise planning capabilities of IBM Cognos Planning in a Web-based client. Other features include an "Undo/Redo" capability that lets users store a collection of data value changes and walk back through the actions to undo them; pick lists for cells to reduce the amount of data entry performed; sandboxes for creating personal modeling scenarios, and data entry shortcuts.

References

[1] http://www-01.ibm.com/software/analytics/cognos/

External links

- IBM Cognos TM1 Web Site (http://www-01.ibm.com/software/data/cognos/products/tm1/)

BusinessObjects OLAP Intelligence

BusinessObjects OLAP Intelligence is an On Line Analytical Processing (OLAP) application for analysing business data. It was previously known as Crystal Analysis Professional.

External links

- Product page at Business Objects [1]

References

[1] http://www.businessobjects.com/products/queryanalysis/olapi.asp

Crystal Analysis

Crystal Analysis (a.k.a. **Crystal Analysis Professional**) is an On Line Analytical Processing (OLAP) application for analysing business data originally developed by Seagate Software.

It was first released under the name *Seagate Analysis* as a free application written in Java released in 1999. After disappointing application performance, a decision was made to rewrite using ATL COM in C++. The initial rewrite only supported Microsoft Analysis Services, but support for other vendors soon followed, with Holos cubes in version 8.5, Essbase, IBM DB2 and SAP BW following in later releases. The web client was rewritten using an XSLT abstraction layer for the version 9.0 release, with better standards compliance to support Mozilla based browsers—this work also set the building blocks for support for Safari.

Crystal Analysis relies on Crystal Enterprise for distribution of analytical applications created with it.

Release timeline

- *Seagate Analysis* 1999, by Seagate Software
- *Crystal Analysis Professional* v8.0, 29 May 2001 [1] by Crystal Decisions
- *Crystal Analysis Professional* v8.1, Q4 2001 by Crystal Decisions
- *Crystal Analysis Professional* v8.5 9 July 2002 [2], by Crystal Decisions
- *Crystal Analysis Professional* v9.0 9 April 2003 [3], by Crystal Decisions
- *Crystal Analysis Professional* v10.0 8 January 2004 [4], by Business Objects
- *Crystal Analysis Professional* v11.0 31 January 2005, by Business Objects
- *Crystal Analysis Professional* v11.0 Release 2 30 November 2005 [5], by Business Objects

Future versions will be released under the name, *BusinessObjects OLAP Intelligence*.

External links

- Product page at Business Objects [6]

References

[1] http://www.businessobjects.com/news/presscd/2001/052901.asp
[2] http://www.businessobjects.com/news/presscd/2002/070901.asp
[3] http://www.businessobjects.com/news/presscd/2003/040902.asp
[4] http://www.businessobjects.com/news/press/press2004/prod_crystal_v10_08012003.asp
[5] http://www.businessobjects.com/news/press/press2005/20051130_bo_xir2_prod.asp
[6] http://www.businessobjects.com/products/queryanalysis/olapaccess/crystalanalysis.asp

CubePort

CubePort

Developer(s)	ExoLogic Corporation (2002-current)
Stable release	v2.23 / 03-31-2010
Operating system	Microsoft Windows
Type	Data transformation
License	Proprietary EULA
Website	[1]

CubePort is a commercial software application that converts from Oracle Essbase to the analogous Microsoft product Microsoft Analysis Services, which is built into Microsoft SQL Server. This application achieves this through various analogy mapping techniques, and is a standard client-server application that runs on a Windows computer but may connect to non-Windows servers. CubePort converts the various OLAP structures and syntaxes in the source through an extraction process, interprets, and recreates in the target. The objective is to simulate exactly the behavior of the original source system to the target system.

The application also includes UI phases such as Authentication, Questions, Extraction, and Verification. Substantial testing normally occurs after the conversion process prior to posting the objects to production.

History

The software was released widely in 2005. CubePort represents the first product of its kind in the Business intelligence platform migration space. The product was design and created to provide an organization with the ability to move their Business Intelligence assets from one platform to another.

Unlike the RDBMS (relational databases) world where there exists the commonplace ability to migrate from one RDBMS product type to another, in the Business Intelligence or OLAP product space, this capability is currently not found in the marketplace.

Usage Motivations

Motivations for migrating from one Business Intelligence platform to another, regardless of whether this is achieved through automated or manual methods, vary from implementation to implementation, and may include one or more of the following motivations:

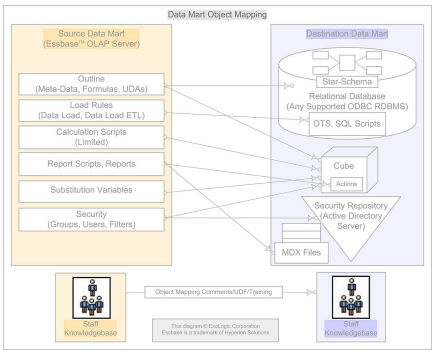
- Reduction of licensing or recurring maintenance expense
- Increase of detail-level of the cube
- Increase of the user count
- Elimination of cube load and calculation time constraints
- Elimination of pervasive vendor lock-in scenario
- Standardization towards Microsoft BI platform, where appropriate

Objects Converted

These standard objects in a cube are converted by the software with varying levels of robustness:

- Cube Outline
- Cube Data Loads and Dimension Builds
- Cube Security Model
- Cube Substitution Variables
- Cube Calculation Scripts
- Cube Report Scripts

The output objects of a successful run of CubePort are: A relational Star schema and/or relational Snowflake schema, OLAP cube, Etl load packages, MDX (Multidimensional Expressions) files, and others.



CubePort Simple Diagram

External links

- SSAS Info Analysis Services Information Page [2]
- Product Preview on CubePort [3]
- Corporate Product Page [1]
- Microsoft's Mosha Pasumansky's Analysis Services Product Page [4]
- Microsoft Solution Page [5]
- Mosha Pasumanky's BI Blog [6]

References

[1] http://www.exologic.com/products.htm
[2] http://www.ssas-info.com/analysis-services-tools/214-exologic-cubeport-migration-from-hyperion-essbase-to-ssas
[3] http://www.theregister.co.uk/2007/06/08/cubeport_essbase
[4] http://www.mosha.com/msolap/util.htm
[5] https://solutionfinder.microsoft.com/Solutions/SolutionDetailsView.aspx?solutionid=a0b39a1ef6d84aa58e9e22f27651aecc&partnerid=844e938e-6c3f-4f92-8535-7cc98a45bccd
[6] http://web.archive.org/web/20060104134906/http://sqljunkies.com/WebLog/mosha/archive/2005/07/02/exologic_cubeport.aspx

Essbase

Essbase

Stable release	11.1.2.3.0 / Apr 2013
Operating system	Microsoft Windows, Linux, AIX, HP-UX , Solaris
Type	Multidimensional database
License	Proprietary
Website	www.oracle.com/technetwork/middleware/essbase [1]

Essbase is a multidimensional database management system (MDBMS) that provides a multidimensional database platform upon which to build analytic applications. Essbase, whose name derives from "**E**xtended **S**pread **S**heet data**BASE**", began as a product of Arbor Software, which merged with Hyperion Software in 1998. Oracle Corporation acquired Hyperion Solutions Corporation in 2007, as of 2009[2] it markets Essbase as "Oracle Essbase". Until late 2005 IBM also marketed the product — as **DB2 OLAP Server**.[3]

The database researcher E. F. Codd coined the term "on-line analytical processing" (OLAP) in a whitepaper that set out twelve rules for analytic systems (an allusion to his earlier famous set of twelve rules defining the relational model). This whitepaper, published by Computerworld, was somewhat explicit in its reference to Essbase features, and when it was later discovered that Codd had been sponsored by Arbor Software, Computerworld withdrew the paper.[4]

In contrast to "on-line transaction processing" (OLTP), OLAP defines a database technology optimized for processing human queries rather than transactions. The results of this orientation were that MDBMS oriented their performance requirements around a different set of benchmarks (Analytic Performance Benchmark, APB-1) than that of RDBMS (Transaction Processing Performance Council (TPC)).

Hyperion renamed many of its products in 2005, giving Essbase an official name of **Hyperion System 9 BI+ Analytic Services**, but the new name was largely ignored by practitioners. The Essbase brand was later returned to the official product name for marketing purposes, but the server software still carried the "Analytic Services" title until it was incorporated into Oracle's Business Intelligence product suite. [5]

In August 2005, *Information Age* magazine named Essbase as one of the 10 most influential technology innovations of the previous 10 years,[6] along with Netscape, the BlackBerry, Google, virtualization, Voice Over IP (VOIP), Linux, XML, the Pentium processor and ADSL. Editor Kenny MacIver said: "Hyperion Essbase was the multi-dimensional database technology that put online analytical processing on the business intelligence map. It has spurred the creation of scores of rival OLAP products – and billions of OLAP cubes".

History and motivation

Although Essbase has been categorisedWikipedia:Manual of Style/Words to watch#Unsupported attributions as a general-purpose multidimensional database, it was originally developed to address the scalability issues associated with spreadsheets such as Lotus 1-2-3 and Microsoft Excel. Indeed, the patent covering Essbase uses spreadsheets as a motivating example to illustrate the need for such a system.[7]

In this context, "multi-dimensional" refers to the representation of financial data in spreadsheet format. A typical spreadsheet may display time intervals along column headings, and account names on row headings. For example:

	Jan	Feb	Mar	Total
Quantity	1000	2000	3000	**6000**
Sales	$100	$200	$300	**$600**
Expenses	$80	$160	$240	**$480**
Profit	$20	$40	$60	**$120**

If a user wants to break down these values by region, for example, this typically involves the duplication of this table on multiple spreadsheets:

	Jan	Feb	Mar	Total
Quantity	240	1890	50	**2180**
Sales	$24	$189	$5	**$218**
Expenses	$20	$150	$3	**$173**
Profit	$4	$39	$2	**$45**

	Jan	Feb	Mar	Total
Quantity	760	110	2950	**3820**
Sales	$76	$11	$295	**$382**
Expenses	$60	$10	$237	**$307**
Profit	$16	$1	$58	**$75**

	Jan	Feb	Mar	Total
Quantity	1000	2000	3000	**6000**
Sales	$100	$200	$300	**$600**
Expenses	$80	$160	$240	**$480**
Profit	$20	$40	$60	**$120**

An alternative representation of this structure would require a three-dimensional spreadsheet grid, giving rise to the idea that "Time", "Account", and "Region" are dimensions. As further dimensions are added to the system, it becomes very difficult to maintain spreadsheets that correctly represent the multi-dimensional values. Multidimensional databases such as Essbase provide a data store for values that exist, at least conceptually, in a multi-dimensional "hypercube".

Sparsity

As the number and size of dimensions increases, developers of multidimensional databases increasingly face technical problems in the physical representation of data. Say the above example was extended to add a "Customer" and "Product" dimension:

Dimension	Number of dimension values
Accounts	4
Time	4
Region	3
Customer	10,000
Product	5,000

If the multidimensional database reserved storage space for every possible value, it would need to store 2,400,000,000 (4 × 4 × 3 × 10,000 × 5,000) cells. If the software maps each cell as a 64-bit floating point value, this equates to a memory requirement of at least 17 gigabytes (exactly 19.2GB). In practice, of course, the number of combinations of "Customer" and "Product" that contain meaningful values will be a tiny subset of the total space. This property of multi-dimensional spaces is referred to as sparsity.

Aggregation

OLAP systems generally provide for multiple levels of detail within each dimension by arranging the members of each dimension into one or more hierarchies. A time dimension, for example, may be represented as a hierarchy starting with "Total Time", and breaking down into multiple years, then quarters, then months. An Accounts dimension may start with "Profit", which breaks down into "Sales" and "Expenses", and so on.

In the example above, if "Product" represents individual product SKUs, analysts may also want to report using aggregations such as "Product Group", "Product Family", "Product Line", etc. Similarly, for "Customer", natural aggregations may arrange customers according to geographic location or industry.

The number of aggregate values implied by a set of input data can become surprisingly large. If the Customer and Product dimensions are each in fact six "generations" deep, then 36 (6 × 6) aggregate values are affected by a single data point. It follows that if all these aggregate values are to be stored, the amount of space required is proportional to the product of the depth of all aggregating dimensions. For large databases, this can cause the effective storage requirements to be many hundred times the size of the data being aggregated.

Block storage (Essbase Analytics)

Since version 7, Essbase has supported two "storage options" which take advantage of sparsity to minimize the amount of physical memory and disk space required to represent large multidimensional spaces. The Essbase patent describes the original method, which aimed to reduce the amount of physical memory required without increasing the time required to look up closely related values. With the introduction of alternative storage options, marketing materials called this the **Block Storage Option** (**Essbase BSO**), later referred to as **Essbase Analytics**.

Put briefly, Essbase requires the developer to tag dimensions as "dense" or "sparse". The system then arranges data to represent the hypercube into "blocks", where each block comprises a multi-dimensional array made up of "dense" dimensions, and space is allocated for every potential cell in that block. Sparsity is exploited because the system only creates blocks when required. In the example above, say the developer has tagged "Accounts" and "Time" as "dense", and "Region", "Customer", and "Product" as "sparse". If there are, say, 12,000 combinations of Region, Customer and Product that contain data, then only 12,000 blocks will be created, each block large enough to store every possible combination of Accounts and Time. The number of cells stored is therefore 192000 (4 × 4 × 12000), requiring under 2 gigabytes of memory (exact 1,536MB), plus the size of the index used to look up the appropriate blocks.

Because the database hides this implementation from front-end tools (i.e., a report that attempts to retrieve data from non-existent cells merely sees "null" values), the full hypercube can be navigated naturally, and it is possible to load values into any cell interactively.

Calculation engine

Users can specify calculations in Essbase BSO as:

- the aggregation of values through dimensional hierarchies;
- stored calculations on dimension members;
- "dynamically calculated" dimension members; or
- procedural "calculation scripts" that act on values stored in the database.

The first method (dimension aggregation) takes place implicitly through addition, or by selectively tagging branches of the hierarchy to be subtracted, multiplied, divided or ignored. Also, the result of this aggregation can be stored in the database, or calculated dynamically on demand—members must be tagged as "Stored" or "Dynamic Calc." to specify which method is to be used.

The second method (stored calculations) uses a formula against each calculated dimension member — when Essbase calculates that member, the result is stored against that member just like a data value.

The third method (dynamic calculation) is specified in exactly the same format as stored calculations, but calculates a result when a user accesses a value addressed by that member; the system does not store such calculated values.

The fourth method (calculation scripts) uses a procedural programming language specific to the Essbase calculation engine. This type of calculation may act upon any data value in the hypercube, and can therefore perform calculations that cannot be expressed as a simple formula.

A calculation script must also be executed to trigger the calculation of aggregated values or stored calculations as described above—a built-in calculation script (called the "default calculation") can be used to execute this type of calculation.

Aggregate storage (Enterprise Analytics)

Although block storage effectively minimizes storage requirements without impacting retrieval time, it has limitations in its treatment of aggregate data in large applications, motivating the introduction of a second storage engine, named **Aggregate Storage Option** (**Essbase ASO**) or more recently, **Enterprise Analytics**. This storage option makes the database behave much more like an OLAP database, such as SQL Server Analysis Services.

Following a data load, Essbase ASO does not store any aggregate values, but instead calculates them on demand. For large databases, where the time required to generate these values may become inconvenient, the database can materialize one or more aggregate "views", made up of one aggregate level from each dimension (for example, the database may calculate all combinations of the fifth generation of Product with the third generation of Customer), and these views are then used to generate other aggregate values where possible. This process can be partially automated, where the administrator specifies the amount of disk space that may be used, and the database generates views according to actual usage.

This approach has a major drawback in that the cube cannot be treated for calculation purposes as a single large hypercube, because aggregate values cannot be directly controlled, so write-back from front-end tools is limited, and complex calculations that cannot be expressed as MDX expressions are not possible.

Calculation engine

Essbase ASO can specify calculations as:

- the aggregation of values through dimensional hierarchies; or
- dynamically calculated dimension members.

The first method (dimension aggregation) basically duplicates the algorithm used by Essbase BSO.

The second method (dynamic calculations) evaluates MDX expressions against dimension members.

User interface

Many users work with Essbase data using as their interface an add-in for Microsoft Excel (previously also Lotus 1-2-3). The add-in adds a menu to the spreadsheet application that can be used to connect to Essbase databases, retrieve data, and navigate the cube's dimensions ("Zoom in", "Pivot", etc.).[8]

With the release of System 9, Hyperion provided a new user interface add-in for Essbase called "Smart View for Microsoft Office". Smart View provides access to Essbase and other System 9 content for Microsoft Powerpoint, Microsoft Word, Microsoft Outlook as well as supplanting the previous add-in for Microsoft Excel.

In 2005, Hyperion began to offer a visualization tool called Tableau under the name "Hyperion Visual Explorer" [9] (2005). Tableau originated at Stanford University as a government-sponsored research project to investigate new ways for users to interact with relational and OLAP databases.

Other user-facing applications with support for Essbase databases include:

- Hyperion Analyzer (aka Hyperion System 9 BI+ Web Analysis)

- Hyperion Reports (aka Hyperion System 9 BI+ Financial Reporting)
- Hyperion Enterprise Reporting
- Hyperion Intelligence (aka Hyperion System 9 BI+ Interactive Reporting)
- Hyperion SQR (aka Hyperion System 9 BI+ Production Reporting)
- Alphablox
- Arcplan dynaSight (aka Arcplan Enterprise)
- Oracle Business Intelligence Suite Enterprise Edition (aka OBIEE, Siebel Analytics)
- Applied OLAP Dodeca [10]
- CXO-Cockpit Reporting Suite [11]

The previous offerings from Hyperion acquired new names as given below:

Hyperion's previous offerings	Hyperion System 9 BI+ offerings
Hyperion Essbase ASO	Enterprise Analytics
Hyperion Essbase BSO	Essbase Analytics
Hyperion Analyzer	Web Analysis
Hyperion Reports	Financial Reporting
Hyperion Intelligence	Interactive Reporting
Hyperion SQR	Production Reporting
Hyperion Metrics Builder	Enterprise Metrics

APIs are available for C, Visual Basic and Java, and embedded scripting support is available for Perl. The standardised XML for Analysis protocol can query Essbase data sources using the MDX language.

In 2007, Oracle Corporation began bundling Hyperion BI tools into Oracle Business Intelligence Enterprise Edition Plus.

Administrative interface

A number of standard interfaces can administer of Essbase applications:

- *ESSCMD*, the original command line interface for administration commands;
- *MaxL*, a "multi-dimensional database access language" which provides both a superset of ESSCMD commands, but with a syntax more akin to SQL, as well as support for MDX queries;
- *Essbase Application Manager*, the original Microsoft Windows GUI administration client, compatible with versions of Essbase before 7.0;
- *Essbase Administration Services*, later renamed *Analytic Administration Services*, and then back to 'Essbase Administration Services' in v. 9.3.1, the currently supported GUI administration client; and
- *Essbase Integration Server* for maintaining the structure and content of Essbase databases based on data models derived from relational or file-based data sources.

Competitors

There are several significant competitors among the OLAP, analytics products to that of Essbase (HOLAP/MOLAP) on the market, among them Microsoft SQL Server Microsoft Analysis Services, (MOLAP, HOLAP, ROLAP), IBM Cognos (ROLAP), IBM/Cognos/Applix TM1 (MOLAP), Oracle OLAP (ROLAP/MOLAP), MicroStrategy (ROLAP), and EXASolution (ROLAP).

Also note that of the above competitors, including Essbase, all use heterogenous relational (Microsoft SQL Server, Oracle, IBM DB/2, TeraData, Access, etc.) or non-relational data sourcing (Excel, text Files, CSV Files, etc.) to feed the cubes (facts and dimensional data), except for Oracle OLAP which may only use Oracle relational sourcing.

Export and/or product migration of Essbase

As of 2009[2] two options can export Essbase cubes into other formats:

1. CubePort, a commercial conversion application, converts Essbase cubes to the Microsoft SQL Server Analysis Services product. This product performs an object-to-object translation that make up an Essbase cube, including: outline, member formulas, calc scripts, data loading (load rules), report scripts to MDX queries, substitution variables, and security model. It can extract from any platform version of Essbase, including Oracle/Hyperion Essbase on Windows, Unix, AIX, HP UX, Solaris, IBM DB/2 OLAP, or AS/400 Showcase Essbase.
2. OlapUnderground Outline Extractor performs a pure, rudimentary, export of the outline, though it does not directly create any new objects. The output is a simple text file that can be pulled indirectly into other OLAP products, among other uses, such as synchronizing outlines. The Outline Extractor [12] is now maintained, supported and distributed free of charge by Applied OLAP, Inc.

References

v11.1.2.3 documentation: * http://docs.oracle.com/cd/E40248_01/nav/portal_3.htm

v11.1.1.3 documentation: * http://download.oracle.com/docs/cd/E12825_01/nav/portal_3.htm

v9.3.1 documentation: * http://download.oracle.com/docs/cd/E10530_01/doc/index.htm

[1] http://www.oracle.com/technetwork/middleware/essbase
[2] http://en.wikipedia.org/w/index.php?title=Essbase&action=edit
[3] http://www-306.ibm.com/software/data/db2/db2olap/
[4] http://www.regdeveloper.com/2007/01/26/olap_speed/
[5] http://www.oracle.com/technetwork/middleware/essbase/overview/index.html
[6] http://web.archive.org/web/20070927190115/http://www.hyperion.com/company/news/news_releases/press_release_2005_000512.cfm
[7] Earle, Robert J. (1992) "Method and apparatus for storing and retrieving multi-dimensional data in computer memory" (http://patft.uspto.gov/netacgi/nph-Parser?u=/netahtml/srchnum.htm&Sect1=PTO1&Sect2=HITOFF&p=1&r=1&l=50&f=G&d=PALL&s1=5359724.PN.&OS=PN/5359724&RS=PN/5359724). United States Patent 5,359,724 assigned to Arbor Software Corporation.
[8] Hyperion Solutions Corporation (2006). Essbase Database Administrator's Guide (http://dev.hyperion.com/techdocs/essbase/essbase_712/Docs/dbag/dba_html.htm).
[9] http://www.tableausoftware.com/about/press-releases/2007/tableau-software-lands-global-oem-deal-hyperion
[10] http://www.appliedolap.com/products/dodeca
[11] http://essbase.cxo-cockpit.com
[12] http://www.appliedolap.com/free-tools/outline-extractor

External links

- Oracle EPM, BI & Data Warehousing (http://www.oracle.com/bi)
- Oracle Essbase (http://www.oracle.com/technology/products/bi/essbase/index.html)
- Hyperion at Oracle (http://www.oracle.com/hyperion)
- Russian Essbase Site (http://essbase.ru)
- Essbase Custom Relational Calculation (http://essbase.ru/archives/6273)

Microsoft Analysis Services

Microsoft SQL Server Analysis Services (SSAS)

Developer(s)	Microsoft
Stable release	Analysis Services 2012 / December 21, 2010
Operating system	Microsoft Windows
Type	OLAP, Data Mining
License	Microsoft EULA
Website	[1]

Microsoft SQL Server Analysis Services, SSAS, is an Online Analytical Processing, OLAP, data mining and reporting tool in Microsoft SQL Server. SSAS is used as a tool by organizations to analyze and make sense of information possibly spread out across multiple databases, or in disparate tables. Microsoft has included a number of services in SQL Server related to business intelligence and data warehousing. These services include Integration Services and Analysis Services. Analysis Services includes a group of OLAP and data mining capabilities.

History

In 1996, Microsoft began its foray into the OLAP Server business by acquiring the OLAP software technology from Israel-based Panorama Software. Just over two years later, in 1998, Microsoft released OLAP Services as part of SQL Server 7. OLAP Services supported MOLAP, ROLAP, and HOLAP architectures, and it used OLE DB for OLAP as the client access API and MDX as a query language. It could work in client-server mode or offline mode with local cube files.

In 2000, Microsoft released Analysis Services 2000. It was renamed from "OLAP Services" due to the inclusion of data mining services. Analysis Services 2000 was considered an evolutionary release, since it was built on the same architecture as OLAP Services and was therefore backward compatible with it. Major improvements included more flexibility in dimension design through support of parent child dimensions, changing dimensions, and virtual dimensions. Another feature was a greatly enhanced calculation engine with support for unary operators, custom rollups, and cell calculations. Other features were dimension security, distinct count, connectivity over HTTP, session cubes, grouping levels, and many others.

In 2005, Microsoft released the next generation of OLAP and data mining technology as Analysis Services 2005. It maintained backward compatibility on the API level: although applications written with OLE DB for OLAP and MDX continued to work, the architecture of the product was completely different. The major change came to the model in the form of UDM - Unified Dimensional Model.Wikipedia:Please clarify

Timeline

The key events in the history of Microsoft Analysis Services cover a period starting in 1996.

Microsoft Analysis Services Events

Date	Event
1996-07-01	Microsoft opens new team to build an OLAP product, codenamed Plato (permutation of letters from OLAP)
1996-07-15	Panorama Software delegation meets with Microsoft
1996-10-27	Microsoft announces acquisition of Panorama Software development team
1998-11	OLAP Services 7.0 (codename Sphinx) ships
2000-08	Analysis Services 2000 (codename Shiloh) ships
2001-11	XML for Analysis Software Development Kit 1.0 ships
2003-04	ADOMD.NET and XML for Analysis SDK 1.1 ship
2005-10-28	Analysis Services 2005 (codename Yukon) ships
2008-08-06	Analysis Services 2008 (codename Katmai) ships
2012-03-06	Analysis Services 2012

Storage modes

Microsoft Analysis Services takes a neutral position in the MOLAP vs. ROLAP arguments among OLAP products. It allows all the flavors of MOLAP, ROLAP and HOLAP to be used within the same model.

Partition storage modes

- MOLAP - Multidimensional OLAP - Both fact data and aggregations are processed, stored, and indexed using a special format optimized for multidimensional data.
- ROLAP - Relational OLAP - Both fact data and aggregations remain in the relational data source, eliminating the need for special processing.
- HOLAP - Hybrid OLAP - This mode uses the relational data source to store the fact data, but pre-processes aggregations and indexes, storing these in a special format, optimized for multidimensional data.

Dimension storage modes

- MOLAP - dimension attributes and hierarchies are processed and stored in the special format
- ROLAP - dimension attributes are not processed and remain in the relational data source.

APIs and object models

Microsoft Analysis Services supports different sets of APIs and object models for different operations and in different programming environments.

Querying

- XML for Analysis - The lowest level API. It can be used from any platform and in any language that supports HTTP and XML
- OLE DB for OLAP - Extension of OLEDB. COM based and suitable for C/C++ programs on Windows platform.
- ADOMD - Extension of ADO. COM Automation based and suitable for VB programs on Windows platform.
- ADOMD.NET - Extension of ADO.NET. .NET based and suitable for managed code programs on CLR platforms.
- ADO.NET Entity Framework - Entity Framework and LINQ can be used on top of ADOMD.NET (SSAS Entity Framework Provider is required)

Administration and management

- DSO - For AS 2000. COM Automation based and suitable for VB programs on Windows platform.
- AMO - For AS 2005. .NET based and suitable for managed code programs on CLR platforms.

Query languages

Microsoft Analysis Services supports the following query languages

Data definition language (DDL)

DDL in Analysis Services is XML based and supports commands such as <Create>, <Alter>, <Delete>, and <Process>. For data mining models import and export, it also supports PMML.

Data manipulation language (DML)

- MDX - for querying OLAP cubes
- LINQ - for querying OLAP cubes from .NET using ADO.NET Entity Framework and Language INtegrated Query (SSAS Entity Framework Provider is required)
- SQL - small subset of SQL for querying OLAP cubes and dimensions as if they were tables
- DMX - for querying Data Mining models

....

References

[1] http://technet.microsoft.com/en-us/sqlserver/cc510300.aspx

Bibliography

- Sivakumar Harinath, Stephen Quinn: *Professional SQL Server Analysis Services 2005 with MDX*. ISBN 0-7645-7918-5
- Teo Lachev: *Applied Microsoft Analysis Services 2005 : And Microsoft Business Intelligence Platform*. ISBN 0-9766353-0-5
- Reed Jacobson: *Microsoft(r) SQL Server(tm) 2000 Analysis Services Step by Step*. ISBN 0-7356-0904-7
- Claude Seidman: Data Mining with Microsoft SQL Server 2000 Technical Reference." ISBN 0-7356-1271-4
- George Spofford: *MDX-Solutions*. Wiley, 2001, ISBN 0-471-40046-7
- Mosha Pasumansky, Mark Whitehorn, Rob Zare: *Fast Track to MDX*. ISBN 1-84628-174-1
- ZhaoHui Tang, Jamie MacLennan: *Data Mining with SQL Server 2005*. ISBN 0-471-46261-6
- Edward Melomed, Irina Gorbach, Alexander Berger, Py Bateman: *Microsoft SQL Server 2005 Analysis Services*. ISBN 0-672-32782-1
- Chris Webb, Marco Russo, Alberto Ferrary: *Expert Cube Development with Microsoft SQL Server 2008 Analysis Services*. ISBN 1-84719-722-1

External links

- Microsoft Analysis Services (http://www.microsoft.com/sql/technologies/analysis/default.mspx)
- Microsoft OLAP Information (http://www.mosha.com/msolap)
- Microsoft Data Mining Information (http://www.sqlserverdatamining.com)
- Analysis Services public forum (http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=83&SiteID=1)
- Microsoft Analysis Services information hub - FAQs, tools, blogs, forums (http://ssas-info.com)
- Microsoft Analysis Services WIKI (http://ssas-wiki.com)

Mondrian OLAP server

Mondrian OLAP Server

Developer(s)	Pentaho Corporation
Stable release	3.5.0 / November 29, 2012
Operating system	Cross-platform (JVM)
Type	OLAP Server
License	EPL
Website	mondrian.pentaho.org [1]

Mondrian is an open source OLAP (online analytical processing) server, written in Java. It supports the MDX (multidimensional expressions) query language and the XML for Analysis and olap4j [2] interface specifications. It reads from SQL and other data sources and aggregates data in a memory cache.

Mondrian is used for:

- High performance, interactive analysis of large or small volumes of information
- Dimensional exploration of data, for example analyzing sales by product line, by region, by time period
- Parsing the MDX language into Structured Query Language (SQL) to retrieve answers to dimensional queries
- High-speed queries through the use of aggregate tables in the RDBMS
- Advanced calculations using the calculation expressions of the MDX language

Mondrian History

The first public release of Mondrian was on Aug. 9th, 2002.

A release history is here [3]

Mondrian frontends

- JPivot [4]
- Saiku [5]
- Pentaho Analyzer [6]
- Pivot4J [7]
- EasyBI [8]

References

[1] http://mondrian.pentaho.org/
[2] http://www.olap4j.org
[3] http://mondrian.pentaho.com/documentation/roadmap.php#Release_0.3
[4] http://jpivot.sourceforge.net
[5] http://www.analytical-labs.com/
[6] http://www.pentaho.com/products/analysis/
[7] http://mysticfall.github.io/pivot4j/
[8] https://eazybi.com/

OLE DB for OLAP

OLE DB for OLAP (Object Linking and Embedding, Database for Online Analytical Processing abbreviated **ODBO**) is a Microsoft published specification and an industry standard for multi-dimensional data processing. ODBO is the standard application programming interface (API) for exchanging metadata and data between an OLAP server and a client on a Windows platform. ODBO extends the ability of OLE DB to access multi-dimensional (OLAP) data stores.[*citation needed*]

Description

ODBO is the most widely supported, multi-dimensional API to date. Platform-specific to Microsoft Windows, ODBO was specifically designed for Online Analytical Processing (OLAP) systems by Microsoft as an extension to Object Linking and Embedding Database (OLE DB). ODBO uses Microsoft's Component Object Model.[*citation needed*]

ODBO permits independent software vendors (ISVs) and corporate developers to create a single set of standard interfaces that allow OLAP clients to access multi-dimensional data, regardless of vendor or data source. ODBO is currently supported by a wide spectrum of server and client tools.[*citation needed*]

When exposing the ODBO interface, the underlying multi-dimensional database must also support the MDX Query Language. XML for Analysis is a newer interface to MDX Data Sources that is often supported in parallel with ODBO.[*citation needed*]

References

External links

- Microsoft (http://www.microsoft.com/) – Developed ODBO standard
- MSDN (http://msdn2.microsoft.com/en-us/library/ms145506.aspx/) – Multidimensional Expressions Reference
- The OLAP Report (http://www.olapreport.com/) – Independent research resource for organizations buying and implementing OLAP applications
- XML for Analysis (http://www.xmlforanalysis.com/) – Informational web site on ODBO, XMLA, and MDX

Oracle OLAP

The **Oracle Database OLAP Option** implements On-line Analytical Processing (OLAP) within an Oracle database environment. Oracle Corporation markets the Oracle Database OLAP Option as an extra-cost option to supplement the "Enterprise Edition" of its database. (Oracle offers Essbase for customers without the Oracle Database or who require multiple data-sources to load their cubes.)

As of Oracle Database 11g, the Oracle database optimizer can transparently redirect SQL queries to levels within the OLAP Option cubes. The cubes are managed and can take the place of multi-dimensional materialized views, simplifying Oracle data-warehouse management and speeding up query response.

Logical components

The Oracle Database OLAP Option offers:

- an OLAP analytic engine
- workspaces
- an analytic workspace manager (AWM)[1]
- a worksheet environment
- OLAP DML for DDL and DML
- an interface from SQL
- an analytic workspace Java API
- a Java-based OLAP API

Physical implementation

The Oracle database tablespace `CWMLITE` stores OLAPSYS schema objects and integrates Oracle Database OLAP Option with the Oracle Warehouse Builder (OWB). The CWMLITE name reflects the use of CWM — the Common Warehouse Metamodel, which Oracle Corporation refers to as "Common Warehouse Metadata".

External links

- Oracle Technology Network [2] on Oracle OLAP, retrieved 2006-11-30

References

[1] "Introduction to the OLAP Java API" (http://download.oracle.com/docs/cd/B28359_01/olap.111/b28127/intro.htm#sthref19) in *Oracle OLAP Java API Developer's Guide, 11g Release 1 (11.1)*, 2007. Chapter 1.

[2] http://www.oracle.com/technology/products/bi/olap/index.html

Palo (OLAP database)

Palo

Developer(s)	Jedox AG
Stable release	5.0 SR3 / February 7, 2014
Operating system	Cross-platform, Linux, Windows
Type	OLAP
License	Multi-licensing, GPL v2, LGPL, Proprietary
Website	Jedox.com [1]

Palo is a memory resident multidimensional (online analytical processing (OLAP) or multidimensional online analytical processing (MOLAP)) database server and typically used as a business intelligence tool for controlling and budgeting purposes with spreadsheet software acting as the user interface. Beyond the multidimensional data concept, Palo enables multiple users to share one centralised data storage (single version of the truth).

This type of database is suitable to handle complex data models for business management and statistics. Apart from multidimensional queries, data can also be written back and consolidated real-time. To give rapid access to all data, Palo stores them in the memory during run time. The server is available as open source and proprietary software.

Jedox was founded by Kristian Raue in 2002 and developed by Jedox AG, a company based in Freiburg, Germany. The firm currently employs approximately 100 people.

Features

Palo for Excel is an open source plug-in for Microsoft Excel. There is also an open source plug-in for OpenOffice.org named PalOOCa, with Java and web client also available from the JPalo project.[2] Palo can also be integrated into other systems via its client libraries for Java, PHP, C/C++, or .NET Framework. It is fairly easy to communicate with Palo OLAP Server, since it uses representational state transfer (REST).

Starting in October 2008, Palo supports XML for Analysis and MultiDimensional eXpressions (MDX) APIs for connectivity, and OLE DB for OLAP interface which allows standard Excel pivot tables to serve as a client tool. Starting September 2011, Palo supports SDX [3] dialect of LINQ.

Palo also provides a web-based spreadsheet interface called Palo Web.[4]

Architecture

Palo Suite is a tightly integrated framework consisting of: Palo MOLAP Server, Palo ETL Server, Palo Web (Palo Spreadsheet - Connection, User, ETL, File and Report Manager), Palo for Excel, Palo Supervision Server and the Palo Client Libraries.

The Data in Palo database is stored as a cube in the Palo MOLAP server. The Palo Excel Add-In component is used as a service to communicate between the Excel and the Palo MOLAP Server.

References

[1] http://www.jedox.com/en/
[2] Tools, Consulting and Service for Palo (http://www.jpalo.com/en/)
[3] http://ssasefprovider.codeplex.com/wikipage?title=sdx&referringTitle=Glossary
[4] Palo Web (http://www.palo.net/index.php?id=5)

Additional sources

- Bernd Held, Hartmut Erb: *Advanced Controlling mit Excel. Unternehmenssteuerung mit OLAP und PALO, m. CD-ROM.*, Franzis, Poing 2006, ISBN 978-3-7723-7585-9 (in German)
- Stefan Müller, Leif Mergener: *Business Intelligence im Vertrieb auf Basis von Open-Source-Lösungen.* In: Ronald Gleich, Andreas Klein (Hrsg.): Marketing- und Vertriebs-Controlling (Der Controlling-Berater Bd. 11). Haufe-Lexware, Freiburg 2010, ISBN 978-3-648-00489-0. (in German)
- Palo Documentation (http://www.jedox.com/en/jedox-downloads/jedox-documentation-archive.html) - Documentation to Palo (pdf) in the download section of the website
- Online Knowledgebase (http://knowledgebase.jedox.com) Free Online Knowledgebase with full-text search and export as PDF functionality
- Feature Voting Tool (http://feedback.jedox.com) Feedback/Feature Voting Tool
- *b-eye network* article (http://www.b-eye-network.co.uk/view-articles/6759) - Open Source OLAP in the Retail Environment. John Hobson, Feb 2008

External links

- Jedox (http://www.jedox.com/) - Open-Source MOLAP-Server for Windows/Linux with Microsoft Excel-interface, documentation, demos and forum
- Palo project on SourceForge.net (http://sf.net/projects/palo/) - Palo project page on SourceForge.net
- Palo project page on ohloh.net (https://www.ohloh.net/p/p4155/) - Palo project page on ohloh.net

Panorama Software

Panorama Software

Type	Privately held company
Industry	Business intelligence software
Founded	1993
Founder(s)	Rony Ross founded the original co. in 1993; assets were sold to Microsoft in 1996; the current company was founded in 2003 by Janice Anderson and Rony Ross
Headquarters	Toronto, Canada
Area served	Global
Key people	Eynav Azarya, CEO Rony Ross, Executive Chairman and CTO, Oudi Antebi , VP Strategy and Marketing, Kobi Averbuch, VP R&D
Products	Novaview
Website	http://www.panorama.com/

Panorama Software is a Canadian software and consulting company specializing in business intelligence. The company was founded by Rony Ross in Israel in 1993; it relocated its worldwide headquarters to Toronto, Canada in 2003.

Panorama sold its OLAP technology to Microsoft in 1996, which was built into Microsoft OLAP Services and later SQL Server Analysis Services, an integrated component of the Microsoft SQL Server platform.

Panorama supports over 1,600 customers worldwide in industries such as financial services, manufacturing, retail, healthcare, telecommunications and life sciences. Panorama has a wide eco-system of partners in 30 countries, and maintains offices throughout North America, EMEA and Asia.

Products

Necto

The company's main product is a business intelligence suite named **Panorama Necto™**. Deemed a "Business Intelligence 3.0" platform with enhanced user interfaces, increased user adoption, and better self-service for reduced operational costs, Necto presents data in a socially enabled setting. Necto offers the users self-serve data mining and report generation, allowing them to generate their own custom views of the data without having to wait on someone else to run a report for them. It lets users create collaborative "workboards" and visual presentations to provide better context as they put together ad-hoc business teams to address internal issues. The users are able to follow other leaders in the organization and discover those who are attempting to analyze similar data sets. The company contends this focus on social analysis leads, starting with business data and connecting it with the people who are involved in this data.

Necto is a BI application based upon understanding of user behavior, one-click reporting, and collaborative decision making. Panorama positions the product upon three strengths of social business intelligence, insights, and intuitive self-service. The platform supports the social sharing of data, similar to sharing found on consumer oriented social networking sites. Data analysis is treated as "conversations" which can themselves be followed and analyzed. Panorama encourages Necto enterprise users to form cross-departmental teams based on data research behaviors. The solution introduces intelligence to the BI engine through tracking of user behavior and making corresponding adjustments.

Necto is an end-to-end BI suite that includes analytics, custom reporting, intuitive dashboards, and integration with Microsoft Office/SharePoint. The solution is compatible running on top of multiple data sources including spreadsheets, in-memory, OLAP, or relational databases. Integration on Microsoft Azure and optimization with Microsoft SQL Server 2012 platform is also available. Necto™ also integrates with SharePoint and adds a social connection to this integration, combining structured data from Necto's analysis services with unstructured data from SharePoint. It can be scaled up to manage thousands of users and several terabytes of data.

Panorama and Microsoft

Panorama Software is the original developer of the online analytical-processing OLAP technology that Microsoft acquired in 1996 and rebranded as SQL Server Analysis Services, an integrated component of the Microsoft SQL Server platform. Since this acquisition, Panorama has actively offered solutions to support the Microsoft platform as a Microsoft Gold ISV Competency partner. Whether on-premises or as a software-as-a-service, Panorama integrates with Microsoft to meet the demands from the largest of enterprise users.

References

External links

- Panorama Software web site (http://www.panorama.com)
- Free Necto Trial (http://www.panorama.com/trial/necto-trial-1.php)
- Extending the Reach of Business Intelligence (http://www.dmreview.com/article_sub.cfm?articleId=6952) - *DM Review*, July 2003 profile of Rony Ross and history of Panorama Software

ProClarity

ProClarity Corporation was a software company specializing in business intelligence and data analysis applications.

The company was founded in 1995 as Knosys Inc. in Boise, Idaho. The company was renamed ProClarity Corporation after its primary commercial software product, "ProClarity", in 2001.

ProClarity's software products integrated tightly with Microsoft Analysis Services.

Among ProClarity's more than 2,000 global clients were AT&T, Ericsson, Hewlett-Packard, Home Depot, Pennzoil QuakerState, Reckitt Benckiser, Roche, Siemens, USDA, Verizon, and Wells Fargo.

On April 3, 2006, Microsoft announced the acquisition of ProClarity.[1] The company was gradually folded into Microsoft's Business Division while a final major version, ProClarity 6.3, was released in 2007 [2] Additional business intelligence components, such as PerformancePoint Services for SharePoint 2010, and business intelligence improvements in existing products, as in Excel 2013, were released by the division in subsequent years.

References

[1] " Microsoft Agrees to Acquire ProClarity, Enhancing Business Intelligence Offering (http://www.microsoft.com/presspass/press/2006/apr06/04-03ProClarityPR.mspx)", Microsoft press release, April 3, 2006

[2] " ProClarity Analytics Server 6.3 is now available (http://support.microsoft.com/kb/934052/en-us)", Microsoft support article, March 21, 2007

External links

- Microsoft business intelligence (http://www.microsoft.com/bi)

SAP BI Accelerator

In computing, the **SAP BW Accelerator** is a computer appliance - preinstalled software on predefined hardware - which is used to speed up OLAP queries. The software was initially known as the BI Accelerater.

SAP BW Accelerator includes indexes that are vertically inverted reproductions of all the data included in InfoCubes (i.e., fact and dimension tables as well as master data). Note that there is no relational or other database management systems in BW Accelerator. There is only a file system, and indexes are essentially held as flat files. The second primary component of SAP BW Accelerator is the engine that processes the queries in memory - it uses the SAP TREX search engine. The software is running on an expandable rack of blade servers. The operating system used for BW Accelerator is 64-bit SUSE Linux Enterprise Server (SLES).

Hardware partners

The software is optimized for specific hardware and operating system combinations.

The list of partners which deliver the appliance is:

- IBM BW Accelerator solution [1]
- HP
- Fujitsu Siemens
- Sun BI Accelerator Offering [2]

The product is now going out to the free market; i.e. it is not only the partners that are allowed to offer this product. As always when becoming more open markets, lower prices might therefore be expected.

References

[1] http://www.ibm.com/solutions/sap/us/en/landing/N367059H83793W50.html?cm_sp=MTE8587

[2] http://www.sun.com/third-party/global/sap/solutions/bia.jsp

SAP NetWeaver Business Intelligence

SAP NetWeaver Business Warehouse (**SAP NetWeaver BW**) is business intelligence (BI), analytical, reporting and data warehousing software produced by SAP AG. It was originally named SAP BIW (Business Information Warehouse), then abbreviated to SAP BW, but is now known as "SAP BI" at the end user level. In contrast, "BW" is still used to describe the underlying data warehouse area and accelerator components. It is often used by companies who run their business on SAP's operational systems.

BW is part of the SAP NetWeaver technology. Other components of SAP NetWeaver include SAP Enterprise Portal (EP, called SAP NetWeaver Portal as of Release 7.0), Web Application Server (WAS), SAP Process Integration (PI, or previously XI, i.e. eXchange Infrastructure) and Master Data Management (MDM). It also includes end-user reporting tools such as Report Designer, BEx Query Designer, BEx Web Application Designer and BEx Analyzer.

Structure

SAP's BI products have layers:

- Extract, transform, load (ETL) layer - responsible for extracting data from a specific source, applying transformation rules, and loading it into the Data Warehouse Area.
- Data Warehouse Area - responsible for storing the information in various types of structures (e.g. Data Store Objects, InfoObjects and multidimensional structures called InfoCubes that follows star schema design).
- Reporting - for accessing the information in data warehouse area and presenting it in a user-friendly manner to the analyst or business user.
- Planning and analysis - Provides capabilities for the user to run simulations and perform tasks such as budget calculations.

SAP BW contains pre-defined business content in the form of InfoCubes, Info Objects, authorization roles, and queries. The business content can be modified to meet an organization's specific requirements; however, this requires a longer process of customization of the pre-defined elements.

Security

User management

The following types of general user profiles exist:

- Executives and knowledge workers
- Information consumers

However, roles and authorizations can be customized significantly.

Authentications and Single Sign-On

The following are the most common forms of authentication:

- User ID and Password
- Secure Network Communications (SNC)
- SAP Logon Ticket
- Client Certificates (e.g., x.509)

SAP NetWeaver Single Sign-On Environment

The SAP NetWeaver Portal is the main entry point within SAP NetWeaver. In order to integrate SAP NetWeaver Business Intelligence, the following two conditions must be satisfied: (note that SAP logon tickets are being used in this example) 1) BI trusts SAP logon tickets from EP because the public key of the EP certificate has been imported into BI. 2) EP trusts SAP logon tickets from BI because the public key of the BI certificate has been imported into EP.

Authorizations

Companies have to define who has access to which data. An authorization allows a user to perform a certain activity on a certain object in the BI system. There are two authorizations concepts to consider for BI: standard authorizations[1] and analysis authorizations.[2]

Communication channel security

The communication channel used depends on different cases[3]

- Front end and application server uses RFC
- Application server and application server uses RFC
- SAP J2EE Engine and application server uses RFC
- SAP router and application server uses RFC
- Connection to database uses RFC
- Web browser and application server uses HTTP, HTTPS, and SOAP

Encrypted communications

RFC communications is not encrypted. In order to encrypt RFC communications, the SAP environment must use Secure Network Communications (SNC) or the SAP Cryptographic Library.[4] SAP recommends the usage of x.509 certificates.[5]

Data storage

Data can be protected from being accessed by an authorized end user by assigning analysis authorizations. Data is not protected under BI default settings.

Transactional data is stored in a **Datastore** or an **InfoCube**. A Datastore serves as a storage location for transaction data at an atomic level. The data in a datastore is stored in transparent flat database tables.

An InfoCube is a set of relational tables arranged according to the star schema: A large fact table in the middle surrounded by several dimension tables.5616

History

The 7.0 version of BW was released in June 2006 as part of the SAP NetWeaver 7.0 (also known as 2004s). This release included features such as next-generation reporting and analytical features, data warehousing enhancements, and a memory resident option for improving query performance called "BI Accelerator" (it has since been renamed BW Accelerator). The BW Accelerator comes as an external appliance, i.e. complete hardware with pre-installed software, and requires a separate licence fee. BW Accelerator is licenced per blade server and 16 GB increments of memory.

SAP acquired Business Objects, one of the premier business intelligence software providers, via tender offers executed December 2007-January 2008.[6] SAP has indicated that Business Objects will operate as an independent entity to preserve the principle of application agnosticism, but also promised a tighter integration between SAP BI and Business Objects. A new BI roadmap was recently released by the combined entity.[7]

References

[1] SAP Library: Standard Authorizations (http://help.sap.com/saphelp_nw70ehp1/helpdata/en/be/076f3b6c980c3be10000000a11402f/content.htm)
[2] SAP Library: Analysis Authorizations (http://help.sap.com/saphelp_nw2004s/helpdata/en/66/019441b8972e7be10000000a1550b0/content.htm)
[3] SAP NetWeaver BI Security Guide (http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/80be74ea-7d55-2a10-22a0-ff664d1454fc?QuickLink=index&overridelayout=true)
[4] SAP Cryptographic Library (http://help.sap.com/saphelp_nw70/helpdata/en/ca/cbca6b937ea344a9a3be78a128a803/content.htm)
[5] Single Sign-On Technology for SAP Enterprises (http://www.itsecuritystandard.com/blog/?p=1612)
[6] SAP Acquires Business Objects in Friendly Takeover (http://www.sap.com/about/investor/bobj/index.epx)
[7] Business Intelligence Platform Roadmap (http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/b050d131-7ebc-2a10-57a3-99f7554953bb?QuickLink=index&overridelayout=true&26345329490306)

Further reading

- Shiralkar; Shreekant, Amol Palekar, Bharat (2010). ***A-Practical-Guide-to-SAP-NetWeaver-Business-Warehouse-7.0*** (First Edition ed.). SAP Press. ISBN 978-1-59229-323-0.
- McDonald; Wilmsmeier, Dixon, Inmon (2006). *Mastering the SAP Business Information Warehouse* (Second Edition ed.). Wiley. ISBN 0-471-21971-1.
- Mehrwald, Christian (2003). *SAP Business Information Warehouse 3*. Heidelberg: dpunkt-Verlag. ISBN 3-89864-179-1.
- Scott, Peter (2006). *SAP Business Explorer (BEx) Tools*. SAP Press. ISBN 1-59229-086-8.
- Scott, Peter (2009). *SAP Business Explorer (BEx) Tools 2nd Edition*. SAP Press. ISBN 1-59229-279-8.
- Boeke, Joerg (2009). *SAP BW 7.x Reporting - Visualize your data*. Createspace USA 2009. ISBN 978-1-4486-0626-9.
- Shiralkar; Shreekant, Amol Palekar (2012). ***Supply Chain Analytics With SAP NetWeaver Business Warehouse*** (First Edition ed.). Tata McGraw-Hill Education. ISBN 978-1-2590-0608-1.

External links

- SAP NetWeaver Business Intelligence (http://www.sap.com/solutions/netweaver/components/bi/index.epx)

NEVOD DMB

NEVOD

Company / developer	RELEX Group
License	commercial
Official website	[1]

Data Marts Builder **NEVOD** a product of RELEX Group, created in 1996.

Description

DMB[2] **NEVOD** - object-oriented repository with possibility of search and analysis of structured information (Ad hoc inquiries and semantic net visualization). It is used mainly by security services of commercial organisations and divisions of the Ministry of Internal Affairs of the Russian Federation. For ten consecutive years DMB NEVOD has been presented at the international tradeshow of computer technologies "SofTool", at the RELEX Group exposition.[3]

Technical Characteristics of NEVOD DMB

Parameter	**Value**
Maximum number of accounted object types	65000
Maximum number of objects	1 000 000 000
Maximum number of single-valued attributes for an object	124
Maximum number of many-valued attributes for an object	65000
Maximum number of values in a many-valued attribute	1 000 000 000
Maximum number of users	65000
Possible attribute types	Text (up to 512 Unicode characters) Dictionary value (up to 512 Unicode characters) Date Time Integer Floating-point real number Fixed-point real number Attribute group Object reference Multimedia (up to 2 Gb)
Maximum number of dictionaries	32000
Dictionary views	Linear list Synonym tree Hierarchic dictionary

Types of conditions imposed on attributes and used in queries	Equal Not equal More Less More or equal Less or equal Between According to the hierarchic dictionary Empty Begin with End with Contains Template
Types of conditions imposed on connections and used in queries	Use connection type Mandatory connection Implied connection
Types of logical operations connecting the conditions imposed on attributes	AND, OR, NOT
Maximum number of conditions in a query	32000
Maximum number of queries stored	65000
Maximum number of query results stored	65000
Maximum number of input/output forms stored	65000
Maximum number of connection charts stored	65000
Maximum number of statistic reports stored	65000
Maximum number of XSL reports stored	65000
Maximum number of servers for executing a query	4000
Maximum number of identification rules	65000
Import/export format	XML

References

[1] http://linter.ru/en/other/nevod/

[2] http://toolserver.org/%7Edispenser/cgi-bin/dab_solver.py?page=NEVOD_DMB&editintro=Template:Disambiguation_needed/editintro&client=Template:Dn

[3] CeBIT-2007.ru Partnerland Russland (http://www.cebit-2007.ru/firmen/morenews.php?iditem=3)

External links

- Official site of RELEX Group (http://www.relex.ru/main_eng.php)

Data Transforamtion

Extract, transform, load

In computing, **extract, transform, and load** (**ETL**) refers to a process in database usage and especially in data warehousing that:

- Extracts data from outside sources
- Transforms it to fit operational needs, which can include quality levels
- Loads it into the end target (database, more specifically, operational data store, data mart, or data warehouse)

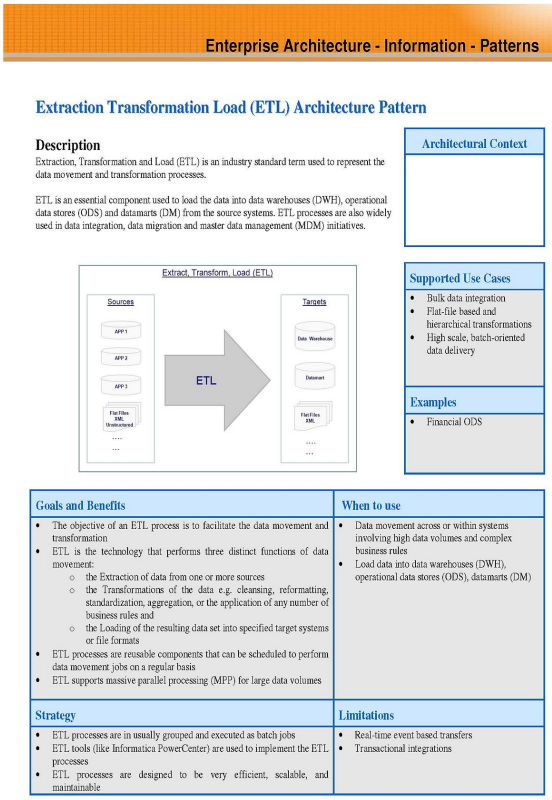
ETL systems are commonly used to integrate data from multiple applications, typically developed and supported by different vendors or hosted on separate computer hardware. The disparate systems containing the original data are frequently managed and operated by different employees. For example a cost accounting system may combine data from payroll, sales and purchasing.

Extract

The first part of an ETL process involves extracting the data from the source systems. In many cases this is the most challenging aspect of ETL, since extracting data correctly sets the stage for how subsequent processes go further.

Most data warehousing projects consolidate data from different source systems. Each separate system may also use a different data organization and/or format. Common data source formats are relational databases and flat files, but may include non-relational database structures such as Information Management System (IMS) or other data structures such as Virtual Storage Access Method (VSAM) or Indexed Sequential Access Method (ISAM), or even fetching from outside sources such as through web spidering or screen-scraping. The streaming of the extracted data source and load on-the-fly to the destination database is another way of performing ETL when no intermediate data storage is required. In general, the goal of the extraction phase is to convert the data into a single format appropriate for transformation processing.

An intrinsic part of the extraction involves the parsing of extracted data, resulting in a check if the data meets an expected pattern or structure. If not, the data may be rejected entirely or in part.



ETL Architecture Pattern

Transform

The transform stage applies a series of rules or functions to the extracted data from the source to derive the data for loading into the end target. Some data sources require very little or even no manipulation of data. In other cases, one or more of the following transformation types may be required to meet the business and technical needs of the target database:

- Selecting only certain columns to load (or selecting null columns not to load). For example, if the source data has three columns (also called attributes), roll_no, age, and salary, then the selection may take only roll_no and salary. Similarly, the selection mechanism may ignore all those records where salary is not present (salary = null).
- Translating coded values (*e.g.*, if the source system stores 1 for male and 2 for female, but the warehouse stores M for male and F for female)
- Encoding free-form values (*e.g.*, mapping "Male" to "M")
- Deriving a new calculated value (*e.g.*, sale_amount = qty * unit_price)
- Sorting
- Joining data from multiple sources (*e.g.*, lookup, merge) and deduplicating the data
- Aggregation (for example, rollup — summarizing multiple rows of data — total sales for each store, and for each region, etc.)
- Generating surrogate-key values
- Transposing or pivoting (turning multiple columns into multiple rows or vice versa)
- Splitting a column into multiple columns (*e.g.*, converting a comma-separated list, specified as a string in one column, into individual values in different columns)
- Disaggregation of repeating columns into a separate detail table (*e.g.*, moving a series of addresses in one record into single addresses in a set of records in a linked *address* table)
- Lookup and validate the relevant data from tables or referential files for slowly changing dimensions.
- Applying any form of simple or complex data validation. If validation fails, it may result in a full, partial or no rejection of the data, and thus none, some or all the data is handed over to the next step, depending on the rule design and exception handling. Many of the above transformations may result in exceptions, for example, when a code translation parses an unknown code in the extracted data.

Load

The load phase loads the data into the end target, usually the data warehouse (DW). Depending on the requirements of the organization, this process varies widely. Some data warehouses may overwrite existing information with cumulative information; frequently, updating extracted data is done on a daily, weekly, or monthly basis. Other data warehouses (or even other parts of the same data warehouse) may add new data in a historical form at regular intervals—for example, hourly. To understand this, consider a data warehouse that is required to maintain sales records of the last year. This data warehouse overwrites any data older than a year with newer data. However, the entry of data for any one year window is made in a historical manner. The timing and scope to replace or append are strategic design choices dependent on the time available and the business needs. More complex systems can maintain a history and audit trail of all changes to the data loaded in the data warehouse.

As the load phase interacts with a database, the constraints defined in the database schema — as well as in triggers activated upon data load — apply (for example, uniqueness, referential integrity, mandatory fields), which also contribute to the overall data quality performance of the ETL process.

- For example, a financial institution might have information on a customer in several departments and each department might have that customer's information listed in a different way. The membership department might list the customer by name, whereas the accounting department might list the customer by number. ETL can bundle all this data and consolidate it into a uniform presentation, such as for storing in a database or data warehouse.

- Another way that companies use ETL is to move information to another application permanently. For instance, the new application might use another database vendor and most likely a very different database schema. ETL can be used to transform the data into a format suitable for the new application to use.
- An example of this would be an Expense and Cost Recovery System (ECRS) such as used by accountancies, consultancies and lawyers. The data usually ends up in the time and billing system, although some businesses may also utilize the raw data for employee productivity reports to Human Resources (personnel dept.) or equipment usage reports to Facilities Management.

Real-life ETL cycle

The typical real-life ETL cycle consists of the following execution steps:

1. Cycle initiation
2. Build reference data
3. Extract (from sources)
4. Validate
5. Transform (clean, apply business rules, check for data integrity, create aggregates or disaggregates)
6. Stage (load into staging tables, if used)
7. Audit reports (for example, on compliance with business rules. Also, in case of failure, helps to diagnose/repair)
8. Publish (to target tables)
9. Archive
10. Clean up

Challenges

ETL processes can involve considerable complexity, and significant operational problems can occur with improperly designed ETL systems.

The range of data values or data quality in an operational system may exceed the expectations of designers at the time validation and transformation rules are specified. Data profiling of a source during data analysis can identify the data conditions that must be managed by transform rules specifications. This leads to an amendment of validation rules explicitly and implicitly implemented in the ETL process.

Data warehouses are typically assembled from a variety of data sources with different formats and purposes. As such, ETL is a key process to bring all the data together in a standard, homogeneous environment.

Design analysts should establish the scalability of an ETL system across the lifetime of its usage. This includes understanding the volumes of data that must be processed within service level agreements. The time available to extract from source systems may change, which may mean the same amount of data may have to be processed in less time. Some ETL systems have to scale to process terabytes of data to update data warehouses with tens of terabytes of data. Increasing volumes of data may require designs that can scale from daily batch to multiple-day micro batch to integration with message queues or real-time change-data capture for continuous transformation and update.

Performance

ETL vendors benchmark their record-systems at multiple TB (terabytes) per hour (or ~1 GB per second) using powerful servers with multiple CPUs, multiple hard drives, multiple gigabit-network connections, and lots of memory. The fastest ETL record is currently held by Syncsort,[1] Vertica and HP at 5.4TB in under an hour, which is more than twice as fast as the earlier record held by Microsoft and Unisys.

In real life, the slowest part of an ETL process usually occurs in the database load phase. Databases may perform slowly because they have to take care of concurrency, integrity maintenance, and indices. Thus, for better performance, it may make sense to employ:

- Direct Path Extract method or bulk unload whenever is possible (instead of querying the database) to reduce the load on source system while getting high speed extract
- Most of the transformation processing outside of the database
- Bulk load operations whenever possible.

Still, even using bulk operations, database access is usually the bottleneck in the ETL process. Some common methods used to increase performance are:

- Partition tables (and indices). Try to keep partitions similar in size (watch for `null` values that can skew the partitioning).
- Do all validation in the ETL layer before the load. Disable integrity checking (`disable constraint ...`) in the target database tables during the load.
- Disable triggers (`disable trigger ...`) in the target database tables during the load. Simulate their effect as a separate step.
- Generate IDs in the ETL layer (not in the database).
- Drop the indices (on a table or partition) before the load - and recreate them after the load (SQL: `drop index ...; create index ...`).
- Use parallel bulk load when possible — works well when the table is partitioned or there are no indices. Note: attempt to do parallel loads into the same table (partition) usually causes locks — if not on the data rows, then on indices.
- If a requirement exists to do insertions, updates, or deletions, find out which rows should be processed in which way in the ETL layer, and then process these three operations in the database separately. You often can do bulk load for inserts, but updates and deletes commonly go through an API (using SQL).

Whether to do certain operations in the database or outside may involve a trade-off. For example, removing duplicates using `distinct` may be slow in the database; thus, it makes sense to do it outside. On the other side, if using `distinct` significantly (x100) decreases the number of rows to be extracted, then it makes sense to remove duplications as early as possible in the database before unloading data.

A common source of problems in ETL is a big number of dependencies among ETL jobs. For example, job "B" cannot start while job "A" is not finished. One can usually achieve better performance by visualizing all processes on a graph, and trying to reduce the graph making maximum use of parallelism, and making "chains" of consecutive processing as short as possible. Again, partitioning of big tables and of their indices can really help.

Another common issue occurs when the data is spread between several databases, and processing is done in those databases sequentially. Sometimes database replication may be involved as a method of copying data between databases - and this can significantly slow down the whole process. The common solution is to reduce the processing graph to only three layers:

- Sources
- Central ETL layer
- Targets

This allows processing to take maximum advantage of parallel processing. For example, if you need to load data into two databases, you can run the loads in parallel (instead of loading into 1st - and then replicating into the 2nd).

Sometimes processing must take place sequentially. For example, dimensional (reference) data is needed before one can get and validate the rows for main "fact" tables.

Parallel processing

A recent[2] development in ETL software is the implementation of parallel processing. This has enabled a number of methods to improve overall performance of ETL processes when dealing with large volumes of data.

ETL applications implement three main types of parallelism:

- **Data**: By splitting a single sequential file into smaller data files to provide parallel access.
- **Pipeline**: Allowing the simultaneous running of several components on the same data stream. For example: looking up a value on record 1 at the same time as adding two fields on record 2.
- **Component**: The simultaneous running of multiple processes on different data streams in the same job, for example, sorting one input file while removing duplicates on another file.

All three types of parallelism usually operate combined in a single job.

An additional difficulty comes with making sure that the data being uploaded is relatively consistent. Because multiple source databases may have different update cycles (some may be updated every few minutes, while others may take days or weeks), an ETL system may be required to hold back certain data until all sources are synchronized. Likewise, where a warehouse may have to be reconciled to the contents in a source system or with the general ledger, establishing synchronization and reconciliation points becomes necessary.

Rerunnability, recoverability

Data warehousing procedures usually subdivide a big ETL process into smaller pieces running sequentially or in parallel. To keep track of data flows, it makes sense to tag each data row with "row_id", and tag each piece of the process with "run_id". In case of a failure, having these IDs help to roll back and rerun the failed piece.

Best practice also calls for *checkpoints*, which are states when certain phases of the process are completed. Once at a checkpoint, it is a good idea to write everything to disk, clean out some temporary files, log the state, and so on.

Virtual ETL

As of 2010[2] data virtualization had begun to advance ETL processing. The application of data virtualization to ETL allowed solving the most common ETL tasks of data migration and application integration for multiple dispersed data sources. So-called Virtual ETL operates with the abstracted representation of the objects or entities gathered from the variety of relational, semi-structured and unstructured data sources. ETL tools can leverage object-oriented modeling and work with entities' representations persistently stored in a centrally located hub-and-spoke architecture. Such a collection that contains representations of the entities or objects gathered from the data sources for ETL processing is called a metadata repository and it can reside in memory[3] or be made persistent. By using a persistent metadata repository, ETL tools can transition from one-time projects to persistent middleware, performing data harmonization and data profiling consistently and in near-real time.[*citation needed*]

Dealing with keys

Keys are some of the most important objects in all relational databases, as they tie everything together. A primary key is a column that identifies a given entity, where a foreign key is a column in another table that refers a primary key. These keys can also be made of several columns, in which case they are composite keys. In many cases the primary key is an auto generated integer that has no meaning for the business entity being represented, but solely exists for the purpose of the relational database - commonly referred to as a surrogate key.

As there is usually more than one data source being loaded into the warehouse, the keys are an important concern to be addressed.
Your customers might be represented in several data sources, and in one their SSN (Social Security Number) might be the primary key, their phone number in another and a surrogate in the third. All of the customers information needs to be consolidated into one dimension table.

A recommended way to deal with the concern is to add a warehouse surrogate key, which is used as a foreign key from the fact table.[4]

Usually updates occur to a dimension's source data, which obviously must be reflected in the data warehouse.
If the primary key of the source data is required for reporting, the dimension already contains that piece of information for each row. If the source data uses a surrogate key, the ware house must keep track of it even though it is never used in queries or reports.

That is done by creating a lookup table that contains the warehouse surrogate key and the originating key.[5] This way the dimension is not polluted with surrogates from various source systems, while the ability to update is preserved.

The lookup table is used in different ways depending on the nature of the source data. There are 5 types to consider,[6] where three selected ones are included here:

Type 1:

- The dimension row is simply updated to match the current state of the source system. The warehouse does not capture history. The lookup table is used to identify the dimension row to update or overwrite.

Type 2:

- A new dimension row is added with the new state of the source system. A new surrogate key is assigned. Source key is no longer unique in the lookup table.

Fully logged:

- A new dimension row is added with the new state of the source system, while the previous dimension row is updated to reflect it is no longer active and record time of deactivation.

Tools

Programmers can set up ETL processes using almost any programming language, but building such processes from scratch can become complex. Increasingly, companies are buying ETL tools to help in the creation of ETL processes.[7]

By using an established ETL framework, one may increase one's chances of ending up with better connectivity and scalability.[*citation needed*] A good ETL tool must be able to communicate with the many different relational databases and read the various file formats used throughout an organization. ETL tools have started to migrate into Enterprise Application Integration, or even Enterprise Service Bus, systems that now cover much more than just the extraction, transformation, and loading of data. Many ETL vendors now have data profiling, data quality, and metadata capabilities. A common use case for ETL tools include converting CSV files to formats readable by relational databases. A typical translation of millions of records is facilitated by ETL tools that enable users to input csv-like data feeds/files and import it into a database with as little code as possible.

ETL Tools are typically used by a broad range of professionals - from students in computer science looking to quickly import large data sets to database architects in charge of company account management, ETL Tools have become a convenient tool that can be relied on to get maximum performance. ETL tools in most cases contain a GUI that helps users conveniently transform data as opposed to writing large programs to parse files and modify data types—which ETL tools facilitate as much as possible.[*citation needed*]

References

[1] "New ETL World Record: 5.4 TB Loaded in Under 1 Hour - Syncsort" (http://www.syncsort.com/Portals/0/Resources/Solution/DMX_Solution_WorldRecord.pdf)
[2] http://en.wikipedia.org/w/index.php?title=Extract,_transform,_load&action=edit
[3] Virtual ETL (http://itnewscast.com/etl-architecture-and-business-models)
[4] (Kimball, The Data Warehouse Lifecycle Toolkit, p 332)
[5] Golfarelli/Rizzi, Data Warehouse Design, p 291
[6] Golfarelli/Rizzi, Data Warehouse Design, p 291
[7] ETL poll produces unexpected results (http://www.etltool.com/nieuws/2715_ETL_poll_produces_unexpected_results.htm)

Staging (data)

A **staging area**, or **landing zone**, is an intermediate storage area used for data processing during the extract, transform and load (ETL) process. The data staging area sits between the data source(s) and the data target(s), which are often data warehouses, data marts or other data repositories.[1]

Data staging areas are often transient in nature, with their contents being erased prior to running an ETL process or immediately following successful completion of an ETL process. There are staging area architectures, however, which are designed to hold data for extended periods of time for archival or troubleshooting purposes.

Implementation

Staging areas can be implemented in the form of tables in relational databases, text-based flat files (or XML files) stored in file systems or proprietary formatted binary files stored in file systems.[2] Staging area architectures range in complexity from a set of simple relational tables in a target database to self-contained database instances or file systems.[3] Though the source systems and target systems supported by ETL processes are often relational databases, the staging areas that sit between data sources and targets need not also be relational databases.[4]

Functions

Staging areas can be designed to provide many benefits, but the primary motivations for their use are to increase efficiency of ETL processes, ensure data integrity and support data quality operations. The functions of the staging area include the following:

Consolidation

One of the primary functions performed by a staging area is consolidation of data from multiple source systems. In performing this function the staging area acts as a large "bucket" in which data from multiple source systems can be temporarily placed for further processing. It is common to tag data in the staging area with additional metadata indicating the source of origin and timestamps indicating when the data was placed in the staging area.

Alignment

Aligning data includes standardization of reference data across multiple source systems and validation of relationships between records and data elements from different sources. Data alignment in the staging area is a function closely related to, and acting in support of, master data management capabilities.[5]

Minimizing contention

The staging area and ETL processes it supports are often designed with a goal of minimizing contention within source systems. Copying required data from source systems to the staging area in one shot is often more efficient than retrieving individual records (or small sets of records) on a one-off basis. The former method takes advantage of technical efficiencies, such as data streaming technologies, reduced overhead through minimizing the need to break and re-establish connections to source systems and optimization of concurrency lock management on multi-user source systems. By copying the source data from the source systems and waiting to perform intensive processing and transformation in the staging area, the ETL process exercises a great degree of control over concurrency issues during processing.

Independent scheduling/multiple targets

The staging area can support hosting of data to be processed on independent schedules, and data that is meant to be directed to multiple targets. In some instances data might be pulled into the staging area at different times to be held and processed all at once. This situation might occur when enterprise processing is done across multiple time zones each night, for instance. In other cases data might be brought into the staging area to be processed at different times; or the staging area may be used to push data to multiple target systems. As an example, daily operational data might be pushed to an operational data store (ODS) while the same data may be sent in a monthly aggregated form to a data warehouse.

Change detection

The staging area supports efficient change detection operations against target systems. This functionality is particularly useful when the source systems do not support reliable forms of change detection, such as system-enforced timestamping, change tracking or change data capture (CDC).

Cleansing data

Data cleansing includes identification and removal (or update) of invalid data from the source systems. The ETL process utilizing the staging area can be used to implement business logic to identify and handle "invalid" data. Invalid data is often defined through a combination of business rules and technical limitations. Technical constraints may additionally be placed on staging area structures (such as table constraints in a relational database) to enforce data validity rules.

Aggregate precalculation

Precalculation of aggregates, complex calculations and application of complex business logic may be done in a staging area to support highly responsive service level agreements (SLAs) for summary reporting in target systems.

Data archiving and troubleshooting

Data archiving can be performed in, or supported by, a staging area. In this scenario the staging area can be used to maintain historical records during the load process, or it can be used to push data into a target archive structure. Additionally data may be maintained within the staging area for extended periods of time to support technical troubleshooting of the ETL process.

References

[1] Oracle 9i Data Warehousing Guide, *Data Warehousing Concepts* (http://docs.oracle.com/cd/B10501_01/server.920/a96520/concept.htm), Oracle Corp.
[2] **Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals**, p. 137-138, Paulraj Ponniah, 2001.
[3] *BI Experts: Big Data and Your Data Warehouse's Data Staging Area* (http://tdwi.org/articles/2012/07/10/big-data-staging-area.aspx), The Data Warehousing Institute, Phillip Russom, 2012.
[4] Is Data Staging Relational? (http://www.kimballgroup.com/1998/04/02/is-data-staging-relational), Ralph Kimball, 1998.
[5] **Master Data Management in Practice: Achieving True Customer MDM**, Dalton Cervo and Mark Allen, 2011.

Vocabulary-based transformation

In metadata, a **vocabulary-based transformation (VBT)** is a transformation aided by the use of a semantic equivalence statements within a controlled vocabulary.

Many organizations today require communication between one or more computers. Although many standards exist to exchange data between computers such as HTML or email, there are still much structured information that needs to be exchanged between computers that is not standardized. The process of mapping one source of data into another is often a slow and labor-intensive process.

VBT is a possible way to avoid much of the time and cost of manual data mapping using traditional Extract, transform, load technologies.

History

The term *vocabulary-based transformation* was first defined by Roy Shulte of the Gartner Group around May 2003 and appeared in annual "hype-cycle" for integration.

Application

VBT allows computer systems integrators to more automatically "look up" the definitions of data elements in a centralized data dictionary and use that definition and the equivalent mappings to transform that data element into a foreign namespace.

The Web Ontology Language (OWL) language also support three semantic equivalence statements.

Companies or products

- IONA Technologies
- Contivo [1]
- enLeague Systems
- ItemField
- Unicorn Solutions
- Vitria Technology
- Zonar

External links

- Gartner Glossary of Terms [2] Gartner definition Vocabulary-based transformation
- Gartner Hype Cycle 2003 [3]

References

[1] http://www.contivo.com
[2] http://www.gartner.com/6_help/glossary/GlossaryV.jsp
[3] http://www.sun.com/service/openwork/analyst/Gartner_Hype_Cycle.pdf

Surrogate key

A **surrogate key** in a database is a unique identifier for either an *entity* in the modeled world or an *object* in the database. The surrogate key is *not* derived from application data.

Definition

There are at least two definitions of a surrogate:

Surrogate (1) – Hall, Owlett and Codd (1976)

> A surrogate represents an *entity* in the outside world. The surrogate is internally generated by the system but is nevertheless visible to the user or application.

Surrogate (2) – Wieringa and De Jonge (1991)

> A surrogate represents an *object* in the database itself. The surrogate is internally generated by the system and is invisible to the user or application.

The *Surrogate (1)* definition relates to a data model rather than a storage model and is used throughout this article. See Date (1998).

An important distinction between a surrogate and a primary key depends on whether the database is a current database or a temporal database. Since a *current database* stores only *currently* valid data, there is a one-to-one correspondence between a surrogate in the modeled world and the primary key of the database. In this case the surrogate may be used as a primary key, resulting in the term *surrogate key*. In a temporal database, however, there is a many-to-one relationship between primary keys and the surrogate. Since there may be several objects in the database corresponding to a single surrogate, we cannot use the surrogate as a primary key; another attribute is required, in addition to the surrogate, to uniquely identify each object.

Although Hall *et al.* (1976) say nothing about this, othersWikipedia:Citing sources have argued that a surrogate should have the following characteristics:

- the value is unique system-wide, hence never reused
- the value is system generated
- the value is not manipulable by the user or application
- the value contains no semantic meaning
- the value is not visible to the user or application
- the value is not composed of several values from different domains.

Surrogates in practice

In a current database, the surrogate key can be the primary key, generated by the database management system and *not* derived from any application data in the database. The only significance of the surrogate key is to act as the primary key. It is also possible that the surrogate key exists in addition to the database-generated UUID (for example, an HR number for each employee other than the UUID of each employee).

A surrogate key is frequently a sequential number (e.g. a Sybase or SQL Server "identity column", a PostgreSQL or Informix `serial`, an Oracle `SEQUENCE` or a column defined with `AUTO_INCREMENT` in MySQL) but doesn't have to be. Having the key independent of all other columns insulates the database relationships from changes in data values or database design (making the database more agile) and guarantees uniqueness.

In a temporal database, it is necessary to distinguish between the surrogate key and the business key. Every row would have both a business key and a surrogate key. The surrogate key identifies one unique row in the database, the business key identifies one unique entity of the modeled world. One table row represents a slice of time holding all the entities attributes for a defined timespan. Those slices depict the whole lifespan of one business entity. For example, a table *EmployeeContracts* may hold temporal information to keep track of contracted working hours. The business key for one contract will be identical (non-unique) in both rows however the surrogate key for each row is unique.

SurrogateKey	BusinessKey	EmployeeName	WorkingHoursPerWeek	RowValidFrom	RowValidTo
1	BOS0120	John Smith	40	2000-01-01	2000-12-31
56	P0000123	Bob Brown	25	1999-01-01	2011-12-31
234	BOS0120	John Smith	35	2001-01-01	2009-12-31

Some database designers use surrogate keys systematically regardless of the suitability of other candidate keys, while others will use a key already present in the data, if there is one.

A *surrogate key* may also be called a synthetic key, an entity identifier, a system-generated key, a database sequence number, a factless key, a technical key, or an arbitrary unique identifier.[*citation needed*] Some of these terms describe the way of *generating* new surrogate values rather than the *nature* of the surrogate concept.

Approaches to generating surrogates include:

- Universally Unique Identifiers (UUIDs)
- Globally Unique Identifiers (GUIDs)
- Object Identifiers (OIDs)
- Sybase or SQL Server identity column `IDENTITY` OR `IDENTITY(n,n)`
- Oracle `SEQUENCE`
- PostgreSQL or IBM Informix serial
- MySQL `AUTO_INCREMENT`
- AutoNumber data type in Microsoft Access
- `AS IDENTITY GENERATED BY DEFAULT` in IBM DB2
- Identity column (implemented in DDL) in Teradata

Advantages

Immutability

Surrogate keys do not change while the row exists. This has the following advantages:

- Applications cannot lose their reference to a row in the database (since the identifier never changes).
- The primary or natural key data can always be modified, even with databases that do not support cascading updates across related foreign keys.

Requirement changes

Attributes that uniquely identify an entity might change, which might invalidate the suitability of natural keys. Consider the following example:

> An employee's network user name is chosen as a natural key. Upon merging with another company, new employees must be inserted. Some of the new network user names create conflicts because their user names were generated independently (when the companies were separate).

In these cases, generally a new attribute must be added to the natural key (for example, an *original_company* column). With a surrogate key, only the table that defines the surrogate key must be changed. With natural keys, all tables (and possibly other, related software) that use the natural key will have to change.

Some problem domains do not clearly identify a suitable natural key. Surrogate key avoids choosing a natural key that might be incorrect.

Performance

Surrogate keys tend to be a compact data type, such as a four-byte integer. This allows the database to query the single key column faster than it could multiple columns. Furthermore a non-redundant distribution of keys causes the resulting b-tree index to be completely balanced. Surrogate keys are also less expensive to join (fewer columns to compare) than compound keys.

Compatibility

While using several database application development systems, drivers, and object-relational mapping systems, such as Ruby on Rails or Hibernate, it is much easier to use an integer or GUID surrogate keys for every table instead of natural keys in order to support database-system-agnostic operations and object-to-row mapping.

Uniformity

When every table has a uniform surrogate key, some tasks can be easily automated by writing the code in a table-independent way.

Validation

It is possible to design key-values that follow a well-known pattern or structure which can be automatically verified. For instance, the keys that are intended to be used in some column of some table might be designed to "look differently from" those that are intended to be used in another column or table, thereby simplifying the detection of application errors in which the keys have been misplaced. However, this characteristic of the surrogate keys should never be used to drive any of the logic of the applications themselves, as this would violate the principles of Database normalization.

Disadvantages

Disassociation

The values of generated surrogate keys have no relationship to the real-world *meaning* of the data held in a row. When inspecting a row holding a foreign key reference to another table using a surrogate key, the meaning of the surrogate key's row cannot be discerned from the key itself. Every foreign key must be joined to see the related data item. This can also make auditing more difficult,[*citation needed*] as incorrect data is not obvious.

Surrogate keys are unnatural for data that is exported and shared. A particular difficulty is that tables from two otherwise identical schemas (for example, a test schema and a development schema) can hold records that are equivalent in a business sense, but have different keys. This can be mitigated by not exporting surrogate keys, except as transient data (most obviously, in executing applications that have a "live" connection to the database).

Query optimization

Relational databases assume a unique index is applied to a table's primary key. The unique index serves two purposes: (i) to enforce entity integrity, since primary key data must be unique across rows and (ii) to quickly search for rows when queried. Since surrogate keys replace a table's identifying attributes—the natural key—and since the identifying attributes are likely to be those queried, then the query optimizer is forced to perform a full table scan when fulfilling likely queries. The remedy to the full table scan is to apply indexes on the identifying attributes, or sets of them. Where such sets are themselves a candidate key, the index can be a unique index.

These additional indexes, however, will take up disk space and slow down inserts and deletes.

Normalization

The presence of a surrogate key can result in the database administrator forgetting to establish, or accidentally removing, a secondary unique index on the natural key of the table. Without a unique index on the natural key, duplicate rows can appear and once present can be difficult to identify.

Business process modeling

Because surrogate keys are unnatural, flaws can appear when modeling the business requirements. Business requirements, relying on the natural key, then need to be translated to the surrogate key. A strategy is to draw a clear distinction between the logical model (in which surrogate keys do not appear) and the physical implementation of that model, to ensure that the logical model is correct and reasonably well normalised, and to ensure that the physical model is a correct implementation of the logical model.

Inadvertent disclosure

Proprietary information can be leaked if sequential key generators are used. By subtracting a previously generated sequential key from a recently generated sequential key, one could learn the number of rows inserted during that time period. This could expose, for example, the number of transactions or new accounts per period. There are a few ways to overcome this problem:

- Increase the sequential number by a random amount.
- Generate a completely random primary key. However, to prevent duplication which would cause an insert rejection, a randomly generated primary key must either be queried (to check that it is not already in use), or the key must contain enough entropy that one can be confident that collisions will not happen.

Inadvertent assumptions

One might incorrectly infer from sequentially generated surrogate keys that events with a higher primary key value occurred after events with a lower primary key value. The sequential primary key implies nothing of the kind. It is possible for inserts to fail and leave gaps, and for those gaps to be filled at some later time. A sequential key value is not a reliable indicator of chronology. If chronology is important, rely not upon the sequential key but upon a timestamp. A random key would prevent a person from making the assumption that the key has some bearing to real-world chronology only if the person making the assumption is aware that the key is indeed random and has no bearing upon chronology. A randomly generated primary key must be queried before assigned to prevent duplication and cause an insert rejection.[*citation needed*]

References

This article is based on material taken from the Free On-line Dictionary of Computing prior to 1 November 2008 and incorporated under the "relicensing" terms of the GFDL, version 1.3 or later.

- Nijssen, G.M. (1976). *Modelling in Data Base Management Systems*. North-Holland Pub. Co. ISBN 0-7204-0459-2.
- Engles, R.W.: (1972), *A Tutorial on Data-Base Organization*, Annual Review in Automatic Programming, Vol.7, Part 1, Pergamon Press, Oxford, pp. 1–64.
- Langefors, B (1968). *Elementary Files and Elementary File Records*, Proceedings of File 68, an IFIP/IAG International Seminar on File Organisation, Amsterdam, November, pp. 89–96.
- Wieringa, R.; de Jonge, W. (1991). *The identification of objects and roles: Object identifiers revisited*. CiteSeerX: 10.1.1.16.3195 [1].
- Date, C. J. (1998). "Chapters 11 and 12". *Relational Database Writings 1994–1997*. ASIN 0201398141 [2].
- Carter, Breck. "Intelligent Versus Surrogate Keys" [1]. Retrieved 2006-12-03.
- Richardson, Lee. "Create Data Disaster: Avoid Unique Indexes – (Mistake 3 of 10)" [2]. Retrieved 2008-01-19.
- Berkus, Josh. "Database Soup: Primary Keyvil, Part I" [3]. Retrieved 2006-12-03.

References

[1] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.3195

[2] http://www.amazon.co.uk/dp/0201398141

[3] http://it.toolbox.com/blogs/database-soup/primary-keyvil-part-i-7327

Variable data publishing

Variable-data publishing (VDP) (also known as database publishing) is a term referring to the output of a variable composition system. While these systems can produce both electronically viewable and hard-copy (print) output, the "variable-data publishing" term today often distinguishes output destined for electronic viewing, rather than that which is destined for hard-copy print (e.g. variable data printing).

Essentially the same techniques are employed to perform variable-data publishing, as those utilized with variable data printing. The difference is in the interpretation for output. While variable-data printing may be interpreted to produce various print streams or page-description files (e.g. AFP/IPDS, PostScript, PCL), variable-data publishing produces electronically viewable files, most commonly seen in the forms of PDF, HTML, or XML.

Variable-data composition involves the use of data to conditionally:

- exhibit text (static blocks and/or variable content)
- exhibit images
- select fonts
- select colors
- format page layouts & flows

Variable-data may be as simple as an address block or salutation. However, it can be any or all of the document's textual content—including words, sentences, paragraphs, pages, or the entire document. In other words, it can make up as little or as much of the document as the composer desires. Variable data may also be used to exhibit various images, such as logos, products, or membership photos. Further, variable-data can be used to build rule-based design schemes, including fonts, colors, and page formats. The possibilities are vast.

The variable-data tools available today, make it possible to perform variable-data composition at nearly every stage of document production. However, the level of control that can be achieved varies, based upon how far into the document production process a variable-data tool is deployed. For example, if variable-data insertion occurs just prior to output...it's not likely that the text flow or layout can be altered with nearly as much control as would be available at the time of initial document composition.

Many organizations will produce multiple forms of output (aka: multi-channel output), for the same document. This ensures that the published content is available to recipients via any form of access method they might require. When multi-channel output is utilized, integrity between those output channels often becomes important.

Variable-data publishing may be performed on everything from a personal computer to a mainframe system. However, the speed and practical output volumes which can be achieved are directly affected by the computer power utilized.

Origin of the concept

The term variable-data publishing was likely an offshoot of the term "variable-data printing", first introduced to the printing industry by Frank Romano, Professor Emeritus, School of Print Media, at the College of Imaging Arts and Sciences at Rochester Institute of Technology.[*citation needed*] However, the concept of merging static document elements and variable document elements predates the term and has seen various implementations ranging from simple desktop 'mail merge', to complex mainframe applications in the financial and banking industry. In the past, the term VDP has been most closely associated with digital printing machines. However, in the past 3 years the application of this technology has spread to web pages, emails, and mobile messaging.

Semantic warehousing

In data management, **semantic warehousing** is a methodology of digitalized text data using similar functions to Data warehousing (DW), such as ETL(Extract, transform, load), ODS(Operational data store), and MODEL. Key value operation is less useful for the digitalized text. Semantic warehousing is different from DW in that semantic information base from text(semantic) data.

Semantic warehousing is different from search engine in that semantic information base from text data is stored in the database.(DBMS)

Though data is most important word in computing era, it can not explain human knowledge well yet. Data(numeric data) is key element of computing systems for certain organization (especially companies, enterprises), but no performance oriented organization needs something to gather and use knowledge or human feeling. Semantic warehousing will be equally or more important than data warehousing in the future.

Definition

Semantic warehousing is a conceptual and functional term meaning to gather from a source, semantically defining and providing information from digitalized text based knowledge data.

Background

Data warehousing (DW) is popular these days. Gathering data from systems that generate transactions, data warehouses become a base of information. Key of data warehouse is a model (called datamart) and that model is made up of dimensions(key) and measures(value). Users get information from the models by doing certain operations. Online analytical processing (OLAP) is most the important operation for the users to get information from the DW models. Handling dimensions with pivoting, drilling, slice & dice operations users get numeric values like sales amounts, growth rates, etc. Various areas of this world defined and appeared on the world wide web(Internet), eager to present their contents in a semantic way. Briefly speaking semantic warehousing has datawarehousing boby and search head and ontology features.

Data warehousing contributed to companies' business values and lots of solutions and tools are commercially successful. Analysis of internal data delivers a certain level of business values, on the contrary to this Semantic warehousing environment has not yet matured. Capacity of social data is increasing rapidly and various efforts of finding value from that data are made widely known as Big data, etc. Semantic warehousing can be main stream of treat data and intelligence of social world in the future though it is defined with other keywords.

Practices

Some hospital implement semantic warehousing for clinical information (SWCI). Medical information is now knowledge network level. UMLS define semantic knowledge network of medical language. Currently medical information stored in database and not fully used for clinic. Semantic warehousing is next stage of digitalized medical information.

SWCI is a name of conceptual system of clinical information.
Named by Juhan Kim (SNUH, Seoul National University Hospital) and Bohyon Hwang, YongChan Keum on 2008.

Defined architecture on SWCI ;

1. Semantic-oriented cleansing
2. Semantic-oriented meta management
3. Clinical(Medical) knowledge basement
4. Semantic-oriented user intelligence

Connected area

- Semantic web
- Ontology
- Knowledge
- Medical and healthcare : EMR (Electronic Medical Record), EHR (Electronic Health Record)
- Data warehouse
- AI (artificial intelligence)

References

- **BI** Laboratory of Seoul National University Hospital [1]
- Smith, Barry Kumar, Anand and Schulze-Kremer, Steffen (2004) Revising the UMLS Semantic Network [2], in M. Fieschi, et al. (eds.), Medinfo 2004, Amsterdam: IOS Press, 1700.
- Foundations of Data Warehouse Quality :

```
Data Quality article mentioning that semantically rich DW.
http://www.cs.brown.edu/courses/cs227/Papers/Projects/iq97_dwq.pdf
```

- An Integrative and Uniform Model for Metadata Management in Data Warehousing Environment.

```
Semantic metadata and technical metadata.
http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/paper12.pdf
```

- Effective Query Expansion using Condensed UMLS Metathesaurus for Medical Information Retrieval

http:/ / www. e-hir. org/ journal/ view. html?uid=201& start=& sort=& scale=& key=all& oper=& key_word=UMLS& year1=& year2=& Vol=& Num=& PG=& book=& mod=vol& sflag=& sub_box=Y& aut_box=Y&sos_box=&pub_box=Y&key_box=&abs_box=&year=

- A Study of Effective Unified Medical Language System Concept Indexing in Radiology Reports

http:/ / www. e-hir. org/ journal/ view. html?uid=226& start=& sort=& scale=& key=all& oper=& key_word=UMLS& year1=& year2=& Vol=& Num=& PG=& book=& mod=vol& sflag=& sub_box=Y& aut_box=Y&sos_box=&pub_box=Y&key_box=&abs_box=&year=

- Developing a Reference Terminology Model for Health Care Using an Object-Oriented Approach

http:/ / www. e-hir. org/ journal/ view. html?uid=311& start=& sort=& scale=& key=all& oper=& key_word=UMLS& year1=& year2=& Vol=& Num=& PG=& book=& mod=vol& sflag=& sub_box=Y& aut_box=Y&sos_box=&pub_box=Y&key_box=&abs_box=&year=

- UMLS(Unified Medical Language System)의 증상용어와 국내의무기록에서 사용되는 증상용어와의 비교연구

http:/ / www. e-hir. org/ journal/ view. html?uid=922& start=& sort=& scale=& key=all& oper=& key_word=UMLS& year1=& year2=& Vol=& Num=& PG=& book=& mod=vol& sflag=& sub_box=Y& aut_box=Y&sos_box=&pub_box=Y&key_box=&abs_box=&year=

References

[1] http://www.snubi.org/
[2] http://ontology.buffalo.edu/medo/UMLS_SN.pdf

Scriptella

Scriptella

scriptella	
Stable release	1.1 / 28 December 2012
Operating system	Cross-platform
Type	ETL, Data migration and SQL.
License	Apache Software License
Website	scriptella.javaforge.com [1]

Scriptella is an open source ETL (Extract-Transform-Load) and script execution tool written in Java. Its primary focus is simplicity. It doesn't require the user to learn another complex XML-based language to use it, but allows the use of SQL or another scripting language suitable for the data source to perform required transformations.

Potential users should be aware that Scriptella does not offer any graphical user interface.

Typical use

- Database migration.
- Database creation/update scripts.
- Cross-database ETL operations, import/export.
- Alternative for Ant <sql> task.
- Automated database schema upgrade.

Features

- **Simple XML syntax** for scripts. Add dynamics to your existing SQL scripts by creating a thin wrapper XML file:

```
<!DOCTYPE etl SYSTEM "http://scriptella.javaforge.com/dtd/etl.dtd">
<etl>
    <connection driver="$driver" url="$url" user="$user" password="$password"/>
    <script>
        <include href="PATH_TO_YOUR_SCRIPT.sql"/>
        -- And/or directly insert SQL statements here
    </script>
</etl>
```

- Support for **multiple datasources** (or multiple connections to a single database) in an ETL file.
- Support for many useful **JDBC features**, e.g. parameters in SQL including file blobs and JDBC escaping.
- **Performance.** Performance and low memory usage are one of the primary goals.
- Support for **evaluated expressions and properties** (JEXL syntax)
- Support for **cross-database ETL scripts** by using <dialect> elements

- **Transactional execution**
- **Error handling** via <onerror> elements
- **Conditional scripts/queries execution** (similar to Ant if/unless attributes but more powerful)
- **Easy-to-Use** as a standalone tool or Ant task. No deployment/installation required.
- **Easy-To-Run** ETL files directly from Java code.
- **Built-in adapters for popular databases** for a tight integration. Support for any database with JDBC/ODBC compliant driver.
- Service Provider Interface (SPI) for interoperability with non-JDBC DataSources and integration with scripting languages. Out of the box support for JSR 223 (Scripting for the Java Platform) compatible languages.
- Built-In CSV, TEXT, XML, LDAP, Lucene, Velocity, JEXL and Janino providers. Integration with Java EE, Spring Framework, JMX and JNDI for enterprise ready scripts.

External links

- Scriptella ETL Site [1]
- Discussion forum [2]
- Discussion forum(deprecated) [3]
- Scriptella ETL Author's Blog [4]
- Example code snippets for Scriptella ETL [5]
- Scriptella ETL [6] at Ohloh

References

[1] http://scriptella.javaforge.com
[2] http://groups.google.com/group/scriptella/
[3] http://www.javaforge.com/proj/forum/browseForum.do?forum_id=3126
[4] http://jroller.com/page/ejboy
[5] http://snippets.dzone.com/tag/scriptella
[6] http://www.ohloh.net/projects/4526

Data Quality

Bit rot

Bit rot, also **bit decay**, **data rot**, or **data decay**, is a colloquial computing phrase for the gradual decay of storage media.[*citation needed*]

The *Jargon File,* a compendium of hacker's lore, defines "bit rot" as a jocular explanation for the software rot, the degradation of a software program over time even if "nothing has changed"; an explanations is that bits are subject to decay as if they were radioactive.

Decay of storage media

Bit rot is often defined as the event in which the small electric charge of a bit in memory disperses, possibly altering program code or stored data. The hypothesis that semiconductor RAM may occasionally be altered by cosmic rays is also known as soft error.

Bit rot can also be used to describe the phenomenon of storage media gradually decaying over the duration of many years. The cause of bit rot varies depending on the medium:

- *Solid state media* – such as EPROMs, flash memory and other solid-state drives – stores data using electrical charges, which can slowly leak away due to imperfect insulation. The chip itself is not affected by this, so re-programming it once per decade or so will prevent bit rot. The biggest problem can be finding a clean copy of the chip from which to make the copy; frequently, by the time the user discovers the bit rot, there are no un-damaged chips to use as a master.
- *Magnetic media* – such as floppy disks and magnetic tapes – may experience bit rot as bits lose their magnetic orientation. Periodic refreshing by rewriting the data can alleviate this problem. Also, in warm and humid conditions these media are prone to literal rotting.
- *Optical media* – such as CD-R, DVD-R and BD-R – may experience bit rot from the breakdown of the material onto which the data is stored. This can be mitigated by storing discs in a dark, cool location with low humidity. "Archival quality" discs are also available, but do not necessarily provide a permanent solution to the onset of bit rot or other types of data corruption beyond a certain amount of time. Some media (such as M-DISC) are designed to improve longevity over DVD-R and BD-R.
- *Paper media* – such as punched cards and punched tape – may also experience literal rotting. Mylar punched tape is available for use in this situation.

Component and system failures

Most disk, disk controller and higher level systems are subject to a small degree of unrecoverable failure. With ever-growing disk capacities, file sizes, and increases in the amount of data stored on a disk, the likelihood of the occurrence of bit rot and other forms of uncorrected and undetected data corruption increases.

Higher level software systems may be employed to mitigate the risk of such underlying failures by increasing redundancy and implementing integrity checking and self-repairing algorithms. The ZFS file system was designed to address many of these issues. The Btrfs file system also includes data protection and recovery mechanisms, and so does ReFS.

References

Cleansing and Conforming Data

This process of **Cleansing and Conforming Data** change data on its way from source system(s) to the data warehouse and can also be used to identify and record errors about data. The latter information can be used to fix how the source system(s) work(s).

Good quality source data has to do with “Data Quality Culture” and must be initiated at the top of the organization. It is not just a matter of implementing strong validation checks on input screens, because almost no matter how strong these checks are, they can often still be circumvented by the users.

There is a nine-step guide for organizations that wish to improve data quality:

- Declare a high level commitment to a data quality culture
- Drive process reengineering at the executive level
- Spend money to improve the data entry environment
- Spend money to improve application integration
- Spend money to change how processes work
- Promote end-to-end team awareness
- Promote interdepartmental cooperation
- Publicly celebrate data quality excellence
- Continuously measure and improve data quality

Data Cleansing System

The essential job of this system is to find a suitable balance between fixing dirty data and maintaining the data as close as possible to the original data from the source production system. This is a challenge for the Extract, transform, load architect.

The system should offer an architecture that can cleanse data, record quality events and measure/control quality of data in the data warehouse.

A good start is to perform a thorough data profiling analysis that will help define to the required complexity of the data cleansing system and also give an idea of the current data quality in the source system(s).

Quality Screens

Part of the data cleansing system is a set of diagnostic filters known as quality screens. They each implement a test in the data flow that, if it fails records an error in the Error Event Schema. Quality screens are divided into three categories:

- Column screens. Testing the individual column, e.g. for unexpected values like NULL values; non-numeric values that should be numeric; out of range values; etc.
- Structure screens. These are used to test for the integrity of different relationships between columns (typically foreign/primary keys) in the same or different tables. They are also used for testing that a group of columns is valid according to some structural definition it should adhere.
- Business rule screens. The most complex of the three tests. They test to see if data, maybe across multiple tables, follow specific business rules. An example could be, that if a customer is marked as a certain type of customer, the business rules that define this kind of customer should be adhered.

When a quality screen records an error, it can either stop the dataflow process, send the faulty data somewhere else than the target system or tag the data. The latter option is considered the best solution because the first option requires, that someone has to manually deal with the issue each time it occurs and the second implies that data are missing from the target system (integrity) and it is often unclear, what should happen to these data.

Criticism of existing tools and processes

The main reasons cited are:

- **Project costs**: costs typically in the hundreds of thousands of dollars
- **Time**: lack of enough time to deal with large-scale data-cleansing software
- **Security**: concerns over sharing information, giving an application access across systems, and effects on legacy systems

Error Event Schema

This schema is the place, where all error events thrown by quality screens, are recorded. It consists of an Error Event Fact table with foreign keys to three dimension tables that represent date (when), batch job (where) and screen (who produced error). It also holds information about exactly when the error occurred and the severity of the error. In addition there is an Error Event Detail Fact table with a foreign key to the main table that contains detailed information about in which table, record and field the error occurred and the error condition.

References

Sources

- Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., Becker, B. *The Data Warehouse Lifecycle Toolkit*, Wiley Publishing, Inc., 2008. ISBN 978-0-470-14977-5.
- Olson, J. E. *Data Quality: The Accuracy Dimension", Morgan Kauffman, 2002. ISBN 1-55860-891-5.*

Data auditing

Data auditing is the process of conducting a data audit to assess how company's data is fit for given purpose. This involves profiling the data and assessing the impact of poor quality data on the organization's performance and profits.

Data cleansing

Data cleansing, **data cleaning** or **data scrubbing** is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database. Used mainly in databases, the term refers to identifying incomplete, incorrect, inaccurate, irrelevant, etc. parts of the data and then replacing, modifying, or deleting this dirty data.

After cleansing, a data set will be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores.

Data cleansing differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at entry time, rather than on batches of data.

The actual process of data cleansing may involve removing typographical errors or validating and correcting values against a known list of entities. The validation may be strict (such as rejecting any address that does not have a valid postal code) or fuzzy (such as correcting records that partially match existing, known records).

Some data cleansing solutions will clean data by cross checking with a validated data set. Also data enhancement, where data is made more complete by adding related information, is a common data cleansing practice. For example, appending addresses with phone numbers related to that address.

Data cleansing may also involve activities like, harmonization of data, and standardization of data. For example, harmonization of short codes (St, rd etc.) to actual words (street, road). Standardization of data is a means of changing a reference data set to a new standard, ex, use of standard codes.

Motivation

Administratively, incorrect or inconsistent data can lead to false conclusions and misdirected investments on both public and private scales. For instance, the government may want to analyze population census figures to decide which regions require further spending and investment on infrastructure and services. In this case, it will be important to have access to reliable data to avoid erroneous fiscal decisions.

In the business world, incorrect data can be costly. Many companies use customer information databases that record data like contact information, addresses, and preferences. For instance, if the addresses are inconsistent, the company will suffer the cost of resending mail or even losing customers.

There are packages available so you can cleanse/wash address data while you enter it into your system. This is normally done via an API and will prompt staff as they type the address.

Data quality

High-quality data needs to pass a set of quality criteria. Those include:

- **Validity**: The degree to which the measures conform to defined business rules or constraints (see also Validity (statistics). When modern database technology is used to design data-capture systems, validity is fairly easy to ensure: invalid data arises mainly in legacy contexts (where constraints were not implemented in software) or where inappropriate data-capture technology was used (e.g., spreadsheets, where it is very hard to limit what a user chooses to enter into a cell).Data constraints fall into the following categories:
  - *Data-Type Constraints* – e.g., values in a particular column must be of a particular datatype, e.g., Boolean, numeric (integer or real), date, etc.
  - *Range Constraints:* typically, numbers or dates should fall within a certain range. That is, they have minimum and/or maximum permissible values.
  - *Mandatory Constraints:* Certain columns cannot be empty.
  - *Unique Constraints:* A field, or a combination of fields, must be unique across a dataset. For example, no two persons can have the same social security number.
  - *Set-Membership constraints*: The values for a column come from a set of discrete values or codes. For example, a person's gender may be Female, Male or Unknown (not recorded).
  - *Foreign-key constraints*: This is the more general case of set membership. The set of values in a column is defined in a column of another table that contains unique values. For example, in a US taxpayer database, the "state" column is required to belong to one of the US's defined states or territories: the set of permissible states/territories is recorded in a separate States table. The term foreign key is borrowed from relational database terminology: follow the hyperlink for more details.
  - Regular expression patterns: Occasionally, text fields will have to be validated this way. For example, phone numbers may be required to have the pattern (999) 999-9999.
- **Decleansing** is detecting errors and syntactically removing them for better programming.
  - Cross-field validation: Certain conditions that utilize multiple fields must hold. For example, in laboratory medicine, the sum of the components of the differential white blood cell count must be equal to 100 (since they are all percentages). In a hospital database, a patient's date of discharge from hospital cannot be earlier than the date of admission.
- **Accuracy**: The degree of conformity of a measure to a standard or a true value - see also Accuracy and precision. Accuracy is very hard to achieve through data-cleansing in the general case, because it requires accessing an external source of data that contains the true value: such "gold standard" data is often unavailable. Accuracy has been achieved in some cleansing contexts, notably customer contact data, by using external databases that match up zip codes to geographical locations (city and state), and also help verify that street addresses within these zip codes actually exist.
- **Completeness**: The degree to which all required measures are known (see also Completeness). Incompleteness is almost impossible to fix with data cleansing methodology: one cannot infer facts that were not captured when the data in question was initially recorded. (In some contexts, e.g., interview data, it may be possible to fix incompleteness by going back to the original source of data, i,e., re-interviewing the subject, but even this does not guarantee success because of problems of recall - e.g., in an interview to gather data on food consumption, no one is likely to remember exactly what one ate six months ago. In the case of systems that insist certain columns should not be empty, one may work around the problem by designating a value that indicates "unknown" or "missing", but supplying of default values does not imply that the data has been made complete.
- **Consistency**: The degree to which a set of measures are equivalent in across systems (see also Consistency). Inconsistency occurs when two data items in the data set contradict each other: e.g., a customer is recorded in two different systems as having two different current addresses, and only one of them can be correct. Fixing inconsistency is not always possible: it requires a variety of strategies - e.g., deciding which data were recorded

more recently, which data source is likely to be most reliable (the latter knowledge may be specific to a given organization), or simply trying to find the truth by testing both data items (e.g., calling up the customer).

- **Uniformity**: The degree to which a set data measures are specified using the same units of measure in all systems (see also Unit of measure). In datasets pooled from different locales, weight may be recorded either in pounds or kilos, and must be converted to a single measure using an arithmetic transformation.

The term **Integrity** encompasses accuracy, consistency and some aspects of validation (see also Data integrity) but is rarely used by itself in data-cleansing contexts because it is insufficiently specific. (For example, "referential integrity" is a term used to refer to the enforcement of foreign-key constraints above.)

The process of data cleansing

- **Data auditing**: The data is audited with the use of statistical and database methods to detect anomalies and contradictions: this eventually gives an indication of the characteristics of the anomalies and their locations. Several commercial software packages will let you specify constraints of various kinds (using a grammar that conforms to that of a standard programming language, e.g., JavaScript or Visual Basic) and then generate code that checks the data for violation of these constraints. This process is referred to below in the bullets "workflow specification" and "workflow execution." For users who lack access to high-end cleansing software, Microcomputer database packages such as Microsoft Access or FileMaker Pro will also let you perform such checks, on a constraint-by-constraint basis, interactively with little or no programming required in many cases.
- **Workflow specification**: The detection and removal of anomalies is performed by a sequence of operations on the data known as the workflow. It is specified after the process of auditing the data and is crucial in achieving the end product of high-quality data. In order to achieve a proper workflow, the causes of the anomalies and errors in the data have to be closely considered.
- **Workflow execution**: In this stage, the workflow is executed after its specification is complete and its correctness is verified. The implementation of the workflow should be efficient, even on large sets of data, which inevitably poses a trade-off because the execution of a data-cleansing operation can be computationally expensive.
- **Post-processing and controlling**: After executing the cleansing workflow, the results are inspected to verify correctness. Data that could not be corrected during execution of the workflow is manually corrected, if possible. The result is a new cycle in the data-cleansing process where the data is audited again to allow the specification of an additional workflow to further cleanse the data by automatic processing.

Decleanse

- **Parsing**: for the detection of syntax errors. A parser decides whether a string of data is acceptable within the allowed data specification. This is similar to the way a parser works with grammars and languages.
- **Data transformation**: Data transformation allows the mapping of the data from its given format into the format expected by the appropriate application. This includes value conversions or translation functions, as well as normalizing numeric values to conform to minimum and maximum values.
- **Duplicate elimination**: Duplicate detection requires an algorithm for determining whether data contains duplicate representations of the same entity. Usually, data is sorted by a key that would bring duplicate entries closer together for faster identification.
- **Statistical methods**: By analyzing the data using the values of mean, standard deviation, range, or clustering algorithms, it is possible for an expert to find values that are unexpected and thus erroneous. Although the correction of such data is difficult since the true value is not known, it can be resolved by setting the values to an average or other statistical value. Statistical methods can also be used to handle missing values which can be replaced by one or more plausible values, which are usually obtained by extensive data augmentation algorithms.

Challenges and problems

- **Error correction and loss of information**: The most challenging problem within data cleansing remains the correction of values to remove duplicates and invalid entries. In many cases, the available information on such anomalies is limited and insufficient to determine the necessary transformations or corrections, leaving the deletion of such entries as a primary solution. The deletion of data, though, leads to loss of information; this loss can be particularly costly if there is a large amount of deleted data.
- **Maintenance of cleansed data**: Data cleansing is an expensive and time-consuming process. So after having performed data cleansing and achieving a data collection free of errors, one would want to avoid the re-cleansing of data in its entirety after some values in data collection change. The process should only be repeated on values that have changed; this means that a cleansing lineage would need to be kept, which would require efficient data collection and management techniques.
- **Data cleansing in virtually integrated environments**: In virtually integrated sources like IBM's DiscoveryLink, the cleansing of data has to be performed every time the data is accessed, which considerably decreases the response time and efficiency.
- **Data-cleansing framework**: In many cases, it will not be possible to derive a complete data-cleansing graph to guide the process in advance. This makes data cleansing an iterative process involving significant exploration and interaction, which may require a framework in the form of a collection of methods for error detection and elimination in addition to data auditing. This can be integrated with other data-processing stages like integration and maintenance.

Major players and technologies

- IBM - InfoSphere Information Server is a tool that provides data cleansing and data monitoring services.
- SAS - Integration with DataFlux suite of data integration, cleansing, data governance, and data quality services.
- Oracle – Data quality solutions work with both customer and product data.
- Experian – QAS Clean service provides CASS certification (Coding Accuracy Support System) for address verification services.
- D&B – Offers data management transition and data quality programs
- Equifax – Offers database management, data integration, and analytics solutions

References

Sources

- Han, J., Kamber, M. *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2001. ISBN 1-55860-489-8.
- Kimball, R., Caserta, J. *The Data Warehouse ETL Toolkit*, Wiley and Sons, 2004. ISBN 0-7645-6757-8.
- Muller H., Freytag J., *Problems, Methods, and Challenges in Comprehensive Data Cleansing*, Humboldt-Universitat zu Berlin, Germany.
- Rahm, E., Hong, H. *Data Cleaning: Problems and Current Approaches*, University of Leipzig, Germany.

External links

- *Computerworld: Data Scrubbing* (http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=78230) (February 10, 2003)
- *Information Management: Data Cleansing Legacy Systems* (http://www.information-management.com/infodirect/20041029/1012952-1.html)

Data corruption

Data corruption refers to errors in computer data that occur during writing, reading, storage, transmission, or processing, which introduce unintended changes to the original data. Computer, transmission and storage systems use a number of measures to provide end-to-end data integrity, or lack of errors.

In general, when data corruption occurs, a file containing that data will produce unexpected results when accessed by the system or the related application; results could range from a minor loss of data to a system crash. For example, if a Microsoft Word file is corrupted, when a person tries to open that file with MS Word, they may get an error message, thus the file would not be opened or the file might open with some of the data corrupted. The image to the right is a corrupted jpg file in which most of the information has been lost.

Photo data corruption; in this case, a result of a failed data recovery from a hard disk drive

Some programs can give a suggestion to repair the file automatically (after the error), and some programs cannot repair it. It depends on the level of corruption, and the built-in functionality of the application to handle the error. There are various causes of the corruption.

Overview

There are two types of data corruption associated with computer systems:

Undetected

> Also known as *silent data corruption;* such problems are the most dangerous errors as there is no indication that the data is incorrect.

Detected

> Detected errors may be permanent with the loss of data or maybe temporary where some part of the system is able to detect and correct the error, in this latter case there is no data corruption.

Data corruption can occur at any level in a system, from the host to the storage medium. Modern systems attempt to detect corruption at many layers and then recover or correct the corruption; this is almost always successful but very rarely the information arriving in the systems memory is corrupted and can cause unpredictable results.

Data corruption during transmission has a variety of causes. Interruption of data transmission causes information loss. Environmental conditions can interfere with data transmission, especially when dealing with wireless transmission methods. Heavy clouds can block satellite transmissions. Wireless networks are susceptible to interference from devices such as microwave ovens.

Hardware and software failure are the two main causes for data loss. Background radiation, head crashes, and aging or wear of the storage device fall into the former category, while software failure typically occurs due to bugs in the

code. Cosmic rays cause most soft errors in DRAM.

Silent data corruption

The worst type of errors are those that go unnoticed, and are not even detected by the disk firmware or the host operating system. This is known as *silent corruption*.

There are many error sources beyond the disk storage subsystem itself. For instance, cables might be slightly loose, the power supply might be unreliable, external vibrations such as a loud sound, the network might introduce undetected corruption, cosmic radiation and many other causes of soft memory errors, etc. In 39,000 storage systems that were analyzed, firmware bugs accounted for 5–10% of storage failures. All in all, the error rates as observed by a CERN study on silent corruption are far higher than one in every 10^{16} bits. Webshop Amazon.com confirms these high data corruption rates.

The main problem is that hard disk capacities have increased substantially, but their error rates remain unchanged. The data corruption rate has always been roughly constant in time, meaning that modern disks are not much safer than old disks. In old disks the probability of data corruption was very small because they stored tiny amounts of data. In modern disks the probability is much larger because they store much more data, whilst not being safer. That way, silent data corruption has not been a serious concern while storage devices remained relatively small and slow. Hence, the users of small disks very rarely faced silent corruption, so the data corruption was not considered a problem that required a solution. But in modern times and with the advent of larger drives and very fast RAID setups, users are capable of transferring 10^{16} bits in a reasonably short time, thus easily reaching the data corruption thresholds.

As an example, ZFS creator Jeff Bonwick stated that the fast database at Greenplum – a database software company specializing in large-scale data warehousing and analytics – faces silent corruption every 15 minutes. As another example, a real-life study performed by NetApp on more than 1.5 million HDDs over 41 months found more than 400,000 silent data corruptions, out of which more than 30,000 were not detected by the hardware RAID controller. Another study, performed by CERN over six months and involving about 97 petabytes of data, found about 1.2×10^{-9} of involved data (or about 116 gigabytes) becoming permanently corrupted.

Countermeasures

When data corruption behaves as a Poisson process, where each bit of data has an independently low probability of being changed, data corruption can generally be detected by the use of checksums, and can often be corrected by the use of error correcting codes.

If an uncorrectable data corruption is detected, procedures such as automatic retransmission or restoration from backups can be applied. Certain levels of RAID disk arrays have the ability to store and evaluate parity bits for data across a set of hard disks and can reconstruct corrupted data upon the failure of a single or multiple disks, depending on the level of RAID implemented.

Many errors are detected and corrected by the hard disk drives using the ECC/CRC codes which are stored on disk for each sector. If the disk drive detects multiple read errors on a sector it may make a copy of the failing sector on another part of the disk- remapping the failed sector of the disk to a spare sector without the involvement of the operating system (though this may be delayed until the next write to the sector).

This "silent correction" can lead to other problems if disk storage is not managed well, as the disk drive will continue to remap sectors until it runs out of spares, at which time the temporary correctable errors can turn into permanent ones as the disk drive deteriorates. S.M.A.R.T. provides a standardized way of monitoring the health of a disk drive, and there are tools available for most operating systems to automatically check the disk drive for impending failures by watching for deteriorating SMART parameters.

Some filesystems (including Btrfs and ZFS) provide internal data and metadata checksumming, what is used for detecting silent data corruption; if a corruption is detected that way and internal RAID mechanisms provided by those filesystems are also used, such filesystems can additionally reconstruct corrupted data in a transparent way. This approach allows improved data integrity protection covering the entire data paths, which is usually known as *end-to-end data protection*.

"Data scrubbing" is another method to reduce the likelihood of data corruption, as disk errors are caught and recovered from, before multiple errors accumulate and overwhelm the number of parity bits. Instead of parity being checked on each read, the parity is checked during a regular scan of the disk, often done as a low priority background process. Note that the "data scrubbing" operation activates a parity check. If a user simply runs a normal program that reads data from the disk, then the parity would not be checked unless parity-check-on-read was both supported and enabled on the disk subsystem.

If appropriate mechanisms are employed to detect and remedy data corruption, data integrity can be maintained. This is particularly important in commercial applications (e.g. banking), where an undetected error could either corrupt a database index or change data to drastically affect an account balance, and in the use of encrypted or compressed data, where a small error can make an extensive dataset unusable.

References

Data integrity

Data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes or retrieves data. The term **data integrity** is broad in scope and may have widely different meanings depending on the specific context - even under the same general umbrella of computing. This article provides only a broad overview of some of the different types and concerns of data integrity.

Data integrity is the opposite of data corruption, which is a form of data loss. The overall intent of any data integrity technique is the same: ensure data is recorded exactly as intended (such as a database correctly rejecting mutually exclusive possibilities,) and upon later retrieval, ensure the data is the same as it was when it was originally recorded. In short, data integrity aims to prevent unintentional changes to information. Data integrity is not to be confused with data security, the discipline of protecting data from unauthorized parties.

Any unintended changes to data as the result of a storage, retrieval or processing operation, including malicious intent, unexpected hardware failure, and human error, is failure of data integrity. If the changes are the result of unauthorized access, it may also be a failure of data security. Depending on the data involved this could manifest itself as benign as a single pixel in an image appearing a different color than was originally recorded, to the loss of vacation pictures or a business-critical database, to even catastrophic loss of human life in a Life-critical system.

Physical vs. logical integrity

Data integrity can be roughly divided into two overlapping categories:

Physical integrity - deals with challenges associated with correctly storing and fetching the data itself. Challenges with physical integrity may include electromechanical faults, design flaws, material fatigue, corrosion, power outages, natural disasters, acts of war and terrorism, and other special environmental hazards such as ionizing radiation, extreme temperatures, pressures and g-forces. Ensuring physical integrity includes methods such as redundant hardware, an uninterruptible power supply, certain types of RAID arrays, radiation hardened chips, ECC memory, use of a clustered file system, using file systems that employ block level checksums such as ZFS, storage

arrays that compute parity calculations such as Exclusive or or use a Cryptographic hash function and even having a watchdog timer on critical subsystems.

Physical integrity often makes extensive use of error detecting algorithms known as error-correcting codes. Human induced data integrity errors are often detected through the use of simpler check digits and algorithms used to detect them such as the Damm algorithm or Luhn algorithm. These are used to maintain data integrity after manual transcription from one computer system to another by a human intermediary. Examples include credit card and bank routing numbers. Computer induced transcription errors can be detected through hash functions.

In production systems these techniques are used in combination to ensure various degrees of data integrity. For example a computer file system may configured on a fault tolerant RAID array, but might not provide block level checksums to detect and prevent silent data corruption. A database management system might be ACID compliant, but the raid controller or hard drive's internal write-cache might not be.

Logical integrity - concerned with the correctness or rationality of a piece of data, given a particular context. This includes topics such as referential integrity and entity integrity in a relational database or correctly ignoring impossible sensor data in robotic systems. These concerns involve making certain the data "makes sense" given its environment. Challenges include software bugs, design flaws, human error. Common methods of ensuring logical integrity include things such as a Check constraint, foreign key constraint, program assertion (computing) and other runtime sanity checks.

Both physical and logical integrity often share many common challenges such as human error, design flaws and both must appropriately deal with concurrent requests to record and retrieve data, the later of which is its own subject entirely. See mutex and Copy-on-write.

Databases

Data integrity contains guidelines for data retention, specifying or guaranteeing the length of time of data can be retained in a particular database. It specifies what can be done with data values when its validity or usefulness expires. In order to achieve data integrity, these rules are consistently and routinely applied to all data entering the system, and any relaxation of enforcement could cause errors in the data. Implementing checks on the data as close as possible to the source of input (such as human data entry), causes less erroneous data to enter the system. Strict enforcement of data integrity rules causes the error rates to be lower, resulting in time saved troubleshooting and tracing erroneous data and the errors it causes algorithms.

Data integrity also includes rules defining the relations a piece of data can have, to other pieces of data, such as a *Customer* record being allowed to link to purchased *Products*, but not to unrelated data such as *Corporate Assets*. Data integrity often includes checks and correction for invalid data, based on a fixed schema or a predefined set of rules. An example being textual data entered where a date-time value is required. Rules for data derivation are also applicable, specifying how a data value is derived based on algorithm, contributors and conditions. It also specifies the conditions on how the data value could be re-derived.

Types of integrity constraints

Data integrity is normally enforced in a database system by a series of integrity constraints or rules. Three types of integrity constraints are an inherent part of the relational data model: entity integrity, referential integrity and domain integrity:

- *Entity integrity* concerns the concept of a primary key. Entity integrity is an integrity rule which states that every table must have a primary key and that the column or columns chosen to be the primary key should be unique and not null.
- *Referential integrity* concerns the concept of a foreign key. The referential integrity rule states that any foreign-key value can only be in one of two states. The usual state of affairs is that the foreign key value refers to

a primary key value of some table in the database. Occasionally, and this will depend on the rules of the data owner, a foreign-key value can be null. In this case we are explicitly saying that either there is no relationship between the objects represented in the database or that this relationship is unknown.

- *Domain integrity* specifies that all columns in relational database must be declared upon a defined domain. The primary unit of data in the relational data model is the data item. Such data items are said to be non-decomposable or atomic. A domain is a set of values of the same type. Domains are therefore pools of values from which actual values appearing in the columns of a table are drawn.

If a database supports these features it is the responsibility of the database to insure data integrity as well as the consistency model for the data storage and retrieval. If a database does not support these features it is the responsibility of the applications to ensure data integrity while the database supports the consistency model for the data storage and retrieval.

Having a single, well-controlled, and well-defined data-integrity system increases

- stability (one centralized system performs all data integrity operations)
- performance (all data integrity operations are performed in the same tier as the consistency model)
- re-usability (all applications benefit from a single centralized data integrity system)
- maintainability (one centralized system for all data integrity administration).

As of 2012[1], since all modern databases support these features (see Comparison of relational database management systems), it has become the *de-facto* responsibility of the database to ensure data integrity. Out-dated and legacy systems that use file systems (text, spreadsheets, ISAM, flat files, etc.) for their consistency model lack any[*citation needed*] kind of data-integrity model. This requires organizations to invest a large amount of time, money, and personnel in building data-integrity systems on a per-application basis that effectively just duplicate the existing data integrity systems found in modern databases. Many companies, and indeed many database systems themselves, offer products and services to migrate out-dated and legacy systems to modern databases to provide these data-integrity features. This offers organizations substantial savings in time, money, and resources because they do not have to develop per-application data-integrity systems that must be re-factored each time business requirements change.

Examples

An example of a data-integrity mechanism is the parent-and-child relationship of related records. If a parent record owns one or more related child records all of the referential integrity processes are handled by the database itself, which automatically insures the accuracy and integrity of the data so that no child record can exist without a parent (also called being orphaned) and that no parent loses their child records. It also ensures that no parent record can be deleted while the parent record owns any child records. All of this is handled at the database level and does not require coding integrity checks into each applications.

File systems

Various research results show that neither widespread filesystems (including UFS, Ext, XFS, JFS and NTFS) nor hardware RAID solutions provide sufficient protection against data integrity problems.

Some filesystems (including Btrfs and ZFS) provide internal data and metadata checksumming, what is used for detecting silent data corruption and improving data integrity. If a corruption is detected that way and internal RAID mechanisms provided by those filesystems are also used, such filesystems can additionally reconstruct corrupted data in a transparent way. This approach allows improved data integrity protection covering the entire data paths, which is usually known as end-to-end data protection.

Data storage

Apart from data in databases, standards exist to address the integrity of data on storage devices.

References

[1] http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit

Further reading

- This article incorporates public domain material from the General Services Administration document "Federal Standard 1037C" (http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm) (in support of MIL-STD-188).
- Xiaoyun Wang; Hongbo Yu (2005). "How to Break MD5 and Other Hash Functions" (http://www.infosec.sdu.edu.cn/uploadfile/papers/How to Break MD5 and Other Hash Functions.pdf). *EUROCRYPT*. ISBN 3-540-25910-4.

Data profiling

Data profiling is the process of examining the data available in an existing data source (e.g. a database or a file) and collecting statistics and information about that data. The purpose of these statistics may be to:

1. Find out whether existing data can easily be used for other purposes
2. Improve the ability to search the data by tagging it with keywords, descriptions, or assigning it to a category
3. Give metrics on data quality including whether the data conforms to particular standards or patterns
4. Assess the risk involved in integrating data for new applications, including the challenges of joins
5. Assess whether metadata accurately describes the actual values in the source database
6. Understanding data challenges early in any data intensive project, so that late project surprises are avoided. Finding data problems late in the project can lead to delays and cost overruns.
7. Have an enterprise view of all data, for uses such as master data management where key data is needed, or data governance for improving data quality.

Data Profiling in Relation to Data Warehouse/Business Intelligence Development

Introduction

Data profiling is an analysis of the candidate data sources for a data warehouse to clarify the structure, content, relationships and derivation rules of the data.[1] Profiling helps not only to understand anomalies and to assess data quality, but also to discover, register, and assess enterprise metadata.[2] Thus the purpose of data profiling is both to validate metadata when it is available and to discover metadata when it is not.[3] The result of the analysis is used both strategically, to determine suitability of the candidate source systems and give the basis for an early go/no-go decision, and tactically, to identify problems for later solution design, and to level sponsors' expectations.

How to do Data Profiling

Data profiling utilizes different kinds of descriptive statistics such as minimum, maximum, mean, mode, percentile, standard deviation, frequency, and variation as well as other aggregates such as count and sum. Additional metadata information obtained during data profiling could be data type, length, discrete values, uniqueness, occurrence of null values, typical string patterns, and abstract type recognition.[4][5] The metadata can then be used to discover problems such as illegal values, misspelling, missing values, varying value representation, and duplicates. Different

analyses are performed for different structural levels. E.g. single columns could be profiled individually to get an understanding of frequency distribution of different values, type, and use of each column. Embedded value dependencies can be exposed in cross-columns analysis. Finally, overlapping value sets possibly representing foreign key relationships between entities can be explored in an inter-table analysis. Normally purpose-built tools are used for data profiling to ease the process.[6][7] The computation complexity increases when going from single column, to single table, to cross-table structural profiling. Therefore, performance is an evaluation criterion for profiling tools.

When to Conduct Data Profiling

According to Kimball, data profiling is performed several times and with varying intensity throughout the data warehouse developing process. A light profiling assessment should be undertaken as soon as candidate source systems have been identified right after the acquisition of the business requirements for the DW/BI. The purpose is to clarify at an early stage if the right data is available at the right detail level and that anomalies can be handled subsequently. If this is not the case the project might have to be canceled. More detailed profiling is done prior to the dimensional modeling process in order to see what it will require to convert data into the dimensional model, and extends into the ETL system design process to establish what data to extract and which filters to apply. An additional time to conduct data profiling is during the data warehouse development process after data has been loaded into staging, the data marts, etc. Doing so at these points in time helps assure that data cleaning and transformations have been done correctly according to requirements.

Benefits of Data Profiling

The benefits of data profiling is to improve data quality, shorten the implementation cycle of major projects, and improve understanding of data for the users. Discovering business knowledge embedded in data itself is one of the significant benefits derived from data profiling. Data profiling is one of the most effective technologies for improving data accuracy in corporate databases. Although data profiling is effective, then do remember to find a suitable balance and do not slip into “analysis paralysis”.

References

[1] [Ralph Kimball et al. (2008), “The Data Warehouse Lifecycle Toolkit”, Second Edition, Wiley Publishing, Inc., ISBN 9780470149775], (p. 297) (p. 376)

[2] [David Loshin (2009), “Master Data Management”, Morgan Kaufmann Publishers, ISBN 9780123742254], (p. 94-96)

[3] [David Loshin (2003), “Business Intelligence: The Savvy Manager's Guide, Getting Onboard with Emerging IT”, Morgan Kaufmann Publishers, ISBN 9781558609167], (p. 110-111)]

[4] [Erhard Rahm and Hong Hai Do (2000), “Data Cleaning: Problems and Current Approaches” in “Bulletin of the Technical Committee on Data Engineering”, IEEE Computer Society, Vol. 23, No. 4, December 2000]

[5] [Ranjit Singh, Dr Kawaljeet Singh et al. (2010), “A Descriptive Classification of Causes of Data Quality Problems in Data Warehousing”, IJCSI International Journal of Computer Science Issue, Vol. 7, Issue 3, No. 2, May 2010]

[6] "[Ralph Kimball (2004), “Kimball Design Tip #59: Surprising Value of Data Profiling”, Kimball Group, Number 59, September 14, 2004, (www.rkimball.com/html/designtipsPDF/ KimballDT59 SurprisingValue.pdf)]

[7] [Jack E. Olson (2003), “Data Quality: The Accuracy dimension”, Morgan Kaufmann Publishers], (p.140-142)

Data quality

Data are of high quality "if they are fit for their intended uses in operations, decision making and planning" (J. M. Juran). Alternatively, the data are deemed of high quality if they correctly represent the real-world construct to which they refer. Furthermore, apart from these definitions, as data volume increases, the question of *internal consistency* within data becomes paramount, regardless of fitness for use for any external purpose, e.g. a person's age and birth date may conflict within different parts of a database. The first views can often be in disagreement, even about the same set of data used for the same purpose. This article discusses the concept as it related to business data processing, although of course other data have various quality issues as well.

Definitions

This list is taken from the online book "Data Quality: High-impact Strategies". See also the Glossary of data quality terms [1]

- Degree of excellence exhibited by the data in relation to the portrayal of the actual scenario.
- The state of completeness, validity, consistency, timeliness and accuracy that makes data appropriate for a specific use.[2]
- The totality of features and characteristics of data that bears on their ability to satisfy a given purpose; the sum of the degrees of excellence for factors related to data.[3]
- The processes and technologies involved in ensuring the conformance of data values to business requirements and acceptance criteria.[4]
- Complete, standards based, consistent, accurate and time stamped.[5]

History

Before the rise of the inexpensive server, massive mainframe computers were used to maintain name and address data so that the mail could be properly routed to its destination. The mainframes used business rules to correct common misspellings and typographical errors in name and address data, as well as to track customers who had moved, died, gone to prison, married, divorced, or experienced other life-changing events. Government agencies began to make postal data available to a few service companies to cross-reference customer data with the National Change of Address registry (NCOA). This technology saved large companies millions of dollars compared to manually correcting customer data. Large companies saved on postage, as bills and direct marketing materials made their way to the intended customer more accurately. Initially sold as a service, data quality moved inside the walls of corporations, as low-cost and powerful server technology became available.

Companies with an emphasis on marketing often focus their quality efforts on name and address information, but data quality is recognized as an important property of all types of data. Principles of data quality can be applied to supply chain data, transactional data, and nearly every other category of data found in the enterprise. For example, making supply chain data conform to a certain standard has value to an organization by: 1) avoiding overstocking of similar but slightly different stock; 2) improving the understanding of vendor purchases to negotiate volume discounts; and 3) avoiding logistics costs in stocking and shipping parts across a large organization.

While name and address data has a clear standard as defined by local postal authorities, other types of data have few recognized standards. There is a movement in the industry today to standardize certain non-address data. The non-profit group GS1 is among the groups spearheading this movement.

For companies with significant research efforts, data quality can include developing protocols for research methods, reducing measurement error, bounds checking of the data, cross tabulation, modeling and outlier detection, verifying data integrity, etc.

Overview

There are a number of theoretical frameworks for understanding data quality. A systems-theoretical approach influenced by American pragmatism expands the definition of data quality to include information quality, and emphasizes the inclusiveness of the fundamental dimensions of accuracy and precision on the basis of the theory of science (Ivanov, 1972). One framework, dubbed "Zero Defect Data" (Hansen, 1991) adapts the principles of statistical process control to data quality. Another framework seeks to integrate the product perspective (conformance to specifications) and the service perspective (meeting consumers' expectations) (Kahn et al. 2002). Another framework is based in semiotics to evaluate the quality of the form, meaning and use of the data (Price and Shanks, 2004). One highly theoretical approach analyzes the ontological nature of information systems to define data quality rigorously (Wand and Wang, 1996).

A considerable amount of data quality research involves investigating and describing various categories of desirable attributes (or dimensions) of data. These lists commonly include accuracy, correctness, currency, completeness and relevance. Nearly 200 such terms have been identified and there is little agreement in their nature (are these concepts, goals or criteria?), their definitions or measures (Wang et al., 1993). Software engineers may recognise this as a similar problem to "ilities".

MIT has a Total Data Quality Management program, led by Professor Richard Wang, which produces a large number of publications and hosts a significant international conference in this field (International Conference on Information Quality, ICIQ). This program grew out of the work done by Hansen on the "Zero Defect Data" framework (Hansen, 1991).

In practice, data quality is a concern for professionals involved with a wide range of information systems, ranging from data warehousing and business intelligence to customer relationship management and supply chain management. One industry study estimated the total cost to the US economy of data quality problems at over US$600 billion per annum (Eckerson, 2002). Incorrect data – which includes invalid and outdated information – can originate from different data sources – through data entry, or data migration and conversion projects.[6]

In 2002, the USPS and PricewaterhouseCoopers released a report stating that 23.6 percent of all U.S. mail sent is incorrectly addressed.[7]

One reason contact data becomes stale very quickly in the average database – more than 45 million Americans change their address every year.[8]

In fact, the problem is such a concern that companies are beginning to set up a data governance team whose sole role in the corporation is to be responsible for data quality. In someWikipedia:Avoid weasel words organizations, this data governance function has been established as part of a larger Regulatory Compliance function - a recognition of the importance of Data/Information Quality to organizations.

Problems with data quality don't only arise from *incorrect* data. *Inconsistent* data is a problem as well. Eliminating data shadow systems and centralizing data in a warehouse is one of the initiatives a company can take to ensure data consistency.

Enterprises, scientists, and researchers are starting to participate within data curation communities to improve the quality of their common data.[9]

The market is going some way to providing data quality assurance. A number of vendors make tools for analysing and repairing poor quality data *in situ*, service providers can clean the data on a contract basis and consultants can advise on fixing processes or systems to avoid data quality problems in the first place. Most data quality tools offer a series of tools for improving data, which may include some or all of the following:

1. Data profiling - initially assessing the data to understand its quality challenges
2. Data standardization - a business rules engine that ensures that data conforms to quality rules
3. Geocoding - for name and address data. Corrects data to US and Worldwide postal standards

4. Matching or Linking - a way to compare data so that similar, but slightly different records can be aligned. Matching may use "fuzzy logic" to find duplicates in the data. It often recognizes that 'Bob' and 'Robert' may be the same individual. It might be able to manage 'householding', or finding links between husband and wife at the same address, for example. Finally, it often can build a 'best of breed' record, taking the best components from multiple data sources and building a single super-record.
5. Monitoring - keeping track of data quality over time and reporting variations in the quality of data. Software can also auto-correct the variations based on pre-defined business rules.
6. Batch and Real time - Once the data is initially cleansed (batch), companies often want to build the processes into enterprise applications to keep it clean.

There are several well-known authors and self-styled experts, with Larry English perhaps the most popular guru. In addition, the International Association for Information and Data Quality (IAIDQ) [10] was established in 2004 to provide a focal point for professionals and researchers in this field.

ISO 8000 is the international standard for data quality.

Data quality control

Data quality control is the process of controlling the usage of data with known quality measurement—for an application or a process. This process is usually done after a Data Quality Assurance (QA) process, which consists of discovery of data inconsistency and correction.

Data QA process provides following information to Data Quality Control (QC):

- Severity of inconsistency
- Incompleteness
- Accuracy
- Precision
- Missing / Unknown

The Data QC process uses the information from the QA process, then it decides to use the data for analysis or in an application or business process. For example, if a Data QC process finds the data contains too much error or inconsistency, it rejects the data to be processed. The usage of incorrect data could crucially impact output. For example, providing invalid measurements from several sensors to the automatic pilot feature on an aircraft could cause it to crash. Thus, establishing data QC process provides the protection of usage of data control and establishes safe information usage.

optimum use of data quality

Data Quality is a niche area which is the backbone for the integrity of the data management by covering gaps of data issues. This is the key piece that aids to data governance and monitors data to find exceptions undiscovered by current data management operations. Some data quality checks may translate as future business requirements. Below are the possible sections of data flows that need perennial DQ checks.

Data quality excludes activities done in business logic to avoid duplication of the check (redundancy) and better ROI (return of investment).

Data if sourced from database that has valid not null columns should not under go DQ check for completeness and precision. (redundancy). So it's done infrequently to discover database design issues or Source inconsistencies. (this should be avoided as a regular check). Results are used for DB structure changes

Data if validated against a set of well-defined valid values are atomically performed to discover new valid values or discrepant valid values through the validity DQ check. Results are used for Reference data updates in the master data management.

Data if validated from third party before providing to the internal transformations should be checked for its accuracy (DQ).

Data, either semi static or a golden copy, that's maintained internally (reference data or master data (Master data management: MDM)) is validated for its consistency check to segregate discrepant or/and new data for the MDM process.

Time stamp, that proves the data position (time and location), comparisons are done to verify the SLA of the data from one team to another team and validated against a known maximum constant time difference for its timeliness DQ checks. SLA should be defined and used as source for this DQ check.

Complex logic which is segregated into multiple processes and can be rolled up into a logical result with minimum/maximum values or interrelationships can be validated for reasonableness DQ check. This check may discover outliers of the data and its drift from BAU (business as usual) and eventually provide possible exceptions. And then they may be identified as data issues. This check may be simple generic rules engulfed by large chunk of data or it can be complicated logic pertaining to the core business of the company. This DQ check requires high degree of business knowledge. And discovery of reasonableness issues may aid for policy/strategy changes by either business or data governance or both.

Conformity checks and integrity checks need not covered in all business needs, it's strictly under the database architecture's discretion.

Data Quality Assurance

Data quality assurance is the process of profiling the data to discover inconsistencies, and other anomalies in the data and performing data cleansing activities (e.g. removing outliers, missing data interpolation) to improve the data quality .

These activities can be undertaken as part of data warehousing or as part of the database administration of an existing piece of applications software.

Criticism of existing tools and processes

The main reasons cited are:

- **Project costs**: costs typically in the hundreds of thousands of dollars
- **Time**: lack of enough time to deal with large-scale data-cleansing software
- **Security**: concerns over sharing information, giving an application access across systems, and effects on legacy systems

Professional associations

International Association for Information and Data Quality (IAIDQ) [10]

References

[1] Glossary of data quality terms (http://iaidq.org/main/glossary.shtml) published by IAIDQ (http://iaidq.org/)
[2] Government of British Columbia (http://www.cio.gov.bc.ca/other/daf/IRM_Glossary.htm)
[3] REFERENCE-QUALITY WATER SAMPLE DATA: NOTES ON ACQUISITION, RECORD KEEPING, AND EVALUATION (http://www.go-ship.org/Manual/Swift_DataEval.pdf)
[4] istabg.org Data QualYtI – Do You Trust Your Data? (http://istabg.org/?p=647)
[5] GS1.ORG dqf (http://www.gs1.org/gdsn/dqf)
[6] http://www.information-management.com/issues/20060801/1060128-1.html
[7] http://www.directionsmag.com/article.php?article_id=509
[8] http://ribbs.usps.gov/move_update/documents/tech_guides/PUB363.pdf

[9] E. Curry, A. Freitas, and S. O'Riáin, "The Role of Community-Driven Data Curation for Enterprises," (http://3roundstones.com/led_book/led-curry-et-al.html) in Linking Enterprise Data, D. Wood, Ed. Boston, MA: Springer US, 2010, pp. 25-47.
[10] http://iaidq.org/

Further reading

- Baamann, Katharina, "Data Quality Aspects of Revenue Assurance", Article (http://mitiq.mit.edu/iciq/PDF/DATA QUALITY ASPECTS OF REVENUE ASSURANCE.pdf)
- Eckerson, W. (2002) "Data Warehousing Special Report: Data quality and the bottom line", Article (http://www.adtmag.com/article.asp?id=6321)
- Ivanov, K. (1972) "Quality-control of information: On the concept of accuracy of information in data banks and in management information systems" (http://www.informatik.umu.se/~kivanov/diss-avh.html). The University of Stockholm and The Royal Institute of Technology. Doctoral dissertation.
- Hansen, M. (1991) Zero Defect Data, MIT. Masters thesis (http://dspace.mit.edu/handle/1721.1/13812)
- Kahn, B., Strong, D., Wang, R. (2002) "Information Quality Benchmarks: Product and Service Performance," Communications of the ACM, April 2002. pp. 184–192. Article (http://mitiq.mit.edu/Documents/Publications/TDQMpub/2002/IQ Benchmarks.pdf)
- Price, R. and Shanks, G. (2004) A Semiotic Information Quality Framework, Proc. IFIP International Conference on Decision Support Systems (DSS2004): Decision Support in an Uncertain and Complex World, Prato. Article (http://vishnu.sims.monash.edu.au/dss2004/proceedings/pdf/65_Price_Shanks.pdf)
- Redman, T. C. (2004) Data: An Unfolding Quality Disaster Article (http://www.dmreview.com/article_sub.cfm?articleId=1007211)
- Wand, Y. and Wang, R. (1996) "Anchoring Data Quality Dimensions in Ontological Foundations," Communications of the ACM, November 1996. pp. 86–95. Article (http://web.mit.edu/tdqm/www/tdqmpub/WandWangCACMNov96.pdf)
- Wang, R., Kon, H. & Madnick, S. (1993), Data Quality Requirements Analysis and Modelling, Ninth International Conference of Data Engineering, Vienna, Austria. Article (http://web.mit.edu/tdqm/www/tdqmpub/IEEEDEApr93.pdf)
- Fournel Michel, Accroitre la qualité et la valeur des données de vos clients, éditions Publibook, 2007. ISBN 978-2-7483-3847-8.
- Daniel F., Casati F., Palpanas T., Chayka O., Cappiello C. (2008) "Enabling Better Decisions through Quality-aware Reports", International Conference on Information Quality (ICIQ), MIT. Article (http://dit.unitn.it/~themis/publications/iciq08.pdf)
- Jack E. Olson (2003), "Data Quality: The Accuracy dimension", Morgan Kaufmann Publishers

Data quality assessment

Data quality assessment is the process of exposing technical and business data issues in order to plan data cleansing and data enrichment strategies. Technical quality issues are generally easy to discover and correct, such as:

- Inconsistent standards in structure, format/ values
- Missing data, default values
- Spelling errors, data in wrong fields

Business quality issues are more subjective and are associated with business processes such as generating accurate reports, ensuring that data driven processes are working correctly.

Business data quality measures like accuracy and correctness are subjective and need Subject Matter Expert (SME) involvement to assess data quality.

References

- Feeney, Griffith. Data Assessment. In: Paul Demeny and Geoffrey McNicoll (Eds.). 2003. The Encyclopedia of Population. New York, Macmillan Reference USA, vol.1, 190-193
- Ehling, Manfred, Körner, Thomas. (eds.) Handbook on Data Quality Assessment Methods and Tools [1] Eurostat, European Commission, Wiesbaden, 2007.

References

[1] http://epp.eurostat.ec.europa.eu/portal/page/portal/quality/documents/HANDBOOK%20ON%20DATA%20QUALITY%20ASSESSMENT%20METHODS%20AND%20TOOLS%20%20I.pdf

Data quality assurance

Data are of high quality "if they are fit for their intended uses in operations, decision making and planning" (J. M. Juran). Alternatively, the data are deemed of high quality if they correctly represent the real-world construct to which they refer. Furthermore, apart from these definitions, as data volume increases, the question of *internal consistency* within data becomes paramount, regardless of fitness for use for any external purpose, e.g. a person's age and birth date may conflict within different parts of a database. The first views can often be in disagreement, even about the same set of data used for the same purpose. This article discusses the concept as it related to business data processing, although of course other data have various quality issues as well.

Definitions

This list is taken from the online book "Data Quality: High-impact Strategies". See also the Glossary of data quality terms [1]

- Degree of excellence exhibited by the data in relation to the portrayal of the actual scenario.
- The state of completeness, validity, consistency, timeliness and accuracy that makes data appropriate for a specific use.[2]
- The totality of features and characteristics of data that bears on their ability to satisfy a given purpose; the sum of the degrees of excellence for factors related to data.[3]
- The processes and technologies involved in ensuring the conformance of data values to business requirements and acceptance criteria.[4]
- Complete, standards based, consistent, accurate and time stamped.[5]

History

Before the rise of the inexpensive server, massive mainframe computers were used to maintain name and address data so that the mail could be properly routed to its destination. The mainframes used business rules to correct common misspellings and typographical errors in name and address data, as well as to track customers who had moved, died, gone to prison, married, divorced, or experienced other life-changing events. Government agencies began to make postal data available to a few service companies to cross-reference customer data with the National Change of Address registry (NCOA). This technology saved large companies millions of dollars compared to manually correcting customer data. Large companies saved on postage, as bills and direct marketing materials made their way to the intended customer more accurately. Initially sold as a service, data quality moved inside the walls of corporations, as low-cost and powerful server technology became available.

Companies with an emphasis on marketing often focus their quality efforts on name and address information, but data quality is recognized as an important property of all types of data. Principles of data quality can be applied to supply chain data, transactional data, and nearly every other category of data found in the enterprise. For example, making supply chain data conform to a certain standard has value to an organization by: 1) avoiding overstocking of similar but slightly different stock; 2) improving the understanding of vendor purchases to negotiate volume discounts; and 3) avoiding logistics costs in stocking and shipping parts across a large organization.

While name and address data has a clear standard as defined by local postal authorities, other types of data have few recognized standards. There is a movement in the industry today to standardize certain non-address data. The non-profit group GS1 is among the groups spearheading this movement.

For companies with significant research efforts, data quality can include developing protocols for research methods, reducing measurement error, bounds checking of the data, cross tabulation, modeling and outlier detection, verifying data integrity, etc.

Overview

There are a number of theoretical frameworks for understanding data quality. A systems-theoretical approach influenced by American pragmatism expands the definition of data quality to include information quality, and emphasizes the inclusiveness of the fundamental dimensions of accuracy and precision on the basis of the theory of science (Ivanov, 1972). One framework, dubbed "Zero Defect Data" (Hansen, 1991) adapts the principles of statistical process control to data quality. Another framework seeks to integrate the product perspective (conformance to specifications) and the service perspective (meeting consumers' expectations) (Kahn et al. 2002). Another framework is based in semiotics to evaluate the quality of the form, meaning and use of the data (Price and Shanks, 2004). One highly theoretical approach analyzes the ontological nature of information systems to define data quality rigorously (Wand and Wang, 1996).

A considerable amount of data quality research involves investigating and describing various categories of desirable attributes (or dimensions) of data. These lists commonly include accuracy, correctness, currency, completeness and relevance. Nearly 200 such terms have been identified and there is little agreement in their nature (are these concepts, goals or criteria?), their definitions or measures (Wang et al., 1993). Software engineers may recognise this as a similar problem to "ilities".

MIT has a Total Data Quality Management program, led by Professor Richard Wang, which produces a large number of publications and hosts a significant international conference in this field (International Conference on Information Quality, ICIQ). This program grew out of the work done by Hansen on the "Zero Defect Data" framework (Hansen, 1991).

In practice, data quality is a concern for professionals involved with a wide range of information systems, ranging from data warehousing and business intelligence to customer relationship management and supply chain management. One industry study estimated the total cost to the US economy of data quality problems at over

US$600 billion per annum (Eckerson, 2002). Incorrect data – which includes invalid and outdated information – can originate from different data sources – through data entry, or data migration and conversion projects.[6]

In 2002, the USPS and PricewaterhouseCoopers released a report stating that 23.6 percent of all U.S. mail sent is incorrectly addressed.[7]

One reason contact data becomes stale very quickly in the average database – more than 45 million Americans change their address every year.[8]

In fact, the problem is such a concern that companies are beginning to set up a data governance team whose sole role in the corporation is to be responsible for data quality. In someWikipedia:Avoid weasel words organizations, this data governance function has been established as part of a larger Regulatory Compliance function - a recognition of the importance of Data/Information Quality to organizations.

Problems with data quality don't only arise from *incorrect* data. *Inconsistent* data is a problem as well. Eliminating data shadow systems and centralizing data in a warehouse is one of the initiatives a company can take to ensure data consistency.

Enterprises, scientists, and researchers are starting to participate within data curation communities to improve the quality of their common data.[9]

The market is going some way to providing data quality assurance. A number of vendors make tools for analysing and repairing poor quality data *in situ*, service providers can clean the data on a contract basis and consultants can advise on fixing processes or systems to avoid data quality problems in the first place. Most data quality tools offer a series of tools for improving data, which may include some or all of the following:

1. Data profiling - initially assessing the data to understand its quality challenges
2. Data standardization - a business rules engine that ensures that data conforms to quality rules
3. Geocoding - for name and address data. Corrects data to US and Worldwide postal standards
4. Matching or Linking - a way to compare data so that similar, but slightly different records can be aligned. Matching may use "fuzzy logic" to find duplicates in the data. It often recognizes that 'Bob' and 'Robert' may be the same individual. It might be able to manage 'householding', or finding links between husband and wife at the same address, for example. Finally, it often can build a 'best of breed' record, taking the best components from multiple data sources and building a single super-record.
5. Monitoring - keeping track of data quality over time and reporting variations in the quality of data. Software can also auto-correct the variations based on pre-defined business rules.
6. Batch and Real time - Once the data is initially cleansed (batch), companies often want to build the processes into enterprise applications to keep it clean.

There are several well-known authors and self-styled experts, with Larry English perhaps the most popular guru. In addition, the International Association for Information and Data Quality (IAIDQ) [10] was established in 2004 to provide a focal point for professionals and researchers in this field.

ISO 8000 is the international standard for data quality.

Data quality control

Data quality control is the process of controlling the usage of data with known quality measurement—for an application or a process. This process is usually done after a Data Quality Assurance (QA) process, which consists of discovery of data inconsistency and correction.

Data QA process provides following information to Data Quality Control (QC):

- Severity of inconsistency
- Incompleteness
- Accuracy
- Precision
- Missing / Unknown

The Data QC process uses the information from the QA process, then it decides to use the data for analysis or in an application or business process. For example, if a Data QC process finds the data contains too much error or inconsistency, it rejects the data to be processed. The usage of incorrect data could crucially impact output. For example, providing invalid measurements from several sensors to the automatic pilot feature on an aircraft could cause it to crash. Thus, establishing data QC process provides the protection of usage of data control and establishes safe information usage.

optimum use of data quality

Data Quality is a niche area which is the backbone for the integrity of the data management by covering gaps of data issues. This is the key piece that aids to data governance and monitors data to find exceptions undiscovered by current data management operations. Some data quality checks may translate as future business requirements. Below are the possible sections of data flows that need perennial DQ checks.

Data quality excludes activities done in business logic to avoid duplication of the check (redundancy) and better ROI (return of investment).

Data if sourced from database that has valid not null columns should not under go DQ check for completeness and precision. (redundancy). So it's done infrequently to discover database design issues or Source inconsistencies. (this should be avoided as a regular check). Results are used for DB structure changes

Data if validated against a set of well-defined valid values are atomically performed to discover new valid values or discrepant valid values through the validity DQ check. Results are used for Reference data updates in the master data management.

Data if validated from third party before providing to the internal transformations should be checked for its accuracy (DQ).

Data, either semi static or a golden copy, that's maintained internally (reference data or master data (Master data management: MDM)) is validated for its consistency check to segregate discrepant or/and new data for the MDM process.

Time stamp, that proves the data position (time and location), comparisons are done to verify the SLA of the data from one team to another team and validated against a known maximum constant time difference for its timeliness DQ checks. SLA should be defined and used as source for this DQ check.

Complex logic which is segregated into multiple processes and can be rolled up into a logical result with minimum/maximum values or interrelationships can be validated for reasonableness DQ check. This check may discover outliers of the data and its drift from BAU (business as usual) and eventually provide possible exceptions. And then they may be identified as data issues. This check may be simple generic rules engulfed by large chunk of data or it can be complicated logic pertaining to the core business of the company. This DQ check requires high degree of business knowledge. And discovery of reasonableness issues may aid for policy/strategy changes by either

business or data governance or both.

Conformity checks and integrity checks need not covered in all business needs, it's strictly under the database architecture's discretion.

Data Quality Assurance

Data quality assurance is the process of profiling the data to discover inconsistencies, and other anomalies in the data and performing data cleansing activities (e.g. removing outliers, missing data interpolation) to improve the data quality .

These activities can be undertaken as part of data warehousing or as part of the database administration of an existing piece of applications software.

Criticism of existing tools and processes

The main reasons cited are:

- **Project costs**: costs typically in the hundreds of thousands of dollars
- **Time**: lack of enough time to deal with large-scale data-cleansing software
- **Security**: concerns over sharing information, giving an application access across systems, and effects on legacy systems

Professional associations

International Association for Information and Data Quality (IAIDQ) [10]

References

[1] Glossary of data quality terms (http://iaidq.org/main/glossary.shtml) published by IAIDQ (http://iaidq.org/)
[2] Government of British Columbia (http://www.cio.gov.bc.ca/other/daf/IRM_Glossary.htm)
[3] REFERENCE-QUALITY WATER SAMPLE DATA: NOTES ON ACQUISITION, RECORD KEEPING, AND EVALUATION (http://www.go-ship.org/Manual/Swift_DataEval.pdf)
[4] istabg.org Data QualYtI – Do You Trust Your Data? (http://istabg.org/?p=647)
[5] GS1.ORG dqf (http://www.gs1.org/gdsn/dqf)
[6] http://www.information-management.com/issues/20060801/1060128-1.html
[7] http://www.directionsmag.com/article.php?article_id=509
[8] http://ribbs.usps.gov/move_update/documents/tech_guides/PUB363.pdf
[9] E. Curry, A. Freitas, and S. O'Riáin, "The Role of Community-Driven Data Curation for Enterprises," (http://3roundstones.com/led_book/led-curry-et-al.html) in Linking Enterprise Data, D. Wood, Ed. Boston, MA: Springer US, 2010, pp. 25-47.

Further reading

- Baamann, Katharina, "Data Quality Aspects of Revenue Assurance", Article (http://mitiq.mit.edu/iciq/PDF/DATA QUALITY ASPECTS OF REVENUE ASSURANCE.pdf)
- Eckerson, W. (2002) "Data Warehousing Special Report: Data quality and the bottom line", Article (http://www.adtmag.com/article.asp?id=6321)
- Ivanov, K. (1972) "Quality-control of information: On the concept of accuracy of information in data banks and in management information systems" (http://www.informatik.umu.se/~kivanov/diss-avh.html). The University of Stockholm and The Royal Institute of Technology. Doctoral dissertation.
- Hansen, M. (1991) Zero Defect Data, MIT. Masters thesis (http://dspace.mit.edu/handle/1721.1/13812)
- Kahn, B., Strong, D., Wang, R. (2002) "Information Quality Benchmarks: Product and Service Performance," Communications of the ACM, April 2002. pp. 184–192. Article (http://mitiq.mit.edu/Documents/Publications/TDQMpub/2002/IQ Benchmarks.pdf)

- Price, R. and Shanks, G. (2004) A Semiotic Information Quality Framework, Proc. IFIP International Conference on Decision Support Systems (DSS2004): Decision Support in an Uncertain and Complex World, Prato. Article (http://vishnu.sims.monash.edu.au/dss2004/proceedings/pdf/65_Price_Shanks.pdf)
- Redman, T. C. (2004) Data: An Unfolding Quality Disaster Article (http://www.dmreview.com/article_sub.cfm?articleId=1007211)
- Wand, Y. and Wang, R. (1996) “Anchoring Data Quality Dimensions in Ontological Foundations,” Communications of the ACM, November 1996. pp. 86–95. Article (http://web.mit.edu/tdqm/www/tdqmpub/WandWangCACMNov96.pdf)
- Wang, R., Kon, H. & Madnick, S. (1993), Data Quality Requirements Analysis and Modelling, Ninth International Conference of Data Engineering, Vienna, Austria. Article (http://web.mit.edu/tdqm/www/tdqmpub/IEEEDEApr93.pdf)
- Fournel Michel, Accroitre la qualité et la valeur des données de vos clients, éditions Publibook, 2007. ISBN 978-2-7483-3847-8.
- Daniel F., Casati F., Palpanas T., Chayka O., Cappiello C. (2008) "Enabling Better Decisions through Quality-aware Reports", International Conference on Information Quality (ICIQ), MIT. Article (http://dit.unitn.it/~themis/publications/iciq08.pdf)
- Jack E. Olson (2003), “Data Quality: The Accuracy dimension”, Morgan Kaufmann Publishers

Data Quality Firewall

A **data quality firewall** is the use of software to protect a computer system from the entry of erroneous, duplicated or poor quality data. Gartner estimates that poor quality data causes failure in up to 50% of customer relationship management systems.[*citation needed*] Older technology required the tight integration of data quality software, whereas this can now be accomplished by loosely coupling technology in a service-oriented architecture.

Features and functionality

A data quality firewall guarantees database accuracy and consistency. This application ensures that only valid and high quality data enter the system, which means that it obliquely protects the database from damage; this is extremely important since database integrity and security are absolutely essential. A data quality firewall provides real time feedback information about the quality of the data submitted to the system.

The main goal of a data quality process consists in capturing erroneous and invalid data, processing them and eliminating duplicates and, lastly, exporting valid data to the user without failing to store a back-up copy into the database. A data quality firewall acts similarly to a network security firewall. It enables packets to pass through specified ports by filtering out data that present quality issues and allowing the remaining, valid data to be stored in the database. In other words, the firewall sits between the data source and the database and works throughout the extraction, processing and loading of data.

It is necessary that data streams be subject to accurate validity checks before they can be considered as being correct or trustworthy. Such checks are of a temporal, formal, logic and forecasting kind.

Data truncation

In databases and computer networking **data truncation** occurs when data or a data stream (such as a file) is stored in a location too short to hold its entire length. Data truncation may occur automatically, such as when a long string is written to a smaller buffer, or deliberately, when only a portion of the data is wanted.

Depending on what type of data validation a program or operating system has, the data may be truncated silently (i.e., without informing the user), or the user may be given an error message.

Data validation

In computer science, **data validation** is the process of ensuring that a program operates on clean, correct and useful data. It uses routines, often called "validation rules" or "check routines", that check for correctness, meaningfulness, and security of data that are input to the system. The rules may be implemented through the automated facilities of a data dictionary, or by the inclusion of explicit application program validation logic.

For business applications, data validation can be defined through declarative data integrity rules, or procedure-based business rules.[1] Data that does not conform to these rules will negatively affect business process execution. Therefore, data validation should start with business process definition and set of business rules within this process. Rules can be collected through the requirements capture exercise.[2]

The simplest data validation verifies that the characters provided come from a valid set. For example, telephone numbers should include the digits and possibly the characters +, −, (, and) (plus, minus, and parentheses). A more sophisticated data validation routine would check to see the user had entered a valid country code, i.e., that the number of digits entered matched the convention for the country or area specified.

Incorrect data validation can lead to data corruption or a security vulnerability. Data validation checks that data are valid, sensible, reasonable, and secure before they are processed.

A validation process involves two distinct steps: (a) Validation Check and (b) Post-Check action. The check step uses one or more computational rules (see section below) to determine if the data is valid. The Post-validation action sends feedback to help enforce validation.

Validation methods

Allowed character checks Checks that ascertain that only expected characters are present in a field. For example a numeric field may only allow the digits 0-9, the decimal point and perhaps a minus sign or commas. A text field such as a personal name might disallow characters such as < and >, as they could be evidence of a markup-based security attack. An e-mail address might require at least one @ sign and various other structural details. Regular expressions are effective ways of implementing such checks. (See also data type checks below)

Batch totals Checks for missing records. Numerical fields may be added together for all records in a batch. The batch total is entered and the computer checks that the total is correct, e.g., add the 'Total Cost' field of a number of transactions together.

Cardinality check Checks that record has a valid number of related records. For example if Contact record classified as a Customer it must have at least one associated Order (Cardinality > 0). If order does not exist for a "customer" record then it must be either changed to "seed" or the order must be created. This type of rule can be complicated by additional conditions. For example if contact record in Payroll database is marked as "former employee", then this record must not have any associated salary payments after the date on which employee left organization (Cardinality = 0).

Check digits Used for numerical data. An extra digit is added to a number which is calculated from the digits. The computer checks this calculation when data are entered. For example the last digit of an ISBN for a book is a check digit calculated modulus 10.[3]

Consistency checks Checks fields to ensure data in these fields corresponds, e.g., If Title = "Mr.", then Gender = "M".

Control totals This is a total done on one or more numeric fields which appears in every record. This is a meaningful total, e.g., add the total payment for a number of Customers.

Cross-system consistency checks Compares data in different systems to ensure it is consistent, e.g., The address for the customer with the same id is the same in both systems. The data may be represented differently in different systems and may need to be transformed to a common format to be compared, e.g., one system may store customer name in a single Name field as 'Doe, John Q', while another in three different fields: First_Name (John), Last_Name (Doe) and Middle_Name (Quality); to compare the two, the validation engine would have to transform data from the second system to match the data from the first, for example, using SQL: Last_Name || ', ' || First_Name || substr(Middle_Name, 1, 1) would convert the data from the second system to look like the data from the first 'Doe, John Q'

Data type checks Checks the data type of the input and give an error message if the input data does not match with the chosen data type, e.g., In an input box accepting numeric data, if the letter 'O' was typed instead of the number zero, an error message would appear.

File existence check Checks that a file with a specified name exists. This check is essential for programs that use file handling.

Format or picture check Checks that the data is in a specified format (template), e.g., dates have to be in the format DD/MM/YYYY. Regular expressions should be considered for this type of validation.

Hash totals This is just a batch total done on one or more numeric fields which appears in every record. This is a meaningless total, e.g., add the Telephone Numbers together for a number of Customers.

Limit check Unlike range checks, data are checked for one limit only, upper OR lower, e.g., data should not be greater than 2 (<=2).

Logic check Checks that an input does not yield a logical error, e.g., an input value should not be 0 when it will divide some other number somewhere in a program.

Presence check Checks that important data is actually present and have not been missed out, e.g., customers may be required to have their telephone numbers listed.

Range check Checks that the data lie within a specified range of values, e.g., the month of a person's date of birth should lie between 1 and 12.

Referential integrity In modern Relational database values in two tables can be linked through foreign key and primary key. If values in the primary key field are not constrained by database internal mechanism,[4] then they should be validated. Validation of the foreign key field checks that referencing table must always refer to a valid row in the referenced table.[5]

Spelling and grammar check Looks for spelling and grammatical errors.

Uniqueness check Checks that each value is unique. This can be applied to several fields (i.e. Address, First Name, Last Name).

Table Look Up Check A table look up check takes the entered data item and compares it to a valid list of entries that are stored in a database table.

Post-validation actions

Enforcement Action

Enforcement action typically rejects the data entry request and requires the input actor to make a change that brings the data into compliance. This is most suitable for interactive use, where a real person is sitting on the computer and making entry. It also works well for batch upload, where a file input may be rejected and a set of messages sent back to the input source for why the data is rejected. Another form of enforcement action involves automatically changing the data and saving a conformant version instead of the original version. This is most suitable for cosmetic change. For example, converting an [all-caps] entry to a [Pascal case] entry does not need user input. An inappropriate use of automatic enforcement would be in situations where the enforcement leads to loss of business information. For example, saving a truncated comment if the length is longer than expected. This is not typically a good thing since it may result in loss of significant data.

Advisory Action

Advisory actions typically allow data to be entered unchanged but sends a message to the source actor indicating those validation issues that were encountered. This is most suitable for non-interactive system, for systems where the change is not business critical, for cleansing steps of existing data and for verification steps of an entry process.

Verification Action

Verification actions are special cases of advisory actions. In this case, the source actor is asked to verify that this data is what they would really want to enter, in the light of a suggestion to the contrary. Here, the check step suggests an alternative (e.g.: a check of your mailing address returns a different way of formatting that address or suggests a different address altogether). You would want in this case, to give the user the option of accepting the recommendation or keeping their version. This is not a strict validation process, by design and is useful for capturing addresses to a new location or to a location that is not yet supported by the validation database.

References

[1] Data Validation, Data Integrity, Designing Distributed Applications with Visual Studio .NET (http://msdn.microsoft.com/en-us/library/aa291820(VS.71).aspx)

[2] Arkady Maydanchik (2007), "Data Quality Assessment", Technics Publications, LLC

Data verification

Data **Verification** is a process where in different types of data are checked for accuracy and inconsistencies after data migration is done.[1]

It helps to determine whether data was accurately translated when data is transferred from one source to another, is complete, and supports processes in the new system. During verification, there may be a need for a parallel run of both systems to identify areas of disparity and forestall erroneous data loss.

A type of Data Verification is double entry and proofreading data. Proofreading data involves someone checking the data entered against the original document. This is also time consuming and costly.

References

[1] http://www.datacap.com/products/features/verify/

External links

- PC Guide article (http://www.pcguide.com/care/bu/howVerification-c.html)

Database integrity

Database integrity ensures that data entered into the database is accurate, valid, and consistent. Any applicable integrity constraints and data validation rules must be satisfied before permitting a change to the database.

Three basic types of database integrity constraints are:

- Entity integrity, not allowing multiple rows to have the same identity within a table.
- Domain integrity, restricting data to predefined data types, e.g.: dates.
- Referential integrity, requiring the existence of a related row in another table, e.g. a customer for a given customer ID.

Database preservation

Database preservation usually involves converting the information stored in a database to a form likely to be accessible in the long term as technology changes, without losing the initial characteristics (Context, Content, Structure, Appearance and Behaviour) of the data.

Database preservation projects

In the past different research groups have contributed to the solutions of the problems of database preservation. Research projects carried out in the past in this regard include:

- Software independent archival of relational databases (SIARD)[1]
- Repository of Authentic Digital Objects (RODA)[2]
- Digital Preservation Testbed[3]
- Lots of Copies Keep Stuff Safe (LOCKSS)[4]

References

[1] http://arxiv.org/abs/cs/0408054v1
[2] http://repositorium.sdum.uminho.pt/bitstream/1822/8226/1/RodaAndCrib.pdf
[3] http://www.digitaleduurzaamheid.nl/bibliotheek/docs/volatility-permanence-databases-en.pdf
[4] http://www.lockss.org

Declarative Referential Integrity

Declarative Referential Integrity (DRI) is one of the techniques in the SQL database programming language to ensure data integrity.

Meaning in SQL

A table (called the child table) can refer to a column (or a group of columns) in another table (the parent table) by using a foreign key. The referenced column(s) in the parent table must be under a unique constraint, such as a primary key. Also, self-references are possible (not fully implemented in MS SQL Server though). On inserting a new row into the child table, the relational database management system (RDBMS) checks if the entered key value exists in the parent table. If not, no insert is possible. It is also possible to specify DRI actions on UPDATE and DELETE, such as CASCADE (forwards a change/delete in the parent table to the child tables), NO ACTION (if the specific row is referenced, changing the key is not allowed) or SET NULL / SET DEFAULT (a changed/deleted key in the parent table results in setting the child values to NULL or to the DEFAULT value if one is specified).

Product specific meaning

In Microsoft SQL Server the term DRI also applies to the assigning of permissions to users on a database object. Giving DRI permission to a database user allows them to add foreign key constraints on a table.

References

External links

- DRI versus Triggers (http://www.cvalde.net/document/declaRefIntegVsTrig.htm) (archived (http://web.archive.org/web/20110723065232/http://www.cvalde.net/document/declaRefIntegVsTrig.htm))

Digital continuity

Digital continuity is the ability to maintain the digital information of a creator in such a way that the information will continue to be available, as needed, despite changes in digital storage technology. It focuses on making sure that information is complete, available and therefore usable. Activities involved with managing digital continuity include information management, information risk assessment and managing technical environments, including file format conversion. Digital continuity management is particularly important to organisations that have a duty to maintain accountability, and to act transparently and legally, such as government and infrastructure companies. Digital continuity is also an important issue for organisations responsible for maintaining repositories of information in digital form over time, such as archives and libraries.

Focus of digital continuity

The focus of digital continuity differs from that of digital preservation and business continuity. While there is some overlap among these areas, they should be treated as related but separate issues. Business continuity focuses on making sure that critical business functions will be available to customers, suppliers, regulators, and other entities that must have access to those functions. Digital preservation focuses on long term strategies and requirements for storing digital information in an effort to stabilize the collection of digital records. Digital continuity is concerned with the ability to make digitally preserved information continuously usable. What constitutes usable will be different depending on each organization's needs. For the purposes of digital continuity, The National Archives believes digital information is usable when one can:

- find it when needed
- open it as needed
- work with it in the way needed
- understand what it is and what it is about
- trust that it is what it says it is.

Digital continuity projects

The National Archives in the United Kingdom began a digital continuity project for public use in 2007. The project is based on a four-stage process for managing digital information effectively in organisations:

- Step 1: Plan for action: All employees in the organization need to be aware of the digital continuity project and have an understanding of what digital continuity means for the information they create. This step also includes assessing existing work practices in order to ensure that people working within the system have the potential to deliver continuity.

- Step 2: Define your digital continuity requirements: Ensure that an effective records management system is in place so that it is understood what information needs to be kept and how it will be used over time, regardless of change.
- Step 3: Assess and manage risks to digital continuity: This step involves a digital continuity risk assessment which highlights areas of concern. This may include identifying file formats in current use which are susceptible to obsolescence. Identifying risks will help mitigate potential problems.
- Step 4: Maintain digital continuity: This step ensures that your organization will be able to continue to use the digital information it produces in the future through changes in technology. This involves embedding the concept of digital continuity into existing business processes.

At Archives New Zealand, the digital continuity project is entitled the *Digital Continuity Action Plan*. The Archives New Zealand project is focused on ensuring digital information is available in the future, that the information remains authentic and reliable, and that the public has continuous access through a proactive approach to maintaining digital information.

The Digital Preservation Coalition (DPC), which concerns itself with digital preservation issues including digital continuity, has published reports advocating the assessment of digital preservation needs in the UK, and has been consulted in the creation of the *Digital Continuity Action Plan* at Archives New Zealand. The National Archives is also a member of the DPC.

Other digital continuity projects are underway at the Welsh Assembly Government in conjunction with the University of Wales, Newport., and the National Library of Australia.

References

Digital preservation

In library and archival science, **digital preservation** is a formal endeavor to ensure that digital information of continuing value remains accessible and usable. It involves planning, resource allocation, and application of preservation methods and technologies,[1] and it combines policies, strategies and actions to ensure access to reformatted and "born-digital" content, regardless of the challenges of media failure and technological change. The goal of digital preservation is the accurate rendering of authenticated content over time.[2]

Challenges of digital preservation

Society's heritage has been presented on many different materials, including stone, vellum, bamboo, silk, and paper. Now a large quantity of information exists in digital forms, including emails, blogs, social networking websites, national elections websites, web photo albums, and sites which change their content over time. With digital media it is easier to create content and keep it up-to-date, but at the same time there are many challenges in the preservation of this content, both technical and economic.

Unlike traditional analog objects such as books or photographs where the user has unmediated access to the content, a digital object always needs a software environment to render it. These environments keep evolving and changing at a rapid pace, threatening the continuity of access to the content. Physical storage media, data formats, hardware, and software all become obsolete over time, posing significant threats to the survival of the content. This process can be referred to as digital obsolescence.

In the case of born-digital content (e.g., institutional archives, Web sites, electronic audio and video content, born-digital photography and art, research data sets, observational data), the enormous and growing quantity of content presents significant scaling issues to digital preservation efforts. Rapidly changing technologies can hinder

digital preservationists work and techniques due to outdated and antiquated machines or technology. This has become a common problem and one that is a constant worry for a digital archivist—how to prepare for the future.

Digital content can also present challenges to preservation because of its complex and dynamic nature, e.g., interactive Web pages, virtual reality and gaming environments, learning objects, social media sites. In many cases of emergent technological advances there are substantial difficulties in maintaining the authenticity, fixity, and integrity of objects over time deriving from the fundamental issue of experience with that particular digital storage medium and while particular technologies may prove to be more robust in terms of storage capacity, there are issues in securing a framework of measures to ensure that the object remains fixed while in stewardship.

For the preservation of software as digital content, a specific challenge is the typically non-availability of the source code as commercial software is normally distributed only in compiled binary form. Without the source code an adaption (Porting) on modern computing hardware or operating system is most often impossible, therefore the original hardware and software context needs to be emulated. Another potential challenge for software preservation can be the copyright who prohibits often the bypassing of copy protection mechanisms (Digital Millennium Copyright Act) incase software has become a orphaned work (Abandonware). An exemption from the United States Digital Millennium Copyright Act to permit to bypass copy protection was approved in 2003 for a period of 3 years to the Internet Archive who created an archive of "vintage software", as a way to preserve them. The exemption was renewed in 2006, and as of 27 October 2009[3], has been indefinitely extended pending further rulemakings "for the purpose of preservation or archival reproduction of published digital works by a library or archive."

Another challenge surrounding preservation of digital content resides in the issue of scale. The amount of digital information being created along with the "proliferation of format types" makes creating trusted digital repositories with adequate and sustainable resources a challenge. The Web is only one example of what might be considered the "data deluge". For example, the Library of Congress currently amassed 170 billion tweets between 2006 and 2010 totaling 133.2 terabytes[4] and each Tweet is composed of 23 fields of metadata.[5]

The economic challenges of digital preservation are also great. Preservation programs require significant up front investment to create, along with ongoing costs for data ingest, data management, data storage, and staffing. One of the key strategic challenges to such programs is the fact that, while they require significant current and ongoing funding, their benefits accrue largely to future generations.

Intellectual foundations of digital preservation

"*Preserving Digital Information* (1996)"

The challenges of long-term preservation of digital information have been recognized by the archival community for years. In December 1994, the Research Libraries Group (RLG) and Commission on Preservation and Access (CPA) formed a Task Force on Archiving of Digital Information with the main purpose of investigating what needed to be done to ensure long-term preservation and continued access to the digital records. The final report published by the Task Force (Garrett, J. and Waters, D., ed. (1996). “Preserving digital information: Report of the task force on archiving of digital information.”) became a fundamental document in the field of digital preservation that helped set out key concepts, requirements, and challenges.

The Task Force proposed development of a national system of digital archives that would take responsibility for long-term storage and access to digital information; introduced the concept of trusted digital repositories and defined their roles and responsibilities; identified five features of digital information integrity (content, fixity, reference, provenance, and context) that were subsequently incorporated into a definition of Preservation Description Information in the Open Archival Information System Reference Model; and defined migration as a crucial function of digital archives. The concepts and recommendations outlined in the report laid a foundation for subsequent research and digital preservation initiatives.

OAIS

To standardize digital preservation practice and provide a set of recommendations for preservation program implementation, the Reference Model for an Open Archival Information System (OAIS) was developed. OAIS is concerned with all technical aspects of a digital object's life cycle: ingest, archival storage, data management, administration, access and preservation planning. The model also addresses metadata issues and recommends that five types of metadata be attached to a digital object: reference (identification) information, provenance (including preservation history), context, fixity (authenticity indicators), and representation (formatting, file structure, and what "imparts meaning to an object's bitstream").[6]

Trusted Digital Repository Model

In March 2000, the Research Libraries Group (RLG) and Online Computer Library Center (OCLC) began a collaboration to establish attributes of a digital repository for research organizations, building on and incorporating the emerging international standard of the Reference Model for an Open Archival Information System (OAIS). In 2002, they published "Trusted Digital Repositories: Attributes and Responsibilities." In that document a "Trusted Digital Repository" (TDR) is defined as "one whose mission is to provide reliable, long-term access to managed digital resources to its designated community, now and in the future." The TDR must include the following seven attributes: compliance with the reference model for an Open Archival Information System (OAIS), administrative responsibility, organizational viability, financial sustainability, technological and procedural suitability, system security, procedural accountability. The Trusted Digital Repository Model outlines relationships among these attributes. The report also recommended the collaborative development of digital repository certifications, models for cooperative networks, and sharing of research and information on digital preservation with regard to intellectual property rights.[7]

In 2004 Henry M. Gladney proposed another approach to digital object preservation that called for the creation of "Trustworthy Digital Objects" (TDOs). TDOs are digital objects that can speak to their own authenticity since they incorporate a record maintaining their use and change history, which allows the future users to verify that the contents of the object are valid.

InterPARES

International Research on Permanent Authentic Records in Electronic Systems (InterPARES) is a collaborative research initiative led by the University of British Columbia that is focused on addressing issues of long-term preservation of authentic digital records. The research is being conducted by focus groups from various institutions in North America, Europe, Asia, and Australia, with an objective of developing theories and methodologies that provide the basis for strategies, standards, policies, and procedures necessary to ensure the trustworthiness, reliability, and accuracy of digital records over time.

The project began in 1999 with the first phase, InterPARES 1, which ran to 2001 and focused on establishing requirements for authenticity of inactive records generated and maintained in large databases and document management systems created by government agencies. InterPARES 2 (2002 – 2007) concentrated on issues of reliability, accuracy and authenticity of records throughout their whole life cycle, and examined records produced in dynamic environments in the course of artistic, scientific and online government activities. The third five-year phase (InterPARES 3) was initiated in 2007. Its goal is to utilize theoretical and methodological knowledge generated by InterPARES and other preservation research projects for developing guidelines, action plans, and training programs on long-term preservation of authentic records for small and medium-sized archival organizations.

Strategies

In 2006, the Online Computer Library Center developed a four-point strategy for the long-term preservation of digital objects that consisted of:

- Assessing the risks for loss of content posed by technology variables such as commonly used proprietary file formats and software applications.
- Evaluating the digital content objects to determine what type and degree of format conversion or other preservation actions should be applied.
- Determining the appropriate metadata needed for each object type and how it is associated with the objects.
- Providing access to the content.[8]

There are several additional strategies that individuals and organizations may use to actively combat the loss of digital information.

Refreshing

Refreshing is the transfer of data between two types of the same storage medium so there are no bitrot changes or alteration of data. For example, transferring census data from an old preservation CD to a new one. This strategy may need to be combined with migration when the software or hardware required to read the data is no longer available or is unable to understand the format of the data. Refreshing will likely always be necessary due to the deterioration of physical media.

Migration

Migration is the transferring of data to newer system environments (Garrett et al., 1996). This may include conversion of resources from one file format to another (e.g., conversion of Microsoft Word to PDF or OpenDocument) or from one operating system to another (e.g., Windows to GNU/Linux) so the resource remains fully accessible and functional. Two significant problems face migration as a plausible method of digital preservation in the long terms. Due to the fact that digital objects are subject to a state of near continuous change, migration may cause problems in relation to authenticity and migration has proven to be time-consuming and expensive for "large collections of heterogeneous objects, which would need constant monitoring and intervention.

Replication

Creating duplicate copies of data on one or more systems is called *replication*. Data that exists as a single copy in only one location is highly vulnerable to software or hardware failure, intentional or accidental alteration, and environmental catastrophes like fire, flooding, etc. Digital data is more likely to survive if it is replicated in several locations. Replicated data may introduce difficulties in refreshing, migration, versioning, and access control since the data is located in multiple places.

Emulation

Emulation is the replicating of functionality of an obsolete system. According to van der Hoeven, "Emulation does not focus on the digital object, but on the hard- and software environment in which the object is rendered. It aims at (re)creating the environment in which the digital object was originally created.".[9] Examples are having the ability to replicate or imitate another operating system. Examples include emulating an Atari 2600 on a Windows system or emulating WordPerfect 1.0 on a Macintosh. Emulators may be built for applications, operating systems, or hardware platforms. Emulation has been a popular strategy for retaining the functionality of old video game systems, such as with the MAME project. The feasibility of emulation as a catch-all solution has been debated in the academic community. (Granger, 2000)

Raymond A. Lorie has suggested a Universal Virtual Computer (UVC) could be used to run any software in the future on a yet unknown platform. The UVC strategy uses a combination of emulation and migration. The UVC strategy has not yet been widely adopted by the digital preservation community.

Jeff Rothenberg, a major proponent of Emulation for digital preservation in libraries, working in partnership with Koninklijke Bibliotheek and National Archief of the Netherlands, developed a software program called Dioscuri, a modular emulator that succeeds in running MS-DOS, WordPerfect 5.1, DOS games, and more.

Another example of emulation as a form of digital preservation can be seen in the example of Emory University and the Salman Rushdie's papers. Rushdie donated an outdated computer to the Emory University library, which was so old that the library was unable to extract papers from the harddrive. In order to procure the papers, the library emulated the old software system and was able to take the papers off his old computer.[10]

Encapsulation

This method maintains that preserved objects should be self-describing, virtually "linking content with all of the information required for it to be deciphered and understood". The files associated with the digital object would have details of how to interpret that object by using "logical structures called "containers" or "wrappers" to provide a relationship between all information components[11] that could be used in future development of emulators, viewers or converters through machine readable specifications.[12] The method of encapsulation is usually applied to collections that will go unused for long periods of time

Persistent Archives Concept

Developed by the San Diego Supercomputing Center and funded by the National Archives and Records Administration, this method requires the development of comprehensive and extensive infrastructure that enables "the preservation of the organisation of collection as well as the objects that make up that collection, maintained in a platform independent form". A persistent archive includes both the data constituting the digital object and the context that the defines the provenance, authenticity, and structure of the digital entities.[13] This allows for the replacement of hardware or software components with minimal effect on the preservation system. This method can be based on virtual data grids and resembles OAIS Information Model (specifically the Archival Information Package).

Metadata attachment

Metadata is data on a digital file that includes information on creation, access rights, restrictions, preservation history, and rights management.[14] Metadata attached to digital files may be affected by file format obsolescence. ASCII is considered to be the most durable format for metadata [15] because it is widespread, backwards compatible when used with Unicode, and utilizes human-readable characters, not numeric codes. It retains information, but not the structure information it is presented in. For higher functionality, SGML or XML should be used. Both markup languages are stored in ASCII format, but contain tags that denote structure and format.

Digital sustainability

Digital sustainability encompasses a range of issues and concerns that contribute to the longevity of digital information.[16] Unlike traditional, temporary strategies, and more permanent solutions, digital sustainability implies a more active and continuous process. Digital sustainability concentrates less on the solution and technology and more on building an infrastructure and approach that is flexible with an emphasis on interoperability, continued maintenance and continuous development.[17] Digital sustainability incorporates activities in the present that will facilitate access and availability in the future.[18][19]

Preservation repository assessment and certification

A few of the major frameworks for digital preservation repository assessment and certification are described below. A more detailed list is maintained by the U.S. Center for Research Libraries.

Specific tools and methodologies

TRAC

In 2007, CRL/OCLC published Trustworthy Repositories Audit & Certification: Criteria & Checklist (TRAC), a document allowing digital repositories to assess their capability to reliably store, migrate, and provide access to digital content. TRAC is based upon existing standards and best practices for trustworthy digital repositories and incorporates a set of 84 audit and certification criteria arranged in three sections: Organizational Infrastructure; Digital Object Management; and Technologies, Technical Infrastructure, and Security.

TRAC "provides tools for the audit, assessment, and potential certification of digital repositories, establishes the documentation requirements required for audit, delineates a process for certification, and establishes appropriate methodologies for determining the soundness and sustainability of digital repositories".

DRAMBORA

Digital Repository Audit Method Based On Risk Assessment (DRAMBORA), introduced by the Digital Curation Centre (DCC) and Digital Preservation Europe (DPE) in 2007, offers a methodology and a toolkit for digital repository self-assessment.

The DRAMBORA process is arranged in six stages and concentrates on evaluation of likelihood and potential impact of risks on the repository. The auditor is required to describe and document the repository's role, objectives, policies, activities and assets, in order to identify and assess the risks associated with these activities and assets and define appropriate measures to manage them.

European Framework for Audit and Certification of Digital Repositories

The European Framework for Audit and Certification of Digital Repositories [20] was defined in a memorandum of understanding signed in July 2010 between Consultative Committee for Space Data Systems (CCSDS), Data Seal of Approval (DSA) Board and German Institute for Standardization (DIN) "Trustworthy Archives – Certification" Working Group.

The framework is intended to help organizations in obtaining appropriate certification as a trusted digital repository and establishes three increasingly demanding levels of assessment:

1. Basic Certification: self-assessment using 16 criteria of the Data Seal of Approval (DSA).
2. Extended Certification: Basic Certification and additional externally reviewed self-audit against ISO 16363 or DIN 31644 requirements.
3. Formal Certification: validation of the self-certification with a third-party official audit based on ISO 16363 or DIN 31644.

nestor Catalogue of Criteria

A German initiative, *nestor* [21] (the Network of Expertise in Long-Term Storage of Digital Resources) sponsored by the German Ministry of Education and Research, developed a catalogue of criteria for trusted digital repositories in 2004. In 2008 the second version of the document was published. The catalogue, aiming primarily at German cultural heritage and higher education institutions, establishes guidelines for planning, implementing, and self-evaluation of trustworthy long-term digital repositories.

The *nestor* catalogue of criteria conforms to the OAIS reference model terminology and consists of three sections covering topics related to Organizational Framework, Object Management, and Infrastructure and Security.

PLANETS Project

In 2002 the *Preservation and Long-term Access through Networked Services* (PLANETS) project, part of the EU Framework Programmes for Research and Technological Development 6, addressed core digital preservation challenges. The primary goal for *Planets* was to build practical services and tools to help ensure long-term access to digital cultural and scientific assets. The outputs of the project are now sustained by the follow-on organisation, the Open Planets Foundation.

PLATTER

Planning Tool for Trusted Electronic Repositories (PLATTER) is a tool released by DigitalPreservationEurope (DPE) to help digital repositories in identifying their self-defined goals and priorities in order to gain trust from the stakeholders.

PLATTER is intended to be used as a complementary tool to DRAMBORA, NESTOR, and TRAC. It is based on ten core principles for trusted repositories and defines nine Strategic Objective Plans, covering such areas as acquisition, preservation and dissemination of content, finance, staffing, succession planning, technical infrastructure, data and metadata specifications, and disaster planning. The tool enables repositories to develop and maintain documentation required for an audit.[:49]

Audit and Certification of Trustworthy Digital Repositories (ISO 16363)

Audit and Certification of Trustworthy Digital Repositories (ISO 16363:2012), developed by the Consultative Committee for Space Data Systems (CCSDS), was approved as a full international standard in March 2012. Extending the OAIS Reference Model and based largely on the TRAC checklist, the standard is designed for all types of digital repositories. It provides a detailed specification of criteria against which the trustworthiness of a digital repository should be evaluated.

The CCSDS Repository Audit and Certification Working Group has also developed and submitted for approval a second standard, Requirements for Bodies Providing Audit and Certification of Candidate Trustworthy Digital Repositories (ISO 16919), that defines the external auditing process and requirements for organizations responsible for assessment and certification of digital repositories.

Digital preservation best practices

Although preservation strategies vary for different types of materials and between institutions, adhering to nationally and internationally recognized standards and practices is a crucial part of digital preservation activities. Best or recommended practices define strategies and procedures that may help organizations to implement existing standards or provide guidance in areas where no formal standards have been developed.

Best practices in digital preservation continue to evolve and may encompass processes that are performed on content prior to or at the point of ingest into a digital repository as well as processes performed on preserved files post-ingest over time. Best practices may also apply to the process of digitizing analog material and may include the creation of specialized metadata (such as technical, administrative and rights metadata) in addition to standard descriptive metadata. The preservation of born-digital content may include format transformations to facilitate long-term preservation or to provide better access.

Audio preservation

Various best practices and guidelines for digital audio preservation have been developed, including:

- *Capturing Analog Sound for Digital Preservation: Report of a Roundtable Discussion of Best Practices for Transferring Analog Discs and Tapes* (2006), which defined procedures for reformatting sound from analog to digital and provided recommendations for best practices for digital preservation
- *Digital Audio Best Practices* (2006) prepared by the Collaborative Digitization Program Digital Audio Working Group, which covers best practices and provides guidance both on digitizing existing analog content and on creating new digital audio resources
- *Sound Directions: Best Practices for Audio Preservation* (2007) published by the Sound Directions Project, which describes the audio preservation workflows and recommended best practices and has been used as the basis for other projects and initiatives
- Documents developed by the International Association of Sound and Audiovisual Archives (IASA), the European Broadcasting Union (EBU), the Library of Congress, and the Digital Library Federation (DLF).

The Audio Engineering Society (AES) also issues a variety of standards and guidelines relating to the creation of archival audio content and technical metadata.

Moving Image Preservation

The term “moving images” includes analog film and video and their born-digital forms: digital video, digital motion picture materials, and digital cinema. As analog videotape and film become obsolete, digitization has become a key preservation strategy, although many archives do continue to perform photochemical preservation of film stock.

"Digital preservation" has a double meaning for audiovisual collections: analog originals are preserved through digital reformatting, with the resulting digital files preserved; and born-digital content is collected, most often in proprietary formats that pose problems for future digital preservation.

There is currently no broadly accepted standard target digital preservation format for analog moving images.

The following resources offer information on analog to digital reformatting and preserving born-digital audiovisual content.

- The Library of Congress tracks the sustainability of digital formats, including moving images.[22]
- *The Digital Dilemma 2: Perspectives from Independent Filmmakers, Documentarians and Nonprofit Audiovisual Archives (2012)*. The section on nonprofit archives reviews common practices on digital reformatting, metadata, and storage. There are four case studies.
- Federal Agencies Digitization Guidelines Initiative (FADGI). Started in 2007, this is a collaborative effort by federal agencies to define common guidelines, methods, and practices for digitizing historical content. As part of this, two working groups are studying issues specific to two major areas, Still Image and Audio Visual.
- PrestoCenter publishes general audiovisual information and advice at a European level. Its online library has research and white papers on digital preservation costs and formats.[23]
- The Association of Moving Image Archivists (AMIA) sponsors conferences, symposia, and events on all aspects of moving image preservation, including digital. The *AMIA Tech Review* [24] contains articles reflecting current thoughts and practices from the archivists’ perspectives. *Video Preservation for the Millennia (2012)*, published in the *AMIA Tech Review*, details the various strategies and ideas behind the current state of video preservation.

Email preservation

Email poses special challenges for preservation: email client software varies widely; there is no common structure for email messages; email often communicates sensitive information; individual email accounts may contain business and personal messages intermingled; and email may include attached documents in a variety of file formats. Email messages can also carry viruses or have spam content. While email transmission is standardized, there is no formal standard for the long-term preservation of email messages.

Approaches to preserving email may vary according to the purpose for which it is being preserved. For businesses and government entities, email preservation may be driven by the need to meet retention and supervision requirements for regulatory compliance and to allow for legal discovery. (Additional information about email archiving approaches for business and institutional purposes may be found under the separate article, Email archiving.) For research libraries and archives, the preservation of email that is part of born-digital or hybrid archival collections has as its goal ensuring its long-term availability as part of the historical and cultural record.

Several projects developing tools and methodologies for email preservation have been conducted based on various preservation strategies: normalizing email into XML format, migrating email to a new version of the software and emulating email environments: Memories Using Email [25] (MUSE), Collaborative Electronic Records Project [26] (CERP), E-Mail Collection And Preservation [27] (EMCAP), PeDALS Email Extractor Software [28] (PeDALS), XML Electronic Normalizing of Archives tool [29] (XENA).

Some best practices and guidelines for email preservation can be found in the following resources:

- *Curating E-Mails: A Life-cycle Approach to the Management and Preservation of E-mail Messages* (2006) by Maureen Pennock.
- *Technology Watch Report 11-01: Preserving Email* (2011) by Christopher J Prom.
- *Best Practices: Email Archiving* by Jo Maitland.

Video game preservation

In 2007 the *Keeping Emulation Environments Portable* (KEEP) project, part of the EU Framework Programmes for Research and Technological Development 7, developed tools and methodologies to keep digital software objects available in their original context. Digital software objects as video games might get lost because of digital obsolescence and non-availability of required legacy hardware or operating system software; such software is referred to as abandonware. Because the source code is often not available any longer, emulation is the only preservation opportunity. KEEP provided an emulation framework to help the creation of such emulators. KEEP was developed by Vincent Joguin, first launched in February 2009 and was coordinated by Elisabeth Freyre of the French National Library.

Personal Archiving

There are many things consumers and artists can do themselves to help care for their collections at home.

- "Resource Center: Caring For Your Treasures" by American Institute for Conservation of Historic and Artistic Works details simple strategies for artists and consumers to care for and preserve their work themselves.

The Library of Congress also hosts a list for the self-preserver which includes direction toward programs and guidelines from other institutions that will help the user preserve social media, email, and formatting general guidelines (such as caring for CDs).[30] Some of the programs listed include:

- HTTrack Website Copier [31]: A site which allows the user to download a World Wide Web site from the Internet to a local directory, building recursively all directories, getting HTML, images, and other files from the server to their computer.
- Muse [25]: Muse (short for Memories Using Email) is a program that helps users revive memories, using their long-term email archives, run by Stanford University.

Education for digital preservation

Digital Preservation Outreach and Education (DPOE)

The Digital Preservation Outreach and Education (DPOE), as part of the Library of Congress, serves to foster preservation of digital content through a collaborative network of instructors and collection management professionals working in cultural heritage institutions. Composed of Library of Congress staff, the National Trainer Network, the DPOE Steering Committee, and a community of Digital Preservation Education Advocates, as of 2013 the DPOE has 24 working trainers across the six regions of the United States.

In 2010 the DPOE conducted an assessment, reaching out to archivists, librarians, and other information professionals around the country. A working group of DPOE instructors then developed a curriculum [32] based on the assessment results and other similar digital preservation curricula designed by other training programs, such as LYRASIS, Educopia Institute, MetaArchive Cooperative, University of North Carolina, DigCCurr (Digital Curation Curriculum) and Cornell University-ICPSR Digital Preservation Management Workshops. The resulting core principles are also modeled on the principles outlined in "A Framework of Guidance for Building Good Digital Collections" by the National Information Standards Organization (NISO).[33]

Examples of digital preservation initiatives

- The Library of Congress operates the National Digital Stewardship Alliance [34]
- The British Library is responsible for several programmes in the area of **digital preservation** and is a founding member of the Digital Preservation Coalition [35]. Their digital preservation strategy [36] is publicly available. The National Archives of the United Kingdom have also pioneered various initiatives in the field of **digital preservation**.

Digitization at the British Library of a Dunhuang manuscript for the International Dunhuang Project

A number of open source products have been developed to assist with digital preservation, including DSpace, Fedora, EPrints and Research-Output Repository Platform. The commercial sector also offers digital preservation software tools, such as Ex Libris Ltd.'s *Rosetta*, Tessella Ltd.'s *Safety Deposit Box* and cloud based *Preservica*, CONTENTdm, Digital Commons, Equella, intraLibrary, Open Repository and Vital.

Large-scale digital preservation initiatives (LSDIs)

Many research libraries and archives have begun or are about to begin Large-Scale digital preservation initiatives (LSDIs). The main players in LSDIs are cultural institutions, commercial companies such as Google and Microsoft, and non-profit groups including the Open Content Alliance (OCA), the Million Book Project (MBP), and HathiTrust. The primary motivation of these groups is to expand access to scholarly resources.

LSDIs: library perspective

Approximately 30 cultural entities, including the 12-member Committee on Institutional Cooperation (CIC), have signed digitization agreements with either Google or Microsoft. Several of these cultural entities are participating in the Open Content Alliance (OCA) and the Million Book Project (MBP). Some libraries are involved in only one initiative and others have diversified their digitization strategies through participation in multiple initiatives. The three main reasons for library participation in LSDIs are: Access, Preservation and Research and Development. It is hoped that digital preservation will ensure that library materials remain accessible for future generations. Libraries have a perpetual responsibility for their materials and a commitment to archive their digital materials. Libraries plan

to use digitized copies as backups for works in case they go out of print, deteriorate, or are lost and damaged.

Footnotes

[1] Day, Michael. "The long-term preservation of Web content". Web archiving (Berlin: Springer, 2006), pp. 177-199. ISBN 3-540-23338-5.

[2] Evans, Mark; Carter, Laura. (December 2008). The Challenges of Digital Preservation. Presentation at the Library of Parliament, Ottawa.

[3] http://en.wikipedia.org/w/index.php?title=Digital_preservation&action=edit

[4] http://www.loc.gov:8081/today/pr/2013/files/twitter_report_2013jan.pdf

[5] http://articles.forensicfocus.com/2012/04/25/key-twitter-and-facebook-metadata-fields-forensic-investigators-need-to-be-aware-of/

[6] Cornell University Library. (2005) Digital Preservation Management: Implementing Short-term Strategies for Long-term Problems (http://dpworkshop.org/dpm-eng/eng_index.html/)

[7] Research Libraries Group. (2002). Trusted Digital Repositories: Attributes and Responsibilities (http://www.oclc.org/programs/ourwork/past/trustedrep/repositories.pdf)

[8] Online Computer Library Center, Inc. (2006). OCLC Digital Archive Preservation Policy and Supporting Documentation (http://www.oclc.org/support/documentation/digitalarchive/preservationpolicy.pdf), p. 5

[9] http://www.ijdc.net/index.php/ijdc/article/view/50/35

[10] http://marbl.library.emory.edu/innovations/salman-rushdie

[11] Digital Preservation: Planning, Process and Approaches for Libraries Teena KapoorJaypee Institute of Information TechnologyA-10, Sector-62, Noida UP

[12] SOLUTIONS WALKTHROUGH REPORT José Miguel Araújo Ferreira Department of Information Systems University of Minho 4800-058 Guimarães, Portugal

[13] Moore, Reagan W., Andre Merzky. Persistent Archive Research Group. Dec. 25, 2003.

[14] NISO Framework Advisory Group. (2007). A Framework of Guidance for Building Good Digital Collections, 3rd edition (http://www.niso.org/publications/rp/framework3.pdf), p. 57,

[15] National Initiative for a Networked Cultural Heritage. (2002). NINCH Guide to Good Practice in the Digital Representation and Management of Cultural Heritage Materials (http://www.nyu.edu/its/humanities/ninchguide/)

[16] Bradley, K. (Summer 2007). Defining digital sustainability. Library Trends v. 56 no 1 p. 148-163.

[17] Sustainability of Digital Resources. (2008). *TASI: Technical Advisory Service for Images.* (http://www.tasi.ac.uk/advice/managing/sust.html)

[18] Towards a Theory of Digital Preservation. (2008). *International Journal of Digital Curation* (http://ijdc.net/index.php/ijdc/article/view/63)

[19] *Electronic Archives Preservation Policy* (http://wiki.lib.sun.ac.za/index.php/SUNScholar/Digital_Preservation/Electronic_Archives_Preservation_Policy)

[20] http://www.trusteddigitalrepository.eu/Site/Trusted%20Digital%20Repository.html

[21] http://www.langzeitarchivierung.de/Subsites/nestor/DE/Home/home_node.html

[22] http://www.digitalpreservation.gov/formats/content/video.shtml

[23] https://www.prestocentre.org/library

[24] http://www.amiatechreview.com/

[25] http://mobisocial.stanford.edu/muse/

[26] http://siarchives.si.edu/cerp/

[27] http://www.records.ncdcr.gov/EmailPreservation/default.htm

[28] http://sourceforge.net/projects/pedalsemailextr/

[29] http://xena.sourceforge.net/

[30] http://www.digitalpreservation.gov/personalarchiving/padKit/resources.html

[31] http://www.httrack.com/

[32] DPOE Curriculum. 2013 (http://www.digitalpreservation.gov/education/curriculum.html)

[33] DPOE Background. 2013 (http://www.digitalpreservation.gov/education/background.html)

[34] http://www.digitalpreservation.gov/ndsa

[35] http://www.dpconline.org/

[36] http://www.bl.uk/aboutus/stratpolprog/collectioncare/discovermore/digitalpreservation/strategy/dpstrategy.html

References

- Garrett, J., D. Waters, H. Gladney, P. Andre, H. Besser, N. Elkington, H. Gladney, M. Hedstrom, P. Hirtle, K. Hunter, R. Kelly, D. Kresh, M. Lesk, M. Levering, W. Lougee, C. Lynch, C. Mandel, S. Mooney, A. Okerson, J. Neal, S. Rosenblatt, and S. Weibe (1996). "Preserving digital information: Report of the task force on archiving of digital information" (http://www.rlg.org/legacy/ftpd/pub/archtf/final-report.pdf) (PDF). *Commission on Preservation and Access and the Research Libraries Group*. Retrieved 2009-06-23.
- Gladney, H. M.; Lorie, R. A. (2005). "Trustworthy 100-year digital objects: durable encoding for when it's too late to ask" (http://portal.acm.org/citation.cfm?id=1080343.1080346). *ACM Transactions on Information Systems* **23** (3): 299–324. doi: 10.1145/1080343.1080346 (http://dx.doi.org/10.1145/1080343.1080346).
- Gladney, H. M. (2006). "Principles for digital preservation" (http://portal.acm.org/citation.cfm?id=1113034.1113038&coll=GUIDE&dl=ACM&CFID=22202827&CFTOKEN=41873705). *Communications of the ACM* **49** (2): 111–116. doi: 10.1145/1113034.1113038 (http://dx.doi.org/10.1145/1113034.1113038).
- Granger, Stewart (2000). "Emulation as a Digital Preservation Strategy" (http://www.dlib.org/dlib/october00/granger/10granger.html). *D-Lib Magazine* **6** (10). doi: 10.1045/october2000-granger (http://dx.doi.org/10.1045/october2000-granger).
- Edwards, Eli (2004). "Ephemeral to Enduring: The Internet Archive and Its Role in Preserving Digital Media". *Information Technology & Libraries* **23** (1).
- Hedstrom, M., Ross, S., Ashley, K., Christensen-Dalsgaard, B., Duff, W., Gladney, H., Huc, C., Kenney, A.R., Moore, R., Neuhold, E. (2003). "Invest to Save: Report and Recommendations of the NSF-DELOS Working Group on Digital Archiving and Preservation" (http://delos-noe.iei.pi.cnr.it/activities/internationalforum/Joint-WGs/digitalarchiving/Digitalarchiving.pdf). *Nsf/Delos* (Pisa & Washington DC, USA).
- Jantz, R. & Giarlo, M.J. (2005). "Digital preservation: Architecture and technology for trusted digital repositories". *D-Lib Magazine* **11** (6). doi: 10.1045/june2005-jantz (http://dx.doi.org/10.1045/june2005-jantz).
- Ross, S (2000). *Changing Trains at Wigan: Digital Preservation and the Future of Scholarship* (http://www.bl.uk/blpac/pdf/wigan.pdf). London, UK: National Preservation Office (British Library). ISBN 0-7123-4717-8.
- Ross, S. and Gow, A. (1999). *Digital archaeology? Rescuing Neglected or Damaged Data Resources* (http://www.ukoln.ac.uk/services/elib/papers/supporting/pdf/p2.pdf). Bristol & London: British Library and Joint Information Systems Committee. ISBN 1-900508-51-6.
- Rossi, Christian (2009). "From distribution to preservation of digital documents" (http://hal.inria.fr/hal-00809433/). *TUGboat* **30** (2): 274–280.
- Rothenberg, Jeff (1995). "Ensuring the Longevity of Digital Documents". *Scientific American* **272** (1): 42. doi: 10.1038/scientificamerican0195-42 (http://dx.doi.org/10.1038/scientificamerican0195-42).
- Rothenberg, Jeff (1999). *Ensuring the Longevity of Digital Information* (http://www.clir.org/pubs/archives/ensuring.pdf). Expanded version of *Ensuring the Longevity of Digital Documents*.
- Milne, Ronald -- moderator: Webcast panel discussion, "Economics," (http://www.lib.umich.edu/mdp/symposium/economics.html) *Scholarship and Libraries in Transition: A Dialogue about the Impacts of Mass Digitization Projects* (2006), Symposium sponsored by the University of Michigan Library and the National Commission on Libraries and Information Science (US).
- Dobratz, S. et al. (2009). "Catalogue of Criteria for Trusted Digital Repositories" (http://files.d-nb.de/nestor/materialien/nestor_mat_08_eng.pdf). nestor materials, Deutsche Nationalbibliothek, Frankfurt (Main), Germany. Retrieved October 2, 2012.

External links

- National Digital Information Infrastructure and Preservation Program (http://www.digitalpreservation.gov) at the Library of Congress
- DPOE - Digital Preservation Outreach & Education (http://www.digitalpreservation.gov/education/) at Library of Congress
- Digital Preservation page (http://www.diglib.org/preserve.htm) from the Digital Library Federation
- "Thirteen Ways of Looking at...Digital Preservation" (http://dlib.org/dlib/july04/lavoie/07lavoie.html)
- Cornell University Library's *Digital Imaging Tutorial* (http://www.library.cornell.edu/preservation/tutorial/contents.html)
- What is Digital Preservation? (http://www.digitalpreservationeurope.eu/what-is-digital-preservation/) - an introduction to digital preservation by Digital Preservation Europe
- Macroscopic 10-Terabit–per–Square-Inch Arrays from Block Copolymers with Lateral Order. (http://www.sciencemag.org/cgi/content/abstract/323/5917/1030) Science magazine article about prospective usage of sapphire in digital storage media technology
- Animations introducing digital preservation and curation (http://www.youtube.com/watch?v=pbBa6Oam7-w)
- Capture Your Collections: Planning and Implementing Digitization Projects (http://www.pro.rcip-chin.gc.ca/sommaire-summary/planification_numerisation-digitization_planning-eng.jsp) A CHIN (Canadian Heritage Information Network) Resource
- Digitales Archiv Hessen (http://www.hauptstaatsarchiv.hessen.de/irj/HHStAW_Internet?cid=69e5c1d5b0eb5a25b5961e83962b068c) Digital preservation page by Hessisches Hauptstaatsarchiv Wiesbaden
- “Land of the lost” : a discussion of what can be preserved through digital preservation." (http://www.nla.gov.au/openpublish/index.php/nlasp/article/viewArticle/1677) Nick del Pozo, Andrew Stawowczyk Long, David Pearson.
- Various activities in digital preservation at the University of Cologne (professorship for Applied Computer Science in the Humanities) (http://www.hki.uni-koeln.de/preservation)
- Challenges in AV Digitization and Digital Preservation (http://www.nationalvideo.com.au/challenges-in-audio-visual-digitization-and-digital-preservation)

User:TheAmazing0and1/draftdigipres

Editing this page here to potentially reorganize it.

Digital preservation is the set of processes, activities and management of digital information over time to ensure its long term accessibility. The goal of digital preservation is to preserve materials resulting from digital reformatting, and particularly information that is born-digital with no analog counterpart. Because of the relatively short lifecycle of digital information, preservation is an ongoing process.

In the language of digital imaging and electronic resources, preservation is no longer just the product of a program but an ongoing process. In this regard the way digital information is stored is important in ensuring its longevity. The long-term storage of digital information is assisted by the inclusion of preservation metadata.

Digital preservation is defined as: long-term, error-free storage of digital information, with means for retrieval and interpretation, for the entire time span the information is required for. Long-term is defined as "long enough to be concerned with the impacts of changing technologies, including support for new media and data formats, or with a changing user community. Long Term may extend indefinitely".[1] "Retrieval" means obtaining needed digital files from the long-term, error-free digital storage, without possibility of corrupting the continued error-free storage of the digital files. "Interpretation" means that the retrieved digital files, files that, for example, are of texts, charts, images or sounds, are decoded and transformed into usable representations. This is often interpreted as "rendering", i.e. making it available for a human to access. However, in many cases it will mean able to be processed by computational means.

Digital Format Preservation Concerns

Society's heritage has been presented on many different materials, including stone, vellum, bamboo, silk, and paper. Now a large quantity of information exists in digital forms, including emails, blogs, social networking websites, national elections websites, web photo albums, and sites which change their content over time. According an article by Brewster Kahle, in 1996 founder of Internet Archive, "Preserving the Internet", Scientific American, the average life of a URL was, in 1997, 44 days.[2]

The unique characteristic of digital forms makes it easy to create content and keep it up-to-date, but at the same time brings many difficulties in the preservation of this content. Margaret Hedstrom points out that "...digital preservation raises challenges of a fundamentally different nature which are added to the problems of preserving traditional format materials."[3]

Physical deterioration

The media on which digital contents are stored are more vulnerable to deterioration and catastrophic loss than some analog media such as paper. While acid paper is prone to deterioration, becoming brittle and yellowing with age, the deterioration may not become apparent for some decades and progresses slowly. It remains possible to retrieve information without loss once deterioration is noticed. Digital data recording media may deteriorate more rapidly and once the deterioration starts, in most cases there may already be data loss. This characteristic of digital forms leaves a very short time frame for preservation decisions and actions.

Digital obsolescence

Another challenge is the issue of long-term access to data. Digital technology is developing quickly and retrieval and playback technologies can become obsolete in a matter of years. When faster, more capable and less expensive storage and processing devices are developed, older versions may be quickly replaced. When a software or decoding technology is abandoned, or a hardware device is no longer in production, records created with such technologies are at great risk of loss, simply because they are no longer accessible. This process is known as digital obsolescence.

This challenge is exacerbated by a lack of established standards, protocols and proven methods for preserving digital information.[4] We used to save copies of data on tapes, but media standards for tapes have changed considerably over the last five to ten years, and there is no guarantee that tapes will be readable in the future.[5] Recovering these materials may require special tools Hedstrom further explained that almost all digital library researches have been focused on "...architectures and systems for information organization and retrieval, presentation and visualization, and administration of intellectual property rights" and that "...digital preservation remains largely experimental and replete with the risks associated with untested methods".

Strategies

In 2006, the Online Computer Library Center developed a four-point strategy for the long-term preservation of digital objects that consisted of:

- Assessing the risks for loss of content posed by technology variables such as commonly used proprietary file formats and software applications.
- Evaluating the digital content objects to determine what type and degree of format conversion or other preservation actions should be applied.
- Determining the appropriate metadata needed for each object type and how it is associated with the objects.
- Providing access to the content.[6]

There are several additional strategies that individuals and organizations may use to actively combat the loss of digital information.

Refreshing

Refreshing is the transfer of data between two types of the same storage medium so there are no bitrate changes or alteration of data.[7] For example, transferring census data from an old preservation CD to a new one. This strategy may need to be combined with migration when the software or hardware required to read the data is no longer available or is unable to understand the format of the data. Refreshing will likely always be necessary due to the deterioration of physical media.

Migration

Migration is the transferring of data to newer system environments (Garrett et al., 1996). This may include conversion of resources from one file format to another (e.g., conversion of Microsoft Word to PDF or OpenDocument), from one operating system to another (e.g., Windows to Linux) or from one programming language to another (e.g., C to Java) so the resource remains fully accessible and functional. Resources that are migrated run the risk of losing some type of functionality since newer formats may be incapable of capturing all the functionality of the original format, or the converter itself may be unable to interpret all the nuances of the original format. The latter is often a concern with proprietary data formats.

The US National Archives Electronic Records Archives and Lockheed Martin are jointly developing a migration system that will preserve any type of document, created on any application or platform, and delivered to the archives on any type of digital media. In the system, files are translated into flexible formats, such as XML; they will therefore be accessible by technologies in the future. Lockheed Martin argues that it would be impossible to develop an emulation system for the National Archives ERA because the volume of records and cost would be prohibitive.

Replication

Creating duplicate copies of data on one or more systems is called *replication*. Data that exists as a single copy in only one location is highly vulnerable to software or hardware failure, intentional or accidental alteration, and environmental catastrophes like fire, flooding, etc. Digital data is more likely to survive if it is replicated in several locations. Replicated data may introduce difficulties in refreshing, migration, versioning, and access control since the data is located in multiple places.

Emulation

Emulation is the replicating of functionality of an obsolete system. Examples include emulating an Atari 2600 on a Windows system or emulating WordPerfect 1.0 on a Macintosh. Emulators may be built for applications, operating systems, or hardware platforms. Emulation has been a popular strategy for retaining the functionality of old video game systems, such as with the MAME project. The feasibility of emulation as a catch-all solution has been debated in the academic community. (Granger, 2000)

Raymond A. Lorie has suggested a Universal Virtual Computer (UVC) could be used to run any software in the future on a yet unknown platform. The UVC strategy uses a combination of emulation and migration. The UVC strategy has not yet been widely adopted by the digital preservation community.

Jeff Rothenberg, a major proponent of Emulation for digital preservation in libraries, working in partnership with Koninklijke Bibliotheek and National Archief of the Netherlands, has recently helped launch Dioscuri, a modular emulator that succeeds in running MS-DOS, WordPerfect 5.1, DOS games, and more.

Metadata attachment

Metadata is data on a digital file that includes information on creation, access rights, restrictions, preservation history, and rights management.[8] Metadata attached to digital files may be affected by file format obsolescence. ASCII is considered to be the most durable format for metadata [9] because it is widespread, backwards compatible when used with Unicode, and utilizes human-readable characters, not numeric codes. It retains information, but not the structure information it is presented in. For higher functionality, SGML or XML should be used. Both markup languages are stored in ASCII format, but contain tags that denote structure and format.

Trustworthy digital objects

Digital objects that can speak to their own authenticity are called *trustworthy digital objects* (TDOs). TDOs were proposed by Henry M. Gladney to enable digital objects to maintain a record of their change history so future users can know with certainty that the contents of the object are authentic. Other preservation strategies like replication and migration are necessary for the long-term preservation of TDOs.

Digital sustainability

Digital sustainability encompasses a range of issues and concerns that contribute to the longevity of digital information.[10] Unlike traditional, temporary strategies and more permanent solutions, digital sustainability implies a more active and continuous process. Digital sustainability concentrates less on the solution and technology and more on building an infrastructure and approach that is flexible with an emphasis on interoperability, continued maintenance and continuous development.[11] Digital sustainability incorporates activities in the present that will facilitate access and availability in the future.

Digital preservation standards

To standardize digital preservation practice and provide a set of recommendations for preservation program implementation, the Reference Model for an Open Archival Information System (OAIS) was developed. The reference model (ISO 14721:2003) includes the following responsibilities that an OAIS archive must abide by:

- Negotiate for and accept appropriate information from information Producers.
- Obtain sufficient control of the information provided to the level needed to ensure Long-Term Preservation.
- Determine, either by itself or in conjunction with other parties, which communities should become the Designated Community and, therefore, should be able to understand the information provided.
- Ensure that the information to be preserved is Independently Understandable to the Designated Community. In other words, the community should be able to understand the information without needing the assistance of the experts who produced the information.
- Follow documented policies and procedures which ensure that the information is preserved against all reasonable contingencies, and which enable the information to be disseminated as authenticated copies of the original, or as traceable to the original.
- Make the preserved information available to the Designated Community.[12]

OAIS is concerned with all technical aspects of a digital object's life cycle: ingest into and storage in a preservation infrastructure, data management, accessibility, and distribution. The model also addresses metadata issues and recommends that five types of metadata be attached to a digital object: reference (identification) information, provenance (including preservation history), context, fixity (authenticity indicators), and representation (formatting, file structure, and what "imparts meaning to an object's bitstream").[13]

Prior to Gladney's proposal of TDOs was the Research Library Group's (RLG) development of "attributes and responsibilities" that denote the practices of a "Trusted Digital Repository" (TDR) The seven attributes of a TDR are: "compliance with the Reference Model for an Open Archival Information System (OAIS), Administrative responsibility, Organizational viability, Financial sustainability, Technological and procedural suitability, System security, Procedural accountability." Among RLG's attributes and responsibilities were recommendations calling for the collaborative development of digital repository certifications, models for cooperative networks, and sharing of research and information on digital preservation with regards to intellectual property rights.[14]

Digital sound preservation standards

In January 2004, the Council on Library and Information Resources (CLIR) hosted a roundtable meeting of audio experts discussing best practices, which culminated in a report delivered March 2006. This report investigated procedures for reformatting sound from analog to digital, summarizing discussions and recommendations for best practices for digital preservation. Participants made a series of 15 recommendations for improving the practice of analog audio transfer for archiving:

- Develop core competencies in audio preservation engineering. Participants noted with concern that the number of experts qualified to transfer older recordings is shrinking and emphasized the need to find a way to ensure that the technical knowledge of these experts can be passed on.
- Develop arrangements among smaller institutions that allow for cooperative buying of esoteric materials and supplies.
- Pursue a research agenda for magnetic-tape problems that focuses on a less destructive solution for hydrolysis than baking, relubrication of acetate tapes, and curing of cupping.
- Develop guidelines for the use of automated transfer of analog audio to digital preservation copies.
- Develop a web-based clearinghouse for sharing information on how archives can develop digital preservation transfer programs.
- Carry out further research into nondestructive playback of broken audio discs.

- Develop a flowchart for identifying the composition of various types of audio discs and tapes.
- Develop a reference chart of problematic media issues.
- Collate relevant audio engineering standards from organizations.
- Research safe and effective methods for cleaning analog tapes and discs.
- Develop a list of music experts who could be consulted for advice on transfer of specific types of musical content (e.g., determining the proper key so that correct playback speed can be established).
- Research the life expectancy of various audio formats.
- Establish regional digital audio repositories.
- Cooperate to develop a common vocabulary within the field of audio preservation.
- Investigate the transfer of technology from such fields as chemistry and materials science to various problems in audio preservation.[15]

Updated technical guidelines on the creation and preservation of digital audio have been prepared by the International Association of Sound and Audiovisual Archives (IASA).[16]

Examples of digital preservation initiatives

- The Library of Congress operates the National Digital Information Infrastructure and Preservation Program [17]
- The British Library is responsible for several programmes in the area of **digital preservation**. The National Archives of the United Kingdom have also pioneered various initiatives in the field of **digital preservation**.
- The Safety Deposit Box [18] software from Tessella is being widely adopted by National Archives and has been selected by the National Archives in the UK, Netherlands, Switzerland, Finland, Estonia, Malaysia and Austria as well as FamilySearch and the Wellcome Collection. It was recently awarded the Queen's Awards for Enterprise in the UK.
- **Ex Libris Rosetta** is commercial software helping memory institutions to collect, manage, archive and preserve their digital collections, ensuring its data integrity and access over time. The system enables managing digital entities end to end—from submission to dissemination. A rule-based workflow engine and open architecture allow institutions using the system to develop unique plug-in tools and applications to enhance the system's ingest, management, preservation and delivery processes.
- The MetaArchive Cooperative is a library-run, collaborative approach to digital preservation that embeds digital preservation infrastructure and knowledge in each of its constituent member institutions. Comprised mainly of University libraries, the Cooperative functions as a network wherein each preserved file is replicated seven times, is stored in geographically distinct locations across four countries, and is carefully managed from ingest (as a SIP) to dissemination (as a DIP).

Large-scale digital preservation initiatives (LSDIs)

Many research libraries and archives have begun or are about to begin Large-Scale digital preservation initiatives (LSDI's). The main players in LSDIs are cultural institutions, commercial companies such as Google and Microsoft, and non-profit groups including the Open Content Alliance (OCA), the Million Book Project (MBP), and HathiTrust. The primary motivation of these groups is to expand access to scholarly resources.

LSDIs: library perspective

Approximately 30 cultural entities, including the 12-member Committee on Institutional Cooperation (CIC), have signed digitization agreements with either Google or Microsoft. Several of these cultural entities are participating in the Open Content Alliance (OCA) and the Million Book Project (MBP). Some libraries are involved in only one initiative and others have diversified their digitization strategies through participation in multiple initiatives. The three main reasons for library participation in LSDIs are: Access, Preservation and Research and Development. It is hoped that digital preservation will ensure that library materials remain accessible for future generations. Libraries

have a perpetual responsibility for their materials and a commitment to archive their digital materials. Libraries plan to use digitized copies as backups for works in case they go out of print, deteriorate, or are lost and damaged.

Footnotes

[1] Consultative Committee for Space Data Systems. (2002). Reference Model for an Open Archival Information System (OAIS). Washington, DC: CCSDS Secretariat, p. 1-1

[2] Brewster Kahle *Preserving the Internet*. «Scientific American», 276 (1997), n. 3, p. 72-74. (http://web.archive.org/web/19980627072808/http://www.sciam.com/0397issue/0397kahle.html/) retrieved on 2011-02-06

[3] Hedstrom, M. (1997). Digital preservation: a time bomb for Digital Libraries. Retrieved on December 4th, 2007, from http://www.uky.edu/~kiernan/DL/hedstrom.html.

[4] Levy, D. M. & Marshall, C. C. (1995). Going digital: a look at assumptions underlying digital libraries," Communications of the ACM, 58, No. 4: 77-84.

[5] Flugstad, Myron. (2007). Website Archiving: the Long-Term Preservation of Local Born Digital Resources. Arkansas Libraries v. 64 no. 3 (Fall 2007) p. 5-7

[6] Online Computer Library Center, Inc. (2006). OCLC Digital Archive Preservation Policy and Supporting Documentation (http://www.oclc.org/support/documentation/digitalarchive/preservationpolicy.pdf), p. 5

[7] Cornell University Library. (2005) Digital Preservation Management: Implementing Short-term Strategies for Long-term Problems. (http://www.library.cornell.edu/iris/tutorial/dpm/eng_index.html/)

[8] NISO Framework Advisory Group. (2007). A Framework of Guidance for Building Good Digital Collections, 3rd edition (http://www.niso.org/publications/rp/framework3.pdf), p. 57,

[9] National Initiative for a Networked Cultural Heritage. (2002). NINCH Guide to Good Practice in the Digital Representation and Management of Cultural Heritage Materials (http://www.nyu.edu/its/humanities/ninchguide/)

[10] Bradley, K. (Summer 2007). Defining digital sustainability. Library Trends v. 56 no 1 p. 148-163.

[11] Sustainability of Digital Resources. (2008). *TASI: Technical Advisory Service for Images.* (http://www.tasi.ac.uk/advice/managing/sust.html)

[12] Consultative Committee for Space Data Systems. (2002). Reference Model for an Open Archival Information System (OAIS). Washington, DC: CCSDS Secretariat, p. 3-1

[13] Cornell University Library. (2005) Digital Preservation Management: Implementing Short-term Strategies for Long-term Problems (http://www.library.cornell.edu/iris/tutorial/dpm/eng_index.html/)

[14] Research Libraries Group. (2002). Trusted Digital Repositories: Attributes and Responsibilities (http://www.oclc.org/programs/ourwork/past/trustedrep/repositories.pdf)

[15] Council on Library and Information Resources. Publication 137: *Capturing Analog Sound for Digital Preservation: Report of a Roundtable Discussion of Best Practices for Transferring Analog Discs and Tapes* March 2006 (http://www.clir.org/pubs/abstract/pub137abst.html)

[16] IASA (2009). Guidelines on the Production and Preservation of Digital Audio Objects (http://www.iasa-web.org/tc04/audio-preservation)

[17] http://www.digitalpreservation.gov

[18] http://www.digital-preservation.com

References

- Garrett, J., D. Waters, H. Gladney, P. Andre, H. Besser, N. Elkington, H. Gladney, M. Hedstrom, P. Hirtle, K. Hunter, R. Kelly, D. Kresh, M. Lesk, M. Levering, W. Lougee, C. Lynch, C. Mandel, S. Mooney, A. Okerson, J. Neal, S. Rosenblatt, and S. Weibe (1996). "Preserving digital information: Report of the task force on archiving of digital information" (http://www.rlg.org/legacy/ftpd/pub/archtf/final-report.pdf) (PDF). *Commission on Preservation and Access and the Research Libraries Group*. Retrieved 2009-06-23.
- Gladney, H. M.; Lorie, R. A. (2005). "Trustworthy 100-year digital objects: durable encoding for when it's too late to ask" (http://portal.acm.org/citation.cfm?id=1080343.1080346). *ACM Transactions on Information Systems* **23** (3): 299–324. doi: 10.1145/1080343.1080346 (http://dx.doi.org/10.1145/1080343.1080346).
- Gladney, H. M. (2006). "Principles for digital preservation" (http://portal.acm.org/citation.cfm?id=1113034.1113038&coll=GUIDE&dl=ACM&CFID=22202827&CFTOKEN=41873705). *Communications of the ACM* **49** (2): 111–116. doi: 10.1145/1113034.1113038 (http://dx.doi.org/10.1145/1113034.1113038).
- Granger, Stewart (2000). "Emulation as a Digital Preservation Strategy" (http://www.dlib.org/dlib/october00/granger/10granger.html). *D-Lib Magazine* **6** (10). doi: 10.1045/october2000-granger (http://dx.doi.org/10.1045/october2000-granger).

- Edwards, Eli (2004). "Ephemeral to Enduring: The Internet Archive and Its Role in Preserving Digital Media". *Information Technology & Libraries* **23** (1).
- Hedstrom, M., Ross, S., Ashley, K., Christensen-Dalsgaard, B., Duff, W., Gladney, H., Huc, C., Kenney, A.R., Moore, R., Neuhold, E. (2003). "Invest to Save: Report and Recommendations of the NSF-DELOS Working Group on Digital Archiving and Preservation" (http://delos-noe.iei.pi.cnr.it/activities/internationalforum/Joint-WGs/digitalarchiving/Digitalarchiving.pdf). *Nsf/Delos* (Pisa & Washington DC, USA).
- Jantz, R. & Giarlo, M.J. (2005). "Digital preservation: Architecture and technology for trusted digital repositories". *D-Lib Magazine* **11** (6). doi: 10.1045/june2005-jantz (http://dx.doi.org/10.1045/june2005-jantz).
- Ross, S (2000). *Changing Trains at Wigan: Digital Preservation and the Future of Scholarship* (http://www.bl.uk/blpac/pdf/wigan.pdf). London, UK: National Preservation Office (British Library). ISBN 0-7123-4717-8.
- Ross, S. and Gow, A. (1999). *Digital archaeology? Rescuing Neglected or Damaged Data Resources* (http://www.ukoln.ac.uk/services/elib/papers/supporting/pdf/p2.pdf). Bristol & London: British Library and Joint Information Systems Committee. ISBN 1-900508-51-6.
- Rothenberg, Jeff (1995). "Ensuring the Longevity of Digital Documents". *Scientific American* **272** (1).
- Rothenberg, Jeff (1999). *Ensuring the Longevity of Digital Information* (http://www.clir.org/pubs/archives/ensuring.pdf). Expanded version of *Ensuring the Longevity of Digital Documents*.
- Milne, Ronald -- moderator: Webcast panel discussion, "Economics," (http://www.lib.umich.edu/mdp/symposium/economics.html) *Scholarship and Libraries in Transition: A Dialogue about the Impacts of Mass Digitization Projects* (2006), Symposium sponsored by the University of Michigan Library and the National Commission on Libraries and Information Science (US).

External links

- National Digital Information Infrastructure and Preservation Program (http://www.digitalpreservation.gov) at the Library of Congress
- Digital Preservation page (http://www.diglib.org/preserve.htm) from the Digital Library Federation
- "Thirteen Ways of Looking at...Digital Preservation" (http://dlib.org/dlib/july04/lavoie/07lavoie.html)
- Cornell University Library's *Digital Imaging Tutorial* (http://www.library.cornell.edu/preservation/tutorial/contents.html)
- What is Digital Preservation? (http://www.digitalpreservationeurope.eu/what-is-digital-preservation/) - an introduction to digital preservation by Digital Preservation Europe
- Macroscopic 10-Terabit−per−Square-Inch Arrays from Block Copolymers with Lateral Order. (http://www.sciencemag.org/cgi/content/abstract/323/5917/1030) Science magazine article about prospective usage of sapphire in digital storage media technology
- Animations introducing digital preservation and curation (http://www.youtube.com/watch?v=pbBa6Oam7-w)
- Capture Your Collections: Planning and Implementing Digitization Projects (http://www.pro.rcip-chin.gc.ca/sommaire-summary/planification_numerisation-digitization_planning-eng.jsp) A CHIN (Canadian Heritage Information Network) Resource
- Digitales Archiv Hessen (http://www.hauptstaatsarchiv.hessen.de/irj/HHStAW_Internet?cid=69e5c1d5b0eb5a25b5961e83962b068c) Digital preservation page by Hessisches Hauptstaatsarchiv Wiesbaden

Category:Digital libraries]] Category:Preservation (library and archival science)]] Category:Archival science]] Category:Conservation-restoration]] Category:Data quality]]

Dirty data

Dirty data is inaccurate, incomplete or erroneous data, especially in a computer system or database.

In reference to databases, this is data that contain errors. Unclean data can contain such mistakes as spelling or punctuation errors, incorrect data associated with a field, incomplete or outdated data, or even data that has been duplicated in the database.

References

Entity integrity

In the relational data model, **entity integrity** is one of the three inherent integrity rules. Entity integrity is an integrity rule which states that every table must have a primary key and that the column or columns chosen to be the primary key should be unique and not null.[1]

Within relational databases using SQL, entity integrity is enforced by adding a primary key clause to a schema definition. The system enforces Entity Integrity by not allowing operation (INSERT, UPDATE) to produce an invalid primary key. Any operation that is likely to create a duplicate primary key or one containing nulls is rejected. The Entity Integrity ensures that the data that you store remains in the proper format as well as comprehensible.

References

[1] Beynon-Davies P. (2004). Database Systems 3rd Edition. Palgrave, Basingstoke, UK. ISBN 1-4039-1601-2

Information quality

Information quality (IQ) is a term to describe the quality of the content of information systems. It is often pragmatically defined as: "The fitness for use of the information provided."

Conceptual problems

Although this pragmatic definition is usable for most everyday purposes, specialists often use more complex models for information quality. Most information system practitioners use the term synonymously with data quality. However, as many academics make a distinction between data and information,[1] some will insist on a distinction between data quality and information quality. This distinction would be akin to the distinction between syntax and semantics where for example, the semantic value of "one" could be expressed in different syntaxes like 00001; 1.0000; 01.0; or 1. Thus a data difference may not necessarily represent poor information quality.

Information quality assurance is the process to guarantee confidence that particular information meets some context specific quality requirements. It has been suggested, however, that higher the quality the greater will be the confidence in meeting more general, less specific contexts.[2]

Dimensions and metrics of Information Quality

"Information quality" is a measure of the value which the information provides to the user of that information. "Quality" is often perceived as subjective and the quality of information can then vary among users and among uses of the information. Nevertheless, a high degree of quality increases its objectivity or at least the intersubjectivity. Accuracy can be seen as just one element of IQ but, depending upon how it is defined, can also be seen as encompassing many other dimensions of quality.

If not, it is perceived that often there is a trade-off between accuracy and other dimensions, aspects or elements of the information determining its suitability for any given tasks. Wang and Strong propose a list of dimensions or elements used in assessing Information Quality is:[3]

- Intrinsic IQ: Accuracy, Objectivity, Believability, Reputation
- Contextual IQ: Relevancy, Value-Added, Timeliness, Completeness, Amount of information
- Representational IQ: Interpretability, Format, Coherence, Compatibility
- Accessibility IQ: Accessibility, Access security

Other authors propose similar but different lists of dimensions for analysis, and emphasize measurement and reporting as information quality metrics. Larry English prefers the term "characteristics" to dimensions.[4]

While information as a distinct term has various ambiguous definitions, there's one which is more general, such as "description of events". While the occurrences being described cannot be subjectively evaluated for quality, since they're very much autonomous events in space and time, their description can—since it possesses a garnishment attribute, unavoidably attached by the medium which carried the information, from the initial moment of the occurrences being described.

In an attempt to deal with this natural phenomenon, qualified professionals primarily representing the researchers' guild, have at one point or another identified particular metrics for information quality. They could also be described as 'quality traits' of information, since they're not so easily quantified, but rather subjectively identified on an individual basis.

Proposed quality metrics

- **Authority/Verifiability**

Authority refers to the expertise or recognized official status of a source. Consider the reputation of the author and publisher. When working with legal or government information, consider whether the source is the official provider of the information. Verifiability refers to the ability of a reader to verify the validity of the information irresepective of how authoritative the source is. To verify the facts is part of the duty of care of the journalistic deontology, as well as, where possible, to provide the sources of information so that they can be verified

- **Scope of coverage**

Scope of coverage refers to the extent to which a source explores a topic. Consider time periods, geography or jurisdiction and coverage of related or narrower topics.

- **Composition and Organization**

Composition and Organization has to do with the ability of the information source to present it's particular message in a coherent, logically sequential manner.

- **Objectivity**

Objectivity is the bias or opinion expressed when a writer interprets or analyze facts. Consider the use of persuasive language, the source's presentation of other viewpoints, its reason for providing the information and advertising.

- **Integrity**

1. Adherence to moral and ethical principles; soundness of moral character
2. The state of being whole, entire, or undiminished

- **Comprehensiveness**

1. Of large scope; covering or involving much; inclusive: a comprehensive study.
2. Comprehending mentally; having an extensive mental grasp.
3. Insurance. covering or providing broad protection against loss.

- **Validity**

Validity of some information has to do with the degree of obvious truthfulness which the information carries

- **Uniqueness**

As much as 'uniqueness' of a given piece of information is intuitive in meaning, it also significantly implies not only the originating point of the information but also the manner in which it is presented and thus the perception which it conjures. The essence of any piece of information we process consists to a large extent of those two elements.

- **Timeliness**

Timeliness refers to information that is current at the time of publication. Consider publication, creation and revision dates. Beware of Web site scripting that automatically reflects the current day's date on a page.

- **Reproducibility** (utilized primarily when referring to instructive information)

Means that documented methods are capable of being used on the same data set to achieve a consistent result.

Education

University of Arkansas at Little Rock offers graduate degrees in the field of Information Quality.[5] University level Information Quality courses around the world are listed in [6]

Professional Associations

International Association for Information and Data Quality (IAIDQ)[7]

IAIDQ is a not-for-profit, vendor neutral, professional association formed in 2004, dedicated to building the information and data quality profession.

Information Quality conferences

A number of major conferences relevant to information quality are held annually:

Data Governance and Information Quality Conference[8]

Commercial conferences held each year in the USA

Data Quality Asia Pacific[9]

Commercial conference held annually or Sydney or Melbourne, Australia

Enterprise Data and Business Intelligence Conference Europe[10]

Commercial conferences held annually in London, England.

Information and Data Quality Conference[11]

Not for profit conference run annually by IAIDQ in the USA

International Conference on Information Quality[12]

Academic Conference launched through MITIQ held annually at a University

Master Data Management & Data Governance Conferences[13]

Six major conferences are run annually by the MDM Institute in venues such as London, San Francisco, Sydney, Toronto, Madrid, Frankfurt, Shanghai and New York City.

References

[1] For a scientific and philosophical unraveling of these concept see Churchman, C.W. (1971) *The design of inquiring systems*, New York: Basic Books.

[2] See Ivanov, K. (1972) "Quality-control of information: On the concept of accuracy of information in data banks and in management information systems" (http://www.informatik.umu.se/~kivanov/diss-avh.html). The University of Stockholm and The Royal Institute of Technology. Doctoral dissertation. Further details are found in Ivanov, K. (1995). A subsystem in the design of informatics: Recalling an archetypal engineer. In B. Dahlbom (Ed.), *The infological equation: Essays in honor of Börje Langefors* (http://www.informatik.umu.se/~kivanov/BLang80.html), (pp. 287-301). Gothenburg: Gothenburg University, Dept. of Informatics (ISSN 1101-7422).

[3] Wang, R. & Strong, D. (1996) "Beyond Accuracy: What Data Quality Means to Data Consumers". "Journal of Management Information Systems", 12(4), p. 5-34.

[4] English, Larry P. (2009) "Information Quality Applied", Wiley Publishing, Indianapolis. ISBN 978-0-470-13447-4

[5] UALR Information Quality (http://ualr.edu/informationquality/)

[6] https://spreadsheets.google.com/a/ualr.edu/lv?key=0Aqdli_lt_048dDNRczIzbGJ2OUY3R0ZSazY3cEpuc0E&hl=en_GB&authkey=CLXWyWM&f=0&rm=full#gid=1

[7] International Association for Information and Data Quality (http://iaidq.org/)

[8] Data Governance and Information Quality Conference (http://dgiq-conference.com/)

[9] Data Quality Asia Pacific (http://www.dqasiapacific.com/)

[10] Data Governance Conference Europe (http://www.irmuk.co.uk/)

[11] Information and Data Quality Conference (http://idq-conference.com/)

[12] http://www.iciq2013.org/

[13] MDM SUMMIT Conference (http://www.tcdii.com/events/cdimdmsummitseries.html)

Link rot

Link rot (or **linkrot**), also known as **link death** or **link breaking**, is an informal term for the process by which hyperlinks (either on individual websites or the Internet in general) point to web pages, servers or other resources that have become permanently unavailable. The phrase also describes the effects of failing to update out-of-date web pages that clutter search engine results. A link that does not work any more is called a **broken link**, **dead link**, or **dangling link**.

Causes

A link may become broken for several reasons. The simplest and most common reason is that the web page it links to doesn't exist anymore. The most common result of a dead link is a 404 error, which indicates that the web server responded, but the specific page could not be found.

Some news sites contribute to the problem of link rot by keeping only recent news articles freely accessible, then removing them or moving them to a paid subscription area. This causes a heavy loss of supporting links in sites discussing newsworthy events and using news sites as references.[*citation needed*]

Another type of dead link occurs when the server that hosts the target page stops working or relocates to a new domain name. In this case the browser may return a DNS error, or it may display a site unrelated to the content sought. The latter can occur when a domain name is allowed to lapse and is subsequently reregistered by another party. Domain names acquired in this manner are attractive to those who wish to take advantage of the stream of unsuspecting surfers that will inflate hit counters and PageRanking.

A link might also be broken because of some form of blocking such as content filters or firewalls. Dead links commonplace on the Internet can also occur on the authoring side, when website content is assembled, copied, or deployed without properly verifying the targets, or simply not kept up to dateWikipedia:Please clarify. Dead links can also occur when a website without Clean URLs is "re-organized".

Prevalence

The 404 "Not Found" response is familiar to even the occasional web user. A number of studies have examined the prevalence of link rot on the web, in academic literature, and in digital libraries. In a 2003 experiment, Fetterly et al. discovered that about one link out of every 200 disappeared each week from the internet. McCown et al. (2005) discovered that half of the URLs cited in D-Lib Magazine articles were no longer accessible 10 years after publication, and other studies have shown link rot in academic literature to be even worse (Spinellis, 2003, Lawrence et al., 2001). Nelson and Allen (2002) examined link rot in digital libraries and found that about 3% of the objects were no longer accessible after one year.

Discovering

Detecting link rot for a given URL is difficult using automated methods. If a URL is accessed and returns an HTTP 200 (OK) response, it may be considered accessible, but the contents of the page may have changed and may no longer be relevant. Some web servers also return a soft 404, a page returned with a 200 (OK) response (instead of a 404 that indicates the URL is no longer accessible). Bar-Yossef et al. (2004) developed a heuristic for automatically discovering soft 404s.

Combating

Due to the unprofessional image that dead links bring to both sites linking and linked to, there are multiple solutions that are available to tackle them: some working to prevent them in the first place, and others trying to resolve them when they have occurred. There are several tools that have been developed to help combat link rot.

Server side

- Avoiding unmanaged hyperlink collections
- Avoiding links to pages deep in a website ("deep linking")
- Using redirection mechanisms (e.g. "301: Moved Permanently") to automatically refer browsers and crawlers to the new location of a URL
- Content management systems may offer builtin solutions to the management of links, e.g. links are updated when content is changed or moved on the site.
- WordPress guards against link rot by replacing non-canonical URLs with their canonical versions.
- IBM's Peridot attempts to automatically fix broken links.
- Permalinking stops broken links by guaranteeing that the content will not move for the foreseeable future. Another form of permalinking is linking to a permalink that then redirects to the actual content, ensuring that even though the real content may be moved etc., links pointing to the resources stay intact.

User side

- The Linkgraph widget gets the URL of the correct page based upon the old broken URL by using historical location information.
- The Google 404 Widget employs Google technology to 'guess' the correct URL, and also provides the user a Google search box to find the correct page.
- When a user receives a 404 response, the Google Toolbar attempts to assist the user in finding the missing page.
- Deadurl.com gathers and ranks alternate urls for a broken link using Google Cache, the Internet Archive, and user submissions. Typing deadurl.com/ left of a broken link in the browser's address bar and pressing enter loads a ranked list of alternate urls, or (depending on user preference) immediately forwards to the best one.

Web archiving

To combat link rot, web archivists are actively engaged in collecting the Web or particular portions of the Web and ensuring the collection is preserved in an archive, such as an archive site, for future researchers, historians, and the public. The largest web archiving organization is the Internet Archive[*citation needed*], whose goal is to maintain an archive of the entire Web, taking periodic snapshots of pages that can then be accessed for free via the Wayback Machine and without registration many years later simply by typing in the URL, or automatically by using browser extensions.WP:NOTRS National libraries, national archives and various consortia of organizations are also involved in archiving culturally important Web content.

Individuals may also use a number of tools that allow them to archive web resources that may go missing in the future:

- WebCite, a tool specifically for scholarly authors, journal editors and publishers to permanently archive "on-demand" and retrieve cited Internet references (Eysenbach and Trudel, 2005).
- Archive-It, a subscription service that allows institutions to build, manage and search their own web archive
- Some social bookmarking websites such as peeep.us, or evernote.com allow users to make online clones of any web page on the internet, creating a copy at an independent url which remains online even if the original page goes down.
- Google keeps a text-based cache (temporary copy) of the pages it has crawled, which can be used to read the information of recently removed pages. However, unlike in archiving services, cached pages are not stored

permanently.
- The WayBack Machine, at the Internet Archive, is a free website that archives old web pages. It does not archive websites whose owners have stated they do not want their website archived.
- archive.is [1], a personal WayBack Machine, which saves text and screenshot of a webpage and keeps it online even if the original page is rot. In most aspects it is a WebCite analog, but also saves images and can save pages from Web 2.0 sites (like Twitter).

Authors citing URLs

A number of studies have shown how widespread link rot is in academic literature (see below). Authors of scholarly publications have also developed best practices for combating link rot in their work:

- Avoiding URL citations that point to resources on a researcher's personal home page (McCown et al., 2005)
- Using Persistent Uniform Resource Locators (PURLs) and digital object identifiers (DOIs) whenever possible
- Using web archiving services (e.g. WebCite) to permanently archive and retrieve cited Internet references (Eysenbach and Trudel, 2005).

Further reading

Link rot on the Web

- Eysenbach, Gunther; Trudel, Mathieu (2005). "Going, going, still there: using the WebCite service to permanently archive cited web pages" [2]. *Journal of Medical Internet Research* **7** (5): e60. doi:10.2196/jmir.7.5.e60 [3]. PMC 1550686 [2]. PMID 16403724 [4].
- Koehler, Wallace (2004). "A longitudinal study of web pages continued: A consideration of document persistence" [5]. *Information Research* **9** (2).
- Bar-Yossef, Ziv; Broder, Andrei Z.; Kumar, Ravi; Tomkins, Andrew (2004). "Sic transit gloria telae: towards an understanding of the Web's decay". *Proceedings of the 13th international conference on World Wide Web*. pp. 328–337. doi:10.1145/988672.988716 [6].
- Fetterly, Dennis; Manasse, Mark; Najork, Marc; Wiener, Janet (2003). "A large-scale study of the evolution of web pages" [7]. *Proceedings of the 12th international conference on World Wide Web*. Retrieved 2010-09-14.
- Markwell, John; Brooks, David W. (2002). "Broken Links: The Ephemeral Nature of Educational WWW Hyperlinks". *Journal of Science Education and Technology* **11** (2): 105–108. doi:10.1023/A:1014627511641 [8].
- Berners-Lee, Tim (1998). *Cool URIs Don't Change* [9]. Retrieved 2010-09-14.

In academic literature

- Gomes, Daniel; Silva, Mário J. (2006). "Modelling Information Persistence on the Web" [10]. *Proceedings of The 6th International Conference on Web Engineering*. ICWE'06. Retrieved 2010-09-14.
- Dellavalle, Robert P.; Hester, Eric J.; Heilig, Lauren F.; Drake, Amanda L.; Kuntzman, Jeff W.; Graber, Marla; Schilling, Lisa M. (2003). "Going, Going, Gone: Lost Internet References". *Science* **302** (5646): 787–788. doi:10.1126/science.1088234 [11]. PMID 14593153 [12].
- Lawrence, Steve; Pennock, David M.; Flake, Gary William; Krovetz, Robert; Coetzee, Frans M.; Glover, Eric; Nielsen, Finn Arup; Kruger, Andries; Giles, C. Lee (2001). "Persistence of Web References in Scientific Research". *Computer* **34** (2): 26–31. doi:10.1109/2.901164 [13]. CiteSeerX: 10.1.1.97.9695 [14].
- Koehler, Wallace (1999). "An Analysis of Web Page and Web Site Constancy and Permanence". *Journal of the American Society for Information Science* **50** (2): 162–180. doi:10.1002/(SICI)1097-4571(1999)50:2<162::AID-ASI7>3.0.CO;2-B [15].
- McCown, Frank; Chan, Sheffan; Nelson, Michael L.; Bollen, Johan (2005). "The Availability and Persistence of Web References in D-Lib Magazine" [16]. *Proceedings of the 5th International Web Archiving Workshop and*

Digital Preservation (IWAW'05).

- Sellitto, Carmine (2005). "The impact of impermanent Web-located citations: A study of 123 scholarly conference publications" [17]. *Journal of the American Society for Information Science and Technology* **56** (7): 695–703. doi:10.1002/asi.20159 [18].
- Spinellis, Diomidis (2003). "The Decay and Failures of Web References" [19]. *Communications of the ACM* **46** (1): 71–77. doi:10.1145/602421.602422 [20].

In digital libraries

- Nelson, Michael L.; Allen, B. Danette (2002). "Object Persistence and Availability in Digital Libraries". *D-Lib Magazine* **8** (1). doi:10.1045/january2002-nelson [21].

References

[1] http://archive.is/
[2] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1550686
[3] http://dx.doi.org/10.2196%2Fjmir.7.5.e60
[4] http://www.ncbi.nlm.nih.gov/pubmed/16403724
[5] http://informationr.net/ir/9-2/paper174.html
[6] http://dx.doi.org/10.1145%2F988672.988716
[7] http://www2003.org/cdrom/papers/refereed/p097/P97%20sources/p97-fetterly.html
[8] http://dx.doi.org/10.1023%2FA%3A1014627511641
[9] http://www.w3.org/Provider/Style/URI.html
[10] http://xldb.di.fc.ul.pt/daniel/docs/papers/gomes06urlPersistence.pdf
[11] http://dx.doi.org/10.1126%2Fscience.1088234
[12] http://www.ncbi.nlm.nih.gov/pubmed/14593153
[13] http://dx.doi.org/10.1109%2F2.901164
[14] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.9695
[15] http://dx.doi.org/10.1002%2F%28SICI%291097-4571%281999%2950%3A2%3C162%3A%3AAID-ASI7%3E3.0.CO%3B2-B
[16] http://www.iwaw.net/05/papers/iwaw05-mccown1.pdf
[17] http://doi.wiley.com/10.1002/asi.20159
[18] http://dx.doi.org/10.1002%2Fasi.20159
[19] http://www.spinellis.gr/pubs/jrnl/2003-CACM-URLcite/html/urlcite.html
[20] http://dx.doi.org/10.1145%2F602421.602422
[21] http://dx.doi.org/10.1045%2Fjanuary2002-nelson

External links

- Future-Proofing Your URIs (http://www.wrox.com/WileyCDA/Section/id-301495.html)
- Jakob Nielsen, "Fighting Linkrot" (http://www.useit.com/alertbox/980614.html), *Jakob Nielsen's Alertbox*, June 14, 1998.

One-for-one checking

In systems auditing, **one-for-one checking** is a control process that is frequently used to ensure that specific elements between two or more sources of data are consistent. The control process can also reduce the chances of human error by typos and miskeyed information.

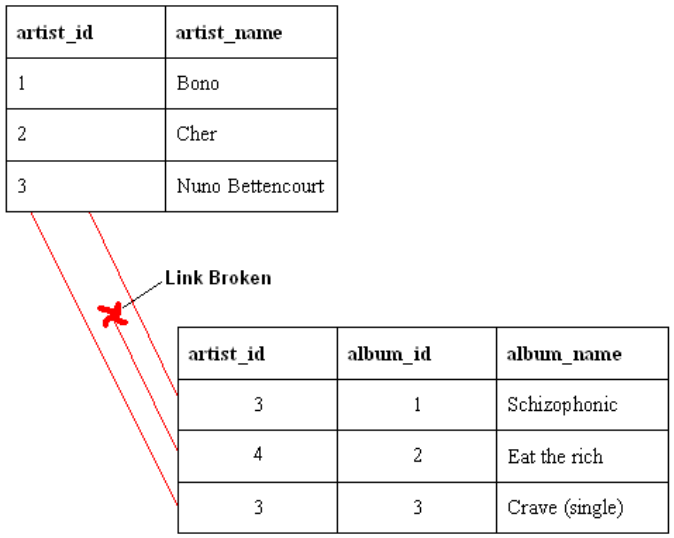
An operations manager might use one-for-one checking of cheques and receivables in order to verify that cash collected is properly reflected by the receivable accounts with regard to the collected cash (i.e., each cheque is associated with an invoice).

References

- *Accounting Information Systems.* Gelinas, Dull. 7th ed. 2008 Thomson Higher Education. ISBN 0-324-37882-3.

Referential integrity

Referential integrity is a property of data which, when satisfied, requires every value of one attribute (column) of a relation (table) to exist as a value of another attribute in a different (or the same) relation (table).



An example of a database that has not enforced **referential integrity**. In this example, there is a foreign key (`artist_id`) value in the album table that references a non-existent artist — in other words there is a foreign key value with no corresponding primary key value in the referenced table. What happened here was that there was an artist called "Aerosmith", with an `artist_id` of `4`, which was deleted from the artist table. However, the album "Eat the Rich" referred to this artist. With referential integrity enforced, this would not have been possible.

For referential integrity to hold in a relational database, any field in a table that is declared a foreign key can contain either a null value, or only values from a parent table's primary key or a candidate key.[1] In other words, when a foreign key value is used it must reference a valid, existing primary key in the parent table. For instance, deleting a record that contains a value referred to by a foreign key in another table would break referential integrity. Some relational database management systems (RDBMS) can enforce referential integrity, normally either by deleting the foreign key rows as well to maintain integrity, or by returning an error and not performing the delete. Which method is used may be determined by a referential integrity constraint defined in a data dictionary.

"Referential" the adjective describes the action that a foreign key performs, 'referring' to a link field in another table. In simple terms, 'referential integrity' is a guarantee that the target it 'refers' to will be found. A lack of referential integrity in a database can lead relational databases to return incomplete data, usually with no indication of an error. A common problem occurs with relational database tables linked with an 'inner join' which requires non-NULL values in both tables, a requirement that can only be met through careful design and referential integrity.

Formalization

An **inclusion dependency** over two (possibly identical) predicates R and S from a schema is written $R[A_1, ..., A_n] \subseteq S[B_1, ..., B_n]$, where the A_i, B_i are distinct attributes (column names) of R and S. It implies that the tuples of values appearing in columns $A_1, ..., A_n$ for facts of R must also appear as a tuple of values in columns $B_1, ..., B_n$ for some fact of S.

Logical implication between inclusion dependencies can be axiomatized by inference rules [2] and can be decided by a PSPACE algorithm. The problem can be shown to be PSPACE-complete by reduction from the acceptance problem for a linear bounded automaton.[3] However, logical implication between dependencies that can be inclusion dependencies or functional dependencies is undecidable by reduction from the word problem for monoids.[4]

References

[1] Coronel et al. (2013). Database Systems 10th ed. Cengage Learning, ISBN 978-1-111-96960-8

[2] Abiteboul, Hull, Vianu. *Foundations of Databases* Addison-Wesley, 1994. Section 9.1, p. 193. Freely available online (http://webdam.inria.fr/Alice/).

[3] ibid., p. 196

[4] ibid., p. 199

Soft error

In electronics and computing, a **soft error** is a type of error where a signal or datum is wrong. Errors may be caused by a defect, usually understood either to be a mistake in design or construction, or a broken component. A soft error is also a signal or datum which is wrong, but is not assumed to imply such a mistake or breakage. After observing a soft error, there is no implication that the system is any less reliable than before. In the spacecraft industry this kind of error is called a single-event upset.

In a computer's memory system, a soft error changes an instruction in a program or a data value. Soft errors typically can be remedied by cold booting the computer. A soft error will not damage a system's hardware; the only damage is to the data that is being processed.

There are two types of soft errors:

Chip-level soft error

> These errors occur when the radioactive atoms in the chip's material decay and release alpha particles into the chip.
>
> Because an alpha particle contains a positive charge and kinetic energy, the particle can hit a memory cell and cause the cell to change state to a different value. The atomic reaction is so tiny that it does not damage the actual structure of the chip.

System-level soft error

> These errors occur when the data being processed is hit with a noise phenomenon, typically when the data is on a data bus. The computer tries to interpret the noise as a data bit, which can cause errors in addressing or processing program code. The bad data bit can even be saved in memory and cause problems at a later time.

If detected, a soft error may be corrected by rewriting correct data in place of erroneous data. Highly reliable systems use error correction to correct soft errors on the fly. However, in many systems, it may be impossible to determine the correct data, or even to discover that an error is present at all. In addition, before the correction can occur, the system may have crashed, in which case the recovery procedure must include a reboot.

Soft errors involve changes to data — the electrons in a storage circuit, for example — but not changes to the physical circuit itself, the atoms. If the data is rewritten, the circuit will work perfectly again.

Soft errors can occur on transmission lines, in digital logic, analog circuits, magnetic storage, and elsewhere, but are most commonly known in semiconductor storage.

Critical charge

Whether or not a circuit experiences a soft error depends on the energy of the incoming particle, the geometry of the impact, the location of the strike, and the design of the logic circuit. Logic circuits with higher capacitance and higher logic voltages are less likely to suffer an error. This combination of capacitance and voltage is described by the **critical charge** parameter, $\mathbf{Q_{crit}}$, the minimum electron charge disturbance needed to change the logic level. A higher Q_{crit} means fewer soft errors. Unfortunately, a higher Q_{crit} also means a slower logic gate and a higher power dissipation. Reduction in chip feature size and supply voltage, desirable for many reasons, decreases Q_{crit}. Thus, the importance of soft errors increases as chip technology advances.

In a logic circuit, Q_{crit} is defined as the minimum amount of induced charge required at a circuit node to cause a voltage pulse to propagate from that node to the output and be of sufficient duration and magnitude to be reliably latched. Since a logic circuit contains many nodes that may be struck, and each node may be of unique capacitance and distance from output, Q_{crit} is typically characterized on a per-node basis.

Causes of soft errors

Alpha particles from package decay

Soft errors became widely known with the introduction of dynamic RAM in the 1970s. In these early devices, chip packaging materials contained small amounts of radioactive contaminants. Very low decay rates are needed to avoid excess soft errors, and chip companies have occasionally suffered problems with contamination ever since. It is extremely hard to maintain the material purity needed. Controlling alpha particle emission rates for critical packaging materials to less than a level of 0.001 counts per hour per cm^2 (cph/cm^2) is required for reliable performance of most circuits. For comparison, the count rate of a typical shoe's sole is between 0.1 and 10 cph/cm^2.

Package radioactive decay usually causes a soft error by alpha particle emission. The positively charged alpha particle travels through the semiconductor and disturbs the distribution of electrons there. If the disturbance is large enough, a digital signal can change from a 0 to a 1 or vice versa. In combinational logic, this effect is transient, perhaps lasting a fraction of a nanosecond, and this has led to the challenge of soft errors in combinational logic mostly going unnoticed. In sequential logic such as latches and RAM, even this transient upset can become stored for an indefinite time, to be read out later. Thus, designers are usually much more aware of the problem in storage circuits.

A 2011 'Black Hat' paper discusses the real-life security implications of such bit-flips in the internet's DNS system. The paper found up to 3,434 incorrect requests a day due to bit-flip changes for various common domains. Many of these bit-flips would probably be attributable to hardware problems, but some could be attributed to alpha particles.

Isaac Asimov received a letter congratulating him on an accidental prediction of alpha-particle RAM errors in a 1950s novel.[1]

Cosmic rays creating energetic neutrons and protons

Once the electronics industry had determined how to control package contaminants, it became clear that other causes were also at work. James F. Ziegler led a program of work at IBM which culminated in the publication of a number of papers (Ziegler and Lanford, 1979) demonstrating that cosmic rays also could cause soft errors. Indeed, in modern devices, cosmic rays may be the predominant cause. Although the primary particle of the cosmic ray does not generally reach the Earth's surface, it creates a shower of energetic secondary particles. At the Earth's surface approximately 95% of the particles capable of causing soft errors are energetic neutrons with the remainder

composed of protons and pions. IBM estimated in 1996 that one error per month per 256 MiB of ram was expected for a desktop computer. This flux of energetic neutrons is typically referred to as "cosmic rays" in the soft error literature. Neutrons are uncharged and cannot disturb a circuit on their own, but undergo neutron capture by the nucleus of an atom in a chip. This process may result in the production of charged secondaries, such as alpha particles and oxygen nuclei, which can then cause soft errors.

Cosmic ray flux depends on altitude. For the common reference location of 40.7° N, 74° W at sea level (New York City, NY, USA) the flux is approximately 14 neutrons/cm^2/hour. Burying a system in a cave reduces the rate of cosmic-ray induced soft errors to a negligible level. In the lower levels of the atmosphere, the flux increases by a factor of about 2.2 for every 1000 m (1.3 for every 1000 ft) increase in altitude above sea level. Computers operated on top of mountains experience an order of magnitude higher rate of soft errors compared to sea level. The rate of upsets in aircraft may be more than 300 times the sea level upset rate. This is in contrast to package decay induced soft errors, which do not change with location. As chip density increases, Intel expects the errors caused by cosmic rays to increase and be a limiting factor in design.[]

The average rate of cosmic-ray soft errors is *inversely* proportional to sunspot activity. That is, the average number of cosmic-ray soft errors decreases during the active portion of the sunspot cycle and increases during the quiet portion. This counterintuitive result occurs for two reasons. The sun does not generally produce cosmic ray particles with energy above 1 GeV that are capable of penetrating to the Earth's upper atmosphere and creating particle showers, so the changes in the solar flux do not directly influence the number of errors. Further, the increase in the solar flux during an active sun period does have the effect of reshaping the Earth's magnetic field providing some additional shielding against higher energy cosmic rays, resulting in a decrease in the number of particles creating showers. The effect is fairly small in any case resulting in a ±7% modulation of the energetic neutron flux in New York City. Other locations are similarly affected.

Energetic neutrons produced by cosmic rays may lose most of their kinetic energy and reach thermal equilibrium with their surroundings as they are scattered by materials. The resulting neutrons are simply referred to as thermal neutrons and have an average kinetic energy of about 25 millielectron-volts at 25°C. Thermal neutrons are also produced by environmental radiation sources such as the decay of naturally occurring uranium or thorium. The thermal neutron flux from sources other than cosmic-ray showers may still be noticeable in an underground location and an important contributor to soft errors for some circuits.

Thermal neutrons

Neutrons that have lost kinetic energy until they are in thermal equilibrium with their surroundings are an important cause of soft errors for some circuits. At low energies many neutron capture reactions become much more probable and result in fission of certain materials creating charged secondaries as fission byproducts. For some circuits the capture of a thermal neutron by the nucleus of the ^{10}B isotope of boron is particularly important. This nuclear reaction is an efficient producer of an alpha particle, ^{7}Li nucleus and gamma ray. Either of the charged particles (alpha or ^{7}Li) may cause a soft error if produced in very close proximity, approximately 5 μm, to a critical circuit node. The capture cross section for ^{11}B is 6 orders of magnitude smaller and does not contribute to soft errors.

Boron has been used in BPSG, the insulator in the interconnection layers of integrated circuits, particularly in the lowest one. The inclusion of boron lowers the melt temperature of the glass providing better reflow and planarization characteristics. In this application the glass is formulated with a boron content of 4% to 5% by weight. Naturally occurring boron is 20% ^{10}B with the remainder the ^{11}B isotope. Soft errors are caused by the high level of ^{10}B in this critical lower layer of some older integrated circuit processes. Boron-11, used at low concentrations as a p-type dopant, does not contribute to soft errors. Integrated circuit manufacturers eliminated borated dielectrics by the 150 nm process node, largely due to this problem.

In critical designs, depleted boron — consisting almost entirely of boron-11 — is used, to avoid this effect and therefore to reduce the soft error rate. Boron-11 is a by-product of the nuclear industry.

For applications in medical electronic devices this soft error mechanism may be extremely important. Neutrons are produced during high energy cancer radiation therapy using photon beam energies above 10 MeV. These neutrons are moderated as they are scattered from the equipment and walls in the treatment room resulting in a thermal neutron flux that is about 40×10^6 higher than the normal environmental neutron flux. This high thermal neutron flux will generally result in a very high rate of soft errors and consequent circuit upset. [2]

Other causes

Soft errors can also be caused by random noise or signal integrity problems, such as inductive or capacitive crosstalk. However, in general, these sources represent a small contribution to the overall soft error rate when compared to radiation effects.

Designing around soft errors

Soft error mitigation

A designer can attempt to minimize the rate of soft errors by judicious device design, choosing the right semiconductor, package and substrate materials, and the right device geometry. Often, however, this is limited by the need to reduce device size and voltage, to increase operating speed and to reduce power dissipation. The susceptibility of devices to upsets is described in the industry using the JEDEC JESD-89 standard.

One technique that can be used to reduce the soft error rate in digital circuits is called radiation hardening. This involves increasing the capacitance at selected circuit nodes in order to increase its effective Q_{crit} value. This reduces the range of particle energies to which the logic value of the node can be upset. Radiation hardening is often accomplished by increasing the size of transistors who share a drain/source region at the node. Since the area and power overhead of radiation hardening can be restrictive to design, the technique is often applied selectively to nodes which are predicted to have the highest probability of resulting in soft errors if struck. Tools and models that can predict which nodes are most vulnerable are the subject of past and current research in the area of soft errors.

Detecting soft errors

There has been work addressing soft errors in processor and memory resources using both hardware and software techniques. Several research efforts addressed soft errors by proposing error detection and recovery via hardware-based redundant multi-threading. These approaches used special hardware to replicate an application execution to identify errors in the output, which increased hardware design complexity and cost including high performance overhead.

Correcting soft errors

Designers can choose to accept that soft errors will occur, and design systems with appropriate error detection and correction to recover gracefully. Typically, a semiconductor memory design might use forward error correction, incorporating redundant data into each word to create an error correcting code. Alternatively, roll-back error correction can be used, detecting the soft error with an error-detecting code such as parity, and rewriting correct data from another source. This technique is often used for write-through cache memories.

Soft errors in logic circuits are sometimes detected and corrected using the techniques of fault tolerant design. These often include the use of redundant circuitry or computation of data, and typically come at the cost of circuit area, decreased performance, and/or higher power consumption. The concept of triple modular redundancy (TMR) can be employed to ensure very high soft-error reliability in logic circuits. In this technique, three identical copies of a circuit compute on the same data in parallel and outputs are fed into majority voting logic, returning the value that occurred in at least two of three cases. In this way, the failure of one circuit due to soft error is discarded assuming the other two circuits operated correctly. In practice, however, few designers can afford the greater than 200% circuit

area and power overhead required, so it is usually only selectively applied. Another common concept to correct soft errors in logic circuits is temporal (or time) redundancy, in which one circuit operates on the same data multiple times and compares subsequent evaluations for consistency. This approach, however, often incurs performance overhead, area overhead (if copies of latches are used to store data), and power overhead, though is considerably more area-efficient than modular redundancy.

Traditionally, DRAM has had the most attention in the quest to reduce, or work-around soft errors, due to the fact that DRAM has comprised the majority-share of susceptible device surface area in desktop, and server computer systems (ref. the prevalence of ECC RAM in server computers). Hard figures for DRAM susceptibility are hard to come by, and vary considerably across designs, fabrication processes, and manufacturers. 1980s technology 256 kilobit DRAMS could have clusters of five or six bits flip from a single alpha particle. Modern DRAMs have much smaller feature sizes, so the deposition of a similar amount of charge could easily cause many more bits to flip.

The design of error detection and correction circuits is helped by the fact that soft errors usually are localised to a very small area of a chip. Usually, only one cell of a memory is affected, although high energy events can cause a multi-cell upset. Conventional memory layout usually places one bit of many different correction words adjacent on a chip. So, even a *multi-cell upset* leads to only a number of separate *single-bit upsets* in multiple correction words, rather than a *multi-bit upset* in a single correction word. So, an error correcting code needs only to cope with a single bit in error in each correction word in order to cope with all likely soft errors. The term 'multi-cell' is used for upsets affecting multiple cells of a memory, whatever correction words those cells happen to fall in. 'Multi-bit' is used when multiple bits in a single correction word are in error.

Soft errors in combinational logic

The three natural masking effects in combinational logic that determine whether a single event upset (SEU) will propagate to become a soft error are electrical masking, logical masking, and temporal (or timing-window) masking. An SEU is *logically masked* if its propagation is blocked from reaching an output latch because off-path gate inputs prevent a logical transition of that gate's output. An SEU is *electrically masked* if the signal is attenuated by the electrical properties of gates on its propagation path such that the resulting pulse is of insufficient magnitude to be reliably latched. An SEU is *temporally masked* if the erroneous pulse reaches an output latch, but it does not occur close enough to when the latch is actually triggered to hold.

If all three masking effects fail to occur, the propagated pulse becomes latched and the output of the logic circuit will be an erroneous value. In the context of circuit operation, this erroneous output value may be considered a soft error event. However, from a microarchitectural-level standpoint, the affected result may not change the output of the currently-executing program. For instance, the erroneous data could be overwritten before use, masked in subsequent logic operations, or simply never be used. If erroneous data does not affect the output of a program, it is considered to be an example of *microarchitectural masking*.

Soft error rate

Soft error rate (SER) is the rate at which a device or system encounters or is predicted to encounter soft errors. It is typically expressed as either number of failures-in-time (FIT), or mean time between failures (MTBF). The unit adopted for quantifying failures in time is called FIT, equivalent to 1 error per billion hours of device operation. MTBF is usually given in years of device operation. To put it into perspective, a one-year MTBF equals to approximately 114,077 FIT (approximately $\frac{1{,}000{,}000{,}000}{24 \times 365.25}$).

While many electronic systems have an MTBF that exceeds the expected lifetime of the circuit, the SER may still be unacceptable to the manufacturer or customer. For instance, many failures per million circuits due to soft errors can be expected in the field if the system does not have adequate soft error protection. The failure of even a few products in the field, particularly if catastrophic, can tarnish the reputation of the product and company that designed it. Also,

in safety- or cost-critical applications where the cost of system failure far outweighs the cost of the system itself, a 1% chance of soft error failure per lifetime may be too high to be acceptable to the customer. Therefore, it is advantageous to design for low SER when manufacturing a system in high-volume or requiring extremely high reliability.

References

[1] Gold (1995): "This letter is to inform you and congratulate you on another remarkable scientific prediction of the future; namely your foreseeing of the dynamic random-access memory (DRAM) logic upset problem caused by alpha particle emission, first observed in 1977, but written about by you in Caves of Steel in 1957." [Note: Actually, 1952.] ... "These failures are caused by trace amounts of radioactive elements present in the packaging material used to encapsulate the silicon devices ... in your book, Caves of Steel, published in the 1950s, you use an alpha particle emitter to 'murder' one of the robots in the story, by destroying ('randomizing') its positronic brain. This is, of course, as good a way of describing a logic upset as any I've heard ... our millions of dollars of research, culminating in several international awards for the most important scientific contribution in the field of reliability of semiconductor devices in 1978 and 1979, was predicted in substantially accurate form twenty years [Note: twenty-five years, actually] before the events took place

[2] Franco, L., Gómez, F., Iglesias, A., Pardo, J., Pazos, A., Pena, J., Zapata, M., SEUs on commercial SRAM induced by low energy neutrons produced at a clinical linac facility, RADECS Proceedings, Sept. 2005

Further reading

- Ziegler, J. F.; Lanford, W. A. (1979). "Effect of Cosmic Rays on Computer Memories". *Science* **206** (4420): 776–788. doi: 10.1126/science.206.4420.776 (http://dx.doi.org/10.1126/science.206.4420.776). ISSN 0036-8075 (http://www.worldcat.org/issn/0036-8075).
- Mukherjee, S, "Architecture Design for Soft Errors," Elsevier, Inc., Feb. 2008.
- Mukherjee, S, "Computer Glitches from Soft Errors: A Problem with Multiple Solutions," Microprocessor Report, May 19, 2008.

External links

- Book on "Architecture Design for Soft Errors" by Shubu Mukherjee, published by Elsevier, Inc. (http://www.amazon.com/dp/0123695295) Book review by Max Baron of Microprocessor Report (May 27, 2008), “Dr. Shubu Mukherjee’s book is a welcome surprise: books by architecture leaders in major companies are few and far between. Written from the viewpoint of a working engineer, the book describes sources of soft errors and solutions involving device, logic, and architecture design to reduce the effects of soft errors”
- Ionizing Radiation Effects in MOS Devices and Circuits by Tso Ping Ma and PAUL V. Dressendorfer (http://www.borders.com.au/book/ionizing-radiation-effects-in-mos-devices-and-circuits/2733970/), The first comprehensive overview describing the effects of ionizing radiation on MOS devices, as well as how to design, fabricate, and test integrated circuits intended for use in a radiation environment.
- Radiation Effects And Soft Errors In Integrated Circuits And Electronic Devices by Dan Fleetwood and Ron D Schrimpf (http://www.amazon.com/dp/9812389407), Vanderbilt University, Nashville, Tennessee, USA A collection of the most important concepts in Radiation Effects by two pioneers in this field.
- Soft Errors in Electronic Memory - A White Paper (http://www.tezzaron.com/about/papers/soft_errors_1_1_secure.pdf) - A good summary paper with many references - Tezzaron Jan 2004. Concludes that 1000–5000 FIT per Mbit (0.2–1 error per day per Gbyte) is a typical DRAM soft error rate.
- Benefits of Chipkill-Correct ECC for PC Server Main Memory (http://www-1.ibm.com/servers/eserver/pseries/campaigns/chipkill.pdf) - A 1997 discussion of SDRAM reliability - some interesting information on "soft errors" from cosmic rays, especially with respect to Error-correcting code schemes
- Soft errors' impact on system reliability (http://www.edn.com/article/CA454636.html) - Ritesh Mastipuram and Edwin C Wee, Cypress Semiconductor, 2004
- Scaling and Technology Issues for Soft Error Rates (http://www.nepp.nasa.gov/DocUploads/40D7D6C9-D5AA-40FC-829DC2F6A71B02E9/Scal-00.pdf) - A Johnston - 4th Annual Research Conference

on Reliability Stanford University, October 2000
- Evaluation of LSI Soft Errors Induced by Terrestrial Cosmic rays and Alpha Particles (http://www.rcnp.osaka-u.ac.jp/~annurep/2001/genkou/sec3/kobayashi.pdf) - H. Kobayashi, K. Shiraishi, H. Tsuchiya, H. Usuki (all of Sony), and Y. Nagai, K. Takahisa (Osaka University), 2001.
- SELSE Workshop Website (http://www.selse.org/) - Website for the workshop on the System Effects of Logic Soft Errors
- Effects of Ionizing Radiation at Books Free Delivery (http://www.booksfreedelivery.com/book/US/9789211422634/Effects_of_Ionizing_Radiation_United_Nations_Scientific_Committee_on_the_Effects_of_Atomic_Radiation_Unscear_2006_Re - Effects of ionizing radiation: United Nations Scientific Committee on the effects of atomic radiation: Unscear 2006 Report, Report ot the General Ass

Two pass verification

Two-pass verification, also called **double data entry**, is a data entry quality control method that was originally employed when data records were entered onto sequential 80-column Hollerith cards with a keypunch. In the first pass through a set of records, the data keystrokes were entered onto each card as the data entry operator typed them. On the second pass through the batch, an operator at a separate machine, called a *verifier*, entered the same data. The verifier compared the second operator's keystrokes with the contents of the original card. If there were no differences, a verification notch was punched on the right edge of the card.

The later IBM 129 keypunch also could operate as a verifier. In that mode, it read a completed card (record) and loaded the 80 keystrokes into a buffer. A data entry operator reentered the record and the keypunch compared the new keystrokes with those loaded into the buffer. If a discrepancy occurred the operator was given a chance to reenter that keystroke and ultimately overwrite the entry in the buffer. If all keystrokes matched the original card, it was passed through and received a verification punch. If corrections were required, then the operator was prompted to discard the original card and insert a fresh card on which corrected keystrokes were typed. The corrected record (card) was passed through and received a corrected verification punch.

Modern use

While this method of quality control clearly is not proof against systematic errors or operator misread entries from a source document, it is very useful in catching and correcting random miskeyed strokes which occur even with experienced data entry operators. However, it proved to be a fatally tragic flaw in the Therac 25 incident. This method has survived the keypunch and is available in some currently available data entry programs (e.g. PSPP/SPSS Data Entry). At least one study suggests that single-pass data entry with range checks and skip rules approaches the reliability of two-pass data entry;[1] however, it is desirable to implement both systems in a data entry application.

References

[1] Controlled Clinical Trials from sometime in the 1990s - Control Clin Trials. 1998 Feb; 19(1):15-24.?

Validation rule

A **Validation rule** is a criterion used in the process of data validation, carried out after the data has been encoded onto an input medium and involves a data vet or validation program. This is distinct from formal verification, where the operation of a program is determined to be that which was intended, and that meets the purpose.

The method is to check that data falls the appropriate parameters defined by the systems analyst. A judgement as to whether data is valid is made possible by the validation program, but it cannot ensure complete accuracy. This can only be achieved through the use of all the clerical and computer controls built into the system at the design stage. The difference between data validity and accuracy can be illustrated with a trivial example. A company has established a Personnel file and each record contains a field for the Job Grade. The permitted values are A, B, C, or D. An entry in a record may be valid and accepted by the system if it is one of these characters, but it may not be the correct grade for the individual worker concerned. Whether a grade is correct can only be established by clerical checks or by reference to other files. During systems design, therefore, data definitions are established which place limits on what constitutes valid data. Using these data definitions, a range of software validation checks can be carried out.

Criteria

An example of a validation check is the procedure used to verify an ISBN.[1]

- Size. The number of characters in a data item value is checked; for example, an ISBN must consist of 10 characters only (in the previous version—the standard for 1997 and later has been changed to 13 characters.)
- Format checks. Data must conform to a specified format. Thus, the first 9 characters must be the digits 0 through 9' the 10th must be either those digits or an *X*
- Consistency. Codes in the data items which are related in some way can thus be checked for the consistency of their relationship. The first number of the ISBN designates the language of publication. for example, books published in French-speaking countries carry the digit "2". This must match the address of the publisher, as given elsewhere in the record. .
- Range. Does not apply to ISBN, but typically data must lie within maximum and minimum preset values. For example, customer account numbers may be restricted within the values 10000 to 20000, if this is the arbitrary range of the numbers used for the system.
- Check digit. An extra digit calculated on, for example, an account number, can be used as a self-checking device. When the number is input to the computer, the validation program carries out a calculation similar to that used to generate the check digit originally and thus checks its validity. This kind of check will highlight transcription errors where two or more digits have been transposed or put in the wrong order. The 10th character of the 10-character ISBN is the check digit.

Test Notes----

References

[1] *Frequently Asked Questions about the new ISBN standard* (http://www.lac-bac.gc.ca/iso/tc46sc9/isbn.htm) ISO.

- *Information integrity : a structure for its definition and management.* Becker, Hal B. New York : McGraw-Hill, 1983. ISBN 0070041911.
- ValidationRule Class (http://msdn2.microsoft.com/en-us/library/microsoft.visualstudio.testtools.webtesting.validationrule(VS.80).aspx) at Microsoft
- Validation Rule Property (http://docs.codecharge.com/studio31/html/index.html?http://docs.codecharge.com/studio31/html/Components/Properties/ValidationRule.html) at CodeCharge Studio

- Create a validation rule to validate data in a field (http://office.microsoft.com/en-us/access/HA100963121033.aspx) at Microsoft

XML

Semi-structured data

Semi-structured data[1] is a form of structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. Therefore, it is also known as self-describing structure.

In the semi-structured data, the entities belonging to the same class may have different attributes even though they are grouped together, and the attributes' order is not important.

Semi-structured data is increasingly occurring since the advent of the Internet where full-text documents and databases are not the only forms of data any more and different applications need a medium for exchanging information. In object-oriented databases, one often finds semi-structured data.

Types of Semi-structured data

XML

XML,[2] other markup languages, email, and EDI are all forms of semi-structured data. OEM (Object Exchange Model) [3] was created prior to XML as a means of self-describing a data structure. XML has been popularized by web services that are developed utilizing SOAP principles.

Some types of data described here as "semi-structured", especially XML, suffer from the impression that they are incapable of structural rigor at the same functional level as Relational Tables and Rows. Indeed, the view of XML as inherently semi-structured (previously, it was referred to as "unstructured") has handicapped its use for a widening range of data-centric applications. Even documents, normally thought of as the epitome of semi-structure, can be designed with virtually the same rigor as database schema, enforced by the XML schema and processed by both commercial and custom software programs without reducing their usability by human readers.

In view of this fact, XML might be referred to as having "flexible structure" capable of human-centric flow and hierarchy as well as highly rigorous element structure and data typing.

The concept of XML as "human-readable", however, can only be taken so far. Some implementations/dialects of XML, such as the XML representation of the contents of a Microsoft Word document, as implemented in Office 2007 and later versions, utilize dozens or even hundreds of different kinds of tags that reflect a particular problem domain - in Word's case, formatting at the character and paragraph and document level, definitions of styles, inclusion of citations, etc. - which are nested within each other in complex ways. Understanding even a portion of such an XML document by reading it, let alone catching errors in its structure, is impossible without a very deep prior understanding of the specific XML implementation, along with assistance by software that understands the XML schema that has been employed. Such text is not "human-understandable" any more than a book written in Swahili (which uses the Latin alphabet) would be to an American or Western European who does not know a word of that language: the tags are symbols that are meaningless to a person unfamiliar with the domain.

JSON

JSON or JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML. JSON has been popularized by web services developed utilizing REST principles.

There is a new breed of databases such as MongoDB and Couchbase that store data natively in JSON format, leveraging the pros of semi-structured data architecture.

Pros and Cons of Using a Semi-structured Data Format

Pros

- Programmers persisting objects from their application to a database do not need to worry about object-relational impedance mismatch, but can often serialize objects via a light-weight library.
- Support for nested or hierarchical data often simplifies data models representing complex relationships between entities.
- Support for lists of objects simplifies data models by avoiding messy translations of lists into a relational data model.

Cons

- The traditional relational data model has a ready-made query language, SQL, which is not applicable to semi-structured data without the extensive use of non-standard features.
- Prone to "garbage in, garbage out"; by removing restraints from the data model, there is less fore-thought that is necessary to operate a data application.

References

[1] Tutorial on semi-structured data by Peter Buneman from Symposium on Principles of Database Systems, 1997 (http://www.cis.upenn.edu/~db/abstracts/semistructured.html)

[2] The Penn database group has semi-structured and XML data project (http://db.cis.upenn.edu/research/SS_XML.html)

[3] Stanford Universities Lore DBMS (http://infolab.stanford.edu/lore/home/index.html)

Semi-structured model

The **semi-structured** model is a database model where there is no separation between the data and the schema, and the amount of structure used depends on the purpose.

The advantages of this model are the following:

- It can represent the information of some data sources that cannot be constrained by schema.
- It provides a flexible format for data exchange between different types of databases.
- It can be helpful to view structured data as semi-structured (for browsing purposes).
- The schema can easily be changed.
- The data transfer format may be portable.

The primary trade-off being made in using a semi-structured database model is that queries cannot be made as efficient as in a more constrained structure, such as in the relational model. Typically the records in a semi-structured database are stored with unique IDs that are referenced with pointers to their location on disk. This makes navigational or path-based queries quite efficient, but for doing searches over many records (as is typical in SQL), it is not as efficient because it has to seek around the disk following pointers.

The Object Exchange Model (OEM) is one standard to express semi-structured data, another way is XML.

Standard Generalized Markup Language

Standard Generalized Generic Markup Language

Filename extension	`.sgml`
Internet media type	`application/sgml`, `text/sgml`
Uniform Type Identifier	public.xmlWikipedia:Please clarify
Developed by	ISO
Type of format	Markup Language
Extended from	GML
Extended to	HTML, XML
Standard(s)	ISO 8879 [1]

The **Standard Generalized Markup Language** (**SGML**; ISO 8879:1986) is an ISO-standard technology for defining generalized markup languages for documents. ISO 8879 Annex A.1 defines generalized markup:

> Generalized markup is based on two novel postulates:

- Markup should be declarative: it should describe a document's structure and other attributes, rather than specify the processing to be performed on it. Declarative markup is less likely to conflict with unforeseen future processing needs and techniques.
- Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and databases can be used for processing documents as well.

  HTML was theoretically an example of an SGML-based language until HTML 5, which admits that browsers can't parse it as SGML (for compatibility reasons) and codifies exactly what they must do instead.

  DocBook SGML and LinuxDoc are better examples, as they were used almost exclusively with actual SGML tools.

Standard versions

SGML is an ISO standard: "ISO 8879:1986 Information processing — Text and office systems — Standard Generalized Markup Language (SGML)", of which there are three versions:

- Original *SGML*, which was accepted in October 1986, followed by a minor Technical Corrigendum.
- *SGML (ENR)*, in 1996, resulted from a Technical Corrigendum to add *extended naming rules* allowing arbitrary-language and -script markup.
- *SGML (ENR+WWW or WebSGML)*, in 1998, resulted from a Technical Corrigendum [2] to better support XML and WWW requirements.

SGML is part of a trio of enabling ISO standards for electronic documents developed by ISO/IEC JTC1/SC34 (ISO/IEC Joint Technical Committee 1, Subcommittee 34 – Document description and processing languages) :

- SGML (ISO 8879)—generalized markup language
  - SGML was reworked in 1998 into XML, a successful profile of SGML. Full SGML is rarely found or used in new projects.
- DSSSL (ISO/IEC 10179)—document processing and styling language based on Scheme.
  - DSSSL was reworked intoWikipedia:Please clarify W3C XSLT and XSL-FO which use an XML syntax. Nowadays, DSSSL is rarely used in new projects apart from Linux documentation.
- HyTime—Generalized hypertext and scheduling.[3]
  - HyTime was partially reworked into W3C XLink. HyTime is rarely used in new projects.

SGML is supported by various technical reports, in particular

- ISO/IEC TR 9573 – Information processing – SGML support facilities – Techniques for using SGML
  - Part 13: Public entity sets for mathematics and science
    - In 2007, the W3C MathML working group agreed to assume the maintenance of these entity sets.

History

SGML descended from IBM's Generalized Markup Language (GML), which Charles Goldfarb, Edward Mosher, and Raymond Lorie developed in the 1960s. Goldfarb, editor of the international standard, coined the “GML” term using their surname initials. The syntax of SGML is closer to the COCOA format.Wikipedia:Please clarify As a document markup language, SGML was originally designed to enable the sharing of machine-readable large-project documents in government, law, and industry. Many such documents must remain readable for several decades—a long time in the information technology field. SGML also was extensively applied by the military, and the aerospace, technical reference, and industrial publishing industries. The advent of the XML profile has made SGML suitable for widespread application for small-scale, general-purpose use.

Document validity

A fragment of the OED (1985), showing SGML markup

SGML (ENR+WWW) defines two kinds of validity. According to the revised Terms and Definitions of ISO 8879 (from the public draft[4]):

A conforming SGML document must be either a type-valid SGML document, a tag-valid SGML document, or both. Note: A user may wish to enforce additional constraints on a document, such as whether a document instance is integrally-stored or free of entity references.

A **type-valid** SGML document is defined by the standard as

An SGML document in which, for each document instance, there is an associated document type declaration (DTD) to whose DTD that instance conforms.

A **tag-valid** SGML document is defined by the standard as

An SGML document, all of whose document instances are fully tagged. There need not be a document type declaration associated with any of the instances. Note: If there is a document type declaration, the instance can be parsed with or without reference to it.

Terminology

Tag-validity was introduced in SGML (ENR+WWW) to support XML which allows documents with no DOCTYPE declaration but which can be parsed without a grammar or documents which have a DOCTYPE declaration that makes no XML Infoset contributions to the document. The standard calls this *fully tagged*. *Integrally stored* reflects the XML requirement that elements end in the same entity in which they started. *Reference-free* reflects the HTML requirement that entity references are for special characters and do not contain markup. SGML validity commentary, especially commentary that was made before 1997 or that is unaware of SGML (ENR+WWW), covers *type-validity* only.

The SGML emphasis on validity supports the requirement for generalized markup that *markup should be rigorous.* (ISO 8879 A.1)

Syntax

An SGML document may have three parts:

1. the SGML Declaration,
2. the Prologue, containing a DOCTYPE declaration with the various *markup declarations* that together make a Document Type Definition (DTD), and
3. the instance itself, containing one top-most element and its contents.

An SGML document may be composed from many entities (discrete pieces of text). In SGML, the entities and element types used in the document may be specified with a DTD, the different character sets, features, delimiter sets, and keywords are specified in the SGML Declaration to create the *concrete syntax* of the document.

Although full SGML allows implicit markup and some other kinds of tags, the XML specification (s4.3.1) states:

Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A

document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup.

For introductory information on a basic, modern SGML syntax, see XML. The following material concentrates on features not in XML and is not a comprehensive summary of SGML syntax.

Optional features

SGML generalizes and supports a wide range of markup languages as found in the mid 1980s. These ranged from terse Wiki-like syntaxes to RTF-like bracketed languages to HTML-like matching-tag languages. SGML did this by a relatively simple default *reference concrete syntax* augmented with a large number of optional features that could be enabled in the SGML Declaration. Not every SGML parser can necessarily process every SGML document. Because each processor's *System Declaration* can be compared to the document's *SGML Declaration* it is always possible to know whether a document is supported by a particular processor.

Many SGML features relate to markup minimization. Other features relate to parallel asynchronous markup (CONCUR), to linking processing attributes (LINK), and to embedding SGML documents within SGML documents (SUBDOC).

The notion of customizable features was not appropriate for Web use, so one goal of XML was to minimize optional features. However XML's well-formedness rules cannot support Wiki-like languages, leaving them unstandardized and difficult to integrate with non-text information systems.

Concrete and abstract syntaxes

The usual (default) SGML *concrete syntax* resembles this example, which is the default HTML concrete syntax:

```
<QUOTE TYPE="example">
  typically something like <ITALICS>this</ITALICS>
</QUOTE>
```

SGML provides an *abstract syntax* that can be implemented in many different types of *concrete syntax*. Although the markup norm is using angle brackets as start- and end- tag delimiters in an SGML document (per the standard-defined *reference concrete syntax*), it is possible to use other characters—provided a suitable *concrete syntax* is defined in the document's SGML declaration. For example, an SGML interpreter might be programmed to parse GML, wherein the tags are delimited with a left colon and a right full stop, thus, an *:e* prefix denotes an end tag: `:xmp.Hello, world:exmp.`. According to the reference syntax, letter-case (upper- or lower-) is not distinguished in tag names, thus the three tags: (i) `<quote>`, (ii) `<QUOTE>`, and (iii) `<quOtE>` are equivalent. (*NOTE:* A concrete syntax might *change* this rule via the NAMECASE NAMING declarations).

Markup Minimization

SGML has features for reducing the number of characters required to mark up a document, which must be enabled in the SGML Declaration. SGML processors need not support every available feature, thus allowing applications to tolerate many types of inadvertent markup omissions; however, SGML systems usually are intolerant of invalid structures. XML is intolerant of syntax omissions, and does not require a DTD for validation.

OMITTAG

Both start tags and end tags may be omitted from a document instance, provided:

1. the OMITTAG feature is enabled in the SGML Declaration,
2. the DTD indicates that the tags are permitted to be omitted,
3. (for start tags) the element has no associated required (`#REQUIRED`) attributes, and
4. the tag can be unambiguously inferred by context.

For example, if OMITTAG YES is specified in the SGML Declaration (enabling the OMITTAG feature), and the DTD includes the following declarations:

```
<!ELEMENT chapter - - (title, section+)>
<!ELEMENT title o o (#PCDATA)>
<!ELEMENT section - - (title, subsection+)>
```

then this excerpt:

```
<chapter>Introduction to SGML
<section>The SGML Declaration
<subsection>
...
```

which omits two `<title>` tags and two `</title>` tags, would represent valid markup.

Note also that omitting tags is optional - the same excerpt could be tagged like this:

```
<chapter><title>Introduction to SGML</title>
<section><title>The SGML Declaration</title>
<subsection>
...
```

and would still represent valid markup.

Note: The OMITTAG feature is unrelated to the tagging of elements whose declared content is `EMPTY` as defined in the DTD:

```
<!ELEMENT image - o EMPTY>
```

Elements defined like this have no end tag, and specifying one in the document instance would result in invalid markup. This is syntactically different than XML empty elements in this regard.

SHORTREF

Tags can be replaced with delimiter strings, for a terser markup, via the SHORTREF feature. This markup style is now associated with wiki markup, e.g. wherein two equals-signs (==), at the start of a line, are the "heading start-tag", and two equals signs (==) after that are the "heading end-tag".

SHORTTAG

SGML markup languages whose concrete syntax enables the SHORTTAG VALUE feature, do not require attribute values containing only alphanumeric characters to be enclosed within quotation marks—either double " " (LIT) or single ' ' (LITA)—so that the previous markup example could be written:

```
<QUOTE TYPE=example>
  typically something like <ITALICS>this</>
</QUOTE>
```

One feature of SGML markup languages is the "presumptuous empty tagging", such that the empty end tag </> in <ITALICS>this</> "inherits" its value from the nearest previous full start tag, which, in this example, is <ITALICS> (in other words, it closes the most recently opened item). The expression is thus equivalent to <ITALICS>this</ITALICS>.

NET

Another feature is the *NET* (Null End Tag) construction: <ITALICS/this/, which is structurally equivalent to <ITALICS>this</ITALICS>.

Other features

Additionally, the SHORTTAG NETENABL IMMEDNET feature allows shortening tags surrounding an empty text value, but forbids shortening full tags:

```
<QUOTE></QUOTE>
```

can be written as:

```
<QUOTE// <!-- not a typo! -->
```

Wherein the first slash (/) stands for the NET-enabling "start-tag close" (NESTC), and the second slash stands for the NET. **NOTE:** XML defines NESTC with a **/**, and NET with an **>** (angled bracket)—hence the corresponding construct in XML appears as **<QUOTE/>**.

The third feature is 'text on the same line', allowing a markup item to be ended with a line-end; especially useful for headings and such, requiring using either SHORTREF or DATATAG minimization. For example, if the DTD includes the following declarations:

```
<!ELEMENT lines (line*)
<!ELEMENT line O - (#PCDATA)>
<!ENTITY   line-tagc  "</line>">
<!SHORTREF one-line "&#RE;&#RS;" line-tagc>
<!USEMAP   one-line line>
```

(and "&#RE;&#RS;" is a short-reference delimiter in the concrete syntax), then:

```
<lines>
first line
second line
```

```
</lines>
```

is equivalent to:

```
<lines>
<line>first line</line>
<line>second line</line>
</lines>
```

Formal characterization

SGML has many features that defied convenient description with the popular formal automata theory and the contemporary parser technology of the 1980s and the 1990s. The standard warns in Annex H:

> The SGML *model group* notation was deliberately designed to resemble the regular expression notation of automata theory, because automata theory provides a theoretical foundation for some aspects of the notion of conformance to a content model. No assumption should be made about the general applicability of automata to content models.

A report on an early implementation of a parser for basic SGML, the Amsterdam SGML Parser, notes

> the DTD-grammar in SGML must conform to a notion of unambiguity which closely resembles the LL(1) conditions

and specifies various differences.

There appears to be no definitive classification of full SGML against a known class of formal grammar. Plausible classes may include tree-adjoining grammars and adaptive grammars.

XML is described as being generally parsable like a two-level grammar for non-validated XML and a Conway-style pipeline of coroutines (lexer, parser, validator) for valid XML. The SGML productions in the ISO standard are reported to be LL(3) or LL(4). XML-class subsets are reported to be expressible using a W-grammar. According to one paper, and probably considered at an *information set* or parse tree level rather than a character or delimiter level:

> The class of documents that conform to a given SGML document grammar forms an LL(1) language. ...
> The SGML document grammars by themselves are, however, not LL(1) grammars.

The SGML standard does not define SGML with formal data structures, such as parse trees, however, an SGML document is constructed of a rooted directed acyclic graph (RDAG) of physical storage units known as “entities”, which is parsed into a RDAG of structural units known as “elements”. The physical graph is loosely characterized as an *entity tree*, but entities might appear multiple times. Moreover, the structure graph is also loosely characterized as an *element tree*, but the ID/IDREF markup allows arbitrary arcs.

The results of parsing can also be understood as a data tree in different notations; where the document is the root node, and entities in other notations (text, graphics) are child nodes. SGML provides apparatus for linking to and annotating external non-SGML entities.

The SGML standard describes it in terms of *maps* and *recognition modes* (s9.6.1). Each entity, and each element, can have an associated *notation* or *declared content type*, which determines the kinds of references and tags which will be recognized in that entity and element. Also, each element can have an associated *delimiter map* (and *short reference map*), which determines which characters are treated as delimiters in context. The SGML standard characterizes parsing as a state machine switching between recognition modes. During parsing, there is a stack of maps that configure the scanner, while the tokenizer relates to the recognition modes.

Parsing involves traversing the dynamically-retrieved entity graph, finding/implying tags and the element structure, and validating those tags against the grammar. An unusual aspect of SGML is that the grammar

(DTD) is used both passively — to *recognize* lexical structures, and actively — to *generate* missing structures and tags that the DTD has declared optional. End- and start- tags can be omitted, because they can be inferred. Loosely, a series of tags can be omitted only if there is a single, possible path in the grammar to imply them. It was this active use of grammars that made concrete SGML parsing difficult to formally characterize.

SGML uses the term *validation* for both recognition and generation. XML does not use the grammar (DTD) to change delimiter maps or to inform the parse modes, and does not allow tag omission; consequently, XML validation of elements is not active in the sense that SGML validation is active. SGML *without* a DTD (e.g. simple XML), is a grammar or a language; SGML *with* a DTD is a metalanguage. SGML with an SGML declaration is, perhaps, a meta-metalanguage, since it is a metalanguage whose declaration mechanism *is* a metalanguage.

SGML has an abstract syntax implemented by many possible concrete syntaxes, however, this is not the same usage as in an abstract syntax tree and as in a concrete syntax tree. In the SGML usage, a concrete syntax is a set of specific delimiters, while the abstract syntax is the set of names for the delimiters. The XML Infoset corresponds more to the programming language notion of abstract syntax introduced by John McCarthy.

Derivatives

XML

The W3C XML (Extensible Markup Language) is a profile (subset) of SGML designed to ease the implementation of the parser compared to a full SGML parser, primarily for use on the World Wide Web. In addition to disabling many SGML options present in the reference syntax (such as omitting tags and nested subdocuments) XML adds a number of additional restrictions on the kinds of SGML syntax. For example, despite enabling SGML shortened tag forms, XML does not allow unclosed start or end tags. It also relied on many of the additions made by the WebSGML Annex. XML currently is more widely used than full SGML. XML has lightweight internationalization based on Unicode. Applications of XML include XHTML, XQuery, XSLT, XForms, XPointer, JSP, SVG, RSS, Atom, XML-RPC, RDF/XML, and SOAP.

HTML

While HTML was developed partially independently and in parallel with SGML, its creator Tim Berners-Lee, intended it to be an application of SGML. The design of HTML (Hyper Text Markup Language) was therefore inspired by SGML tagging, but, since no clear expansion and parsing guidelines were established, most actual HTML documents are not valid SGML documents. Later, HTML was reformulated (version 2.0) to be more of an SGML application, however, the HTML markup language has many legacy- and exception- handling features that differ from SGML's requirements. HTML 4 is an SGML application that fully conforms to ISO 8879 – SGML.

The charter for the recently revived World Wide Web Consortium HTML Working Group says, "the Group will not assume that an SGML parser is used for 'classic HTML'". Although HTML syntax closely resembles SGML syntax with the default *reference concrete syntax*, HTML5 abandons any attempt to define HTML as an SGML application, explicitly defining its own parsing rules [5] which more closely match existing implementations and documents. It does, however, define an alternative XHTML serialization, which conforms to XML and therefore to SGML as well.

OED

The second edition of the *Oxford English Dictionary* (OED) is entirely marked up with an SGML-based markup language.

The third edition is marked up as XML.

Others

Other document markup languages are partly related to SGML and XML, but — because they cannot be parsed or validated or other-wise processed using standard SGML and XML tools — they are not considered either SGML or XML languages; the Z Format markup language for typesetting and documentation is an example.

Several modern programming languages support tags as primitive token types, or now support Unicode and regular expression pattern-matching. An example is the Scala programming language.

Applications

Document markup languages defined using SGML are called "applications" by the standard; many pre-XML SGML applications were proprietary property of the organizations which developed them, and thus unavailable in the World Wide Web. The following list is of pre-XML SGML applications.

- TEI (Text Encoding Initiative) is an academic consortium that designs, maintains, and develops technical standards for digital-format textual representation applications.
- DocBook is a markup language originally created as an SGML application, designed for authoring technical documentation; DocBook currently is an XML application.
- CALS (Continuous Acquisition and Life-cycle Support) is a US Department of Defense (DoD) initiative for electronically capturing military documents and for linking related data and information.
- EDGAR (Electronic Data-Gathering, Analysis, and Retrieval) system effects automated collection, validation, indexing, acceptance, and forwarding of submissions, by companies and others, who are legally required to file data and information forms with the US Securities and Exchange Commission (SEC).
- LinuxDoc. Documentation for Linux packages has used the LinuxDoc SGML DTD and Docbook XML DTD.

Open source implementations

Significant open source implementations of SGML have included:

- ASP-SGML [6]
- ARC-SGML [7], by Standard Generalized Markup Language Users', 1991, C language
- SGMLS [8], by James Clark, 1993, C language
- Project YAO [9], by Yuan-ze Institute of Technology, Taiwan, with Charles Goldfarb, 1994, object
- SP [10] by James Clark, C++ language

SP and Jade, the associated DSSSL processors, are maintained by the OpenJade [11] project, and are common parts of Linux distributions. A general archive of SGML software and materials resides at SUNET [12]. The original HTML parser class, in Sun System's implementation of Java, is a limited-features SGML parser, using SGML terminology and concepts.

References

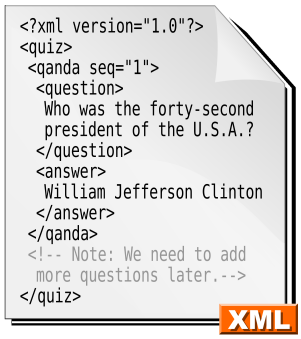
[1] http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387
[2] http://www1.y12.doe.gov/capabilities/sgml/wg8/document/1929.htm
[3] ISO/IEC 10744 (http://www1.y12.doe.gov/capabilities/sgml/wg8/document/n1920/pdf/n1920.pdf) – Hytime (http://www.hytime.org/)
[4] Terms and Definitions of ISO 8879 draft (http://www1.y12.doe.gov/capabilities/sgml/wg8/document/1929.htm)
[5] http://dev.w3.org/html5/spec/parsing.html
[6] http://ftp.sunet.se/pub/text-processing/sgml/ASP-SGML/
[7] http://ftp.sunet.se/pub/text-processing/sgml/ARC-SGML/
[8] http://ftp.sunet.se/pub/text-processing/sgml/SGMLS/
[9] http://ftp.sunet.se/pub/text-processing/sgml/YAO/
[10] http://www.jclark.com/sp/index.htm
[11] http://openjade.sourceforge.net/
[12] http://ftp.sunet.se/pub/text-processing/sgml/

External links

- Overview of SGML Resources (http://www.w3.org/MarkUp/SGML/) at W3C's website.
- Introduction and Examples of Software Documentation in SGML (http://www.the-software-experts.de/e_dta-sw-doku-principles.htm)
- SC34 Committee Records (http://purl.umn.edu/42431), Charles Babbage Institute – Collection on the development of **SGML** and other standards influential in the development of current XML tools; documents include early drafts of SGML administrative materials, documentation, working group papers, and standards for computer languages.
- SGML Syntax Summary by Charles Goldfarb (http://xml.coverpages.org/sgmlsyn/sgmlsyn.htm#C6)
- SGML document introducing you to SGML (http://www.users.cloud9.net/~bradmcc/WhatIsSGML.html); Some reasons why SGML is important (http://www.users.cloud9.net/~bradmcc/sgmlnote.html)
- The SGML Declaration (http://www.is-thought.co.uk/book/sgml-4.htm#Fig4-2), in SGML and HTML Explained, Martin Bryan (1997)
- SGML Declarations (http://xml.coverpages.org/wlw11.html) Wayne Wohler, IBM Corporation, 1994.
- ISO 9069:1988 – Information processing – SGML support facilities – SGML Document Interchange Format (SDIF) (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16643)
- ISO/IEC 9070:1991 – Information technology – SGML support facilities – Registration procedures for public text owner identifiers (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16645)

XML

XML



Filename extension	`.xml`
Internet media type	`application/xml` `text/xml`
Uniform Type Identifier	public.xml
UTI conforms to	public.text
Developed by	World Wide Web Consortium
Type of format	Markup language
Extended from	SGML
Extended to	Numerous, including: XHTML, RSS, Atom, KML
Standard(s)	1.0 (Fifth Edition) [1] November 26, 2008 1.1 (Second Edition) [2] August 16, 2006
Open format?	Yes

Extensible Markup Language (XML)

Current status	Published
Year started	1996
Editors	Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan
Related standards	XML Schema
Domain	Data Serialization
Abbreviation	XML
Website	XML 1.0 [3]

Extensible Markup Language (**XML**) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all free open standards.

The design goals of XML emphasize simplicity, generality, and usability over the Internet. It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services.

Many application programming interfaces (APIs) have been developed to aid software developers with processing XML data, and several schema systems exist to aid in the definition of XML-based languages.

Applications of XML

As of 2009[4], hundreds of document formats using XML syntax have been developed, including RSS, Atom, SOAP, and XHTML. XML-based formats have become the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork. XML has also been employed as the base language for communication protocols, such as XMPP. Applications for the Microsoft .NET Framework use XML files for configuration. Apple has an implementation of a registry based on XML.[5]

XML has come into common use for the interchange of data over the Internet. RFC 3023 gives rules for the construction of Internet Media Types for use when sending XML. It also defines the media types *application/xml* and *text/xml*, which say only that the data are in XML, and nothing about its semantics. The use of *text/xml* has been criticized as a potential source of encoding problems and it has been suggested that it should be deprecated.

RFC 3023 also recommends that XML-based languages be given media types ending in *+xml*; for example *image/svg+xml* for SVG.

Further guidelines for the use of XML in a networked context may be found in RFC 3470, also known as IETF BCP 70; this document is very wide-ranging and covers many aspects of designing and deploying an XML-based language.

Key terminology

The material in this section is based on the XML Specification. This is not an exhaustive list of all the constructs that appear in XML; it provides an introduction to the key constructs most often encountered in day-to-day use.

(Unicode) character

> By definition, an XML document is a string of characters. Almost every legal Unicode character may appear in an XML document.

Processor and application

> The *processor* analyzes the markup and passes structured information to an *application*. The specification places requirements on what an XML processor must do and not do, but the application is outside its scope. The processor (as the specification calls it) is often referred to colloquially as an *XML parser*.

Markup and content

> The characters making up an XML document are divided into *markup* and *content*, which may be distinguished by the application of simple syntactic rules. Generally, strings that constitute markup either begin with the character `<` and end with a `>`, or they begin with the character `&` and end with a `;`. Strings of characters that are not markup are content. However, in a CDATA section, the delimiters `<![CDATA[` and `]]>` are classified as markup, while the text between them is classified as content. In addition, whitespace before and after the outermost element is classified as markup.

Tag

> A markup construct that begins with `<` and ends with `>`. Tags come in three flavors:

- *start-tags*; for example: `<section>`
- *end-tags*; for example: `</section>`
- *empty-element tags*; for example: `<line-break />`

Element

A logical document component which either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag. The characters between the start- and end-tags, if any, are the element's *content*, and may contain markup, including other elements, which are called *child elements*. An example of an element is `<Greeting>Hello, world.</Greeting>` (see hello world). Another is `<line-break />`.

Attribute

A markup construct consisting of a name/value pair that exists within a start-tag or empty-element tag. In the example (below) the element *img* has two attributes, *src* and *alt*:

```
<img src="madonna.jpg" alt='Foligno Madonna, by Raphael' />
```

Another example would be

```
<step number="3">Connect A to B.</step>
```

where the name of the attribute is "number" and the value is "3".

An XML attribute can only have a single value and each attribute can appear at most once on each element. In the common situation where a list of multiple values is desired, this must be done by encoding the list into a well-formed XML attribute[6] with some format beyond what XML defines itself. Usually this is either a comma or semi-colon delimited list or, if the individual values are known not to contain spaces,[7] a space-delimited list can be used.

```
<div class="inner greeting-box" >Hello!</div>
```

where the attribute "class" has both the value "inner greeting-box", indicating the CSS class names "inner" and "greeting-box".

XML declaration

XML documents may begin by declaring some information about themselves, as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Characters and escaping

XML documents consist entirely of characters from the Unicode repertoire. Except for a small number of specifically excluded control characters, any character defined by Unicode may appear within the content of an XML document.

XML includes facilities for identifying the *encoding* of the Unicode characters that make up the document, and for expressing characters that, for one reason or another, cannot be used directly.

Valid characters

Unicode code points in the following ranges are valid in XML 1.0 documents:

- U+0009, U+000A, U+000D: these are the only C0 controls accepted in XML 1.0;
- U+0020–U+D7FF, U+E000–U+FFFD: this excludes *some* (not all) non-characters in the BMP (all surrogates, U+FFFE and U+FFFF are forbidden);
- U+10000–U+10FFFF: this includes *all* code points in supplementary planes, including non-characters.

XML 1.1 extends the set of allowed characters to include all the above, plus the remaining characters in the range U+0001–U+001F. At the same time, however, it restricts the use of C0 and C1 control characters other than U+0009, U+000A, U+000D, and U+0085 by requiring them to be written in escaped form (for example U+0001 must be written as &#x01; or its equivalent). In the case of C1 characters, this restriction is a backwards incompatibility; it was introduced to allow common encoding errors to be detected.

The code point U+0000 is the only character that is not permitted in any XML 1.0 or 1.1 document.

Encoding detection

The Unicode character set can be encoded into bytes for storage or transmission in a variety of different ways, called "encodings". Unicode itself defines encodings that cover the entire repertoire; well-known ones include UTF-8 and UTF-16. There are many other text encodings that predate Unicode, such as ASCII and ISO/IEC 8859; their character repertoires in almost every case are subsets of the Unicode character set.

XML allows the use of any of the Unicode-defined encodings, and any other encodings whose characters also appear in Unicode. XML also provides a mechanism whereby an XML processor can reliably, without any prior knowledge, determine which encoding is being used. Encodings other than UTF-8 and UTF-16 will not necessarily be recognized by every XML parser.

Escaping

XML provides *escape* facilities for including characters which are problematic to include directly. For example:

- The characters "<" and "&" are key syntax markers and may *never* appear in content outside a CDATA section.[8]
- Some character encodings support only a subset of Unicode. For example, it is legal to encode an XML document in ASCII, but ASCII lacks code points for Unicode characters such as "é".
- It might not be possible to type the character on the author's machine.
- Some characters have glyphs that cannot be visually distinguished from other characters: examples are
  - non-breaking space (` `) " "

    compare space (`&#x20;`) " "
  - Cyrillic Capital Letter A (`&#x410;`) "А"

    compare Latin Capital Letter A (`&#x41;`) "A"

There are five predefined entities:

- `<` represents "<"
- `>` represents ">"
- `&` represents "&"
- `&apos;` represents '
- `&quot;` represents "

All permitted Unicode characters may be represented with a *numeric character reference*. Consider the Chinese character "中", whose numeric code in Unicode is hexadecimal 4E2D, or decimal 20,013. A user whose keyboard offers no method for entering this character could still insert it in an XML document encoded either as `&#20013;` or `&#x4e2d;`. Similarly, the string "`I <3 Jörg`" could be encoded for inclusion in an XML document as "`I <3 J&#xF6;rg`".

"`&#0;`" is not permitted, however, because the null character is one of the control characters excluded from XML, even when using a numeric character reference. An alternative encoding mechanism such as Base64 is needed to represent such characters.

Comments

Comments may appear anywhere in a document outside other markup. Comments cannot appear before the XML declaration. Comments start with "`<!--`" and end with "`-->`". The string "`--`" (double-hyphen) is not allowed inside comments; this means comments cannot be nested. The ampersand has no special significance within comments, so entity and character references are not recognized as such, and there is no way to represent characters outside the character set of the document encoding.

An example of a valid comment: "<!-- no need to escape <code> & such in comments -->"

International use

XML 1.0 (Fifth Edition) and XML 1.1 support the direct use of almost any Unicode character in element names, attributes, comments, character data, and processing instructions (other than the ones that have special symbolic meaning in XML itself, such as the less-than sign, "<"). The following is a well-formed XML document including both Chinese and Cyrillic characters:

```
<?xml version="1.0" encoding="UTF-8"?>
<俄语>данные</俄语>
```

Well-formedness and error-handling

The XML specification defines an XML document as a well-formed text – meaning that it satisfies a list of syntax rules provided in the specification. Some key points in the fairly lengthy list include:

- The document contains only properly encoded legal Unicode characters
- None of the special syntax characters such as < and & appear except when performing their markup-delineation roles
- The begin, end, and empty-element tags that delimit the elements are correctly nested, with none missing and none overlapping
- The element tags are case-sensitive; the beginning and end tags must match exactly. Tag names cannot contain any of the characters !"#$%&'()*+,/;<=>?@[\]^`{|}~, nor a space character, and cannot start with -, ., or a numeric digit.
- A single "root" element contains all the other elements

The definition of an *XML document* excludes texts that contain violations of well-formedness rules; they are simply not XML. An XML processor that encounters such a violation is required to report such errors and to cease normal processing. This policy, occasionally referred to as "draconian error handling," stands in notable contrast to the behavior of programs that process HTML, which are designed to produce a reasonable result even in the presence of severe markup errors. XML's policy in this area has been criticized as a violation of Postel's law ("Be conservative in what you send; be liberal in what you accept").

The XML specification defines a **valid XML document** as a well-formed XML document which also conforms to the rules of a Document Type Definition (DTD). By extension, the term can also refer to documents that conform to rules in other schema languages, such as XML Schema (XSD). This term should not be confused with a **well-formed XML document**, which is defined as an XML document that has correct XML syntax according to W3C standards.

Schemas and validation

In addition to being well-formed, an XML document may be *valid*. This means that it contains a reference to a Document Type Definition (DTD), and that its elements and attributes are declared in that DTD and follow the grammatical rules for them that the DTD specifies.

XML processors are classified as *validating* or *non-validating* depending on whether or not they check XML documents for validity. A processor that discovers a validity error must be able to report it, but may continue normal processing.

A DTD is an example of a *schema* or *grammar*. Since the initial publication of XML 1.0, there has been substantial work in the area of schema languages for XML. Such schema languages typically constrain the set of elements that may be used in a document, which attributes may be applied to them, the order in which they may appear, and the allowable parent/child relationships.

Document Type Definition

The oldest schema language for XML is the Document Type Definition (DTD), inherited from SGML.

DTDs have the following benefits:

- DTD support is ubiquitous due to its inclusion in the XML 1.0 standard.
- DTDs are terse compared to element-based schema languages and consequently present more information in a single screen.
- DTDs allow the declaration of standard public entity sets for publishing characters.
- DTDs define a *document type* rather than the types used by a namespace, thus grouping all constraints for a document in a single collection.

DTDs have the following limitations:

- They have no explicit support for newer features of XML, most importantly namespaces.
- They lack expressiveness. XML DTDs are simpler than SGML DTDs and there are certain structures that cannot be expressed with regular grammars. DTDs only support rudimentary datatypes.
- They lack readability. DTD designers typically make heavy use of parameter entities (which behave essentially as textual macros), which make it easier to define complex grammars, but at the expense of clarity.
- They use a syntax based on regular expression syntax, inherited from SGML, to describe the schema. Typical XML APIs such as SAX do not attempt to offer applications a structured representation of the syntax, so it is less accessible to programmers than an element-based syntax may be.

Two peculiar features that distinguish DTDs from other schema types are the syntactic support for embedding a DTD within XML documents and for defining *entities*, which are arbitrary fragments of text and/or markup that the XML processor inserts in the DTD itself and in the XML document wherever they are referenced, like character escapes.

DTD technology is still used in many applications because of its ubiquity.

XML Schema

A newer schema language, described by the W3C as the successor of DTDs, is XML Schema, often referred to by the initialism for XML Schema instances, XSD (XML Schema Definition). XSDs are far more powerful than DTDs in describing XML languages. They use a rich datatyping system and allow for more detailed constraints on an XML document's logical structure. XSDs also use an XML-based format, which makes it possible to use ordinary XML tools to help process them.

xs:schema element that defines a schema:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
</xs:schema>
```

RELAX NG

RELAX NG was initially specified by OASIS and is now also an ISO/IEC International Standard (as part of DSDL). RELAX NG schemas may be written in either an XML based syntax or a more compact non-XML syntax; the two syntaxes are isomorphic and James Clark's conversion tool - 'Trang [9]', can convert between them without loss of information. RELAX NG has a simpler definition and validation framework than XML Schema, making it easier to use and implement. It also has the ability to use datatype framework plug-ins; a RELAX NG schema author, for example, can require values in an XML document to conform to definitions in XML Schema Datatypes.

Schematron

Schematron is a language for making assertions about the presence or absence of patterns in an XML document. It typically uses XPath expressions.

ISO DSDL and other schema languages

The ISO DSDL (Document Schema Description Languages) standard brings together a comprehensive set of small schema languages, each targeted at specific problems. DSDL includes RELAX NG full and compact syntax, Schematron assertion language, and languages for defining datatypes, character repertoire constraints, renaming and entity expansion, and namespace-based routing of document fragments to different validators. DSDL schema languages do not have the vendor support of XML Schemas yet, and are to some extent a grassroots reaction of industrial publishers to the lack of utility of XML Schemas for publishing.

Some schema languages not only describe the structure of a particular XML format but also offer limited facilities to influence processing of individual XML files that conform to this format. DTDs and XSDs both have this ability; they can for instance provide the infoset augmentation facility and attribute defaults. RELAX NG and Schematron intentionally do not provide these.

Related specifications

A cluster of specifications closely related to XML have been developed, starting soon after the initial publication of XML 1.0. It is frequently the case that the term "XML" is used to refer to XML together with one or more of these other technologies which have come to be seen as part of the XML core.

- XML Namespaces enable the same document to contain XML elements and attributes taken from different vocabularies, without any naming collisions occurring. Although XML Namespaces are not part of the XML specification itself, virtually all XML software also supports XML Namespaces.
- XML Base defines the `xml:base` attribute, which may be used to set the base for resolution of relative URI references within the scope of a single XML element.
- The XML Information Set or *XML infoset* describes an abstract data model for XML documents in terms of *information items*. The infoset is commonly used in the specifications of XML languages, for convenience in describing constraints on the XML constructs those languages allow.
- xml:id Version 1.0 asserts that an attribute named `xml:id` functions as an "ID attribute" in the sense used in a DTD.
- XPath defines a syntax named *XPath expressions* which identifies one or more of the internal components (elements, attributes, and so on) included in an XML document. XPath is widely used in other core-XML specifications and in programming libraries for accessing XML-encoded data.
- XSLT is a language with an XML-based syntax that is used to transform XML documents into other XML documents, HTML, or other, unstructured formats such as plain text or RTF. XSLT is very tightly coupled with XPath, which it uses to address components of the input XML document, mainly elements and attributes.
- XSL Formatting Objects, or XSL-FO, is a markup language for XML document formatting which is most often used to generate PDFs.
- XQuery is an XML-oriented query language strongly rooted in XPath and XML Schema. It provides methods to access, manipulate and return XML, and is mainly conceived as a query language for XML databases.
- XML Signature defines syntax and processing rules for creating digital signatures on XML content.
- XML Encryption defines syntax and processing rules for encrypting XML content.

Some other specifications conceived as part of the "XML Core" have failed to find wide adoption, including XInclude, XLink, and XPointer.

Programming interfaces

The design goals of XML include, "It shall be easy to write programs which process XML documents." Despite this, the XML specification contains almost no information about how programmers might go about doing such processing. The XML Infoset specification provides a vocabulary to refer to the constructs within an XML document, but also does not provide any guidance on how to access this information. A variety of APIs for accessing XML have been developed and used, and some have been standardized.

Existing APIs for XML processing tend to fall into these categories:

- Stream-oriented APIs accessible from a programming language, for example SAX and StAX.
- Tree-traversal APIs accessible from a programming language, for example DOM.
- XML data binding, which provides an automated translation between an XML document and programming-language objects.
- Declarative transformation languages such as XSLT and XQuery.

Stream-oriented facilities require less memory and, for certain tasks which are based on a linear traversal of an XML document, are faster and simpler than other alternatives. Tree-traversal and data-binding APIs typically require the use of much more memory, but are often found more convenient for use by programmers; some include declarative retrieval of document components via the use of XPath expressions.

XSLT is designed for declarative description of XML document transformations, and has been widely implemented both in server-side packages and Web browsers. XQuery overlaps XSLT in its functionality, but is designed more for searching of large XML databases.

Simple API for XML

Simple API for XML (SAX) is a lexical, event-driven interface in which a document is read serially and its contents are reported as callbacks to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

Pull parsing

Pull parsing[10] treats the document as a series of items which are read in sequence using the Iterator design pattern. This allows for writing of recursive-descent parsers in which the structure of the code performing the parsing mirrors the structure of the XML being parsed, and intermediate parsed results can be used and accessed as local variables within the methods performing the parsing, or passed down (as method parameters) into lower-level methods, or returned (as method return values) to higher-level methods. Examples of pull parsers include StAX in the Java programming language, XMLReader in PHP, ElementTree.iterparse in Python, System.Xml.XmlReader in the .NET Framework, and the DOM traversal API (NodeIterator and TreeWalker).

A pull parser creates an iterator that sequentially visits the various elements, attributes, and data in an XML document. Code which uses this iterator can test the current item (to tell, for example, whether it is a start or end element, or text), and inspect its attributes (local name, namespace, values of XML attributes, value of text, etc.), and can also move the iterator to the next item. The code can thus extract information from the document as it traverses it. The recursive-descent approach tends to lend itself to keeping data as typed local variables in the code doing the parsing, while SAX, for instance, typically requires a parser to manually maintain intermediate data within a stack of elements which are parent elements of the element being parsed. Pull-parsing code can be more straightforward to understand and maintain than SAX parsing code.

Document Object Model

The Document Object Model (DOM) is an interface-oriented application programming interface that allows for navigation of the entire document as if it were a tree of node objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

Data binding

Another form of XML processing API is XML data binding, where XML data are made available as a hierarchy of custom, strongly typed classes, in contrast to the generic objects created by a Document Object Model parser. This approach simplifies code development, and in many cases allows problems to be identified at compile time rather than run-time. Example data binding systems include the Java Architecture for XML Binding (JAXB) and XML Serialization in .NET.

XML as data type

XML has appeared as a first-class data type in other languages. The ECMAScript for XML (E4X) extension to the ECMAScript/JavaScript language explicitly defines two specific objects (XML and XMLList) for JavaScript, which support XML document nodes and XML node lists as distinct objects and use a dot-notation specifying parent-child relationships. E4X is supported by the Mozilla 2.5+ browsers (though now deprecated) and Adobe Actionscript, but has not been adopted more universally. Similar notations are used in Microsoft's LINQ implementation for Microsoft .NET 3.5 and above, and in Scala (which uses the Java VM). The open-source xmlsh application, which provides a Linux-like shell with special features for XML manipulation, similarly treats XML as a data type, using the <[ ]> notation. The Resource Description Framework defines a data type `rdf:XMLLiteral` to hold wrapped, canonical XML.

History

XML is an application profile of SGML (ISO 8879).

The versatility of SGML for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the Internet. By the mid-1990s some practitioners of SGML had gained experience with the then-new World Wide Web, and believed that SGML offered solutions to some of the problems the Web was likely to face as it grew. Dan Connolly added SGML to the list of W3C's activities when he joined the staff in 1995; work began in mid-1996 when Sun Microsystems engineer Jon Bosak developed a charter and recruited collaborators. Bosak was well connected in the small community of people who had experience both in SGML and the Web.

XML was compiled by a working group of eleven members,[11] supported by a (roughly) 150-member Interest Group. Technical debate took place on the Interest Group mailing list and issues were resolved by consensus or, when that failed, majority vote of the Working Group. A record of design decisions and their rationales was compiled by Michael Sperberg-McQueen on December 4, 1997. James Clark served as Technical Lead of the Working Group, notably contributing the empty-element "<empty />" syntax and the name "XML". Other names that had been put forward for consideration included "MAGMA" (Minimal Architecture for Generalized Markup Applications), "SLIM" (Structured Language for Internet Markup) and "MGML" (Minimal Generalized Markup Language). The co-editors of the specification were originally Tim Bray and Michael Sperberg-McQueen. Halfway through the project Bray accepted a consulting engagement with Netscape, provoking vociferous protests from Microsoft. Bray was temporarily asked to resign the editorship. This led to intense dispute in the Working

Group, eventually solved by the appointment of Microsoft's Jean Paoli as a third co-editor.

The XML Working Group never met face-to-face; the design was accomplished using a combination of email and weekly teleconferences. The major design decisions were reached in a short burst of intense work between August and November 1996,[12] when the first Working Draft of an XML specification was published. Further design work continued through 1997, and XML 1.0 became a W3C Recommendation on February 10, 1998.

Sources

XML is a profile of an ISO standard SGML, and most of XML comes from SGML unchanged. From SGML comes the separation of logical and physical structures (elements and entities), the availability of grammar-based validation (DTDs), the separation of data and metadata (elements and attributes), mixed content, the separation of processing from representation (processing instructions), and the default angle-bracket syntax. Removed were the SGML declaration (XML has a fixed delimiter set and adopts Unicode as the document character set).

Other sources of technology for XML were the Text Encoding Initiative (TEI), which defined a profile of SGML for use as a "transfer syntax"; and HTML, in which elements were synchronous with their resource, document character sets were separate from resource encoding, the xml:lang attribute was invented, and (like HTTP) metadata accompanied the resource rather than being needed at the declaration of a link. The Extended Reference Concrete Syntax (ERCS) project of the SPREAD (Standardization Project Regarding East Asian Documents) project of the ISO-related China/Japan/Korea Document Processing expert group was the basis of XML 1.0's naming rules; SPREAD also introduced hexadecimal numeric character references and the concept of references to make available all Unicode characters. To support ERCS, XML and HTML better, the SGML standard IS 8879 was revised in 1996 and 1998 with WebSGML Adaptations. The XML header followed that of ISO HyTime.

Ideas that developed during discussion which were novel in XML included the algorithm for encoding detection and the encoding header, the processing instruction target, the xml:space attribute, and the new close delimiter for empty-element tags. The notion of well-formedness as opposed to validity (which enables parsing without a schema) was first formalized in XML, although it had been implemented successfully in the Electronic Book Technology "Dynatext" software; the software from the University of Waterloo New Oxford English Dictionary Project; the RISP LISP SGML text processor at Uniscope, Tokyo; the US Army Missile Command IADS hypertext system; Mentor Graphics Context; Interleaf and Xerox Publishing System.

Versions

There are two current versions of XML. The first (*XML 1.0*) was initially defined in 1998. It has undergone minor revisions since then, without being given a new version number, and is currently in its fifth edition, as published on November 26, 2008. It is widely implemented and still recommended for general use.

The second (*XML 1.1*) was initially published on February 4, 2004, the same day as XML 1.0 Third Edition, and is currently in its second edition, as published on August 16, 2006. It contains features (some contentious) that are intended to make XML easier to use in certain cases. The main changes are to enable the use of line-ending characters used on EBCDIC platforms, and the use of scripts and characters absent from Unicode 3.2. XML 1.1 is not very widely implemented and is recommended for use only by those who need its unique features.

Prior to its fifth edition release, XML 1.0 differed from XML 1.1 in having stricter requirements for characters available for use in element and attribute names and unique identifiers: in the first four editions of XML 1.0 the characters were exclusively enumerated using a specific version of the Unicode standard (Unicode 2.0 to Unicode 3.2.) The fifth edition substitutes the mechanism of XML 1.1, which is more future-proof but reduces redundancy. The approach taken in the fifth edition of XML 1.0 and in all editions of XML 1.1 is that only certain characters are forbidden in names, and everything else is allowed, in order to accommodate the use of suitable name characters in future versions of Unicode. In the fifth edition, XML names may contain characters in the Balinese, Cham, or Phoenician scripts among many others which have been added to Unicode since Unicode 3.2.

Almost any Unicode code point can be used in the character data and attribute values of an XML 1.0 or 1.1 document, even if the character corresponding to the code point is not defined in the current version of Unicode. In character data and attribute values, XML 1.1 allows the use of more control characters than XML 1.0, but, for "robustness", most of the control characters introduced in XML 1.1 must be expressed as numeric character references (and #x7F through #x9F, which had been allowed in XML 1.0, are in XML 1.1 even required to be expressed as numeric character references). Among the supported control characters in XML 1.1 are two line break codes that must be treated as whitespace. Whitespace characters are the only control codes that can be written directly.

There has been discussion of an XML 2.0, although no organization has announced plans for work on such a project. XML-SW (SW for skunkworks), written by one of the original developers of XML,[13] contains some proposals for what an XML 2.0 might look like: elimination of DTDs from syntax, integration of namespaces, XML Base and XML Information Set (*infoset*) into the base standard.

The World Wide Web Consortium also has an XML Binary Characterization Working Group doing preliminary research into use cases and properties for a binary encoding of the XML infoset. The working group is not chartered to produce any official standards. Since XML is by definition text-based, ITU-T and ISO are using the name *Fast Infoset* for their own binary infoset to avoid confusion (see ITU-T Rec. X.891 | ISO/IEC 24824-1).

Criticism

XML and its extensions have regularly been criticized for verbosity and complexity.[14] Mapping the basic tree model of XML to type systems of programming languages or databases can be difficult, especially when XML is used for exchanging highly structured data between applications, which was not its primary design goal. Other criticisms attempt to refute the claim that XML is a self-describing language (though the XML specification itself makes no such claim). JSON, YAML, and S-Expressions are frequently proposed as alternatives (see Comparison of data serialization formats);[15] which focus on representing highly structured data rather than documents, which may contain both highly structured and relatively unstructured content.

Notes

[1] http://www.w3.org/TR/2008/REC-xml-20081126/

[2] http://www.w3.org/TR/2006/REC-xml11-20060816/

[3] http://www.w3.org/TR/rec-xml

[4] http://en.wikipedia.org/w/index.php?title=XML&action=edit

[5] appleexaminer.com: "PLIST files" (http://www.appleexaminer.com/MacsAndOS/Analysis/PLIST/PLIST.html)

[6] i.e., embedded quote characters would be a problem

[7] A common example of this would be for CSS class or identifier names.

[8] It is allowed, but not recommended, to use "<" in XML entity values: Extensible Markup Language (XML) 1.0 (Fifth Edition): EntityValue definition (http://www.w3.org/TR/2008/REC-xml-20081126/#NT-AttValue)

[9] http://www.thaiopensource.com/relaxng/trang.html

[10] Push, Pull, Next! (http://www.xml.com/pub/a/2005/07/06/tr.html) by Bob DuCharme, at XML.com

[11] The working group was originally called the "Editorial Review Board." The original members and seven who were added before the first edition was complete, are listed at the end of the first edition of the XML Recommendation, at http://www.w3.org/TR/1998/REC-xml-19980210.

[12] Jon Bosak: The Birth of XML (http://java.sun.com/xml/birth_of_xml.html)

[13] Tim Bray: *Extensible Markup Language - SW (XML-SW)* (http://www.textuality.com/xml/xmlSW.html). 2002-02-10

[14] Jeff Atwood (2009): XML: The Angle Bracket Tax (http://www.codinghorror.com/blog/2008/05/xml-the-angle-bracket-tax.html)

[15] Stackoverflow: What usable alternatives to XML syntax do you know? (http://stackoverflow.com/questions/51492/what-usable-alternatives-to-xml-syntax-do-you-know)

References

Further reading

- Annex A of ISO 8879:1986 (SGML)
- Lawrence A. Cunningham (2005). "Language, Deals and Standards: The Future of XML Contracts". *Washington University Law Review*. SSRN 900616 (http://ssrn.com/abstract=900616).
- Bosak, Jon; Tim Bray (May 1999). "XML and the Second-Generation Web". *Scientific American*. Online at XML and the Second-Generation Web (http://www.scientificamerican.com/article.cfm?id=xml-and-the-second-genera).
- Kelly, Sean (February 6, 2006). "Making Mistakes with XML" (http://www.developer.com/xml/article.php/10929_3583081_1). *Developer.com*. Retrieved 2010-10-26.
- St. Laurent, Simon (February 12, 2003). "Five years later, XML.." (http://www.oreillynet.com/xml/blog/2003/02/five_years_later_xml.html). *O'Reilly XML Blog*. O'Reilly Media. Retrieved 2010-10-26.
- "W3C XML is Ten!" (http://www.w3.org/2008/02/xml10-pressrelease). World Wide Web Consortium. 12 February 2008. Retrieved 2010-10-26.
- "Introduction to XML" (http://wam.inrialpes.fr/courses/PG-MoSIG12/xml.pdf). *Course Slides*. Pierre Geneves. October 2012.

External links

- W3C XML homepage (http://www.w3.org/XML/)
- XML 1.0 Specification (http://www.w3.org/TR/REC-xml)
- Retrospective on Extended Reference Concrete Syntax (http://xml.ascc.net/en/utf-8/ercsretro.html) by Rick Jelliffe
- *XML, Java and the Future of the Web* (http://www.xml.com/pub/a/w3j/s3.bosak.html) (1997) by Jon Bosak
- http://validator.w3.org/ (http://validator.w3.org/) The Official [W3C] Markup Validation Service

XML database

An **XML database** is a data persistence software system that allows data to be stored in XML format. These data can then be queried, exported and serialized into the desired format. XML databases are usually associated with document-oriented databases.

Two major classes of XML database exist:

1. **XML-enabled**: these may either map XML to traditional database structures (such as a relational database[1]), accepting XML as input and rendering XML as output, or more recently support native XML types within the traditional database. This term implies that the database processes the XML itself (as opposed to relying on middleware).
2. **Native XML** (NXD): the internal model of such databases depends on XML and uses XML documents as the fundamental unit of storage, which are, however, not necessarily stored in the form of text files.

Rationale for XML in databases

O'Connell gives one reason for the use of XML in databases: the increasingly common use of XML for data transport, which has meant that "data is extracted from databases and put into XML documents and vice-versa".[2] It may prove more efficient (in terms of conversion costs) and easier to store the data in XML format. In content-based applications, the ability of the native XML database also minimizes the need for extraction or entry of metadata to support searching and navigation. In a native XML environment, the entire content store becomes metadata through query languages such as XPath and XQuery, including content, attributes and relationships within the XML (find string "XABr" within element <para> containing attribute 123 having value "P" or "Q", only within parent Y and siblings F or G.) While this level of search capability is possible in external metadata, it requires more complex and difficult processing to reproduce the content tree in metadata.

XML Enabled databases

XML enabled databases typically offer one or more of the following approaches to storing XML within the traditional relational structure:

1. XML is stored into a CLOB (Character large object)
2. XML is `shredded` into a series of Tables based on a Schema [3]
3. XML is stored into a native XML Type as defined by the ISO[4]

RDBMS that support the ISO XML Type are:

1. IBM DB2 (pureXML[5])
2. Microsoft SQL Server [6]
3. Oracle Database [7]
4. PostgreSQL [8]

Typically an XML enabled database is best suited where the majority of data are non-XML, for datasets where the majority of data are XML a Native XML Database is better suited.

Example of XML Type Query in IBM DB2 SQL

```
select
   id, vol, xmlquery('$j/name', passing journal as "j") as name
from
   journals
where
```

```
xmlexists('$j[publisher="Elsevier"]', passing journal as "j")
```

Native XML databases

The term "native XML database" (NXD) can lead to confusion. Many NXDs do not function as standalone databases at all, and do not really store the native (text) form.

The formal definition from the XML:DB initiative (which appears to be inactive since 2003) states that a native XML database:

- Defines a (logical) model for an XML document — as opposed to the data in that document — and stores and retrieves documents according to that model. At a minimum, the model must include elements, attributes, PCDATA, and document order. Examples of such models include the XPath data model, the XML Infoset, and the models implied by the DOM and the events in SAX 1.0.
- Has an XML document as its fundamental unit of (logical) storage, just as a relational database has a row in a table as its fundamental unit of (logical) storage.
- Need not have any particular underlying physical storage model. For example, NXDs can use relational, hierarchical, or object-oriented database structures, or use a proprietary storage format (such as indexed, compressed files).

Additionally, many XML databases provide a logical model of grouping documents, called "collections". Databases can set up and manage many collections at one time. In some implementations, a hierarchy of collections can exist, much in the same way that an operating system's directory-structure works.

All XML databases now[9] support at least one form of querying syntax. Minimally, just about all of them support XPath for performing queries against documents or collections of documents. XPath provides a simple pathing system that allows users to identify nodes that match a particular set of criteria.

In addition to XPath, many XML databases support XSLT as a method of transforming documents or query-results retrieved from the database. XSLT provides a declarative language written using an XML grammar. It aims to define a set of XPath filters that can transform documents (in part or in whole) into other formats including plain text, XML, or HTML.

Many XML databases also support XQuery to perform querying. XQuery includes XPath as a node-selection method, but extends XPath to provide transformational capabilities. Users sometimes refer to its syntax as "FLWOR" (pronounced 'Flower') because the query may include the following clauses: 'for', 'let', 'where', 'order by' and 'return'. Traditional RDBMS vendors (who traditionally had SQL-only engines), are now shipping with hybrid SQL and XQuery engines. Hybrid SQL/XQuery engines help to query XML data alongside the relational data, in the same query expression. This approach helps in combining relational and XML data.

Most XML Databases support a common vendor neutral API called the XQuery API for Java (XQJ). The XQJ API was developed at the JCP as a standard interface to an XML/XQuery data source, enabling a Java developer to submit queries conforming to the World Wide Web Consortium (W3C) XQuery 1.0 specification and to process the results of such queries. Ultimately the XQJ API is to XML Databases and XQuery as the JDBC API is to Relational Databases and SQL.

Language features

Name	License	Native Language	XQuery 3.0	XQuery Update	XQuery Full Text	EXPath Extensions	EXQuery Extensions	XSLT 2.0
BaseX	BSD License	Java	Yes	Yes	Yes	Yes	Yes	Yes
eXist	LGPL License	Java	Partial	Proprietary[10]	Proprietary	Yes	Yes	Yes
MarkLogic Server	Commercial	C++	Partial	Proprietary	Proprietary	No	No	Yes
Sedna	Apache License	C++	No	Yes	Yes	No	No	No

Supported APIs

Name	XQJ	XML:DB	RESTful	RESTXQ	WebDAV
BaseX	Yes	Yes	Yes	Yes	Yes
eXist	Yes	Yes	Yes	Yes	Yes
MarkLogic Server	Yes	No	Yes	Yes	Yes
Sedna	Yes	Yes	No	No	Yes

References

[1] Mustafa Atay and Shiyong Lu, “Storing and Querying XML: An Efficient Approach Using Relational Databases”, ISBN 3-639-11581-3, VDM Verlag, 2009.

[2] O'Connell, S. *Advanced Databases Course Notes*, Southampton, University of Southampton, 2005, 9.2

[3] Creating XMLType Tables and Columns Based on XML Schema (http://docs.oracle.com/cd/B19306_01/appdev.102/b14259/xdb05sto.htm#i1042421)

[4] ISO/IEC 9075-14:2011 (http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=53686)

[5] IBM DB2 pureXML overview -- DB2 as an XML database (http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.xml.doc/doc/c0022308.html)

[6] Using XML in SQL Server (http://msdn.microsoft.com/en-us/library/ms190936(v=sql.90).aspx)

[7] XMLType Operations (http://docs.oracle.com/cd/B19306_01/appdev.102/b14259/xdb04cre.htm)

[8] PostgreSQL - Data Types - XML Type (http://www.postgresql.org/docs/9.0/static/datatype-xml.html)

[9] http://en.wikipedia.org/w/index.php?title=XML_database&action=edit

[10] XQuery Update Extension (http://www.exist-db.org/exist/apps/doc/update_ext.xml)

External links

- XML Databases - The Business Case, Charles Foster, June 2008 (http://www.cfoster.net/articles/xmldb-business-case) - Talks about the current state of Databases and data persistence, how the current Relational Database model is starting to crack at the seams and gives an insight into a strong alternative for today's requirements.
- An XML-based Database of Molecular Pathways (2005-06-02) (http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-3717) Speed / Performance comparisons of eXist, X-Hive, Sedna and Qizx/open
- XML Native Database Systems: Review of Sedna, Ozone, NeoCoreXMS (http://swing.felk.cvut.cz/index.php?option=com_docman&task=doc_view&gid=5&Itemid=62) 2006
- XML Data Stores: Emerging Practices (http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/ic/&toc=comp/mags/ic/2005/02/w2toc.xml&DOI=10.1109/MIC.2005.48)
- Bhargava, P.; Rajamani, H.; Thaker, S.; Agarwal, A. (2005) *XML Enabled Relational Databases*, Texas, The University of Texas at Austin.
- Initiative for XML Databases (http://xmldb-org.sourceforge.net)

- XML and Databases, Ronald Bourret, September 2005 (http://www.rpbourret.com/xml/XMLAndDatabases.htm)
- The State of Native XML Databases, Elliotte Rusty Harold, August 13, 2007 (http://cafe.elharo.com/xml/the-state-of-native-xml-databases/)
- Comparing XML database approaches (http://www.ibm.com/developerworks/library/x-comparexmldb/)

XML Schema Language comparison

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntax constraints imposed by XML itself. There are several different languages available for specifying an XML schema. Each language has its strengths and weaknesses.

Note: the W3C defined schema language is called "XML Schema". However, this name can be confusing in the context of referring to a number of **XML schema languages**. As such, throughout this document, references to the term "XML schema" will be any XML schema language where the meaning might be ambiguous, while the term "W3C XML Schema" (referred to in this article as WXS) will be used for the W3C-defined XML schema language.

Overview

Though there are a number of schema languages available, the primary three languages are Document Type Definitions, W3C XML Schema, and RELAX NG. Each language has its own advantages and disadvantages.

This article also covers a brief review of other schema languages.

The primary purpose of a schema language is to specify what the structure of an XML document can be. This means which elements can reside in which other elements, which attributes are and are not legal to have on a particular element, and so forth. A schema is somewhat equivalent to a grammar for a language; a schema defines what the vocabulary for the language may be and what a valid "sentence" is.

Document Type Definitions

Tool Support

DTDs are perhaps the most widely supported schema language for XML. Because DTDs are one of the earliest schema languages for XML, defined before XML even had namespace support, they are widely supported. Internal DTDs are often supported in XML processors; external DTDs are less often supported, but only slightly. Most large XML parsers, ones that support multiple XML technologies, will provide support for DTDs as well.

W3C XML Schema

Advantages over DTDs

Compared to DTDs, W3C XML Schemas are exceptionally powerful. They provide much greater specificity than DTDs could. They are namespace aware, and provide support for types.

W3C XML Schema is written in XML itself, and therefore has a schema of its own (appropriately, written in W3C XML Schema).

W3C XML Schema has a large number of built-in and derived data types. These are specified by the W3C XML Schema specification, so all W3C XML Schema validators and processors must support them.

Due to the nature of the schema language, after an XML document is validated, the entire XML document, both content and structure, can be expressed in terms of the schema itself. This functionality, known as Post-Schema-Validation Infoset (PSVI), can be used to transform the document into a hierarchy of typed objects that can be accessed in a programming language through a neutral interface.

Commonality with RELAX NG

RELAX NG and W3C XML Schema allow for similar mechanisms of specificity. Both allow for a degree of modularity in their languages, going so far as to being able to split the schema into multiple files. And both of them are, or can be, defined in an XML language.

Advantages over RELAX NG

RELAX NG does not have any analog to PSVI. Unlike W3C XML Schema, RELAX NG was designed so that validation and augmentation (adding type information and default values) are separate (See XML_Schema_(W3C)#Criticism).

W3C XML Schema has a formal mechanism for attaching a schema to an XML document, while RELAX NG intentionally avoids such mechanisms for security and interoperability reasons (See XML_Schema_(W3C)#Criticism).

RELAX NG has no ability to apply default attribute data to an element's list of attributes (i.e., changing the XML info set), while W3C XML Schema does. Again, this design is intentional and is to separate validation and augmentation (See XML_Schema_(W3C)#Criticism).[1]

W3C XML Schema has a rich "simple type" system built in (xs:number, xs:date, etc., plus derivation of custom types), while RELAX NG has an extremely simplistic one because it's meant to use type libraries developed independently of RELAX NG, rather than grow its own. This is seen by some as a disadvantage. In practice it's common for a RELAX NG schema to use the predefined "simple types" and "restrictions" (pattern, maxLength, etc.) of W3C XML Schema.

In W3C XML Schema a specific number or range of repetitions of patterns can be expressed more elegantly than under RELAX NG. For large numbers it's practically not possible to specify at all in RELAX NG.

Disadvantages

W3C XML Schema is complex and hard to learn, although that's partially because it tries to do more than mere validation (see PSVI).

Although being written in XML is an advantage, it is also a disadvantage in some ways. The W3C XML Schema language in particular can be quite verbose, while a DTD can be terse and relatively easily editable.

Likewise, WXS's formal mechanism for associating a document with a schema can pose a potential security problem. For WXS validators that will follow a URI to an arbitrary online location, there is the potential for reading something malicious from the other side of the stream.[2]

W3C XML Schema does not implement most of the DTD ability to provide data elements to a document. While technically a comparative deficiency, it also does not have the problems that this ability can create as well, which makes it a strength.

Although W3C XML Schema's ability to add default attributes to elements is an advantage, it is a disadvantage in some ways as well. It means that an XML file may not be usable in the absence of its schema, even if the document would validate against that schema. In effect, all users of such an XML document must also implement the W3C XML Schema specification, thus ruling out minimalist or older XML parsers. It can also dramatically slow down processing of the document, as the processor must potentially download and process a second XML file (the schema).

Tool Support

WXS support exists in a number of large XML parsing packages. Xerces and the .NET Framework's Base Class Library both provide support for WXS validation.

RELAX NG

RELAX NG provides for most of the advantages that W3C XML Schema does over DTDs.

Advantages over W3C XML Schema

While the language of RELAX NG can be written in XML, it also has an equivalent form that is much more like a DTD, but with greater specifying power. This form is known as the compact syntax. Tools can easily convert between these forms with no loss of features or even commenting. Even arbitrary elements specified between RELAX NG XML elements can be converted into the compact form.

RELAX NG provides very strong support for unordered content. That is, it allows the schema to state that a sequence of patterns may appear in any order.

RELAX NG also allows for non-deterministic content models. What this means is that RELAX NG allows the specification of a sequence like the following:

```
<zeroOrMore>
  <ref name="odd" />
  <ref name="even" />
</zeroOrMore>
<optional>
  <ref name="odd" />
</optional>
```

When the validator encounters something that matches the "odd" pattern, it is unknown whether this is the optional last "odd" reference or simply one in the zeroOrMore sequence without looking ahead at the data. RELAX NG allows this kind of specification. W3C XML Schema requires all of its sequences to be fully deterministic, so mechanisms like the above must be either specified in a different way or omitted altogether.

RELAX NG allows attributes to be treated as elements in content models. In particular, this means that one can provide the following:

```
<element name="some_element">
  <choice>
    <attribute name="has_name">
      <value>false</value>
    </attribute>
    <group>
      <attribute name="has_name">
        <value>true</value>
      </attribute>
      <element name="name"><text /></element>
    </group>
  </choice>
</element>
```

This block states that the element "some_element" must have an attribute named "has_name". This attribute can only take true or false as values, and if it is true, the first child element of the element must be "name", which stores text.

If "name" did not need to be the first element, then the choice could be wrapped in an "interleave" element along with other elements. The order of the specification of attributes in RELAX NG has no meaning, so this block need not be the first block in the element definition.

W3C XML Schema cannot specify such a dependency between the content of an attribute and child elements.

RELAX NG's specification only lists two built-in types (string and token), but it allows for the definition of many more. In theory, the lack of a specific list allows a processor to support data types that are very problem-domain specific.

Most RELAX NG schemas can be algorithmically converted into W3C XML Schemas and even DTDs (except when using RELAX NG features not supported by those languages, as above). The reverse is not true. As such, RELAX NG can be used as a normative version of the schema, and the user can convert it to other forms for tools that do not support RELAX NG.

Disadvantages

Most of RELAX NG's disadvantages are covered under the section on W3C XML Schema's advantages over RELAX NG.

Though RELAX NG's ability to support user-defined data types is useful, it comes at the disadvantage of only having two data types that the user can rely upon. Which, in theory, means that using a RELAX NG schema across multiple validators requires either providing those user-defined data types to that validator or using only the two basic types. In practice however, most RELAX NG processors support the W3C XML Schema set of data types.

Tool Support

RELAX NG's tool support is significant, but it is less widespread than W3C XML Schema. The Mono Project's implementation of the .NET Framework includes a RELAX NG validator. The C library libxml2 provides RELAX NG support as well. Sun Microsystems's Multiple Schema Validator for Java also provides RELAX NG support.

Schematron

Schematron is a fairly unique schema language. Unlike the main three, it defines an XML file's syntax as a list of XPath-based rules. If the document passes these rules, then it is valid.

Advantages

Because of its rule-based nature, Schematron's specificity is very strong. It can require that the content of an element be controlled by one of its siblings. It can also request or require that the root element, regardless of what element that happens to be, have specific attributes. It can even specify required relationships between multiple XML files.

Disadvantages

While Schematron is good at relational constructs, its ability to specify the basic structure of a document, that is, which elements can go where, results in a very verbose schema.

The typical way to solve this is to combine Schematron with RELAX NG or W3C XML Schema. There are several schema processors available for both languages that support this combined form. This allows Schematron rules to specify additional constraints to the structure defined by W3C XML Schema or RELAX NG.

Tool Support

Schematron's reference implementation is actually an XSLT transformation that transforms the Schematron document into an XSLT that validates the XML file. As such, Schematron's potential toolset is any XSLT processor, though libxml2 provides an implementation that does not require XSLT. Sun Microsystems's Multiple Schema Validator for Java has an add-on that allows it to validate RELAX NG schemas that have embedded Schematron rules.

Namespace Routing Language (NRL)

This is not technically a schema language. Its sole purpose is to direct parts of documents to individual schemas based on the namespace of the encountered elements. An NRL is merely a list of XML namespaces and a path to a schema that each corresponds to. This allows each schema to be concerned with only its own language definition, and the NRL file routes the schema validator to the correct schema file based on the namespace of that element.

This XML format is schema-language agnostic and works for just about any schema language.

References

[1] While annotations in RELAX NG can support default attribute values, the RELAX NG specification does not mandate that a validator provide this ability to modify an XML infoset as part of validation. The WXS specification does mandate this behavior. An additional specification associated with RELAX NG does provide this ability. See Relax NG DTD Compatibility (default value) (http://www.oasis-open.org/committees/relax-ng/compatibility.html#default-value).

[2] James Clark (co-creator of RELAX NG). RELAX NG and W3C XML Schema (http://www.imc.org/ietf-xml-use/mail-archive/msg00217.html)

- Comparative Analysis of Six XML Schema Languages (http://pike.psu.edu/publications/sigmod-record-00.pdf) by Dongwon Lee, Wesley W. Chu, In ACM SIGMOD Record, Vol. 29, No. 3, page 76-87, September 2000
- Taxonomy of XML Schema Languages using Formal Language Theory (http://pike.psu.edu/publications/toit05.pdf) by Makoto Murata, Dongwon Lee, Murali Mani, Kohsuke Kawaguchi, In ACM Trans. on Internet Technology (TOIT), Vol. 5, No. 4, page 1-45, November 2005

XML schema

An **XML schema** is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. These constraints are generally expressed using some combination of grammatical rules governing the order of elements, Boolean predicates that the content must satisfy, data types governing the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints.

There are languages developed specifically to express XML schemas. The Document Type Definition (DTD) language, which is native to the XML specification, is a schema language that is of relatively limited capability, but that also has other uses in XML aside from the expression of schemas. Two more expressive XML schema languages in widespread use are XML Schema (with a capital *S*) and RELAX NG.

The mechanism for associating an XML document with a schema varies according to the schema language. The association may be achieved via markup within the XML document itself, or via some external means.

Capitalization

There is some confusion as to when to use the capitalized spelling "Schema" and when to use the lowercase spelling. The lowercase form is a generic term and may refer to any type of schema, including DTD, XML Schema (aka XSD), RELAX NG, or others, and should always be written using lowercase except when appearing at the start of a sentence. The form "Schema" (capitalized) in common use in the XML community always refers to W3C XML Schema.

Validation

The process of checking to see if an XML document conforms to a schema is called validation, which is separate from XML's core concept of syntactic well-formedness. All XML documents must be well-formed, but it is not required that a document be valid unless the XML parser is "validating", in which case the document is also checked for conformance with its associated schema. DTD-validating parsers are most common, but some support W3C XML Schema or RELAX NG as well.

Documents are only considered **valid** if they satisfy the requirements of the schema with which they have been associated. These requirements typically include such constraints as:

- Elements and attributes that must/may be included, and their permitted structure
- The structure as specified by a regular expression syntax
- How character data is to be interpreted, e.g. as a number, a date, a URL, a Boolean, etc.

Validation of an instance document against a schema can be regarded as a conceptually separate operation from XML parsing. In practice, however, many schema validators are integrated with an XML parser.

XML schema languages

- Document Content Description facility for XML, an RDF framework
- Document Definition Markup Language (DDML)
- Document Schema Definition Languages (DSDL)
- Document Structure Description (DSD)
- SGML's Document Type Definition (DTD)
- Namespace Routing Language (NRL)
- OASIS CAM Content Assembly Mechanism
- RELAX NG and its predecessors RELAX and TREX
- Schema for Object-Oriented XML (SOX)
- Schematron
- XML-Data Reduced (XDR)
- ASN.1 XML Encoding Rules (XER)
- XML Schema (WXS or XSD)

References

External links

- Comparing XML Schema Languages (http://www.xml.com/pub/a/2001/12/12/schemacompare.html) by Eric van der Vlist (2001)
- Comparative Analysis of Six XML Schema Languages (http://pike.psu.edu/publications/sigmod-record-00.pdf) by Dongwon Lee, Wesley W. Chu, In ACM SIGMOD Record, Vol. 29, No. 3, page 76-87, September 2000
- Taxonomy of XML Schema Languages using Formal Language Theory (http://pike.psu.edu/publications/toit05.pdf) by Makoto Murata, Dongwon Lee, Murali Mani, Kohsuke Kawaguchi, In ACM Trans. on Internet Technology (TOIT), Vol. 5, No. 4, page 1-45, November 2005
- Application of XML Schema in Web Services Security (http://www.w3.org/2005/05/25-schema/guthula.html) by Sridhar Guthula, W3C Schema Experience Report, May 2005
- March 2009 DEVX article "Taking XML Validation to the Next Level: Introducing CAM" by Michael Sorens (http://www.devx.com/xml/Article/41066)

XML validation

XML validation is the process of checking a document written in XML (eXtensible Markup Language) to confirm that it is both well-formed and also "valid" in that it follows a defined structure. A well-formed document follows the basic syntactic rules of XML, which are the same for all XML documents. A valid document also respects the rules dictated by a particular DTD or XML schema, according to the application-specific choices for those particular .

In addition, extended tools are available such as OASIS CAM standard specification that provide contextual validation of content and structure that is more flexible than basic schema validations.

xmllint is a command line XML tool that can perform XML validation. It can be found in UNIX / Linux environments. An example with the use of this program for validation of a file called *example.xml* is

```
xmllint --valid --noout example.xml
```

References

External links

XML toolkit

- The XML C parser and toolkit of Gnome (http://xmlsoft.org/xmldtd.html) – libxml includes xmllint

Online validators for XML files

- http://www.w3.org/2001/03/webdata/xsv
- http://www.stg.brown.edu/service/xmlvalid/

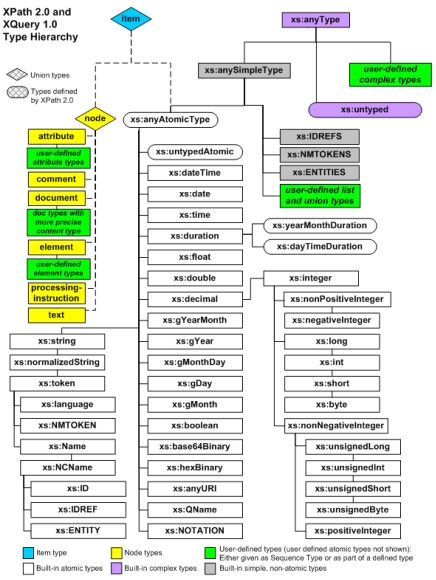
Articles discussing XML validation

- DEVX March, 2009 - Taking XML Validation to the Next Level: Introducing CAM (http://www.devx.com/xml/Article/41066)

Xpath data model

The **XQuery and XPath Data Model** (XDM) is the data model shared by the XPath 2.0, XSLT 2.0 and XQuery programming languages. It is a W3C recommendation and forms an integral part of all three languages. Originally, it was based on the XPath 1.0 data model which in turn is based on the XML Information Set.

The XDM consists of flat sequences of zero or more items of different types. Items can be typed or untyped and include atomic values as well as XML nodes (with elements, attributes and text nodes, etc.). Instances of the XDM can optionally be XML schema-validated.



The XDM type hierarchy

External links

- W3C Recommendation: XQuery 1.0 and XPath 2.0 Data Model [1]
- IBM: XQuery and XPath data model [2]

References

[1] http://www.w3.org/TR/xpath-datamodel/

[2] http://pic.dhe.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=%2Fcom.ibm.db2.luw.xml.doc%2Fdoc%2Fxqrdatam.html

Path expression

In query languages, **path expressions** identify an object by describing how to navigate to it in some graph (possibly implicit) of objects. For example, the path expression `p.Manager.Home.City` might refer the city of residence of someone's manager. Path expressions have been extended to support regular expression-like flexibility. XPath is an example of a path expression language.

In concurrency control, **path expressions** are a mechanism for expressing permitted sequences of execution. For example, a path expression like "`{read}, write`" might specify that either multiple simultaneous executions of `read` or a single execution of `write` but not both are allowed at any point in time.

References

- M. Kifer, W. Kim, and Y. Sagiv (1992). "Querying Object-Oriented Databases". *Proc. of the ACM SIGMOD*. pp. 393–402.
- Elisa Bertino, Mauro Negri, Giuseppe Pelagatti, and Licia Sbattella (June 1992). "Object-Oriented Query Languages: The Notion and the Issues". *IEEE Trans. on Knowledge and Data Engineering* **4** (3): 223–236. doi:10.1109/69.142014 [1].
- R. Campbell and R. Kolstad (1979). "Path Expressions in Pascal". *Proceedings of the 4th International Conference on Software Engineering*. pp. 212–219. Unknown parameter `|Volume=` ignored (`|volume=` suggested) (help)
- Tony Bloom (1979). "Evaluating Synchronization Mechanisms". *Proceedings of the seventh ACM symposium on Operating systems principles*. pp. 24–32.

References

[1] http://dx.doi.org/10.1109%2F69.142014

XQuery

XQuery

Paradigm(s)	declarative, functional, modular
Appeared in	2007
Designed by	W3C
Stable release	1.0 [1] / January 23, 2007
Preview release	3.0 [2] / July 23, 2013
Typing discipline	dynamic or static, strong
Major implementations	Many [3]
Influenced by	XPath, SQL, XSLT
OS	Cross-platform
Usual filename extensions	.xq, .xqy, .xquery
Website	www.w3.org/XML/Query/ [4]
• XQuery at Wikibooks	

XQuery is a query and functional programming language that is designed to query and transform collections of structured and unstructured data, usually in the form of XML, text and with vendor-specific extensions for other data formats (JSON, binary, etc.).

XQuery 3.0 is currently being developed by the XML Query working group of the W3C and has reached Last Call Working Draft status.

XQuery 1.0 was developed by the XML Query working group of the W3C. The work was closely coordinated with the development of XSLT 2.0 by the XSL Working Group; the two groups shared responsibility for XPath 2.0, which is a subset of XQuery 1.0. XQuery 1.0 became a W3C Recommendation on January 23, 2007.

> "The mission of the XML Query project is to provide flexible query facilities to extract data from real and virtual documents on the World Wide Web, therefore finally providing the needed interaction between the Web world and the database world. Ultimately, collections of XML files will be accessed like databases".

Features

XQuery provides the means to extract and manipulate data from XML documents or any data source that can be viewed as XML, such as relational databases or office documents.

XQuery contains a superset of XPath expression syntax to address specific parts of an XML document. It supplements this with a SQL-like "FLWOR expression" for performing joins. A FLWOR expression is constructed from the five clauses after which it is named: FOR, LET, WHERE, ORDER BY, RETURN.

The language also provides syntax allowing new XML documents to be constructed. Where the element and attribute names are known in advance, an XML-like syntax can be used; in other cases, expressions referred to as dynamic node constructors are available. All these constructs are defined as expressions within the language, and can be arbitrarily nested.

The language is based on the XQuery and XPath Data Model (XDM) which uses a tree-structured model of the information content of an XML document, containing seven kinds of nodes: document nodes, elements, attributes, text nodes, comments, processing instructions, and namespaces.

XDM also models all values as sequences (a singleton value is considered to be a sequence of length one). The items in a sequence can either be XML nodes or atomic values. Atomic values may be integers, strings, booleans, and so on: the full list of types is based on the primitive types defined in XML Schema.

XQuery 1.0 does not include features for updating XML documents or databases; it also lacks full text search capability. These features are both under active development for a subsequent version of the language. However, the new standards such as XQuery Update Facility 1.0 supports update feature and XQuery and XPath Full Text 1.0 support full text search in XML documents.

XQuery is a programming language that can express arbitrary XML to XML data transformations with the following features:

1. Logical/physical data independence
2. Declarative
3. High level
4. Side-effect free
5. Strongly typed

Examples

The sample XQuery code below lists the unique speakers in each act of Shakespeare's play Hamlet, encoded in hamlet.xml [5]

```
<html><head/><body>
{
  for $act in doc("hamlet.xml")//ACT
  let $speakers := distinct-values($act//SPEAKER)
  return
    <div>
      <h1>{ string($act/TITLE) }</h1>
      <ul>
      {
        for $speaker in $speakers
        return <li>{ $speaker }</li>
      }
      </ul>
    </div>
}
</body></html>
```

All XQuery constructs for performing computations are expressions. There are no statements, even though some of the keywords appear to suggest statement-like behaviors. To execute a function, the expression within the body is evaluated and its value is returned. Thus to write a function to double an input value, one simply writes:

```
declare function local:doubler($x) { $x * 2 }
```

To write a full query saying 'Hello World', one writes the expression:

```
"Hello World"
```

This style is common in functional programming languages. However, unlike most functional programming languages, XQuery 1.0 doesn't support higher-order functions (they first appear in the drafts for XQuery 3.0).

Applications

Below are a few examples of how XQuery can be used:

1. Extracting information from a database for use in a web service.
2. Generating summary reports on data stored in an XML database.
3. Searching textual documents on the Web for relevant information and compiling the results.
4. Selecting and transforming XML data to XHTML to be published on the Web.
5. Pulling data from databases to be used for the application integration.
6. Splitting up an XML document that represents multiple transactions into multiple XML documents.

XQuery and XSLT compared

Scope

Although XQuery was initially conceived as a query language for large collections of XML documents, it is also capable of transforming individual documents. As such, its capabilities overlap with XSLT, which was designed expressly to allow input XML documents to be transformed into HTML or other formats.

The XSLT 2.0 and XQuery standards were developed by separate working groups within W3C, working together to ensure a common approach where appropriate. They share the same data model (XDM), type system, and function library, and both include XPath 2.0 as a sublanguage.

Origin

The two languages, however, are rooted in different traditions and serve the needs of different communities. XSLT was primarily conceived as a stylesheet language whose primary goal was to render XML for the human reader on screen, on the web (as web template language), or on paper. XQuery was primarily conceived as a database query language in the tradition of SQL.

Because the two languages originate in different communities, XSLT is stronger in its handling of narrative documents with more flexible structure, while XQuery is stronger in its data handling (for example, when performing relational joins).

Versions

XSLT 1.0 appeared as a Recommendation in 1999, whereas XQuery 1.0 only became a Recommendation in early 2007; as a result, XSLT is at present much more widely used. Both languages have similar expressive power, though XSLT 2.0 has many features that are missing from XQuery 1.0, such as grouping, number and date formatting, and greater control over XML namespaces. Many of these features are planned for XQuery 3.0.

Any comparison must take into account the fact that XSLT 1.0 and XSLT 2.0 are very different languages. XSLT 2.0, in particular, has been heavily influenced by XQuery in its move to strong typing and schema-awareness.

Strengths and weaknesses

Usability studies have shown that XQuery is easier to learn than XSLT, especially for users with previous experience of database languages such as SQL.[6] This can be attributed to the fact that XQuery is a smaller language with fewer concepts to learn, and to the fact that programs are more concise. It is also true that XQuery is more orthogonal, in that any expression can be used in any syntactic context. By contrast, XSLT is a two-language system in which XPath expressions can be nested in XSLT instructions but not vice versa.

XSLT is currently stronger than XQuery for applications that involve making small changes to a document (for example, deleting all the NOTE elements). Such applications are generally handled in XSLT by use of a coding pattern that involves an identity template that copies all nodes unchanged, modified by specific templates that modify selected nodes. XQuery has no equivalent to this coding pattern, though in future versions it will be possible to tackle such problems using the update facilities in the language that are under development.

Another facility lacking from XQuery is any kind of mechanism for dynamic binding or polymorphism. The absence of this capability starts to become noticeable when writing large applications, or when writing code that is designed to be reusable in different environments. XSLT offers two complementary mechanisms in this area: the dynamic matching of template rules, and the ability to override rules using `xsl:import`, that make it possible to write applications with multiple customization layers.

The absence of these facilities from XQuery is a deliberate design decision: it has the consequence that XQuery is very amenable to static analysis, which is essential to achieve the level of optimization needed in database query languages. This also makes it easier to detect errors in XQuery code at compile time.

The fact that XSLT 2.0 uses XML syntax makes it rather verbose in comparison to XQuery 1.0. However, many large applications take advantage of this capability by using XSLT to read, write, or modify stylesheets dynamically as part of a processing pipeline. The use of XML syntax also enables the use of XML-based tools for managing XSLT code. By contrast, XQuery syntax is more suitable for embedding in traditional programming languages such as Java[7] or C#. If necessary, XQuery code can also be expressed in an XML syntax called XQueryX. The XQueryX representation of XQuery code is rather verbose and not convenient for humans, but can easily be processed with XML tools, for example transformed with XSLT stylesheets.

Extensions and future work

W3C extensions

Currently, two major extensions to the XQuery are under development by the W3C:

- XQuery 1.0 and XPath 2.0 Full-Text[8]
- XQuery Update Facility

Work on XQuery 3.0 (formerly: 1.1) is well advanced: a Candidate Recommendation was published on 8 January 2013.[9]

A scripting (procedural) extension for XQuery is also being designed.[10]

The EXPath Community Group [11] develops extensions to XQuery and other related standards (XPath, XSLT, XProc, and XForms). There following extensions are currently available:

- Packaging System[12]
- File Module[13]
- Binary Module[14]
- Web Applications[15]

Third-party extensions

JSONiq is an extension of XQuery that adds support to extract and transform data from JSON documents. JSONiq is a superset of XQuery 3.0. It is published under the Creative Commons Attribution-ShareAlike 3.0 license.

The EXQuery project develops standards around creating portable XQuery applications. The following standards are currently available:

- RESTXQ

Further reading

- Querying XML: XQuery, XPath, and SQL/XML in context. Jim Melton and Stephen Buxton. Morgan Kaufmann, 2006. ISBN 1-55860-711-0.
- XQuery. Priscilla Walmsley. O'Reilly Media, 2007. ISBN 0-596-00634-9.
- XQuery: The XML Query Language. Michael Brundage. Addison-Wesley Professional, 2004. ISBN 0-321-16581-0.
- XQuery from the Experts: A Guide to the W3C XML Query Language. Howard Katz (ed). Addison-Wesley, 2004. ISBN 0-321-18060-7
- An Introduction to the XQuery FLWOR [16] Expression. Dr. Michael Kay (W3C XQuery Committee), 2005.

Implementations

- Sirix [17]: XQuery with versioning extensions
- BaseX: BaseX XQuery implementation
- eXist: eXist XQuery implementation
- MarkLogic: MarkLogic XQuery implementation
- RaptorXML Server [18]: Altova's XSLT and XQuery engine
- SAXON: Michael Kay's XSLT and XQuery processor
- XQilla: XQilla xquery implementation
- Zorba: Zorba XQuery and JSONiq processor
- EXPath [19]: XPath/XQuery engines, including a feature matrix
- W3C [3]: XQuery implementations
- SPARQL2XQuery [26]: SPARQL to XQuery translator

References

[1] http://www.w3.org/TR/xquery/
[2] http://www.w3.org/TR/xquery-30/
[3] http://www.w3.org/XML/Query#implementations
[4] http://www.w3.org/XML/Query/
[5] http://www.ibiblio.org/xml/examples/shakespeare/hamlet.xml
[6] Usability of XML Query Languages. Joris Graaumans. SIKS Dissertation Series No 2005-16, ISBN 90-393-4065-X
[7] XQuery API for Java
[8] XQuery and XPath Full Text 1.0 (http://www.w3.org/TR/xquery-full-text/)
[9] XML Query (XQuery) 3.0 (http://www.w3.org/TR/xquery-30/)
[10] XQuery Scripting Extension 1.0 Requirements (http://www.w3.org/TR/xquery-sx-10-requirements/)
[11] EXPath Community Group (http://www.w3.org/community/expath/)
[12] Packaging System (http://expath.org/spec/pkg/20120509)
[13] File Module (http://expath.org/spec/file/20120614)
[14] Binary Module (http://expath.org/spec/binary/20130312)
[15] Web Applications (http://expath.org/spec/webapp/20130401)
[16] http://www.stylusstudio.com/xquery_flwor.html
[17] http://github.com/sirixdb/sirix
[18] http://www.altova.com/raptorxml.html

[19] http://expath.org/wiki/Engines

External links

- W3C XML Query (XQuery) (http://www.w3.org/XML/Query)
- XQuery tutorial (http://www.w3schools.com/xquery/default.asp)
- XQuery API for Java (XQJ) Java Specification Request
- hamlet.xml (http://www.ibiblio.org/xml/examples/shakespeare/hamlet.xml) Hamlet in XML Format
- XQuery (http://www.cafeconleche.org/slides/xmlsig/xquery/index.html) (presentation - as HTML slides)
- FunctX XQuery Functions (http://www.xqueryfunctions.com/xq/) - open source pure XQuery function library, tested with Saxon (http://saxon.sourceforge.net/)
- Discovering XQuery Blog (http://xquery.typepad.com)
- XQuery Development Tools for Eclipse (XQDT) (http://www.xqdt.org/)
- Sirix (http://github.com/sirixdb/sirix/): Beyond XQuery, providing versioning-functions on top of XQuery through Brackit (http://brackit.org).
- BaseX (http://basex.org/): XQuery Update/Full Text implementation with realtime XQuery editor; Live Demo (http://basex.org/products/live-demo/).
- eXist supports an early draft of XQuery Update 1.0: XQuery Update Extensions (http://www.exist-db.org/exist/apps/doc/update_ext.xml)
- XQuery Live Demo of Zorba by the FLWOR-Foundation (http://www.zorba-xquery.com/html/demo)
- SPARQL2XQuery: Translating SPARQL queries to XQuery (http://www.dblab.ntua.gr/~bikakis/SPARQL2XQuery.html)
- XQuery 3.0 Rocks Lightning Talk at FOSDEM 2012 (http://video.fosdem.org/2012/lightningtalks/XQuery_3.0_Rocks.webm)

Portions borrowed with permission from the books "XML Hacks" (O'Reilly Media) and "XQuery" (O'Reilly Media).

Previous version based on an article at the French language Wikipedia

XSA

In computer science, **XSA** (better known as **Cross-Server Attack**) is a networking security intrusion method which allows for a malicious client to compromise security over a website or service on a server by using implemented services on the server that may not be secure.

In general, XSA is demonstrated against websites, yet sometimes it is used in conjunction with other services located on the same server.

Basics

XSA is a method that allows for a malicious client to use services that a remote server implements in order to attack another service on the same server or network.

Most website hosting companies that offer hosting for large or even little amounts of separate websites are vulnerable to this method of attack, because of the amount of access services such as PHP and the webserver itself give to a client that allows the client to access other website configurations, files, passwords and the like.

History

The term 'XSA' was first coined by DeadlyData, a prominent Computer hacker during the early 2000s, over the voice communications software TeamSpeak. While he had not invented or pioneered this method of intrusion, he coined it as a shortened term to describe the act of performing Cross-Server Attacks (XSAs).

It was then used further in the community and now supports for most of the methods and subsets of the method that give both Computer hacker and malicious individuals the terminology to attack websites using software that is located on the same server.

XSIL

XSIL (Extensible Scientific Interchange Language) is an XML-based transport language for scientific data, supporting the inclusion of both in-file data and metadata. The language comes with an extensible Java object model. The language's elementary objects include Param (arbitrary association between a keyword and a value), Array, Table (a set of column headings followed by a set of records), and Stream, which enables one to either encapsulate data inside the XSIL file or point to an external data source.

BFD is an XML dialect based on XSIL.

External links

- XSIL: Extensible Scientific Interchange Language [1]

References

[1] http://authors.library.caltech.edu/28168/

SQL/XML

SQL/XML or **XML-Related Specifications** is an extension to the Structured Query Language (SQL) specification, which defines the use of XML in conjunction with SQL. The XML data type is introduced, as well as several routines, functions, and XML-to-SQL data type mappings to support manipulation and storage of XML in a SQL database.

SQL/XML is defined by **ISO/IEC 9075 Part 14**:

- SQL:2003-14 defined the basic datatypes, mappings, predicates and functions;
- SQL:2006-14, SQL:2008-14, SQL:2011-14 was augmented, as a complementary standard to XQuery .

Specification

The specification defines functions for working with XML, including element construction, mapping data from relational tables, combining XML fragments, and embedding XQuery expressions in SQL statements. Functions which can be embedded include XMLQUERY (which extracts XML or values from an XML field) and XMLEXISTS (which predicates whether an XQuery expression is matched).

Further information and examples of the SQL/XML functions are provided in the external links below.

Standard compliance

The result of Wagner's objective evaluation of the *SQL/XML*:2006 standard compliance of Oracle 11g Release 1, MS SQL Server 2008 and MySQL 5.1.30 is shown in the following table, to which the data for PostgreSQL 9.1,[1][2] and IBM DB2 has been added:

	Oracle 11g Release 1	IBM DB2 9.7	MS SQL Server 2008	MySQL 5.1.30	PostgreSQL 9.1
Datatype XML	partial (Oracle entitles the data type 'XMLType' instead of 'XML')	high	high	no	partial
SQL/XML predicates	high	high	partial	no	partial
SQL/XML functions	high	high	partial	low	high

NOTE: only Oracle, IBM DB2 and MS-SQL-Server have been augmented with XQuery.

Examples

The sample SQLXML query below has SQLXML type as output(tested on DB2 9.7 and Oracle 11g):

```
SELECT XMLELEMENT(NAME "PhoneBook", -- root element name
                  XMLAGG(           -- aggregation over the rows
                  XMLELEMENT(NAME "Contact",

                  XMLATTRIBUTES(cust.FIRST_NAME AS "Name",
                                cust.TEL)
                               )
                        )
                  )
FROM TMP.CUSTOMER AS cust;
```

And the output:

```
<PhoneBook>
    <Contact Name="Daniel" TEL="788255855"/>
    <Contact Name="Martin" TEL="889665447"/>
    <Contact Name="Eva"    TEL="111222333"/>
    <Contact Name="Alena"  TEL="444555666"/>
    <Contact Name="Oliver" TEL="777888999"/>
    <Contact Name="George" TEL="444882446"/>
    <Contact Name="Jamie"  TEL="123456789"/>
</PhoneBook>
```

Samples are taken from javalobby article.

External links

- SQLXML.org [3]
- SQLX.org [4]
- SQL/XML on PostgreSQL [5]

References

[1] PostgreSQL Conformance with ISO 9075-14 (SQL/XML) (http://www.postgresql.org/docs/9.1/static/features.html), at PostgreSQL 9.1 documentation.
[2] PostgreSQL 9.1 XML functions (http://www.postgresql.org/docs/current/static/functions-xml.html), at PostgreSQL 9.1 documentation.
[3] http://sqlxml.org/
[4] http://sqlx.org/
[5] http://www.postgresql.org/docs/current/static/functions-xml.html

Soma File

A specification of modeling architecture (SOMA) file, or **SOMA file** is a proprietary XML-based description of a memory model, created by Denali Software.[1] SOMA files include parameters that configure the timing and protocol requirements for a memory model as specified by the memory vendor. This file is typically encrypted.

SOMA files are read into a proprietary memory modeling object, called an MMAV, that is used in an ASIC simulation environment. This allows the simulation to model any type of memory, if a corresponding SOMA file exists.[2]

Memory vendors typically provide memory models in non-proprietary verilog format and not in SOMA files. For example, Micron memory models [3], and others are readily downloadable from the memory vendors' webpages.

External links

- Soma File [4]
- Free Memory Models [5]
- Micron Memory Model FAQ [6]

References

[1] Soma File (https://www.denali.com/en/ememory)
[2] MMAV (https://www.denali.com/en/products/mmav.jsp)
[3] http://download.micron.com/downloads/models/verilog/sdram/ddr/256meg/256Mb_ddr.zip
[4] https://www.denali.com/en/ememory
[5] http://www.freemodelfoundry.com
[6] http://www.micron.com/faq/answer.aspx?qid=122

Regular Language description for XML

REgular LAnguage description for XML (**RELAX**) is a specification for describing XML-based languages. A description written in RELAX is called a RELAX grammar.

RELAX Core has been approved as an ISO/IEC Technical Report 22250-1 in 2002 (ISO/IEC TR 22250-1:2002). It was developed by ISO/IEC JTC1/SC34 (ISO/IEC Joint Technical Committee 1, Subcommittee 34 - Document description and processing languages).

RELAX was designed by Murata Makoto.

In 2001, an XML schema language RELAX NG was created by unifying of RELAX Core and James Clark's TREX. It was published as ISO/IEC 19757-2 in 2003.

References

External links

- RELAX home page (http://www.xml.gr.jp/relax/)
- ISO/IEC TR 22250-1:2002 - Information technology -- Document description and processing languages -- Regular Language Description for XML (RELAX) -- Part 1: RELAX Core (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34922)

PureXML

pureXML is the native XML storage feature in the IBM DB2 data server. pureXML provides query languages, storage technologies, indexing technologies, and other features to support XML data. The word *pure* in pureXML was chosen to indicate that DB2 natively stores and natively processes XML data in its inherent hierarchical structure, as opposed to treating XML data as plain text or converting it into a relational format.[1]

Technical information

DB2 includes two distinct storage mechanisms: one for efficiently managing traditional SQL data types, and another for managing XML data. The underlying storage mechanism is transparent to users and applications; they simply use SQL (including SQL with XML extensions or SQL/XML) or XQuery to work with the data.

XML data is stored in columns of DB2 tables that have the XML data type. XML data is stored in a parsed format that reflects the hierarchical nature of the original XML data. As such, pureXML uses trees and nodes as its model for storing and processing XML data. If you instruct DB2 to validate XML data against an XML schema prior to storage, DB2 annotates all nodes in the XML hierarchy with information about the schema types; otherwise, it will annotate the nodes with default type information. Upon storage, DB2 preserves the internal structure of XML data, converting its tag names and other information into integer values. Doing so helps conserve disk space and also improves the performance of queries that use navigational expressions. However, users aren't aware of this internal representation. Finally, DB2 automatically splits XML nodes across multiple database pages, as needed.

XML schemas specify which XML elements are valid, in what order these elements should appear in XML data, which XML data types are associated with each element, and so on. pureXML allows you to validate the cells in a column of XML data against no schema, one schema, or multiple schemas. pureXML also provides tools to support evolving XML schemas.

IBM has enhanced its programming language interfaces to support access to its XML data. These enhancements span Java (JDBC), C (embedded SQL and call-level interface), COBOL (embedded SQL), PHP, and Microsoft's .NET Framework (through the DB2.NET provider).

History

pureXML was first included in the DB2 9 for Linux, Unix, and Microsoft Windows release, which was codenamed Viper, in June 2006.[2] It was available on DB2 9 for z/OS in March 2007.[3] In October 2007, IBM released DB2 9.5 with improved XML data transaction performance and improved storage savings.[4] In June 2009, IBM released DB2 9.7 with XML supported for database-partitioned, range-partitioned, and multi-dimensionally clustered tables as well as compression of XML data and indices.[5]

Competition

DB2 is a hybrid data server—it offers data management for traditional relational data, as well as providing native XML data management. Other vendors that offer data management for both relational data and native XML storage include Oracle with its 11g product and Microsoft with its SQL Server product.

pureXML also competes with native XML databases like BaseX, eXist, MarkLogic or Sedna.

User groups

The International DB2 Users Group (IDUG) is an independent, not-for-profit association of IT professionals who use IBM DB2. IDUG provides education, technical resources, peer networking opportunities, online resources and other programs for DB2 users.

Books

IBM International Technical Support Organization (ITSO) has published the following books, which are available in print or as free e-books:

- DB2 9: pureXML Overview and Fast Start [6]
- DB2 9 pureXML Guide [7]

The following books are also available for purchase:

- DB2 pureXML Cookbook: Master the Power of IBM Hybrid Data Server [8]

Education and training

The following pureXML classroom and online courses are available from IBM Education:

- Query and Manage XML Data with DB2 9 [9]. IBM course CG130. Classroom. Duration: 4 days.
- Query XML Data with DB2 9 [10]. IBM course CG100. Classroom. Duration: 2 days (first 2 days of CG130).
- Managing XML Data in DB2 9. IBM course CG160. Classroom. Duration: 2 days (last 2 days of CG130).
- DB2 pureXML [11]. IBM Course CT140. Self-paced study plus Live Virtual Classroom.

References

[1] http://www.ibm.com/developerworks/blogs/page/datastudioteam?entry=purexml_and_purequery_what_s
[2] http://www-03.ibm.com/press/us/en/pressrelease/19781.wss
[3] http://www-03.ibm.com/press/us/en/pressrelease/21189.wss
[4] http://www-03.ibm.com/press/us/en/pressrelease/22455.wss
[5] http://www-03.ibm.com/press/us/en/pressrelease/27279.wss
[6] http://www.redbooks.ibm.com/abstracts/sg247298.html?Open
[7] http://www.redbooks.ibm.com/abstracts/sg247315.html?Open
[8] http://www.amazon.com/DB2-pureXML-Cookbook-Master-Hybrid/dp/0138150478/
[9] http://www-304.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=course_description&courseCode=CG130
[10] http://www-304.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=course_description&courseCode=CG100
[11] http://www-304.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=course_search&sortBy=5&searchType=1&sortDirection=9&includeNotScheduled=15&rowStart=0&rowsToReturn=20&maxSearchResults=200&searchString=CT140&language=en&country=us

External links

- Official website (http://www.ibm.com/software/data/db2/xml)
- pureXML Wiki (http://www.ibm.com/developerworks/wikis/display/db2xml/Home)
- pureXML Forum (http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1423)
- pureXML Team Blog (http://www.ibm.com/developerworks/blogs/page/purexml)
- Native XML Database Blog (http://www.nativexmldatabase.com)
- Blog with pureXML Topics (http://blog.4loeser.net)

Online communities

Online communities allow pureXML users to network with fellow professionals.

- pureXML Group on LinkedIn (http://www.linkedin.com/groups?gid=129185)
- pureXML Group on ChannelDB2 (http://www.channeldb2.com/group/purexml)

List of XML schemas

This is a list of XML schemas in use on the Internet sorted by purpose. XML schemas can be used to create XML documents for a wide range of purposes such as syndication, general exchange, and storage of data in a standard format.

Bookmarks

- XBEL - XML Bookmark Exchange Language

Brewing

- BeerXML: a free XML based data description standard for the exchange of brewing data [1]

Business

- ARTSXML - Retail XML schema specifications by Association for Retail Technology Standards [2]
- PCXML and TXLife - Insurance Industry XML schema specifications by Association for Cooperative Operations Research and Development [3]
- UBL - Defining a common XML library of business documents (purchase orders, invoices, etc.) by Oasis
- HR-XML - Human resources ([4])
- XBRL Extensible Business Reporting Language for International Financial Reporting Standards (IFRS) and United States generally accepted accounting principles (GAAP) business accounting.
- Strategy Markup Language (StratML [5])
- OSCRE Open Standards Consortium for Real Estate format for data exchange within the real estate industry
- IFC-XML Building Information Models for architecture, engineering, construction, and operations.

Elections

- EML, Election Markup Language, is an OASIS standard to support end-to-end management of election processes. It defines over thirty schemas, for example EML 510 for vote count reporting and EML 310 for voter registration.

Financial

- FpML, Financial products Markup Language is the industry-standard protocol for complex financial products. It is based on XML (eXtensible Markup Language), the standard meta-language for describing data shared between applications.
- FIXML, Financial Information eXchange (FIX) protocol is an electronic communications protocol initiated in 1992 for international real-time exchange of information related to the securities transactions and markets.

Geotagging

- KML, Keyhole Markup Language is used for annotation on geographical browsers including Google Earth and NASA's World Wind. These annotations are used to place events such as earthquake warnings, historical events...etc.

Graphical user interfaces

- JAXFront - JAXFront GUI generator (free community edition)
- GLADE - GNOME's User Interface Language (GTK+)
- KParts - KDE's User Interface Language (Qt)
- XUL - XML User Interface Language (Native)
- XForms - XForms
- XAML - Microsoft's Extensible Application Markup Language

Humanities texts

- TEI - Text Encoding Initiative
- EpiDoc - Epigraphic Documents
- Menota [6] - Mediaeval Nordic Texts Archive
- MEI [7] - Music Encoding Initiative

Industrial property

- TM-XML - Trade Mark Information Exchange Standard
- DS-XML - Industrial Design Information Exchange Standard

Math and science

- MathML - Mathematical Markup Language
- ANSI N42.42 or "N42" - NIST data format standard for radiation detectors used for Homeland Security[8]

Metadata

- RDF - Resource Description Framework
- ONIX for Books [9] - ONline Information eXchange, developed and maintained by EDItEUR jointly with Book Industry Communication (UK) and the Book Industry Study Group (US), and with user groups in Australia, Canada, France, Germany, Italy, the Netherlands, Norway, Spain and the Republic of Korea.
- DDML - reformulations XML DTD
- PRISM - Publishing Requirements for Industry Standard Metadata (PRISM) [10]Wikipedia:Link rot

Music playlists

- XSPF - XML Shareable Playlist Format

News syndication

- Atom - Atom
- RSS - Really Simple Syndication

Paper and forest products

- papiNet - XML format for exchange of business documents and product information in the paper and forest products industries.
- EPPML - an XML conceptual model for the interactions between parties of a postal communication system.

Publishing

- NLM DTD, journal publishing DTD from the United States National Library of Medicine [11]
- DITA—Darwin Information Typing Architecture, document authoring system
- DocBook for technical documentation
- PRISM - Publishing Requirements for Industry Standard Metadata (PRISM) [10]

Statistics

- SDMX - SDMX-ML is a format for exchange and sharing of Statistical Data and Metadata.
- DDI - "Data Documentation Initiative" is a format for information describing statistical and social science data (and the lifecycle).

Vector images

- SVG - Scalable Vector Graphics

Notes

[1] http://www.beerxml.com/beerxml.htm
[2] http://www.nrf-arts.org/
[3] http://www.acord.org/
[4] http://www.hr-xml.org/hr-xml/wms/hr-xml-1-org/index.php?language=2
[5] http://xml.fido.gov/stratml/index.htm
[6] http://www.menota.org/
[7] http://music-encoding.org/
[8] http://physics.nist.gov/Divisions/Div846/Gp4/ANSIN4242/xml.html
[9] http://www.editeur.org/onix.html
[10] http://www.idealliance.org/specifications/prism
[11] http://dtd.nlm.nih.gov/

External links

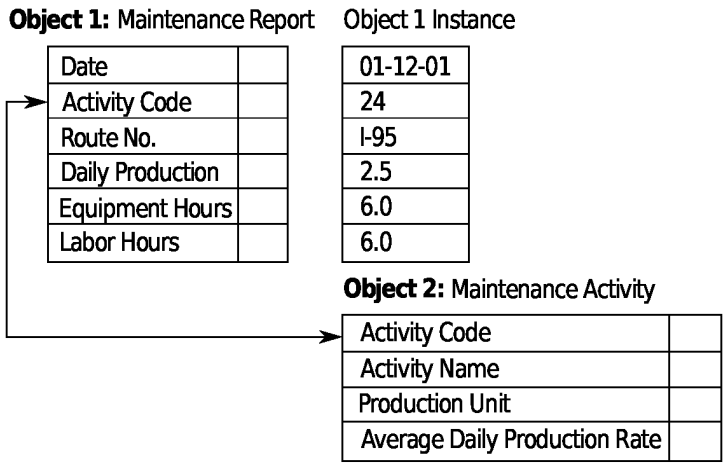
- Schema Documentation Library (http://schemas.liquid-technologies.com/)

Object database

Object database

An **object database** (also **object-oriented database management system**) is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases which are table-oriented. Object-relational databases are a hybrid of both approaches.

Object databases have been considered since the early 1980s.[2]



Example of an object-oriented model[1]

Overview

Object-oriented database management systems (OODBMSs) combine database capabilities with object-oriented programming language capabilities. OODBMSs allow object-oriented programmers to develop the product, store them as objects, and replicate or modify existing objects to make new objects within the OODBMS. Because the database is integrated with the programming language, the programmer can maintain consistency within one environment, in that both the OODBMS and the programming language will use the same model of representation. Relational DBMS projects, by way of contrast, maintain a clearer division between the database model and the application.

As the usage of web-based technology increases with the implementation of Intranets and extranets, companies have a vested interest in OODBMSs to display their complex data. Using a DBMS that has been specifically designed to store data as objects gives an advantage to those companies that are geared towards multimedia presentation or organizations that utilize computer-aided design (CAD).[3]

Some object-oriented databases are designed to work well with object-oriented programming languages such as Delphi, Ruby, Python, Perl, Java, C#, Visual Basic .NET, C++, Objective-C and Smalltalk; others have their own programming languages. OODBMSs use exactly the same model as object-oriented programming languages.

History

Object database management systems grew out of research during the early to mid-1970s into having intrinsic database management support for graph-structured objects. The term "object-oriented database system" first appeared around 1985.[4] Notable research projects included Encore-Ob/Server (Brown University), EXODUS (University of Wisconsin–Madison), IRIS (Hewlett-Packard), ODE (Bell Labs), ORION (Microelectronics and Computer Technology Corporation or MCC), Vodak (GMD-IPSI), and Zeitgeist (Texas Instruments). The ORION project had more published papers than any of the other efforts. Won Kim of MCC compiled the best of those papers

in a book published by The MIT Press.[5]

Early commercial products included Gemstone (Servio Logic, name changed to GemStone Systems), Gbase (Graphael), and Vbase (Ontologic). The early to mid-1990s saw additional commercial products enter the market. These included ITASCA (Itasca Systems), Jasmine (Fujitsu, marketed by Computer Associates), Matisse (Matisse Software), Objectivity/DB (Objectivity, Inc.), ObjectStore (Progress Software, acquired from eXcelon which was originally Object Design), ONTOS (Ontos, Inc., name changed from Ontologic), O_2 [6] (O_2 Technology, merged with several companies, acquired by Informix, which was in turn acquired by IBM), POET (now FastObjects [7] from Versant which acquired Poet Software), Versant Object Database (Versant [8] Corporation), VOSS (Logic Arts) and JADE (Jade [9] Software Corporation). Some of these products remain on the market and have been joined by new open source and commercial products such as InterSystems Caché.

Object database management systems added the concept of persistence to object programming languages. The early commercial products were integrated with various languages: GemStone (Smalltalk), Gbase (LISP), Vbase (COP) and VOSS (Virtual Object Storage System for Smalltalk). For much of the 1990s, C++ dominated the commercial object database management market. Vendors added Java in the late 1990s and more recently, C#.

Starting in 2004, object databases have seen a second growth period when open source object databases emerged that were widely affordable and easy to use, because they are entirely written in OOP languages like Smalltalk, Java, or C#, such as Versant's db4o (db4objects), DTS/S1 from Obsidian Dynamics and Perst (McObject), available under dual open source and commercial licensing.

Timeline

- 1985 – Term Object Database first introduced
- 1988
  - Versant Corporation started (as Object Sciences Corp)
  - Objectivity, Inc. founded
- Early 1990s
  - Gemstone (Smalltalk)-(C++)-(Java)
  - GBase (LISP)
  - VBase (O2- ONTOS – INFORMIX)
  - Objectivity/DB
- Mid 1990's
  - Versant Object Database
  - ObjectStore
  - ODABA
  - ZODB
  - Poet
  - Jade
  - Matisse
- 2000's
  - Caché
  - db4o project started by Carl Rosenberger
  - ObjectDB
- 2001
  - IBM acquires Informix (Illustra) integrates with DB2
  - db4o shipped to first pilot customer
- 2003 - odbpp public release

- 2004 - db4o's commercial launch as db4objects, Inc.
- 2008 - db4o acquired by Versant Corporation
- 2011
  - Wakanda public Developer Preview & Beta versions
  - Gemstone acquired by VMware (at that time property of Springsource)
- 2012
  - Wakanda first production versions with open source and commercial licenses

Adoption of object databases

Object databases based on persistent programming acquired a niche in application areas such as engineering and spatial databases, telecommunications, and scientific areas such as high energy physics and molecular biology.

Another group of object databases focuses on embedded use in devices, packaged software, and real-time systems.

Technical features

Most object databases also offer some kind of query language, allowing objects to be found using a declarative programming approach. It is in the area of object query languages, and the integration of the query and navigational interfaces, that the biggest differences between products are found. An attempt at standardization was made by the ODMG with the Object Query Language, OQL.

Access to data can be faster because joins are often not needed (as in a tabular implementation of a relational database). This is because an object can be retrieved directly without a search, by following pointers.

Another area of variation between products is in the way that the schema of a database is defined. A general characteristic, however, is that the programming language and the database schema use the same type definitions.

Multimedia applications are facilitated because the class methods associated with the data are responsible for its correct interpretation.

Many object databases, for example Gemstone or VOSS, offer support for versioning. An object can be viewed as the set of all its versions. Also, object versions can be treated as objects in their own right. Some object databases also provide systematic support for triggers and constraints which are the basis of active databases.

The efficiency of such a database is also greatly improved in areas which demand massive amounts of data about one item. For example, a banking institution could get the user's account information and provide them efficiently with extensive information such as transactions, account information entries etc. The Big O Notation for such a database paradigm drops from O(n) to O(1), greatly increasing efficiency in these specific cases.

Standards

The Object Data Management Group was a consortium of object database and object-relational mapping vendors, members of the academic community, and interested parties. Its goal was to create a set of specifications that would allow for portable applications that store objects in database management systems. It published several versions of its specification. The last release was ODMG 3.0. By 2001, most of the major object database and object-relational mapping vendors claimed conformance to the ODMG Java Language Binding. Compliance to the other components of the specification was mixed. In 2001, the ODMG Java Language Binding was submitted to the Java Community Process as a basis for the Java Data Objects specification. The ODMG member companies then decided to concentrate their efforts on the Java Data Objects specification. As a result, the ODMG disbanded in 2001.

Many object database ideas were also absorbed into SQL:1999 and have been implemented in varying degrees in object-relational database products.

In 2005 Cook, Rai, and Rosenberger proposed to drop all standardization efforts to introduce additional object-oriented query APIs but rather use the OO programming language itself, i.e., Java and .NET, to express queries. As a result, Native Queries emerged. Similarly, Microsoft announced Language Integrated Query (LINQ) and DLINQ, an implementation of LINQ, in September 2005, to provide close, language-integrated database query capabilities with its programming languages C# and VB.NET 9.

In February 2006, the Object Management Group (OMG) announced that they had been granted the right to develop new specifications based on the ODMG 3.0 specification and the formation of the Object Database Technology Working Group (ODBT WG). The ODBT WG planned to create a set of standards that would incorporate advances in object database technology (e.g., replication), data management (e.g., spatial indexing), and data formats (e.g., XML) and to include new features into these standards that support domains where object databases are being adopted (e.g., real-time systems). The work of the ODBT WG was suspended in March 2009 when, subsequent to the economic turmoil in late 2008, the ODB vendors involved in this effort decided to focus their resources elsewhere.

In January 2007 the World Wide Web Consortium gave final recommendation status to the XQuery language. XQuery uses XML as its data model. Some of the ideas developed originally for object databases found their way into XQuery, but XQuery is not intrinsically object-oriented. Because of the popularity of XML, XQuery engines compete with object databases as a vehicle for storage of data that is too complex or variable to hold conveniently in a relational database. XQuery also allows modules to be written to provide encapsulation features that have been provided by Object-Oriented systems.

Comparison with RDBMSs

An object database stores complex data and relationships between data directly, without mapping to relational rows and columns, and this makes them suitable for applications dealing with very complex data. Objects have a many to many relationship and are accessed by the use of pointers. Pointers are linked to objects to establish relationships. Another benefit of an OODBMS is that it can be programmed with small procedural differences without affecting the entire system.[10]

References

[1] Data Integration Glossary (http://knowledge.fhwa.dot.gov/tam/aashto.nsf/All+Documents/4825476B2B5C687285256B1F00544258/$FILE/DIGloss.pdf), U.S. Department of Transportation, August 2001.

[2] ODBMS.ORG :: Object Database (ODBMS) | Object-Oriented Database (OODBMS) | Free Resource Portal (http://odbms.org/Introduction/history.aspx). ODBMS (2013-08-31). Retrieved on 2013-09-18.

[3] O'Brien, J. A., & Marakas, G. M. (2009). Management Information Systems (9th ed.). New York, NY: McGraw-Hill/Irwin

[4] Three example references from 1985 that use the term: T. Atwood, "An Object-Oriented DBMS for Design Support Applications," *Proceedings of the IEEE COMPINT 85*, pp. 299-307, September 1985; N. Derrett, W. Kent, and P. Lyngbaek, "Some Aspects of Operations in an Object-Oriented Database," *Database Engineering*, vol. 8, no. 4, IEEE Computer Society, December 1985; D. Maier, A. Otis, and A. Purdy, "Object-Oriented Database Development at Servio Logic," *Database Engineering*, vol. 18, no.4, December 1985.

[5] Kim, Won. *Introduction to Object-Oriented Databases.* The MIT Press, 1990. ISBN 0-262-11124-1

[6] Bancilhon, Francois; Delobel,Claude; and Kanellakis, Paris. *Building an Object-Oriented Database System: The Story of O_2.* Morgan Kaufmann Publishers, 1992. ISBN 1-55860-169-4.

[7] http://www.versant.com/developer

[8] http://www.versant.com

[9] http://www.jadeworld.com/jade

[10] Burleson, Donald. (1994). OODBMSs gaining MIS ground but RDBMSs still own the road. Software Magazine, 14(11), 63

External links

- Object DBMS resource portal (http://www.odbms.org/)
- Object-Oriented Databases (http://www.comptechdoc.org/independent/database/basicdb/dataobject.html) – From CompTechDoc.org

Object Definition Language

Object Definition Language (**ODL**) is the specification language defining the interface to object types conforming to the ODMG Object Model. Often abbreviated by the acronym **ODL**.

This language's purpose is to define the structure of an Entity-relationship diagram.

Language

Class declarations

Interface < name > { elements = attributes, relationships, methods }

Element Declarations

attributes (< type > : < name >);

relationships (< rangetype > : < name >);

Example

```
Type Date Tuple (year, day, month) Type year, day, month integer
Class Manager attributes{id : string unique name : string phone : string set employees : Tuple ( [Employee], Start_Date : Date )}
Class Employee attributes{id : string unique name : string Start_Date : Date manager : [Manager]}
```

Object Query Language

Object Query Language (OQL) is a query language standard for object-oriented databases modeled after SQL. OQL was developed by the Object Data Management Group (ODMG). Because of its overall complexity no vendor has ever fully implemented the complete OQL. OQL has influenced the design of some of the newer query languages like JDOQL and EJB QL, but they can't be considered as different flavors of OQL.

General rules

The following rules apply to OQL statements:

- All complete statements must be terminated by a semi-colon.
- A list of entries in OQL is usually separated by commas but not terminated by a comma(,).
- Strings of text are enclosed by matching quotation marks.

Examples

Simple query

The following example illustrates how one might retrieve the CPU-speed of all PCs with more than 64MB of RAM from a fictional PC database:

```
SELECT pc.cpuspeed
FROM PCs pc
WHERE pc.ram > 64;
```

Query with grouping and aggregation

The following example illustrates how one might retrieve the average amount of RAM on a PC, grouped by manufacturer:

```
SELECT manufacturer, AVG(SELECT part.pc.ram FROM partition part)
FROM PCs pc
GROUP BY manufacturer: pc.manufacturer;
```

Note the use of the keyword `partition`, as opposed to aggregation in traditional SQL.

Object-oriented SQL

A functional language, a superset of SQL, used in Hewlett-Packard's OpenODB database system.

SQL1999, formerly known as SQL3, is an Object-Oriented SQL.

SQL:1999 on Wikipedia

Object Exchange Model

The **Object Exchange Model**[1] (OEM) is a model for exchanging semi-structured data between object-oriented databases. It serves as the basic Data model in numerous projects of the Stanford University Database Group, including Tsimmis, Lore, and C3 [2].

Slight variations of OEM have evolved across different Stanford projects. In Lore, labels are actually on parent-child "links" rather than objects. For example, if an OEM object has multiple parents, different parent objects may use different labels to identify that object. An atomic value encoding a person's name might be included in one complex object using the label "Author" and in another complex object using the label "Editor." In C3, additional attributes are required for each object to annotate the changes to the object that have occurred over time [2].

OEM representations

Textual OEM interchange format used in Lore [3] - The goals of this interchange format were to have textual encodings of OEM to be easy to read, easy to edit, and easy to generate or parse by a program.

References

- ^ Papakonstantinou, Y. and Garcia-Molina, H. and Widom, J. (1995). "Object exchange across heterogeneous information sources". *Proceedings of the Eleventh International Conference on Data Engineering*: 251–260. doi:10.1109/ICDE.1995.380386 [4]. ISBN 0-8186-6910-1.
- ^ http://infolab.stanford.edu/~mchughj/oemsyntax/oemsyntax.html

External links

- A Standard Textual Interchange Format for the Object Exchange Model [5]

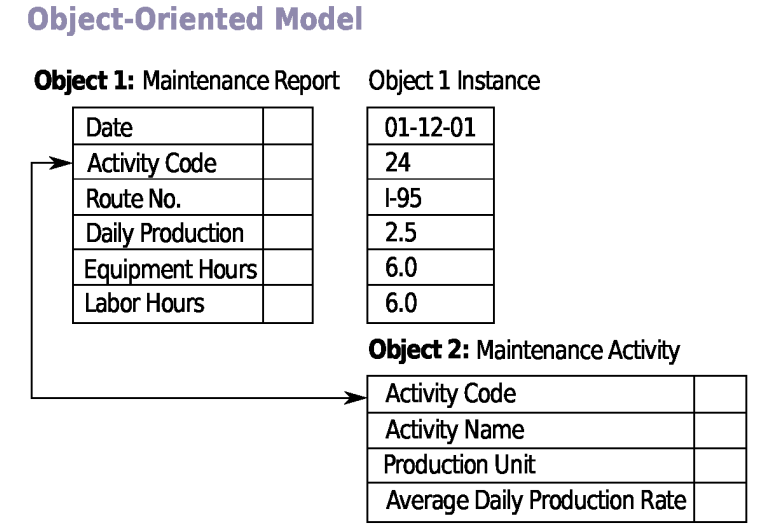
References

[1] http://en.wikipedia.org/wiki/Object_Exchange_Model#endnote_PGW95
[2] http://en.wikipedia.org/wiki/Object_Exchange_Model#endnote_Goldman
[3] http://www.dcs.bbk.ac.uk/~ptw/teaching/ssd/slide9.html
[4] http://dx.doi.org/10.1109%2FICDE.1995.380386
[5] http://infolab.stanford.edu/~mchughj/oemsyntax/oemsyntax.html

Object-relational database

An **object-relational database** (**ORD**), or **object-relational database management system** (**ORDBMS**), is a database management system (DBMS) similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language. In addition, just as with pure relational systems, it supports extension of the data model with custom data-types and methods.

An object-relational database can be said to provide a middle ground between relational databases and *object-oriented databases* (OODBMS). In object-relational databases, the approach is essentially that of relational databases: the data resides in the database and is manipulated collectively with queries in a query language; at the other extreme are OODBMSes in which the database is essentially a persistent object store for software written in an object-oriented programming language, with a programming API for storing and retrieving objects, and little or no specific support for querying.



Example of an object-oriented database model.

Overview

The basic goal for the Object-relational database is to bridge the gap between relational databases and the object-oriented modeling techniques used in programming languages such as Java, C++, Visual Basic .NET or C#. However, a more popular alternative for achieving such a bridge is to use a standard relational database systems with some form of Object-relational mapping (ORM) software. Whereas traditional RDBMS or SQL-DBMS products focused on the efficient management of data drawn from a limited set of data-types (defined by the relevant language standards), an object-relational DBMS allows software developers to integrate their own types and the methods that apply to them into the DBMS.

The ORDBMS (like ODBMS or OODBMS) is integrated with an object-oriented programming language. The characteristic properties of ORDBMS are 1) complex data, 2) type inheritance, and 3) object behavior. **Complex data** creation in most SQL ORDBMSs is based on preliminary schema definition via the user-defined type (UDT). Hierarchy within structured complex data offers an additional property, **type inheritance**. That is, a structured type can have subtypes that reuse all of its attributes and contain additional attributes specific to the subtype. Another advantage, the **object behavior**, is related with access to the program objects. Such program objects have to be storable and transportable for database processing, therefore they usually are named as persistent objects. Inside a database, all the relations with a persistent program object are relations with its object identifier (OID). All of these points can be addressed in a proper relational system, although the SQL standard and its implementations impose arbitrary restrictions and additional complexityWikipedia:Citing sources

In object-oriented programming (OOP) object behavior is described through the methods (object functions). The methods denoted by one name are distinguished by the type of their parameters and type of objects for which they attached (method signature). The OOP languages call this the polymorphism principle, which briefly is defined as

"one interface, many implementations". Other OOP principles, inheritance and encapsulation, are related both to methods and attributes. Method inheritance is included in type inheritance. Encapsulation in OOP is a visibility degree declared, for example, through the PUBLIC, PRIVATE and PROTECTED modifiers.

History

Object-relational database management systems grew out of research that occurred in the early 1990s. That research extended existing relational database concepts by adding object concepts. The researchers aimed to retain a declarative query-language based on predicate calculus as a central component of the architecture. Probably the most notable research project, Postgres (UC Berkeley), spawned two products tracing their lineage to that research: Illustra and PostgreSQL.

In the mid-1990s, early commercial products appeared. These included Illustra[1] (Illustra Information Systems, acquired by Informix Software which was in turn acquired by IBM), Omniscience (Omniscience Corporation, acquired by Oracle Corporation and became the original Oracle Lite), and UniSQL (UniSQL, Inc., acquired by KCOMS). Ukrainian developer Ruslan Zasukhin, founder of Paradigma Software, Inc., developed and shipped the first version of Valentina database in the mid-1990s as a C++ SDK. By the next decade, PostgreSQL had become a commercially viable database and is the basis for several products today which maintain its ORDBMS features.

Computer scientists came to refer to these products as "object-relational database management systems" or ORDBMSs.[2]

Many of the ideas of early object-relational database efforts have largely become incorporated into SQL:1999 via structured types. In fact, any product that adheres to the object-oriented aspects of SQL:1999 could be described as an object-relational database management product. For example, IBM's DB2, Oracle database, and Microsoft SQL Server, make claims to support this technology and do so with varying degrees of success.

Comparison to RDBMS

An RDBMS might commonly involve SQL statements such as these:

```
CREATE TABLE Customers  (
    Id          CHAR(12)     NOT NULL PRIMARY KEY,
    Surname     VARCHAR(32) NOT NULL,
    FirstName   VARCHAR(32) NOT NULL,
    DOB         DATE         NOT NULL
 );
 SELECT InitCap(Surname) || ', ' || InitCap(FirstName)
   FROM Customers
  WHERE Month(DOB) = Month(getdate())
    AND Day(DOB) = Day(getdate())
```

Most current[3] SQL databases allow the crafting of custom functions, which would allow the query to appear as:

```
 SELECT Formal(Id)
   FROM Customers
  WHERE Birthday(DOB) = Today()
```

In an object-relational database, one might see something like this, with user-defined data-types and expressions such as `BirthDay()`:

```
CREATE TABLE Customers (
  Id          Cust_Id     NOT NULL  PRIMARY KEY,
```

```
      Name          PersonName  NOT NULL,
      DOB           DATE        NOT NULL
    );
    SELECT Formal( C.Id )
      FROM Customers C
     WHERE BirthDay ( C.DOB ) = TODAY;
```

The object-relational model can offer another advantage in that the database can make use of the relationships between data to easily collect related records. In an address book application, an additional table would be added to the ones above to hold zero or more addresses for each customer. Using a traditional RDBMS, collecting information for both the user and their address requires a "join":

```
     SELECT InitCap(C.Surname) || ', ' || InitCap(C.FirstName), A.city
       FROM Customers C join Addresses A ON A.Cust_Id=C.Id -- the join
      WHERE A.city="New York"
```

The same query in an object-relational database appears more simply:

```
    SELECT Formal( C.Name )
      FROM Customers C
     WHERE C.address.city="New York" -- the linkage is 'understood' by
the ORDB
```

References

[1] Stonebraker,. Michael with Moore, Dorothy. *Object-Relational DBMSs: The Next Great Wave*. Morgan Kaufmann Publishers, 1996. ISBN 1-55860-397-2.

[2] There was, at the time, a dispute whether the term was coined by Michael Stonebraker of Illustra or Won Kim of UniSQL.

[3] http://en.wikipedia.org/w/index.php?title=Object-relational_database&action=edit

External links

- Savushkin, Sergey (2003), *A Point of View on ORDBMS* (http://savtechno.com/articles/ViewOfORDBMS.html), retrieved 2012-07-21.
- *JPA Performance Benchmark* (http://www.jpab.org/) — comparison of Java JPA ORM Products (Hibernate, EclipseLink, OpenJPA, DataNucleus).
- *PolePosition Benchmark* (http://www.polepos.org/) — shows the performance trade-offs for solutions in the object-relational impedance mismatch context.

Object-relational impedance mismatch

The **object-relational impedance mismatch** is a set of conceptual and technical difficulties that are often encountered when a relational database management system (RDBMS) is being used by a program written in an object-oriented programming language or style; particularly when objects or class definitions are mapped in a straightforward way to database tables or relational schema.

The term *object-relational impedance mismatch* is derived from the electrical engineering term *impedance matching*.

Mismatches

Object-oriented concepts

Encapsulation

Object-oriented programs are designed with techniques that result in encapsulated objects whose representation is hidden. In an object-oriented framework, the underlying properties of a given object are expected to be unexposed to any interface outside of the one implemented alongside the object. However, object-relational mapping necessarily exposes the underlying content of an object to interaction with an interface that the object implementation cannot specify. Hence, object-relational mapping violates the encapsulation of the object.

Accessibility

In relational thinking, "private" versus "public" access is relative to need rather than being an absolute characteristic of the data's state, as in the OO model. The relational and OO models often have conflicts over relativity versus absolutism of classifications and characteristics.

Interface, class, inheritance and polymorphism

Under an object-oriented paradigm, objects have interfaces that together provide the only access to the internals of that object. The relational model, on the other hand, utilizes derived relation variables (views) to provide varying perspectives and constraints to ensure integrity. Similarly, essential OOP concepts for classes of objects, inheritance and polymorphism are not supported by relational database systems.

Mapping to relational concepts

A proper mapping between relational concepts and object-oriented concepts can be made if relational database tables are linked to associations found in object-oriented analysis.

Data type differences

A major mismatch between existing relational and OO languages is the type system differences. The relational model strictly prohibits by-reference attributes (or pointers), whereas OO languages embrace and expect by-reference behavior. Scalar types and their operator semantics are also very often subtly to vastly different between the models, causing problems in mapping.

For example, most SQL systems support string types with varying collations and constrained maximum lengths (open-ended text types tend to hinder performance), while most OO languages consider collation only as an argument to sort routines and strings are intrinsically sized to available memory. A more subtle, but related example is that SQL systems often ignore trailing white space in a string for the purposes of comparison, whereas OO string libraries do not. It is typically not possible to construct new data types as a matter of constraining the possible values of other primitive types in an OO language.

Structural and integrity differences

Another mismatch has to do with the differences in the structural and integrity aspects of the contrasted models. In OO languages, objects can be composed of other objects—often to a high degree—or specialize from a more general definition. This may make the mapping to relational schemas less straightforward. This is because relational data tends to be represented in a named set of global, unnested relation variables. Relations themselves, being sets of tuples all conforming to the same headerWikipedia:Please clarify do not have an ideal counterpart in OO languages. Constraints in OO languages are generally not declared as such, but are manifested as exception raising protection logic surrounding code that operates on encapsulated internal data. The relational model, on the other hand, calls for declarative constraints on scalar types, attributes, relation variables, and the database as a whole.

Manipulative differences

The semantic differences are especially apparent in the manipulative aspects of the contrasted models, however. The relational model has an intrinsic, relatively small and well defined set of primitive operators for usage in the query and manipulation of data, whereas OO languages generally handle query and manipulation through custom-built or lower-level, case and physical access path specific imperative operations. Some OO languages do have support for declarative query sub-languages, but because OO languages typically deal with lists and perhaps hash-tables, the manipulative primitives are necessarily distinct from the set-based operations of the relational model.[*citation needed*]

Transactional differences

The concurrency and transaction aspects are significantly different also. In particular, relational database transactions, as the smallest unit of work performed by databases, are much larger than any operations performed by classes in OO languages. Transactions in relational databases are dynamically bounded sets of arbitrary data manipulations, whereas the granularity of transactions in OO languages is typically individual assignments of primitive typed fields. OO languages typically have no analogue of isolation or durability as well and atomicity and consistency are only ensured for said writes of primitive typed fields.

Solving impedance mismatch

Solving the impedance mismatch problem for object-oriented programs starts with recognition of the differences in the specific logic systems being employed, then either the minimization or compensation of the mismatch.

Minimization

There have been some attempts at building object-oriented database management systems (OODBMS) that would avoid the impedance mismatch problem. They have been less successful in practice than relational databases however, partly due to the limitations of OO principles as a basis for a data model.[1] There has been research performed in extending the database-like capabilities of OO languages through such notions as transactional memory.

One common solution to the impedance mismatch problem is to layer the domain and framework logic. In this scheme, the OO language is used to model certain relational aspects at runtime rather than attempt the more static mapping. Frameworks which employ this method will typically have an analogue for a tuple, usually as a "row" in a "dataset" component or as a generic "entity instance" class, as well as an analogue for a relation. Advantages of this approach may include:

- Straightforward paths to build frameworks and automation around transport, presentation, and validation of domain data.
- Smaller code size; faster compile and load times.
- Ability for the schema to change dynamically.

- Avoids the name-space and semantic mismatch issues.
- Expressive constraint checking
- No complex mapping necessary

Disadvantages may include:

- Lack of static type "safety" checks. Typed accessors are sometimes utilized as one way to mitigate this.
- Possible performance cost of runtime construction and access.
- Inability to natively utilize uniquely OO aspects, such as polymorphism.

Alternative architectures

The rise of XML databases and XML client structures has motivated other alternative architectures to get around the impedance mismatch challenges. These architectures use XML technology in the client (such as XForms) and native XML databases on the server that use the XQuery language for data selection. This allows a single data model and a single data selection language (XPath) to be used in the client, in the rules engines and on the persistence server.[2]

Compensation

The mixing of levels of discourse within OO application code presents problems, but there are some common mechanisms used to compensate. The biggest challenge is to provide framework support, automation of data manipulation and presentation patterns, within the level of discourse in which the domain data is being modeled. To address this, reflection and/or code generation are utilized. Reflection allows code (classes) to be addressed as data and thus provide automation of the transport, presentation, integrity, etc. of the data. Generation addresses the problem through addressing the entity structures as data inputs for code generation tools or meta-programming languages, which produce the classes and supporting infrastructure en masse. Both of these schemes may still be subject to certain anomalies where these levels of discourse merge. For instance, generated entity classes will typically have properties which map to the domain (e. g. Name, Address) as well as properties which provide state management and other framework infrastructure (e. g. IsModified).

Contention

It has been argued, by Christopher J. Date and others, that a truly relational DBMS would pose no such problem, as domains and classes are essentially one and the same thing. A naïve mapping between classes and relational schemata is a fundamental design mistake; and that individual tuples within a database table (relation) ought to be viewed as establishing relationships between entities; not as representations for complex entities themselves. However, this view tends to diminish the influence and role of object oriented programming, using it as little more than a field type management system.

The impedance mismatch in programming between the domain objects and the user interface. Sophisticated user interfaces, to allow operators, managers, and other non-programmers to access and manipulate the records in the database, often require intimate knowledge about the nature of the various database attributes (beyond name and type). In particular, it's considered a good practice (from an end-user productivity point of view) to design user interfaces such that the UI prevents illegal transactions (those which cause a database constraint to be violated) from being entered; to do so requires much of the logic present in the relational schemata to be duplicated in the code.

Certain code-development frameworks can leverage certain forms of logic that are represented in the database's schema (such as referential integrity constraints), so that such issues are handled in a generic and standard fashion through library routines rather than ad hoc code written on a case-by-case basis.

It has been argued that SQL, due to a very limited set of domain types (and other alleged flaws) makes proper object and domain-modelling difficult; and that SQL constitutes a very lossy and inefficient interface between a DBMS and an application program (whether written in an object-oriented style or not). However, SQL is currently the only

widely accepted common database language in the marketplace; use of vendor-specific query languages is seen as a bad practice when avoidable. Other database languages such as Business System 12 and Tutorial D have been proposed; but none of these has been widely adopted by DBMS vendors.

In current versions of mainstream "object-relational" DBMSs like Oracle and Microsoft SQL Server, the above point may be a non-issue. With these engines, the functionality of a given database can be arbitrarily extended through stored code (functions and procedures) written in a modern OO language (Java for Oracle, and a Microsoft.NET language for SQL Server), and these functions can be invoked in-turn in SQL statements in a transparent fashion: that is, the user neither knows nor cares that these functions/procedures were not originally part of the database engine. Modern software-development paradigms are fully supported: thus, one can create a set of library routines that can be re-used across multiple database schemas.

These vendors decided to support OO-language integration at the DBMS back-end because they realized that, despite the attempts of the ISO SQL-99 committee to add procedural constructs to SQL, SQL will never have the rich set of libraries and data structures that today's application programmers take for granted, and it is reasonable to leverage these as directly as possible rather than attempting to extend the core SQL language. Consequently, the difference between "application programming" and "database administration" is now blurred: robust implementation of features such as constraints and triggers may often require an individual with dual DBA/OO-programming skills, or a partnership between individuals who combine these skills. This fact also bears on the "division of responsibility" issue below.

Some, however, would point out that this contention is moot due to the fact that: (1) RDBMSes were never intended to facilitate object modelling, and (2) SQL generally should only be seen as a "lossy" or "inefficient" interface language when one is trying to achieve a solution for which RDBMSes were not designed. SQL is very efficient at doing what it was designed to do, namely, to query, sort, filter, and store large sets of data. Some would additionally point out that the inclusion of OO language functionality in the back-end simply facilitates bad architectural practice, as it admits high-level application logic into the data tier, antithetical to the RDBMS.

Here the "canonical" copy of state is located. The database model generally assumes that the database management system is the only authoritative repository of state concerning the enterprise; any copies of such state held by an application program are just that — temporary copies (which may be out of date, if the underlying database record was subsequently modified by a transaction). Many object-oriented programmers prefer to view the in-memory representations of objects themselves as the canonical data, and view the database as a backing store and persistence mechanism.

The proper division of responsibility between application programmers and database administrators (DBA). It is often the case that needed changes to application code (in order to implement a requested new feature or functionality) require corresponding changes in the database definition; in most organizations, the database definition is the responsibility of the DBA. Due to the need to maintain a production database system 24 hours a day; many DBAs are reluctant to make changes to database schemata that they deem gratuitous or superfluous; and in some cases outright refuse to do so. Use of developmental databases (apart from production systems) can help somewhat; but when the newly developed application "goes live"; the DBA will need to approve any changes. Some programmers view this as intransigence; however the DBA is frequently held responsible if any changes to the database definition cause a loss of service in a production system—as a result, many DBAs prefer to contain design changes to application code, where design defects are far less likely to have catastrophic consequences.

In organizations where the relationship between DBAs and application programmers is not dysfunctional, the above point is a non-issue, and the decision as to whether to change a schema or not is driven by business needs. If new functionality is mandated, and it requires the capture of information that must be persisted, schema enhancements to achieve persistence are the logical first step. Certain schema modifications (including addition of indexes) will also be acceptable if they dramatically improve performance of a critical application. But schema modifications that might serve no purpose beyond making a programmer's life modestly easier would be vetoed if they result in

unacceptable design decisions such as denormalization of transactional schemas.

Philosophical differences

Key philosophical differences between the OO and relational models can be summarized as follows:

- **Declarative vs. imperative interfaces —** Relational thinking tends to use data as interfaces, not behavior as interfaces. It thus has a declarative tilt in design philosophy in contrast to OO's behavioral tilt. (Some relational proponents propose using triggers, stored procedures, etc. to provide complex behavior, but this is not a common viewpoint.)
- **Schema bound —** Objects do not have to follow a "parent schema" for which attributes or accessors an object has, while table rows must follow the entity's schema. A given row must belong to one and only one entity. The closest thing in OO is inheritance, but it is generally tree-shaped and optional. A dynamic form of relational tools that allows ad hoc columns may relax schema bound-ness, but such tools are currently rare.
- **Access rules —** In relational databases, attributes are accessed and altered through predefined relational operators, while OO allows each class to create its own state alteration interface and practices. The "self-handling noun" viewpoint of OO gives independence to each object that the relational model does not permit. This is a "standards versus local freedom" debate. OO tends to argue that relational standards limit expressiveness, while relational proponents suggest the rule adherence allows more abstract math-like reasoning, integrity, and design consistency.
- **Relationship between nouns and actions —** OO encourages a tight association between operations (actions) and the nouns (entities) that the operations operate on. The resulting tightly bound entity containing both nouns and the operations is usually called a class, or in OO analysis, a concept. Relational designs generally do not assume there is anything natural or logical about such tight associations (outside of relational operators).
- **Uniqueness observation —** Row identities (keys) generally have a text-representable form, but objects do not require an externally viewable unique identifier.
- **Object identity —** Objects (other than immutable ones) are generally considered to have a unique identity; two objects which happen to have the same state at a given point in time are not considered to be identical. Relations, on the other hand, have no inherent concept of this kind of identity. That said, it is a common practice to fabricate "identity" for records in a database through use of globally unique candidate keys; though many consider this a poor practice for any database record which does not have a one-to-one correspondence with a real world entity. (Relational, like objects, can use domain keys if they exist in the external world for identification purposes). Relational systems in practice strive for and support "permanent" and inspect-able identification techniques, whereas object identification techniques tend to be transient or situational.
- **Normalization —** Relational normalization practices are often ignored by OO designs. However, this may just be a bad habit instead of a native feature of OO. An alternate view is that a collection of objects, interlinked via pointers of some sort, is equivalent to a network database; which in turn can be viewed as an extremely denormalized relational database.
- **Schema inheritance —** Most relational databases do not support schema inheritance. Although such a feature could be added in theory to reduce the conflict with OOP, relational proponents are less likely to believe in the utility of hierarchical taxonomies and sub-typing because they tend to view set-based taxonomies or classification systems as more powerful and flexible than trees. OO advocates point out that inheritance/subtyping models need not be limited to trees (though this is a limitation in many popular OO languages such as Java), but non-tree OO solutions are seen as more difficult to formulate than set-based variation-on-a-theme management techniques preferred by relational. At the least, they differ from techniques commonly used in relational algebra.
- **Structure vs. behaviour —** OO primarily focuses on ensuring that the structure of the program is reasonable (maintainable, understandable, extensible, reusable, safe), whereas relational systems focus on what kind of behaviour the resulting run-time system has (efficiency, adaptability, fault-tolerance, liveness, logical integrity, etc.). Object-oriented methods generally assume that the primary user of the object-oriented code and its

interfaces are the application developers. In relational systems, the end-users' view of the behaviour of the system is sometimes considered to be more important. However, relational queries and "views" are common techniques to re-represent information in application- or task-specific configurations. Further, relational does not prohibit local or application-specific structures or tables from being created, although many common development tools do not directly provide such a feature, assuming objects will be used instead. This makes it difficult to know whether the stated non-developer perspective of relational is inherent to relational, or merely a product of current practice and tool implementation assumptions.

- **Set vs. graph relationships —** The relationship between different items (objects or records) tend to be handled differently between the paradigms. Relational relationships are usually based on idioms taken from set theory, while object relationships lean toward idioms adopted from graph theory (including trees). While each can represent the same information as the other, the approaches they provide to access and manage information differ.

As a result of the object-relational impedance mismatch, it is often argued by partisans on both sides of the debate that the other technology ought to be abandoned or reduced in scope. Some database advocates view traditional "procedural" languages as more compatible with an RDBMS than many OO languages; or suggest that a less OO-style ought to be used. (In particular, it is argued that long-lived domain objects in application code ought not to exist; any such objects that do exist should be created when a query is made and disposed of when a transaction or task is complete). On the other hand, many OO advocates argue that more OO-friendly persistence mechanisms, such as OODBMS, ought to be developed and used, and that relational technology ought to be phased out. Of course, it should be pointed out that many (if not most) programmers and DBAs do not hold either of these viewpoints; and view the object-relational impedance mismatch as a mere fact of life that information technology has to deal with.

It is also argued that the O/R mapping is paying off in some situations, but is probably oversold: it has advantages besides drawbacks. Skeptics point out that it is worth to think carefully before using it, as it will add little value in some cases.[3]

References

[1] C. J. Date, Relational Database Writings

[2] Dan McCreary, XRX: Simple, Elegant, Disruptive (http://www.oreillynet.com/xml/blog/2008/05/xrx_a_simple_elegant_disruptiv_1.html) on XML.com

[3] *J2EE Design and Development* by Rod Johnson, © 2002 Wrox Press, p. 256.

External links

- The Object-Relational Impedance Mismatch (http://www.agiledata.org/essays/impedanceMismatch.html) - Agile Data Essay
- The Vietnam of Computer Science (http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx) - Examples of mismatch problems
- O/R mapping Why/When (http://www.rgoarchitects.com/Files/ormappin.pdf) - An article explaining the tradeoffs of Object-relational mapping

Object-relational mapping

Not to be confused with Object-Role Modeling.

Object-relational mapping (**ORM**, **O/RM**, and **O/R mapping**) in computer software is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to create their own ORM tools.

Overview

Data management tasks in object-oriented (OO) programming are typically implemented by manipulating objects that are almost always non-scalar values. For example, consider an address book entry that represents a single person along with zero or more phone numbers and zero or more addresses. This could be modeled in an object-oriented implementation by a "Person object" with attributes/fields to hold each data item that the entry comprises: the person's name, a list of phone numbers, and a list of addresses. The list of phone numbers would itself contain "PhoneNumber objects" and so on. The address book entry is treated as a single object by the programming language (it can be referenced by a single variable containing a pointer to the object, for instance). Various methods can be associated with the object, such as a method to return the preferred phone number, the home address, and so on.

However, many popular database products such as structured query language database management systems (SQL DBMS) can only store and manipulate scalar values such as integers and strings organized within tables. The programmer must either convert the object values into groups of simpler values for storage in the database (and convert them back upon retrieval), or only use simple scalar values within the program. Object-relational mapping is used to implement the first approach.

The heart of the problem is translating the logical representation of the objects into an atomized form that is capable of being stored on the database, while somehow preserving the properties of the objects and their relationships so that they can be reloaded as an object when needed. If this storage and retrieval functionality is implemented, the objects are then said to be persistent.

Comparison with traditional data access techniques

Compared to traditional techniques of exchange between an object-oriented language and a relational database, ORM often reduces the amount of code that needs to be written.[1]

Disadvantages of O/R mapping tools generally stem from the high level of abstraction obscuring what is actually happening in the implementation code. Also, heavy reliance on ORM software has been cited as a major factor in producing poorly designed databases.[2]

NoSQL (Not Only SQL) databases

Another approach is to use an object-oriented database management system (OODBMS) or document-oriented databases such as native XML databases that provide more flexibility in data modeling. OODBMSs are databases designed specifically for working with object-oriented values. Using an OODBMS eliminates the need for converting data to and from its SQL form, as the data is stored in its original object representation and relationships are directly represented, rather than requiring join tables/operations.

Document oriented databases also prevent the user from having to "shred" objects into table rows. Many of these systems also support the XQuery query language to retrieve datasets.

Object-oriented databases tend to be used in complex, niche applications. One of the arguments against using an OODBMS is that switching from an SQL DBMS to a purely object-oriented DBMS means that you may lose the capability to create application independent queries for retrieving ad hoc combinations of data without restriction to access path.[*citation needed*] For this reason, many programmers find themselves more at home with an object-SQL mapping system, even though most object-oriented databases are able to process SQL queries to a limited extent. Other OODBMS (such as RavenDB) provide replication to SQL databases, as a means of addressing the need for ad hoc queries, while preserving has well-known query patterns.

Challenges

There are a variety of difficulties that arise when considering how to match an object system to a relational database. These difficulties are referred to as the object-relational impedance mismatch.

An alternative to implementing ORM is use of the native procedural languages provided with every major database on the market. These can be called from the client using SQL statements. The Data Access Object (DAO) design pattern is used to abstract these statements and offer a lightweight object-oriented interface to the rest of the application.

References

[1] Douglas Barry, Torsten Stanienda, "Solving the Java Object Storage Problem," Computer, vol. 31, no. 11, pp. 33-40, Nov. 1998, http://www2.computer.org/portal/web/csdl/doi/10.1109/2.730734, Excerpt at http://www.service-architecture.com/object-relational-mapping/articles/transparent_persistence_vs_jdbc_call-level_interface.html.Lines of code using O/R are only a fraction of those needed for a call-level interface (1:4). *For this exercise, 496 lines of code were needed using the ODMG Java Binding compared to 1,923 lines of code using JDBC.*

[2] Josh Berkus, "Wrecking Your Database", Computer, Aug. 2009, http://it.toolbox.com/blogs/database-soup/wrecking-your-database-33298, Webcast at http://www.youtube.com/watch?v=uFLRc6y_O3s

External links

- About ORM (http://www.artima.com/intv/abstract3.html) by Anders Hejlsberg
- Mapping Objects to Relational Databases: O/R Mapping In Detail (http://www.agiledata.org/essays/mappingObjects.html) by Scott W. Ambler
- Core J2EE Design Pattern: Data Access Objects (http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html)
- Choosing an Object-Relational mapping tool (http://madgeek.com/Articles/ORMapping/EN/mapping.htm)

Polymorphic association

Polymorphic association is a term used in discussions of Object-Relational Mapping with respect to the problem of representing in the relational database domain, a relationship from one class to multiple classes. In statically typed languages such as Java these multiple classes are subclasses of the same superclass. In languages with duck typing, such as Ruby, this is not necessarily the case.

References

Java Persistence with HIBERNATE, Chapter 5, Bauer, Christian & Gavin, King, Manning, copyright 2007, ISBN 1-932394-8-5

External links

- Hibernate Home Page [1]

References

[1] http://www.hibernate.org/

Polyinstantiation

Polyinstantiation in computer science is the concept of type (class, database row or otherwise) being instantiated into multiple independent instances (objects, copies). It may also indicate, such as in the case of database polyinstantiation, that two different instances have the same name (identifier, primary key).

Operating system security

In Operating system security, polyinstantiation is the concept of creating a user or process specific view of a shared resource. I.e. Process **A** cannot affect process **B** by writing malicious code to a shared resource, such as UNIX directory **/tmp**.

Polyinstantiation of shared resources have similar goals as process isolation, an application of virtual memory, where processes are assigned their own isolated virtual address space to prevent process **A** writing into the memory space of process **B**.

Database

In databases, polyinstantiation is database-related SQL (structured query language) terminology. It allows a relation to contain multiple rows with the same primary key; the multiple instances are distinguished by their security levels. It occurs because of mandatory policy. Depending on the security level established, one record contains sensitive information, and the other one does not, that is, a user will see the record's information depending on his/her level of confidentiality previously dictated by the company's policy

Consider the following table, where the primary key is **Name** and **λ(x)** is the security level:

Name	λ(Name)	Age	λ(Age)	λ
Alice	S	18	TS	TS
Bob	S	22	S	S
Bob	S	33	TS	TS
Trudy	TS	15	TS	TS

Although useful from a security standpoint, polyinstantiation raises several problems:

- Moral scrutiny, since it involves lying
- Providing consistent views
- Explosion in the number of rows

Cryptography

In cryptography, polyinstantiation is the existence of a cryptographic key in more than one secure physical location.

References

Single Table Inheritance

Single table inheritance is a way to emulate object-oriented inheritance in a relational database. When mapping from a database table to an object in an object-oriented language, a field in the database identifies what class in the hierarchy the object belongs to.[1] In Ruby on Rails the field in the table called 'type' identifies the name of the class. In .NET Framework, it is called the Discriminator column.

References

[1] Martin Fowler *Patterns of Enterprise Application Architecture* (2003), p. 278

External links

- Single Table Inheritance (http://www.martinfowler.com/eaaCatalog/singleTableInheritance.html)
- Single Table Inheritance in Yii (http://www.yiiframework.com/wiki/198/single-table-inheritance/)

Versant Object Database

Versant Object Database

Developer(s)	Versant Corporation
Stable release	8.0.2.15 / October 1, 2012
Development status	Active
Written in	Java, C, C#, C++, Smalltalk, Python
Operating system	Cross-platform Solaris, Linux, Windows (NT thru Vista), AIX, HP-UX (both 32 and 64 bit for all platforms)
Type	Object Database
License	All rights reserved
Website	www.versant.com [1]

Versant Object Database (VOD) is an object database software product developed by Versant Corporation.

The Versant Object Database enables developers using object oriented languages to transactionally store their information by allowing the respective language to act as the Data Definition Language (DDL) for the database. In other words, the memory model is the database schema model.[2]

In general, persistence in VOD in implemented by declaring a list of classes, then providing a transaction demarcation application programming interface to use cases. Respective language integrations adhere to the constructs of that language, including syntactic and directive sugars.

Additional APIs exist, beyond simple transaction demarcation, providing for the more advanced capabilities necessary to address practical issues found when dealing with performance optimization and scalability for systems with large amounts of data, many concurrent users, network latency, disk bottlenecks, etc.

Feature highlights

- Transparent object persistence from C++, Java and .NET
- Support for standards, e.g., JDO
- Seamless database distribution
- Enterprise-class high availability
- Dynamic schema evolution
- Low administration
- Multithreading, multisession
- End-to-end object architecture
- Fine-grained concurrency control

Supported languages

Primary supported languages are Java, C# and C++. Versant also has language support for Smalltalk and Python.

Query systems

VOD supports queries via a server side indexing and query execution engine. Query support includes both a Versant-specific and a standards-based query language syntax. Versant provides this query capability in a number of forms depending on the developer's chosen language binding. For example, in Java VOD provides VQL (Versant Query Language), JDOQL, EJB QL and OQL. In C++ Versant provides VQL and OQL, with C# support for VQL, OQL and LINQ. VOD will do optimization of query execution based on available attribute indexes. Versant also has support for standard SQL queries against the Versant database using ODBC/JDBC drivers.

Versant Query Language

The native Versant Query Language (VQL) is similar to SQL92. It is a string based implementation which allows parameterized runtime binding. The difference is that instead of targeting tables and columns, it targets classes and attributes.

Other object-oriented elements apply to query processing. For example, a query targeting a super class will return all instances of concrete subclasses that satisfy the query predicate. VOD is a distributed database: a logical database can be composed of many physical database nodes, with queries are performed in parallel.

Versant query support includes most of the core concepts found in relational query languages including: pattern matching, join, set operators, orderby, existence, distinct, projections, numerical expressions, indexing, cursors, etc.

Indexing

VOD supports indexes on large collections. However it is not necessary to have a collection in order to have a queryable object with a usable index. Unlike other OODB implementations, any object in a Versant database is indexable and accessible via query. Indexes can be placed on attributes of classes and those classes can then be the target of a query operation. Indexes can be hash, b-tree, unique, compound, virtual and can be created online either using a utility, via a graphical user interface or via an API call.

Large collection support

VOD provides pagination support for large collections using a special node based implementation. These collections are designed in such a way that access is done so that only nodes needed by the client are brought resident into memory, instead of having to load the entire collection.

These large collections are created and operated on just as other persistent collection classes. The interface is also consistent with the appropriate language constructs. For example C++ Standard Template Library, Java iterators, C# enumerables, etc.

Collections of objects by default are only a collection of object identifiers. So, these can be very large, yet have a small resident memory footprint. To iterate the collection, objects are dereferenced into client memory space in either a configurable batch mode or one at a time. A query on the collection can be done using the “in” operator (or other set based operators like subset_of, superset_of, etc.) without loading the collection to the client memory space.

Data replication

There are several mechanisms for replication on VOD that depend on the motivation behind the replication. It is for high availability or for distribution or integration.

High availability

Versant does synchronous pair replication. Full replication for fault tolerance only requires installation of one configuration file specifying the buddy node names: New connections notice the existence of the replica file and on connect, check the file for a buddy pair and if it exists, connect to both buddies. This could be a distributed database so that there are many buddy pairs. Then all transactional changes are committed synchronously to the buddy database server processes.

If any one of the databases in the buddy pair should become unreachable, the in-flight transactions are handled so that there is no commit failure, instead in-flight transactions on node failure will continue to the node that is still alive in the buddy pair. On the machine where the node is still alive and processing transactions, a new process will start that monitors for the crashed database to become accessible again. Once the previously failed node is alive, the monitoring process starts replicating all changes that have occurred since the time of failure to bring the two buddies back into full synchronization. Once they are in full sync, a flag is set and on the next transaction clients will move back to full synchronous operation. All of this is handled without any user involvement.

In the case of extreme failure, like a broken disk drive, etc., the replicated node can be recreated from an online backup of the live node. Simply install a new disk drive, take an online backup of the live node, restore on the failed machine, start the monitor to sync the last few transactions and restore full replication at clients.

Distribution

Distribution is handled using Versant Asynchronous Replication (VAR), a channel driven, master-slave or peer-to-peer replication framework with rule based conflict detection and resolution.

An administrator uses a utility to define replication channels. Channels are named entities that define a scope of replication within a physical node. The “scope” can be anything from full database replication to something as fine grained as anything definable by a Versant query. Once the channels are defined, applications can register as listeners on these channels, at which point changes from those channel begin to flow to the respective clients.

These channels provide both persistence and reliable messaging. In the event that a connection is lost between a registered listener and a channel, ongoing changes will be guaranteed delivery once the connection is re-established. There are multiple transport protocols that can be configured for optimization in highly reliable LAN networks or high reliability in unreliable WAN type of environments.

In bi-directional channel replication, a set of conflict detection rules are put in place so that conflicting changes can be resolved at runtime without disrupting channel activity. There are other forms of data distribution.

Integration

Usually, integration requires some kind of custom code. Users can connect to both relational and Versant databases using ORM products. They can load objects either from a relational database or Versant and then with some minor code implementation, disconnect those objects from the source and write them to a target. This can be used for import/export in a batch processing mode for integration with other database systems.

Data distribution architecture

VOD handles distributed data processing using a distributed two-phase commit protocol across multiply connected databases. In this process, VOD uses an internal resource manager that is handling the distributed transactions. Versant also supports the XA protocol allowing external transaction monitors to control the transactional context, so for example plug into a CORBA or J2EE application server.

Versant allows object relationships to span physical resource (database) nodes. Shared information referenced from object graphs that reside in other databases and resolution of that information is transparent at runtime. For example, several physical databases may hold user information models that are partitioned by account number holding aggregations on account activities such as trades and then have some more databases holding actual trade models and these users and trades can be related. A query across all of the user databases and return a user (or set of users), then as messages are sent to user objects involving trades, the trade models will automatically be resolved across the distribution. After updates of any of those objects, at commit time Versant will ensure that all changes commit back to their respective physical nodes in a completely ACID 2phase commit process.

Object id's are guaranteed to be unique across all physical nodes. Objects could be "moved" from one physical node to another without any application code changes required.

Schema evolution

Schema evolution is handled via a normal update of the application's class models and then applying those changes to the operational database. Those schema changes can be applied to an existing database either via a utility or API. The result is a versioning of the database schema.

Existing objects in the database are lazily evolved to the latest schema version. No object is actually evolved unless it is made dirty (marked for update) and committed back to the database. In general this means an application with the new schema will not cause evolution, expect for new and updated objects.

There are utilities that can "crawl" a database slowly evolving all instanced to the latest version by grabbing sets of them, marking them dirty, committing. This is sometimes desired for embedded or real-time systems where performance and space needs to be optimized.

In most cases, older clients get patch updates with the new schema in conjunction with updates to the server. The clients schema version is in sync with the database server. Versant's loose schema mapping facility can also be used. This is enabled by a flag in the client so that it does not complain about a mismatch in schema version and instead filters the incoming objects to match the old schema. Using this facility requires some forethought to avoid any unintended side effects.

The process goes as follows:

1. class definitions are updated, i.e. add new subclasses, add attributes, rename attributes, remove attributes, etc. and recompile. When the application connects to a Versant database, a schema version mismatch will be detected and you would normally get an error unless you take some action to avoid the mismatch.
2. The schema mismatch can be avoided using a number of techniques.
    1. a utility can be used to describe the new schema to the database. The utility will show a list of incompatibilities and ask how you want them to be resolved. Your action will depend on whether you are in

development, QA, production, etc. Regardless, actions like dropping the existing class, evolving the schema version and keeping all existing objects, rename and retype, etc, are also possible.

2. the evolution process can be automated via connection options. This is normally used in development mode and allows the schema to automatically evolve any mismatches on connect and continue preserving the existing objects.
3. specific API's can be used to dynamically evolve the database schema. This is an advanced topic, involving what's called Versant runtime classes. Basically, you can create completely dynamic schema structure for the database so that new classes and attributes can be created on the fly.

3. If clients with the older schema continue to operate on the database, loose_schema_mapping in the application profile file should be set to true.
4. Optionally, a utility can be started to crawl the database and force version migration of all existing instances.

The general guidelines for schema evolution are that any schema changes can be made and existing instances preserved, without having to write custom evolution code, with the exception of two things:

1. Changes to the middle of an inheritance hierarchy. Inserting a new class into the middle of a hierarchy is impossible without losing your existing objects, unless custom code is written to do this operation in a series of steps.
2. Incompatible type changes like Array to a String.

All other forms of evolution like renaming attributes, deleting leaf classes, adding leaf classes, adding new classes, adding or removing attributes, etc. can be done online and without custom code. If actions like setting non standard default values for newly added attributes are necessary, this can be done in callback functions within the objects. There are a set of standard object lifecycle callbacks that get invoked in activities like cache load. Those callbacks can be used to check for default values and take action if necessary.

Persistent object lifecycle

The lifecycle of an object load can be controlled on a use case basis.

By default, objects are loaded only when they are sent a message. This includes the default behavior for queries which only return a collection of references to objects that satisfied the query predicate, not the actual objects. When an object is loaded, all it's non-reference attributes (primitives) are also loaded and remaining reference types follow the same pattern as the referencing object.

When a message is sent to an object VOD looks into internal structures to see if the object is already in client memory. If not, VOS does an RPC to load the object. At the time VOD loads the object, it will also look at the connections locking strategy to decide how to deal with locking the object on load. VOD supports both global locking strategies that can be applied to a connection and extremely fine grained control to override behavior for a particular use case.

Once an object is loaded and locked it stays in the client cache, with an equivalent lock in the server, until one of a number of events occurs.

The most common event, the current transaction ends with commit. In the default case, this will release the lock and object from memory. However, note that there are forms of commit that will do combinations of things like, keep the cache and the locks and start a new transaction, keep the cache, but release the locks and start a new transaction. These forms and others are used to optimize cache effectiveness when using non-default locking strategies like optimistic locking or when you have a series of transactions that form a task and operate on the same set of objects.

Another possibility is that your client cache starts to get full. In this case, VOD may decide to swap objects back to the server process to make space and do some work that will have to be done at commit anyway. VOD does this in a fully transactional way, so that even if modified objects get swapped to the server, they will still be undone if the transaction is rolled back. Also, you have the ability to "pin" objects into the client cache to prevent swapping of

important sets of objects, enabling the use of direct memory pointers without concern for memory faults.

Another possible event is a query call which has the option set to flush the cache of objects in the target class, so that changed objects currently in your cache become part of the current query execution evaluation.

Other possibilities include API calls that result in explicit release of the object, like a call to refresh or a call to release.

There are many ways to override the default behavior. Those are in fact commonly used to performance tune on a use case basis. For example, if you are going to iterate over a collection of 1000 objects, you don't want to do 1000 RPC's. Giving the collection of references to a call to groupRead will use a single RPC and load all objects. Similarly, you can make a call to getClosure which will use groupRead behavior to load all referenced objects in a graph from the starting point, down to your specified level of reachability. Further, queries have options to set a lock and load result sets rather than just references or to use cursors. There are API's to explicitly load objects into cache and set higher lock levels than the connection defaults, etc.

Achieving persistence

For users of C++, Versant requires that the uppermost class in an inheritance hierarchy inherit from a base class "PObject", which handles database activities.

Then there is a file setup, `schema.imp`, that declares which classes in the model are to be made persistent and that file is used in a pre-compilation phase where Versant's necessary magicWikipedia:Please clarify is added to the persistent classes. Finally, the resulting `schema.cxx` file is compiled and linked with the application.

The pre-compilation phase is done with a utility though note this is typically automatically set up in one's visual development environment so the process is automatic when a build is done.

When using Java or .NET, this same procedure described above with C++ is accomplished using post-processing byte code enhancement. One sets up a file that declares which classes are to be persistent and then uses a utility, or API, or IDE integration to enhance the classes before running or debugging.

Versant provides other Java APIs based on standards JDO and JPA. In those versions of the API, the system adheres to the standards defined for declaring persistence whether it be some kind of XML or annotation. Enhancement is then done using a utility (similarly with .NET) or more commonly with Eclipse plug-in or Microsoft Visual Studio integration during the build process.

Integration with relational databases

A large percentage of Versant's customers do some form of integration with relational tables. This can be accomplished in a couple of ways depending on the requirements such as: on-line/off-line, batch based, transactional, etc.

XA

Versant supports the XA protocol for distributed transactions. This allows participation in online distributed transactions with relational databases. The interaction with the relational tables can take many forms from custom code to ORM solutions to J2EE application servers (Entity Relationship Modeling) to message passing to ORBs, etc. The XA API allows the Versant database to act as a resource controlled by an external transaction monitor coordinating changes to both Versant and relational databases in the same transactional context.

ORM

Versant can interact with relational databases using Java ORM technology such as JDO (Java Data Objects) and Hibernate JPA. These standards-based implementations have the ability to detach objects from their transactional context and then attach them to another connection. There are restrictions in that Versant requires the application to use a concept known as database identity in order for replication to work with relations intact. Versant does not support the ORM form of application identity in anything other than a disconnected data form.

XML

Versant has tools that enable the import and export of XML data. For example, batch based replication of data can be accomplished by exporting objects from the Versant database as XML, if necessary applying an XSLT transform and then importing into relational tables. The opposite direction is also possible. With Java, the most common approach using XML is to dynamically replicate information using JAXB which runtime converts objects into and out of an XML form. Using JAXB, the Versant database only needs to work with objects rather than importing an XML form. In essence, XML coming from relational databases are converted to objects at runtime using JAXB and those objects are then persisted into the Versant database.

Custom code

Users of C++ are especially challenged in integrating with relational databases. Versant provides consulting to help these customers with their integration challenges, but does not make those solutions, which require customization for each application, available in a productized form.

Transactions

Versant by default is always implicitly in a transaction when connected to the database. In addition, VOD supports the XA protocol and apply that to certain standards based API' such as JDO and JPA which require explicit transaction demarcation. There is a non-implicit form of transaction where transaction begin/end must be declared.

In order to discard from memory objects that have been modified in the current transaction you can either do it globally for the current transaction by issuing a rollback which also implicitly starts another transaction or you can do it in isolation or globally using specific calls within the same transaction.

Locking and caching strategies

Versant by default uses a pessimistic locking strategy to ensure that objects in the database server are in sync with client access in an ACID way. This is done by using a combination of locks against both schema and instance objects.

The database server process maintains lock request queues at the object level to control concurrency of access to the same object. A request for update will establish a queue if there are any existing readers of an object. The request either goes through when all current readers release their locks or times-out (an exception which can be handled by client is thrown). Locks are generally released at transaction boundaries. When a queue is established by an update request, all other subsequent requests fall in queue behind the update request. Once the update request has been filled, all read requests in the queue rush in and get their read lock, return the object, and if there are no other updates, the queue disappears. In this architecture, locks are done at the object level so false waits and false deadlocks do not occur.

Other ways of keeping client caches in sync are, for example, an optimistic locking strategy, using a classic timestamp mechanism. VOD also provides forms of client cache synchronization using multi-cast. Additionally it provides an event mechanism where clients can register for triggering events within the database server to be used for synchronization or for business logic work flow.

Scalability

Storage

Versant supports, multiple file and multiple process configurations. Data storage is done in a single or multiple files, but there are supporting files for the logging subsystem (logical and physical log files). These logging files are used for high performance and scalability under concurrent user loads and for online database backup processes.

Clients

Versant is a multi-user client server database and has production applications with thousands of concurrently connected users. Thus, Versant can also run linked and embedded in the same address space as the application process (so it can be also an embedded database).

Performance

Versant uses internal performance and scalability benchmarks to monitor and measure behavior over time across releases, patches and generations of new hardware.

Versant has done other non-standard benchmarking activities in a public forum.[3] .[4]

Versant ran the 007 benchmarks in the early 90's but currently doesn't provide any comparisons because there are no industry benchmarks that make sense for object databases,

One of the candidates considered was TPC-E, which was supposed to be the new OLTP standard database benchmark with new complex models aimed at being representative of today's computing environment. The TPC-E is based on a financial trading system model. Still, comparative results could not be obtained. The reason is that the TPC specifies requirements regarding what part of the code resides in the "driver" of the benchmark and what part resides in "database" functionality. However, the driver to application logic interface is completely defined at the data level. This means that when measuring relational access you would not incur any overhead for mapping into a C++ object. The mapping of the raw data into what ever form was necessary in the driver to implement the business logic was completely outside of the benchmark measurements. When it comes to the object database, you need to now un-map the C++ objects into the driver data structures and in doing so, measure the cost of that activity as part of the benchmark timings.

But this is the opposite of a real world application where people write object oriented applications resulting in object oriented models. In a relational database, you need to map/un-map from objects to the relational data structures. The TPC-E was written in a way as to exclude the "mapping effect" from the measurements, which by the very nature of how an object database works means the TPC-E was written in a way that forces measurement of an "un-mapping effect", an activity which does not occur in a real world application. Thus with TPC-E, the true cost of computing is removed for relational and even worse added to object databases.

Add-on Modules

Versant provides add-on modules for deployment or access to its Object Database.

- V/Management Center: V/MC delivers real-time views of performance data and analytical information about the Versant Object Database. For example, it alerts administrators about potential issues before the database availability is affected. It's designed as an Eclipse-based RCP client.
- Versant Compact: Online Database Maintenance.
- Versant FTS: High Availability Database Server.
- Versant Async Server: Production Database Replication.
- Versant HA Backup: High Availability Backup Solution.
- Versant SQL: SQL Access & Reporting.

Applications

Usually the “best kind of application” to use a Versant database are those applications requiring an application specific database of an OLTP nature. There are certain application characteristics where Versant technology provides better performance and scalability than traditional relational technology: complex models, large amount of data, large number of concurrent users.

Thus, VOD is found in applications within many different vertical industries: global trading platforms for large stock exchanges, network management for large telecommunications providers, intelligence analytics for defense agencies, reservation systems for large airline/hotel companies, risk management analytics for banking and transportation organizations, Massively multiplayer online game systems, network security and fraud detection, local number portability, advanced simulations, social networking, etc..

References

[1] http://www.versant.com/

[2] "TechView Product Report: Versant Object Database" (http://www.odbms.org/download\048.04 TechView Versant.pdf), *odbms.org*. Retrieved 6 October 2010.

[3] "Poleposition, the open source database benchmark" (http://polepos.sourceforge.net/results/PolePositionClientServer.pdf),*polepos.org*. Retrieved 24 Februar 2011.

[4] "Accelerating IBM WebSphere Application Server Performance with Versant enJin" (http://www.redbooks.ibm.com/abstracts/SG246561.html), *ibm.com*. Retrieved 6 October 2010.

Terminology-oriented database

A **terminology-oriented database** or **terminology-oriented database management system** is a conceptual extension of an object-oriented database. It implements concepts defined in a terminology model. Compared with object-oriented databases, the terminology-oriented database requires some minor conceptual extensions on the schema level as supporting set relations (super-set, subset, intersection etc.), weak-typed collections or shared inheritance.

The data model of a terminology-oriented database is high-level; the terminology-oriented database provides facilities for transforming a terminology model provided by subject area experts completely into a database schema. The target schema might be the database schema for an object-oriented database as well as a relational database schema, or even an XML schema. Typically, terminology-oriented databases are not bound on a specific database type. Since the information content, which can be stored in object-oriented databases and in relational databases, is identical, data for a terminology-oriented database can be stored theoretically in any type of database as well as in an XML file. Thus, terminology-oriented databases may support several database systems for storing application data.

References

Odaba

ODABA is a terminology-oriented database management system, which is a conceptual extension of an object-oriented database system, and implements concepts defined in a terminology model. ODABA supports typical standards and technologies for object-oriented databases, but also terminology-oriented database extensions. ODABA also behaves like an object-relational database management system, i.e. data is seen as being stored in a database rather than accessing persistent objects in a programming environment. ODABA supports active data link[1] (ADL) and provides an ADL-based GUI frame work.

Features

Database access is supported via an application program interface for C++ or .NET programming languages and via the ODABA Script Interface[2] (OSI). Object Definition Language (ODL) and Object Query Language (OQL) provided with OSI are ODMG 3.0 conform.

Beside standard models (object model, functional model and dynamic model), ODABA supports a documentation model and an administration model. In order to be terminology model compliant, several conceptual extensions are supported as set relations, multilingual attributes, weak-typed collections or hierarchical enumerations (classifications).

ODABA supports semi-automatic conversion from terminology models to object models and schema conversion from object model to relational models (MS SQL Server, MySQL, Oracle) which allows storing or mirroring ODABA data in relational databases or in XML files.

References

[1] Active Data Link (http://www.odaba.com/content/downloads/documentation/1.7_ActiveDataLink.pdf)
[2] ODABA Script Interface (http://www.odaba.com/content/downloads/documentation/4.4_OSI.pdf)

External links

- ODABA (http://www.odaba.com/content/products/odaba)

Object Data Management Group

The **Object Data Management Group** (**ODMG**) was conceived in the summer of 1991 at a breakfast with object database vendors that was organized by Rick Cattell of Sun Microsystems. In 1998, the ODMG changed its name from the Object Database Management Group to reflect the expansion of its efforts to include specifications for both object database and object-relational mapping products.

The primary goal of the ODMG was to put forward a set of specifications that allowed a developer to write portable applications for object database and object-relational mapping products. In order to do that, the data schema, programming language bindings, and data manipulation and query languages needed to be portable.

Between 1993 and 2001, the ODMG published five revisions to its specification. The last revision was ODMG version 3.0, after which the group disbanded.

Major components of the ODMG 3.0 specification

- *Object Model.* This was based on the Object Management Group's Object Model. The OMG core model was designed to be a common denominator for object request brokers, object database systems, object programming languages, etc. The ODMG designed a profile by adding components to the OMG core object model.
- *Object Specification Languages.* The ODMG Object Definition Language (ODL) was used to define the object types that conform to the ODMG Object Model. The ODMG Object Interchange Format (OIF) was used to dump and load the current state to or from a file or set of files.
- *Object Query Language (OQL).* The ODMG OQL was a declarative (nonprocedural) language for query and updating. It used SQL as a basis, where possible, though OQL supports more powerful object-oriented capabilities.
- *C++ Language Binding.* This defined a C++ binding of the ODMG ODL and a C++ Object Manipulation Language (OML). The C++ ODL was expressed as a library that provides classes and functions to implement the concepts defined in the ODMG Object Model. The C++ OML syntax and semantics are those of standard C++ in the context of the standard class library. The C++ binding also provided a mechanism to invoke OQL.
- *Smalltalk Language Binding.* This defined the mapping between the ODMG ODL and Smalltalk, which was based on the OMG Smalltalk binding for the OMG Interface Definition Language (IDL). The Smalltalk binding also provided a mechanism to invoke OQL.
- *Java Language Binding.* This defined the binding between the ODMG ODL and the Java programming language as defined by the Java 2 Platform. The Java binding also provided a mechanism to invoke OQL.

Status

ODMG 3.0 was published in book form in 2000.[1] By 2001, most of the major object database and object-relational mapping vendors claimed conformance to the ODMG Java Language Binding. Compliance to the other components of the specification was mixed.[2] In 2001, the ODMG Java Language Binding was submitted to the Java Community Process as a basis for the Java Data Objects specification. The ODMG member companies then decided to concentrate their efforts on the Java Data Objects specification. As a result, the ODMG disbanded in 2001.

In 2004, the Object Management Group (OMG) was granted the right to revise the ODMG 3.0 specification as an OMG specification by the copyright holder, Morgan Kaufmann Publishers. In February 2006, the OMG announced the formation of the Object Database Technology Working Group (ODBT WG) and plans to work on the 4th generation of an object database standard [3].

References

1. ^ *The Object Data Standard: ODMG 3.0.* Edited by R.G.G. Cattell and Douglas K. Barry, with contributions by Mark Berler, Jeff Eastman, David Jordan, Craig L. Russell, Olaf Schadow, Torsten Stanienda, and Fernando Velez. Morgan Kaufmann Publishers, Inc., 2000. ISBN 1-55860-647-5.
2. ^ *Object Storage Fact Books: Object DBMSs and Object-Relational Mapping.* Douglas K. Barry and Joshua Duhl. Barry & Associates, Inc., 2001. Pages showing the ODMG compliance for both object database and object-relational mapping products in 2001. [4]

External links

- ODMG Web site: http://www.odbms.org/ODMG/disbanded since 2001

ODMG Compliant DBMS

- Orient ODBMS: http://www.OrienTechnologies.com
- Objectivity/DB C++, Java and Smalltalk interfaces.

References

[1] http://en.wikipedia.org/wiki/Object_Data_Management_Group#endnote_odmgbook
[2] http://en.wikipedia.org/wiki/Object_Data_Management_Group#endnote_factbook
[3] http://www.odbms.org/About/News/20060218.aspx
[4] http://www.barryandassociates.com/odmg-compliance.html

List of object database management systems

This is a **comparison of notable object database management systems**, showing what fundamental object database features are implemented natively.

Name	Current Stable Version	Language(s)	SQL support	Datatypes	License	Description
Caché	2012.1	ObjectScript (dynamic language), Basic. Java/.NET object mapping supported.	SQL subset. Object notation allowed. Supports embedded SQL, dynamic SQL and xDBC access.		Proprietary	MUMPS ancestry. Includes built-in support for XML, Web/AJAX and an EMB system called Ensemble. Supports embedded, client/server and distributed implementations.
ConceptBase		Telos	CBQL (based on Datalog)	no types but classes	open source, FreeBSD-style license	historical db, active rules, meta-modeling, deductive rules
Db4o	8.0	C#, Java	db4o-sql [1]	.NET and Java data types	GPL, custom,[2] proprietary	Native Queries, LINQ support, automatic schema evolution, Transparent Activation/Persistence, replication to RDBMS, Object Manager plugin for Visual Studio and Eclipse
GemStone/S	3.1.0.5	Smalltalk	None	Objects and code	Proprietary, free version available	Persistent, transactional, multi-user Smalltalk developed by GemTalk Systems.
NeoDatis ODB		C#, Java, Mono			LGPL	Embedded and Client/Server

ObjectDatabase++	3.4	C++, TScript, .NET			Proprietary	Embedded
ObjectDB	2.4.6	Java	None, uses JPA or JDO		Proprietary	
Objectivity/DB	10.2.1	C++, C#, Java, Python, Smalltalk and XML	SQL superset		Proprietary	Distributed, Parallel Query Engine
ObjectStore	7.2 (July 2011)	C++, Java, interoperable with .NET	SQL subset (also has own object query language)		Proprietary	Embedded database supporting efficient, distributed management of C++ and Java objects. Avoids the complexities and limitations of ORM products such as Hibernate by storing objects directly with their relationships intact. Uses a page-based mapping system for fast locking and efficient, distributed, client-side caching.
ODABA		C++, .NET			GPL	Terminology-oriented database
OpenAccess	2.2	C++	no		Proprietary	EDA database
OpenLink Virtuoso	5.0.11	C++, Java/JSP, ASP, ASPX, Mono, RDF, SPARQL, SPARUL, SQL, Perl, Python, PHP, Ruby, XML, ODBC, JDBC, ADO.NET, more	SQL 9x/200x		GPL or proprietary	
Perst	4.2	Java (including Java SE, Java ME & Android), C# (including .NET, .NET Compact Framework, Mono & Silverlight)	JSQL - object-oriented subset of SQL	Java and .NET data types	GPL, Proprietary	Small footprint embedded database. Diverse indexes and specialized collection classes; LINQ; replication; ACID transactions; native full text search; includes Silverlight, Android and Java ME demo apps.
Picolisp	3.1.1	Picolisp			MIT License	DB built into the language
siaqodb [3]	3.7	C#, .NET, Mono, WinRT, Silverlight, Windows Phone, Android, iOS, Unity3D, .NET Compact Framework	LINQ	.NET data types, classes	Commercial	NoSQL embedded database for .NET that runs on .NET, Mono, WinRT, iOS, Android, WindowsPhone, Unity3D,Compact Framework
Twig		Java			Apache license 2.0	Built on Google App Engine's low-level Datastore API
Versant Object Database					Proprietary	
WakandaDB	4	JavaScript, C++	No support. Use REST & SSJS instead	JavaScript and 4D data types	AGPL, proprietary[4]	NoSQL REST / Server-Side JavaScript engine. Integrates Webkit JavaScriptCore engine with HTML5 JS APIs supported on the server. Tables and columns are replaced by JavaScript DataClasses and attributes.

Zope Object Database		Python, C	No support. Object indexing and searching is done through ZCatalog facility.		Zope Public License	

References

[1] http://code.google.com/p/db4o-sql/
[2] http://www.db4o.com/about/company/legalpolicies/docl.aspx
[3] http://siaqodb.com/
[4] Wakanda Commercial license (http://www.wakanda.org/license/commercial)

Comparison of object database management systems

This is a **comparison of notable object database management systems**, showing what fundamental object database features are implemented natively.

Name	Current Stable Version	Language(s)	SQL support	Datatypes	License	Description
Caché	2012.1	ObjectScript (dynamic language), Basic. Java/.NET object mapping supported.	SQL subset. Object notation allowed. Supports embedded SQL, dynamic SQL and xDBC access.		Proprietary	MUMPS ancestry. Includes built-in support for XML, Web/AJAX and an EMB system called Ensemble. Supports embedded, client/server and distributed implementations.
ConceptBase		Telos	CBQL (based on Datalog)	no types but classes	open source, FreeBSD-style license	historical db, active rules, meta-modeling, deductive rules
Db4o	8.0	C#, Java	db4o-sql [1]	.NET and Java data types	GPL, custom,[1] proprietary	Native Queries, LINQ support, automatic schema evolution, Transparent Activation/Persistence, replication to RDBMS, Object Manager plugin for Visual Studio and Eclipse
GemStone/S	3.1.0.5	Smalltalk	None	Objects and code	Proprietary, free version available	Persistent, transactional, multi-user Smalltalk developed by GemTalk Systems.
NeoDatis ODB		C#, Java, Mono			LGPL	Embedded and Client/Server
ObjectDatabase++	3.4	C++, TScript, .NET			Proprietary	Embedded
ObjectDB	2.4.6	Java	None, uses JPA or JDO		Proprietary	
Objectivity/DB	10.2.1	C++, C#, Java, Python, Smalltalk and XML	SQL superset		Proprietary	Distributed, Parallel Query Engine

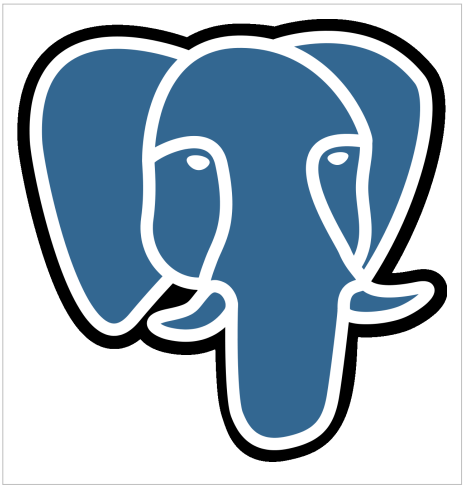
ObjectStore	7.2 (July 2011)	C++, Java, interoperable with .NET	SQL subset (also has own object query language)		Proprietary	Embedded database supporting efficient, distributed management of C++ and Java objects. Avoids the complexities and limitations of ORM products such as Hibernate by storing objects directly with their relationships intact. Uses a page-based mapping system for fast locking and efficient, distributed, client-side caching.
ODABA		C++, .NET			GPL	Terminology-oriented database
OpenAccess	2.2	C++	no		Proprietary	EDA database
OpenLink Virtuoso	5.0.11	C++, Java/JSP, ASP, ASPX, Mono, RDF, SPARQL, SPARUL, SQL, Perl, Python, PHP, Ruby, XML, ODBC, JDBC, ADO.NET, more	SQL 9x/200x		GPL or proprietary	
Perst	4.2	Java (including Java SE, Java ME & Android), C# (including .NET, .NET Compact Framework, Mono & Silverlight)	JSQL - object-oriented subset of SQL	Java and .NET data types	GPL, Proprietary	Small footprint embedded database. Diverse indexes and specialized collection classes; LINQ; replication; ACID transactions; native full text search; includes Silverlight, Android and Java ME demo apps.
Picolisp	3.1.1	Picolisp			MIT License	DB built into the language
siaqodb [3]	3.7	C#, .NET, Mono, WinRT, Silverlight, Windows Phone, Android, iOS, Unity3D, .NET Compact Framework	LINQ	.NET data types, classes	Commercial	NoSQL embedded database for .NET that runs on .NET, Mono, WinRT, iOS, Android, WindowsPhone, Unity3D,Compact Framework
Twig		Java			Apache license 2.0	Built on Google App Engine's low-level Datastore API
Versant Object Database					Proprietary	
WakandaDB	4	JavaScript, C++	No support. Use REST & SSJS instead	JavaScript and 4D data types	AGPL, proprietary[2]	NoSQL REST / Server-Side JavaScript engine. Integrates Webkit JavaScriptCore engine with HTML5 JS APIs supported on the server. Tables and columns are replaced by JavaScript DataClasses and attributes.
Zope Object Database		Python, C	No support. Object indexing and searching is done through ZCatalog facility.		Zope Public License	

References

[1] http://www.db4o.com/about/company/legalpolicies/docl.aspx
[2] Wakanda Commercial license (http://www.wakanda.org/license/commercial)

PostgreSQL

PostgreSQL



Developer(s)	PostgreSQL Global Development Group
Initial release	1 May 1995
Stable release	9.3.3, 9.2.7, 9.1.12, 9.0.16, 8.4.20 / February 20, 2014
Written in	C
Operating system	Cross-platform
Type	ORDBMS
License	PostgreSQL license
Website	www.postgresql.org [1]

PostgreSQL, often simply **"Postgres"**, is an open-source object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance. It is released under the PostgreSQL License, a free/open source software license, similar to the MIT License. PostgreSQL is developed by the PostgreSQL Global Development Group, consisting of a handful of volunteers employed and supervised by companies such as Red Hat and EnterpriseDB. It implements the majority of the SQL:2011 standard, is ACID-compliant, is fully transactional (including all DDL statements), has extensible updateable views, data-types, operators, index methods, functions, aggregates, procedural languages, and has a large number of extensions written by third parties. PostgreSQL runs on many operating-systems including Linux, FreeBSD, Solaris, Microsoft Windows and Mac OS X.

The vast majority of Linux distributions have PostgreSQL available in supplied packages. OS X, starting with Lion, has PostgreSQL server as its standard default database in the server edition, and PostgreSQL client tools in the desktop edition.

Name

PostgreSQL's developers pronounce it /ˈpoʊstgrɛsHelp:IPA for English#Key˛kjuːHelp:IPA for English#Keyˈɛl/; (Audio sample [2], 5.6k MP3). It is abbreviated as *Postgres*, its original name. Because of ubiquitous support for the SQL Standard amongst most relational databases, the community considered changing the name back to Postgres. However, the PostgreSQL Core Team announced in 2007 that the product would continue to use the name PostgreSQL. The name refers to the project's origins as a "post-Ingres" database, being a development from University Ingres DBMS (*Ingres* being an abbreviation for *IN*teractive *G*raphics *RE*trieval *S*ystem).[3]

History

PostgreSQL evolved from the Ingres project at the University of California, Berkeley. In 1982, the project leader, Michael Stonebraker, left Berkeley to make a proprietary version of Ingres. He returned to Berkeley in 1985 and started a post-Ingres project to address the problems with contemporary database systems that had become increasingly clear during the early 1980s. The new project, POSTGRES, aimed to add the fewest features needed to completely support types. These features included the ability to define types and to fully describe relationships – something used widely before but maintained entirely by the user. In Postgres, the database "understood" relationships, and could retrieve information in related tables in a natural way using *rules*. Postgres used many of the ideas of Ingres, but not its code.

Starting in 1986, the team published a number of papers describing the basis of the system, and by 1988 had a prototype version. The team released version 1 to a small number of users in June 1989, then version 2 with a re-written rules system in June 1990. Version 3, released in 1991, again re-wrote the rules system, and added support for multiple storage managers and an improved query engine. By 1993 the great number of users began to overwhelm the project with requests for support and features. After releasing version 4—primarily a cleanup—the project ended.

But open-source developers could obtain copies and develop the system further, because Berkeley had released Postgres under an MIT-style license. In 1994, Berkeley graduate students Andrew Yu and Jolly Chen replaced the Ingres-based QUEL query language interpreter with one for the SQL query language, creating Postgres95. The code was released on the web.

In July 1996, Marc Fournier at Hub.Org Networking Services provided the first non-university development server for the open-source development effort. Along with Bruce Momjian and Vadim B. Mikheev, work began to stabilize the code inherited from Berkeley. The first open-source version was released on August 1, 1996.

In 1996, the project was renamed to PostgreSQL to reflect its support for SQL. The first PostgreSQL release formed version 6.0 in January 1997. Since then, the software has been maintained by a group of database developers and volunteers around the world, coordinating via the Internet.

The PostgreSQL project continues to make major releases (approximately annually) and minor "bugfix" releases, all available under the same license. Code comes from contributions from proprietary vendors, support companies, and open-source programmers at large.

Features

MVCC

PostgreSQL manages concurrency through a system known as multiversion concurrency control (MVCC), which gives each transaction a "snapshot" of the database, allowing changes to be made without being visible to other transactions until the changes are committed. This largely eliminates the need for read locks, and ensures the database maintains the ACID (atomicity, consistency, isolation, durability) principles in an efficient manner. PostgreSQL offers three levels of transaction isolation: Read Committed, Repeatable Read and Serializable. Because PostgreSQL is immune to dirty reads, requesting a Read Uncommitted transaction isolation level provides read committed instead. Prior to PostgreSQL 9.1, requesting Serializable provided the same isolation level as Repeatable Read. PostgreSQL 9.1 and later support full serializability via the serializable snapshot isolation (SSI) technique.

Procedural languages

Procedural languages allow developers to extend the database with custom subroutines (functions), often called *stored procedures*. These functions can be used to build triggers (functions invoked upon modification of certain data) and custom aggregate functions. Procedural languages can also be invoked without defining a function, using the "DO" command at SQL level.

Languages are divided into two groups: "Safe" languages are sandboxed and can be safely used by any user. Procedures written in "unsafe" languages can only be created by superusers, because they allow bypassing the database's security restrictions, but can also access sources external to the database. Some languages like Perl provide both safe and unsafe versions.

PostgreSQL has built-in support for three procedural languages:

- Plain SQL (safe). Simpler SQL functions can get expanded inline into the calling (SQL) query, which saves function call overhead and allows the query optimizer to "see inside" the function.
- PL/pgSQL (safe), which resembles Oracle's PL/SQL procedural language and SQL/PSM.
- C (unsafe), which allows loading custom shared libraries into the database. Functions written in C offer the best performance, but bugs in code can crash and potentially corrupt the database. Most built-in functions are written in C.

In addition, PostgreSQL allows procedural languages to be loaded into the database through extensions. Three language extensions are included with PostgreSQL to support Perl, Python and Tcl. There are external projects to add support for many other languages, including Java, JavaScript (PL/V8), R.

Indexes

PostgreSQL includes built-in support for regular B-tree and hash indexes, and two types of inverted indexes: generalized search trees (GiST) and generalized inverted indexes (GIN). Hash indexes are implemented, but discouraged because they cannot be recovered after a crash or power loss. In addition, user-defined index methods can be created, although this is quite an involved process. Indexes in PostgreSQL also support the following features:

- Expression indexes can be created with an index of the result of an expression or function, instead of simply the value of a column.
- Partial indexes, which only index part of a table, can be created by adding a `WHERE` clause to the end of the `CREATE INDEX` statement. This allows a smaller index to be created.
- The planner is capable of using multiple indexes together to satisfy complex queries, using temporary in-memory bitmap index operations.
- As of PostgreSQL 9.1, *k*-nearest neighbor (also referred to KNN-GiST) indexing provides efficient searching of "closest values" to that specified, useful to finding similar words, or close objects or locations with geospatial

data. This is achieved without exhaustive matching of values.
- In PostgreSQL 9.2 and above, index-only scans often allow the system to fetch data from indexes without ever having to access the main table.

Triggers

Triggers are events triggered by the action of SQL DML statements. For example, an INSERT statement might activate a trigger that checks if the values of the statement are valid. Most triggers are only activated by either INSERT or UPDATE statements.

Triggers are fully supported and can be attached to tables. In PostgreSQL 9.0 and above, triggers can be per-column and conditional, in that UPDATE triggers can target specific columns of a table, and triggers can be told to execute under a set of conditions as specified in the trigger's WHERE clause. As of PostgreSQL 9.1, triggers can be attached to views by utilising the INSTEAD OF condition. Views in versions prior to 9.1 can have rules, though. Multiple triggers are fired in alphabetical order. In addition to calling functions written in the native PL/PgSQL, triggers can also invoke functions written in other languages like PL/Python or PL/Perl.

Schemas

In PostgreSQL, all objects (with the exception of roles and tablespaces) are held within a schema. Schemas effectively act like namespaces, allowing objects of the same name to co-exist in the same database. Schemas are analogous to directories in a file system, except that they cannot be nested, nor is it possible to create a "symbolic link" pointing to another schema or object.

By default, databases are created with the "public" schema, but any additional schemas can be added, and the public schema isn't mandatory. A "search_path" determines the order in which schemas are checked on unqualified objects (those without a prefixed schema), which can be configured on a database or role level. The search path, by default, contains the special schema name of "$user", which first looks for a schema named after the connected database user (e.g. if the user "dave" were connected, it would first look for a schema also named "dave" when referring to any objects). If such a schema is not found, it then proceeds to the next schema. New objects are created in whichever valid schema (one that presently exists) is listed first in the search path.

Data types

A wide variety of native data types are supported, including:

- Boolean
- Arbitrary precision numerics
- Character (text, varchar, char)
- Binary
- Date/time (timestamp/time with/without timezone, date, interval)
- Money
- Enum
- Bit strings
- Text search type
- Composite
- HStore (an extension enabled key-value store within Postgres)
- Arrays (variable length and can be of any data type, including text and composite types) up to 1 GB in total storage size.
- Geometric primitives
- IPv4 and IPv6 addresses
- CIDR blocks and MAC addresses

- XML supporting XPath queries
- UUID
- JSON (versions 9.2 and up)

In addition, users can create their own data types which can usually be made fully indexable via PostgreSQL's GiST infrastructure. Examples of these include the geographic information system (GIS) data types from the PostGIS project for PostgreSQL.

There is also a data type called a "domain", which is the same as any other data type but with optional constraints defined by the creator of that domain. This means any data entered into a column using the domain will have to conform to whichever constraints were defined as part of the domain.

Range Types

Starting with PostgreSQL 9.2, a data type that represents a range of data can be used which are called range types. These can be discrete ranges (e.g. all integer values 1 to 10) or continuous ranges (e.g. any point in time between 10:00am and 11:00am). The built-in range types available include ranges of integers, big integers, decimal numbers, time stamps (with and without time zone) and dates.

Custom range types can be created to make new types of ranges available, such as IP address ranges using the inet type as a base, or float ranges using the float data type as a base. Range types support inclusive and exclusive range boundaries using the [] and () characters respectively. (e.g. '[4,9)' represents all integers starting from and including 4 up to but not including 9.) Range types are also compatible with existing operators used to check for overlap, containment, right of etc.

User-defined objects

New types of almost all objects inside the database can be created, including:

- Casts
- Conversions
- Data types
- Domains
- Functions, including aggregate functions and window functions
- Indexes including custom indexes for custom types
- Operators (existing ones can be overloaded)
- Procedural languages

Inheritance

Tables can be set to inherit their characteristics from a "parent" table. Data in child tables will appear to exist in the parent tables, unless data is selected from the parent table using the ONLY keyword, i.e. `SELECT * FROM ONLY parent_table`. Adding a column in the parent table will cause that column to appear in the child table.

Inheritance can be used to implement table partitioning, using either triggers or rules to direct inserts to the parent table into the proper child tables.

As of 2010[4] this feature is not fully supported yet—in particular, table constraints are not currently inheritable. All check constraints and not-null constraints on a parent table are automatically inherited by its children. Other types of constraints (unique, primary key, and foreign key constraints) are not inherited.

Inheritance provides a way to map the features of generalization hierarchies depicted in Entity Relationship Diagrams (ERD) directly into the PostgreSQL database.

Rules

Rules allow the "query tree" of an incoming query to be rewritten. Rules, or more properly, "Query Re-Write Rules", are attached to a table/class and "Re-Write" the incoming DML (select, insert, update, and/or delete) into one or more queries that either replace the original DML statement or execute in addition to it. Query Re-Write occurs after DML statement parsing, but before query planning.

Replication

Binary replication

PostgreSQL, beginning from version 9.0, includes built-in binary replication, based on shipping the changes (write-ahead logs) to slave systems asynchronously.

Version 9.0 also introduced the ability to run read-only queries against these replicated slaves, where earlier versions would only allow that after promoting them to be a new master. This allows splitting read traffic among multiple nodes efficiently. Earlier replication software that allowed similar read scaling normally relied on adding replication triggers to the master, introducing additional load onto it.

Beginning from version 9.1, PostgreSQL also includes built-in synchronous replication that ensures that, for each write transaction, the master waits until at least one slave node has written the data to its transaction log. Unlike other database systems, the durability of a transaction (whether it's asynchronous or synchronous) can be specified per-database, per-user, per-session or even per-transaction. This can be useful for work loads that don't require such guarantees, and may not be wanted for all data as it will have some negative effect on performance due to the requirement of the confirmation of the transaction reaching the synchronous standby.

There can be a mixture of synchronous and asynchronous standby servers. A list of synchronous standby servers can be specified in the configuration which determines which servers are candidates for synchronous replication. The first in the list which is currently connected and actively streaming is the one that will be used as the current synchronous server. When this fails, it falls to the next in line.

Synchronous multi-master replication is currently not included in the PostgreSQL core. Postgres-XC which is based on PostgreSQL provides scalable synchronous multi-master replication and is licensed under the BSD license.

The community has also written some tools to make managing replication clusters easier, such as repmgr.

Trigger-based replication

There are also several asynchronous trigger-based replication packages for PostgreSQL. These remain useful even after introduction of the expanded core capabilities, for situations where binary replication of an entire database cluster isn't the appropriate approach:

- Slony-I
- Londiste, part of SkyTools (developed by Skype)
- Bucardo multi-master replication (developed by Backcountry.com)
- SymmetricDS multi-master, multi-tier replication

Asynchronous notifications

PostgreSQL provides an asynchronous messaging system that is accessed through the NOTIFY, LISTEN and UNLISTEN commands. A session can issue a NOTIFY command, along with the user-specified channel and an optional payload, to mark a particular event occurring. Other sessions are able to detect these events by issuing a LISTEN command, which can listen to a particular channel. This functionality can be used for a wide variety of purposes, such as letting other sessions know when a table has updated or for separate applications to detect when a particular action has been performed. Such a system prevents the need for continuous polling by applications to see if anything has yet changed, and reducing unnecessary overhead. Notifications are fully transactional, in that

messages aren't sent until the transaction they were sent from is committed. This eliminates the problem of messages being sent for an action being performed which is then rolled back.

Many of the connectors for PostgreSQL provide support for this notification system (including libpq, JDBC, Npgsql, psycopg and node.js) so it can be used by external applications.

Security

Internal

Security within the database is managed on a per-role-basis. A role is generally regarded to be a user (a role that can log in), or a group (a role which other roles are members of). Permissions can be granted or revoked on any object down to the column level, and can also allow/prevent the creation of new objects at the database, schema or table levels.

The sepgsql extension (provided with PostgreSQL as of version 9.1) provides an additional layer of security by integrating with SELinux. This utilises PostgreSQL's SECURITY LABEL feature.

External

PostgreSQL natively supports a broad number of authentication mechanisms including:

- trust (no enforcement)
- password (either MD5 or plain-text)
- GSSAPI
- SSPI
- Kerberos
- ident (maps O/S user name as provided by an ident server to database user name)
- peer (maps local user name to database user name)
- LDAP
- RADIUS
- certificate
- PAM

The GSSAPI, SSPI, Kerberos, peer, ident and certificate methods can also use a specified "map" file that lists which users matched by that authentication system are allowed to connect as a specific database user.

These methods are specified in the cluster's host-based authentication configuration file (pg_hba.conf), which determines what connections are allowed. This allows control over which user can connect to which database, where they can connect from (IP address/IP address range/domain socket), which authentication system will be enforced, and whether the connection must use SSL.

Foreign data wrappers

As of version 9.1, PostgreSQL can link to other systems to retrieve data via foreign data wrappers (FDWs). These can take the form of any data source, such as a file system, another RDBMS, or a web service. This means regular database queries can use these data sources like regular tables, and even join multiple data sources together.

Other features

- Referential integrity constraints including foreign key constraints, column constraints, and row checks
- Views
  - Materialized views
  - Updatable views
  - Recursive views

- Inner, outer (full, left and right), and cross joins
- Sub-selects
  - Correlated sub-queries
- Transactions
- Supports most of the major features of SQL:2008 standard
- Encrypted connections via SSL
- Binary and textual large-object storage
- Online backup
- In-place upgrades with pg_upgrade (available for upgrading to new major versions from 8.3 upwards)
- Domains
- Tablespaces
- Savepoints
- Point-in-time recovery, implemented using write-ahead logging
- Two-phase commit
- TOAST (**T**he **O**versized-**A**ttribute **S**torage **T**echnique) is used to transparently store large table attributes (such as big MIME attachments or XML messages) in a separate area, with automatic compression.
- Regular expressions
- Common table expressions and writable common table expressions
- Embedded SQL is implemented using preprocessor. SQL code is first written embedded into C code. Then code is run through ECPG preprocessor, which replaces SQL with calls to code library. Then code can be compiled using a C compiler. Embedding works also with C++ but it does not recognize all C++ constructs.
- Full text search
- Per-column collation (from 9.1)

Add-ons

- MADlib - an open source analytics library for PostgreSQL providing mathematical, statistical and machine-learning methods for structured and unstructured data.
- MySQL migration wizard - included with EnterpriseDB's PostgreSQL installer. (source code also available)
- Performance Wizard - included with EnterpriseDB's PostgreSQL installer. (source code also available)
- pgRouting - extended PostGIS to provide geospatial routing functionality (GNU GPL)
- PostGIS - a popular add-on which provides support for geographic objects GNU GPL.
- Postgres Enterprise Manager - a non-free tool consisting of a service, multiple agents, and a GUI which provides remote monitoring, management, reporting, capacity planning and tuning.
- ST-Links SpatialKit - Extension for directly connecting to spatial databases.

Upcoming features in 9.4

In order of commit:

- Concurrent refresh for materialised views
- FILTER clause for aggregate functions
- WITH CHECK clause for auto-updatable views
- WITH ORDINALITY support
- Multi-argument UNNEST(), and TABLE() syntax
- ALTER SYSTEM command to change persistent configuration from within database
- pg_prewarm extension to load relation data into the buffer cache of either the OS or PostgreSQL
- Ordered-set (WITHIN GROUP) aggregates
- ALTER TABLESPACE ... MOVE command for moving objects in bulk between tablespaces

- Replication slots for automatic WAL tracking to eliminate replication conflicts
- Many GIN Index enhancements to speed up searching and significantly reduce index size
- Logical decoding of WAL data into logical changes

Interfaces

PostgreSQL has several forms of interface available and is also widely supported among programming language libraries. These include:

Built-in

- libpq - PostgreSQL's official C application interface
- ECPG - An embedded C system

External

- libpqxx - C++ interface
- PostgresDAC - PostgresDAC (for Embarcadero RadStudio/Delphi/CBuilder XE-XE3)
- DBD::Pg - Perl DBI driver
- JDBC - JDBC interface
- Lua - Lua interface
- Npgsql - .NET data provider
- ST-Links SpatialKit - Link Tool to ArcGIS
- node-postgres - Node.js interface
- pgoledb - OLEDB interface
- psqlODBC - ODBC interface
- psycopg - Python interface (also used by HTSQL)
- pgtclng - Tcl interface
- pyODBC - Python library

Platforms

Operating systems

PostgreSQL is available for the following operating systems: Linux (all recent distributions), Windows (Windows 2000 SP4 and later) (compilable by eg. Visual Studio, now with up to most recent 2013 version), DragonFly_BSD, FreeBSD, OpenBSD, NetBSD, Mac OS X, AIX, BSD/OS, HP-UX, IRIX, OpenIndiana, OpenSolaris, SCO OpenServer, SCO UnixWare, Solaris and Tru64 Unix. As of 2012, support for the following obsolete systems was removed: DG/UX, NeXTSTEP, SunOS 4, SVR4, Ultrix 4, and Univel. Most other Unix-like systems should also work.

Instruction set architectures

PostgreSQL works on any of the following instruction set architectures: x86 and x86-64 on Windows and other operatings systems; other than Windows: IA-64 Itanium, PowerPC, PowerPC 64, S/390, S/390x, SPARC, SPARC 64, Alpha, ARMv8-A (64-bit)[5] and older ARM (32-bit), MIPS, MIPSel, M68k, and PA-RISC. It is also known to work on M32R, NS32k, and VAX. In addition to these, it is possible to build PostgreSQL for an unsupported CPU by disabling spinlocks.

Database administration

Open source front-ends and tools

psql

The primary front-end for PostgreSQL is the `psql` command-line program, which can be used to enter SQL queries directly, or execute them from a file. In addition, psql provides a number of meta-commands and various shell-like features to facilitate writing scripts and automating a wide variety of tasks; for example tab completion of object names and SQL syntax.

pgAdmin

The pgAdmin package is a free and open source graphical user interface administration tool for PostgreSQL, which is supported on many computer platforms. The program is available in more than a dozen languages. The first prototype, named pgManager, was written for PostgreSQL 6.3.2 from 1998, and rewritten and released as pgAdmin under the GPL License in later months. The second incarnation (named pgAdmin II) was a complete rewrite, first released on January 16, 2002. The third version, pgAdmin III, was originally released under the Artistic License and then released under the same license as PostgreSQL. Unlike prior versions that were written in Visual Basic, pgAdmin III is written in C++, using the wxWidgets framework allowing it to run on most common operating systems.

phpPgAdmin

phpPgAdmin is a web-based administration tool for PostgreSQL written in PHP and based on the popular phpMyAdmin interface originally written for MySQL administration.

PostgreSQL Studio

PostgreSQL Studio allows users to perform essential PostgreSQL database development tasks from a web-based console. PostgreSQL Studio allows users to work with cloud databases without the need to open firewalls.

TeamPostgreSQL

AJAX/JavaScript-driven web interface for PostgreSQL. Allows browsing, maintaining and creating data and database objects via a web browser. The interface offers tabbed SQL editor with auto-completion, row-editing widgets, click-through foreign key navigation between rows and tables, 'favorites' management for commonly used scripts, among other features. Supports SSH for both the web interface and the database connections. Installers are available for Windows, Mac and Linux, as well as a simple cross-platform archive that runs from a script.

LibreOffice/OpenOffice.org Base

LibreOffice/OpenOffice.org Base can be used as a front-end for PostgreSQL.

pgFouine

The pgFouine PostgreSQL log analyzer generates detailed reports from a PostgreSQL log file and provides VACUUM analysis.

Proprietary front-ends and tools

A number of companies offer proprietary tools for PostgreSQL. They often consist of a universal core that is adapted for various specific database products. These tools mostly share the administration features with the open source tools but offer improvements in data modeling, importing, exporting or reporting.

Benchmarks and performance

Many informal performance studies of PostgreSQL have been done. Performance improvements aimed at improving scalability started heavily with version 8.1. Simple benchmarks between version 8.0 and version 8.4 showed that the latter was more than 10 times faster on read-only workloads and at least 7.5 times faster on both read and write workloads.

The first industry-standard and peer-validated benchmark was completed in June 2007 using the Sun Java System Application Server (proprietary version of GlassFish) 9.0 Platform Edition, UltraSPARC T1-based Sun Fire server and Postgres 8.2. This result of 778.14 SPECjAppServer2004 JOPS@Standard compares favourably with the 874 JOPS@Standard with Oracle 10 on an Itanium-based HP-UX system.

In August 2007, Sun submitted an improved benchmark score of 813.73 SPECjAppServer2004 JOPS@Standard. With the system under test at a reduced price, the price/performance improved from $US 84.98/JOPS to $US 70.57/JOPS.

The default configuration of PostgreSQL uses only a small amount of dedicated memory for performance-critical purposes such as caching database blocks and sorting. This limitation is primarily because older operating systems required kernel changes to allow allocating large blocks of shared memory. PostgreSQL.org provides advice on basic recommended performance practice in a wiki.

In April 2012, Robert Haas of EnterpriseDB demonstrated PostgreSQL 9.2's linear CPU scalability using a server with 64 cores.

Prominent users

- Yahoo! for web user behavioral analysis, storing two petabytes and claimed to be the largest data warehouse using a heavily modified version of PostgreSQL with an entirely different column-based storage engine and different query processing layer. While for performance, storage, and query purposes the database bears little resemblance to PostgreSQL, the front-end maintains compatibility so that Yahoo can use many off-the-shelf tools already written to interact with PostgreSQL.
- In 2009, social networking website MySpace used Aster Data Systems's nCluster database for data warehousing, which was built on unmodified PostgreSQL.
- State Farm uses PostgreSQL on their Aster Data Systems's nCluster Analytics server.
- Geni.com uses PostgreSQL for their main genealogy database.
- OpenStreetMap, a collaborative project to create a free editable map of the world.
- Afilias, domain registries for .org, .info and others.
- Sony Online multiplayer online games.
- BASF, shopping platform for their agribusiness portal.
- Reddit social news website.
- Skype VoIP application, central business databases.
- Sun xVM, Sun's virtualization and datacenter automation suite.
- MusicBrainz, open online music encyclopedia.
- International Space Station for collecting telemetry data in orbit and replicating it to the ground.
- MyYearbook social networking site.
- Instagram, a popular mobile photo sharing service
- Disqus, an online discussion and commenting service

PostgreSQL as a service

Heroku, a platform as a service provider, has supported PostgreSQL since the start in 2007. They offer value-add features like full database "roll-back" (ability to restore a database from any point in time), which is based on WAL-E, open source software developed by Heroku.

In January 2012 EnterpriseDB released a cloud version of both PostgreSQL and their own proprietary Postgres Plus Advanced Server with automated provisioning for failover, replication, load-balancing, and scaling. It runs on Amazon Web Services.

VMware offers vFabric Postgres for private clouds on vSphere since 2012 May.

In 2013 November, Amazon announced that they are adding PostgreSQL to their Relational Database Service offering.

Proprietary derivatives and support

Although the license allowed proprietary products based on Postgres, the code did not develop in the proprietary space at first. The first main offshoot originated when Paula Hawthorn (an original Ingres team member who moved from Ingres) and Michael Stonebraker formed Illustra Information Technologies to make a proprietary product based on Postgres.

In 2000, former Red Hat investors created the company Great Bridge to make a proprietary product based on PostgreSQL and compete against proprietary database vendors. Great Bridge sponsored several PostgreSQL developers and donated many resources back to the community, but by late 2001 closed due to tough competition from companies like Red Hat and to poor market conditions.

In 2001 Command Prompt, Inc. released Mammoth PostgreSQL, a proprietary product based on PostgreSQL. In 2008 Command Prompt, Inc. released the source under the original license. Command Prompt, Inc. continues to support the PostgreSQL community actively through developer sponsorships and projects including PL/Perl, PL/php, and hosting of community projects such as the PostgreSQL build farm.

In January 2005, PostgreSQL received backing by database vendor Pervasive Software, known for its Btrieve product which was ubiquitous on the Novell NetWare platform. Pervasive announced commercial support and community participation and achieved some success. In July 2006, Pervasive left the PostgreSQL support market.

In mid-2005 two other companies announced plans to make proprietary products based on PostgreSQL with focus on separate niche markets. EnterpriseDB added functionality to allow applications written to work with Oracle to be more readily run with PostgreSQL. Greenplum contributed enhancements directed at data warehouse and business intelligence applications, including the BizGres project.

In October 2005 John Loiacono, executive vice president of software at Sun Microsystems, commented: "We're not going to OEM Microsoft but we are looking at PostgreSQL right now," although no specifics were released at that time. By November 2005 Sun had announced support for PostgreSQL. By June 2006 Sun Solaris 10 (6/06 release) shipped with PostgreSQL.

In August 2007, EnterpriseDB announced EnterpriseDB Postgres, a pre-configured distribution of PostgreSQL including many contrib modules and add-on components. EnterpriseDB Postgres was renamed to Postgres Plus in March 2008. Postgres Plus is available in two versions: Postgres Plus Solution Pack (comprising PostgreSQL delivered in a GUI one-click install plus Solution Pack components that include; Postgres Enterprise Manager, Update Monitor, xDB Replication Server, SQL Profiler, SQL Protect, Migration Toolkit and PL/Secure), and Postgres Plus Advanced Server which has all the features of Postgres Plus Solutions Pack plus Oracle compatibility, performance features not available in PostgreSQL, as well as advanced security features not available in PostgreSQL. Both versions are available for download at no cost and are fully supported. The Solution Pack components and Advanced Server are restricted by a "limited use" license for evaluation purposes only unless purchased though a subscription. In 2011, EnterpriseDB announced Postgres Plus Cloud Database, which easily

provisions PostgreSQL and Postgres Plus Advanced Server databases (with Oracle compatibility) in single instances, high availability clusters, or development sandboxes for Database-as-a-Service environments.

In 2011, 2ndQuadrant became a Platinum Sponsor of PostgreSQL, in recognition of their long-standing contributions and developer sponsorship. 2ndQuadrant employ one of the largest teams of PostgreSQL contributors and provide professional support for open source PostgreSQL.

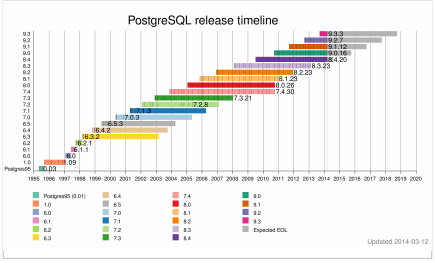
Many other companies have used PostgreSQL as the base for their proprietary database projects. e.g. Truviso, Netezza, ParAccel. In many cases the products have been enhanced so much that the software has been forked, though with some features cherry-picked from later releases.

Release history

Release	First release	Latest minor version	Latest release	Milestones
0.01	1995-05-01	0.03	1995-07-21	Initial release as Postgres95
1.0	1995-09-05	1.09	1996-11-04	Changed copyright to a more liberal license
6.0	1997-01-29	—		Name change from Postgres95 to PostgreSQL, unique indexes, pg_dumpall utility, ident authentication.
6.1	1997-06-08	6.1.1	1997-07-22	Multi-column indexes, sequences, money data type, GEQO (GEnetic Query Optimizer).
6.2	1997-10-02	6.2.1	1997-10-17	JDBC interface, triggers, server programming interface, constraints.
6.3	1998-03-01	6.3.2	1998-04-07	SQL92 subselect capability, PL/pgTCL
6.4	1998-10-30	6.4.2	1998-12-20	VIEWs and RULEs, PL/pgSQL
6.5	1999-06-09	6.5.3	1999-10-13	MVCC, temporary tables, more SQL statement support (CASE, INTERSECT, and EXCEPT)
7.0	2000-05-08	7.0.3	2000-11-11	Foreign keys, SQL92 syntax for joins
7.1	2001-04-13	7.1.3	2001-08-15	Write-ahead log, Outer joins
7.2	2002-02-04	7.2.8	2005-05-09	PL/Python, OIDs no longer required, internationalization of messages
7.3	2002-11-27	7.3.21	2008-01-07	Schema, Internationalization
7.4	2003-11-17	7.4.30	2010-10-04	Optimization all-round
8.0	2005-01-19	8.0.26	2010-10-04	Native server on Microsoft Windows, savepoints, tablespaces, exception handling in functions, point-in-time recovery
8.1	2005-11-08	8.1.23	2010-12-16	Performance optimization, two-phase commit, table partitioning, index bitmap scan, shared row locking, roles
8.2	2006-12-05	8.2.23	2011-09-26	Performance optimization, online index builds, advisory locks, warm standby
8.3	2008-02-04	8.3.23	2013-02-07	Heap-only tuples, full text search, SQL/XML, ENUM types, UUID types
8.4	2009-07-01	8.4.20	2014-02-20	Windowing functions, default and variadic parameters for functions, column-level permissions, parallel database restore, per-database collation, common table expressions and recursive queries
9.0	2010-09-20	9.0.16	2014-02-20	Built-in binary streaming replication, Hot standby, 64-bit Windows, per-column triggers and conditional trigger execution, exclusion constraints, anonymous code blocks, named parameters, password rules
9.1	2011-09-12	9.1.12	2014-02-20	Synchronous replication, per-column collations, unlogged tables, *k*-nearest-neighbor indexing, serializable snapshot isolation, writeable common table expressions, SE-Linux integration, extensions, SQL/MED attached tables (Foreign Data Wrappers), triggers on views
9.2	2012-09-10	9.2.7	2014-02-20	Cascading streaming replication, index-only scans, native JSON support, improved lock management, range types, pg_receivexlog tool, space-partitioned GiST indexes

9.3	2013-09-09	9.3.3	2014-02-20	Custom background workers, data checksums, dedicated JSON operators, LATERAL JOIN, faster pg_dump, new pg_isready server monitoring tool, trigger features, view features, writeable foreign tables, materialized views, replication improvements

Community-supported
Community support ended



References

[1] http://www.postgresql.org/
[2] http://www.postgresql.org/files/postgresql.mp3
[3] http://www.postgresql.org/about/history/
[4] http://en.wikipedia.org/w/index.php?title=PostgreSQL&action=edit
[5] http://lists.debian.org/debian-devel/2012/07/msg00536.html

Further reading

PostgreSQL 9.1–9.2

- Obe, Regina; Hsu, Leo (July 8, 2012). *PostgreSQL: Up and Running* (http://www.postgresonline.com/store.php?asin=1449326331). O'Reilly. ISBN 1-4493-2633-1.

PostgreSQL 9

- Krosing, Hannu; Roybal, Kirk (June 15, 2013). *PostgreSQL Server Programming* (http://www.2ndQuadrant.com/books/). Packt Publishing. ISBN 9781849516983.
- Riggs, Simon; Krosing, Hannu (October 27, 2010). *PostgreSQL 9 Administration Cookbook* (http://www.2ndquadrant.com/books/). Packt Publishing. ISBN 1-84951-028-8.
- Smith, Greg (October 15, 2010). *PostgreSQL 9 High Performance* (http://www.2ndQuadrant.com/books/). Packt Publishing. ISBN 1-84951-030-X.

PostgreSQL 8

- Gilmore, W. Jason; Treat, Robert (February 27, 2006). *Beginning PHP and PostgreSQL 8: From Novice to Professional* (http://www.apress.com/book/view/1590595475). Apress. ISBN 1-59059-547-5. 896 pp.
- Douglas, Korry (August 5, 2005). *PostgreSQL* (http://www.informit.com/store/product.aspx?isbn=0672327562) (Second ed.). Sams. ISBN 0-672-32756-2. 1032 pp.
- Matthew, Neil; Stones, Richard (April 6, 2005). *Beginning Databases with PostgreSQL* (http://www.apress.com/book/view/9781590594780) (Second ed.). Apress. ISBN 1-59059-478-9. 664 pp.

PostgreSQL 7

- Worsley, John C; Drake, Joshua D (January 2002). *Practical PostgreSQL* (http://oreilly.com/catalog/9781565928466/). O'Reilly Media. ISBN 1-56592-846-6. 636 pp.

External links

- Official website (http://www.postgresql.org)
- PGXN (http://www.pgxn.org/) – PostgreSQL Extension Network
- PostgreSQL (http://www.dmoz.org/Computers/Software/Databases/PostgreSQL/) on the Open Directory Project
- Heroku Postgres (https://postgres.heroku.com/)
- Postgres.app (http://www.postgresapp.com/)
- ST-Links SpatialKit (http://www.st-links.com/)

PL/pgSQL

PL/pgSQL (**Procedural Language/PostgreSQL**) is a procedural programming language supported by the PostgreSQL ORDBMS. It closely resembles Oracle's PL/SQL language. With PosgreSQL v9.X some ISO SQL/PSM features, like overloading of SQL-invoked functions and procedures,[1] are supported.

PL/pgSQL, as a fully featured programming language, allows much more procedural control than SQL, including the ability to use loops and other control structures. Functions created in the PL/pgSQL language can be called from an SQL statement, or as the action that a trigger performs.

PL/pgSQL was created to be able to perform more complex operations and computations than SQL, while being easy to use, and is able to be defined as trusted by the server.

PL/pgSQL is the only programming language installed by default for PostgreSQL, but many others are available, including PL/Java [2], PL/Perl [3], PL/php [4], PL/Python [5], PL/R [6], PL/Ruby [7], PL/sh [8], PL/Tcl [9], and PL/Lua [10]. PostgreSQL use Bison in its parser-stage,[11] so it is easy to port many open source languages, as well as reuse code.

Comparing with PSM

The SQL/PSM language is specified by an ISO standard, but also inspired by Oracle's PL/SQL and pgPL/SQL, so there are few differences. The main features of PSM that differ from PL/pgSQL:[12]

- Exception handlers are subroutines (continue handlers);
- Warnings can be handled like an exception;
- Declaration of variables should be based on SQL query result.

References

[1] feature T322 (http://www.postgresql.org/docs/9.0/static/features-sql-standard.html)
[2] http://gborg.postgresql.org/project/pljava/projdisplay.php
[3] http://www.postgresql.org/docs/current/interactive/plperl.html
[4] http://www.commandprompt.com/community/plphp/
[5] http://www.postgresql.org/docs/current/interactive/plpython.html
[6] http://www.joeconway.com/plr/
[7] http://raa.ruby-lang.org/project/pl-ruby
[8] http://plsh.projects.postgresql.org/
[9] http://www.postgresql.org/docs/current/interactive/pltcl.html
[10] http://pllua.projects.postgresql.org/
[11] http://www.postgresql.org/docs/9.0/static/parser-stage.html
[12] Proposal: PL/pgPSM for pg9.3 (http://www.postgresql.org/message-id/CAFj8pRDWFdcjNSnwQB_3j1-rMO6b8=TmLTNBvDCSpRrOW2Dfeg@mail.gmail.com), by P. Stehule; and PosgreSQL-PSM-addon Manual (http://postgres.cz/wiki/SQL/PSM_Manual)

External links

- Official PL/pgSQL documentation (http://www.postgresql.org/docs/current/static/plpgsql.html)
- PL/pgSQL (en) (http://www.pgsql.cz/index.php/PL/pgSQL_(en)), tutorial and examples

PL/Perl

PL/Perl (Procedural Language/Perl) is a procedural language supported by the PostgreSQL RDBMS.

PL/Perl, as an imperative programming language, allows more control than the relational algebra of SQL. Programs created in the PL/Perl language are called functions and can use most of the features that the Perl programming language provides, including common flow control structures and syntax that has incorporated regular expressions directly. These functions can be evaluated as part of a SQL statement, or in response to a trigger or rule.

The design goals of PL/Perl were to create a loadable procedural language that:

- can be used to create functions and trigger procedures,
- adds control structures to the SQL language,
- can perform complex computations,
- can be defined to be either trusted or untrusted [1] by the server,
- is easy to use.

PL/Perl is one of many "PL" languages available for PostgreSQL PL/pgSQL PL/Java [2], plPHP [2], PL/Python [5], PL/R [6], PL/Ruby [3], PL/sh [8], and PL/Tcl [9].

References

- PostgreSQL PL/Perl documentation [4]

References

[1] http://www.postgresql.org/docs/current/static/plperl-trusted.html
[2] http://plphp.commandprompt.com/
[3] http://raa.ruby-lang.org/list.rhtml?name=pl-ruby
[4] http://www.postgresql.org/docs/current/static/plperl.html

JADE (programming language)

JADE is a proprietary object-oriented software development and deployment platform product from the New Zealand-based Jade Software Corporation, first released in 1996. It consists of the JADE programming language, IDE and debugger, integrated application server and object database management system.

Designed as an end-to-end development environment to allow systems to be coded in one language from the database server down to the clients, it also provides APIs for other languages, including .NET Framework, Java, C/C++ and Web services.

As a programming language, its main competitors are Java and C#, while as a database it competes with other object-oriented databases and post-relational databases such as Versant, Caché and Matisse as well as traditional relational database software packages such as Oracle and Microsoft SQL Server.

Although a free limited licence is available for development, using the JADE platform requires per-process fees to be paid.

Language

In syntax, JADE is very similar to Pascal; its syntax is based on the language Modula-2, which was derived from Pascal. While it includes innovations lacking in Pascal or Modula-2, it lacks certain features of other modern object-oriented languages such as C# and Java.

Programming model

Like all of the other popular programming languages used to create database-driven software, JADE is fully object-oriented. JADE was designed to have all the most important features of object-oriented programming, but does not support the overloading of methods or operators, and lacks parameterised constructors.

Classes in JADE are kept together in schemas. Schemas serve the same purpose as Java packages or namespaces in .NET, but have a hierarchy, and inherit classes from superschemas. This becomes useful especially when programming using the model-view-controller methodology, as model classes can be put in one schema, then the controller and view classes can be built on top of the model classes in a subschema.

Program structure

JADE programs are developed using a user interface that allows programmers to visually create classes and define their properties and methods. Instead of locating methods in large files, programmers select the method they would like to edit and only the code for that particular method is displayed. Also instead of compiling all the code of a program at once, in JADE, each method is compiled individually as soon as the method is completed, meaning code can be checked immediately.

All the code for a JADE application is stored in its object-oriented database. This allows for multi-user development as the database maintains concurrency control, and with each piece of the code being a separate object in the database, it is often possible to recode a system while it is live and online as long as the parts of the system being changed are not in use.

Features

The main goal of JADE was to create a seamlessly integrated programming language that would allow developers to create one application that would go from end-to-end instead of having to write three separate applications for the database server, application server and presentation client and then write the code for them to communicate with each other.

Object database

The most striking difference between JADE and other object-oriented programming languages is that its object database is a native part of its language. For example, when creating an object in JADE, it can be created as transient or persistent. Creating an object as *transient* is similar to creating objects in other object-oriented programming languages - the object is simply created in memory, and then lost when the program ends. On the other hand, when an object is created as *persistent*, when the program ends, the object will still exist and be there the next time the program starts up. So, when an object is *persistent* JADE automatically works in the background to store and retrieve the object in the database when necessary. Persistent objects can be distributed across multiple co-operating servers, with JADE automatically handling object caching and cache coherency.

There are very few differences between manipulating transient and persistent objects so JADE makes it *appear* to the programmer as if all the objects in the entire database were in local memory. Most of the time, JADE's object-oriented database is used in a multi-user system, and so JADE makes it appear to the programmer as if all the objects in the database were stored in shared memory that all users connected to the system can access, even from different computers.

With all of the program code centralised on the database server as well the data, all client nodes can be programmed as if they were running on the database server.

JADE's database is inherently object-oriented, and [ACID]-compliant, and has all of the standard features such as atomic transactions, locking, rollback, crash recovery and the ability to keep one or more secondary database servers synchronised with the main database for backup, disaster recovery and performance reasons.

To interoperate with relational databases for reporting, business intelligence and data warehouse purposes JADE since 2010 has provided a "Relational Population Service" that enables automatically replicating objects from its native object-orientated database to one or more relational database. This feature supports Microsoft SQL Server versions 2000, 2005 and 2008.

Three-tier model

Database-driven software, often uses a three-tier methodology with applications being split into three tiers — data storage, processing and presentation. In the data storage and processing tiers, JADE systems are a collection of co-operating servers, called nodes, and multiple nodes may be involved in each tier. Each type of node has a different name and all are capable of manipulating objects and executing business logic. A collection of nodes can be deployed across one or several machines. Database servers handle data storage and can also execute business logic, while application servers handle processing. In a three-tier model, presentation clients provide the user interface. JADE also provides a two-tier client that combines the abilities of an application server and presentation client. Traditionally, these three tiers would be created by combining three programs and having them communicate to form one system. Having the different programs separate like this has many advantages, the main one is that the system becomes scalable, that is, raising the power of the system involves simply adding more nodes.

Designing a system like this gives the programmer a decision to consider every time they want to code in a particular function. They need to decide whether the function would run best on the database server, application server or presentation client before they begin coding as it will be difficult to change that decision once the functionality is coded into one of the tiers.

This is different for JADE applications, as they are coded as one application from end-to-end. When creating a JADE application, the programmer can think as if they were creating a program that will be running on the database server — as if all the data storage, processing and presentation were happening on one computer. When the program runs on three tiers, JADE automatically knows to run all the code by default on the application server, and to send database requests up to the database server and user interface information down to the presentation client. However, it is very easy for the programmer to switch the location at which a particular method is run and move it to a different tier by changing the method signature. Because of this, the decision on where a particular piece of code should run can be made late in the development cycle, and refactoring code to run on different parts of the system ends up being a lot easier because of the way JADE allows end-to-end development of software.

Types of Clients

Programmers have the facility to allow three different kinds of clients to connect to a JADE system. These three types of clients are named:

- JADE Forms
- HTML Documents
- Web Services

In the same schema, a JADE developer can create many completely separate applications which may provide different interfaces to access the same database.

JADE Forms

JADE Forms applications are made up of forms, as the name suggests. Clients need to connect through the JADE Smart Thin Client or Standard Client to be able to run applications that use JADE Forms.

The Smart Thin Client works by connecting to an Application Server which generally does all the processing on behalf of the Smart Thin Client, meaning the thin client only needs to be responsible for displaying forms and taking input. This means the computer running the thin client does not have to be a very powerful computer, and it does not require a fast network connection as it is not loading data from the database — JADE thin clients are often run over a dial-up connection. This is the reason they are called thin clients, as there is not a heavy requirement on computational power to run these clients.

The Standard Client is just the Smart Thin Client combined with the Application Server on one machine. In this case, the node running the client does all of the processing as well as the presentation. Standard clients have greater demands on computational power than thin clients, as they must load data from the database as well as do their own processing.

The advantages of using JADE Forms include:

- Out of the three kinds of clients, JADE Forms provide the shortest development time of JADE applications.
- Allows developers to use the same technology end-to-end.
- Smart thin clients can be packaged up so they can be installed and run on client computers in several clicks.

The disadvantages are:

- Cannot reach a worldwide audience as is possible on the World Wide Web.

JADE Forms have an interesting twist to them though. It is actually possible to run a JADE Forms application through a web browser by changing its mode to *web-enabled*. When this happens, JADE automatically generates HTML code to make pages that resemble the forms and controls, without any modifications to the code. This is a very quick way for programmers that are not competent with HTML and other web technologies to deliver a program through the web.

HTML documents

JADE supports deployment of applications to the web through its HTML documents feature. These work very similarly to ASP.NET, where developers create templates of HTML pages and leave parts in the template for the program to fill in.

The advantages of using HTML documents are:

- Allows the application to reach a worldwide audience.

The disadvantages are:

- When JADE applications use HTML documents, they are no longer using the same technology from end to end. Checking at the front end of the system may be done through JavaScript for example.
- Offloading some of the processing to front-end clients is no longer as easy or secure.

Web services

Web services are used to allow different programs to communicate with each other from remote locations in an object-oriented form. Web services cannot be accessed directly by human users. One of the uses of Web services with JADE is to allow other technologies such as .NET or Java to use JADE as the backend object-oriented database. Web services also allow JADE systems to interoperate with services provided by other non-JADE systems.

Interoperability

In addition to Web services, JADE is also capable of interfacing with other programs through language APIs (including .NET, Java, C/C++), DLL calls, ActiveX/COM objects and .NET assemblies. This allows other programs to access objects and execute methods, and can be used to provide a different interface to a JADE application. JADE 6.2 provided a Java API, .NET Assembly integration and the ability to run Smart Thin Clients on Windows Mobile devices. JADE 6.3 provides an API for .NET languages.

Multilingual abilities

JADE natively supports multilingual programs. It does this in several ways:

- Strings can be marked as *translatable*, which means they will be change depending on the current language.
- Many versions of the same form can be created to suit each language. This means interfaces can look entirely different from one language to the next.
- The developer has methods available to access the current locale of the system and so they can implement their own language-dependent features.

JADE will automatically switch to the language it detects on the system if the language is provided by the developer.

Portability

Currently JADE applications can be run on Windows and Linux. Similar to Java, JADE strives to allow programmers to develop applications once and be able to allow them to run on both of these platforms with minimal changes. JADE 6.2 allows Smart Thin Clients and a specialised Standard Client to run on Windows Mobile devices.

Code Examples

In this section are some short examples of JADE code.

Hello World!

This is the "Hello World!" code in JADE:

```
helloWorld();

begin
   app.msgBox("Hello, World!", "Hello, World!", MsgBox_OK_Only + MsgBox_Information_Icon);
end;
```

or

```
helloWorld();

begin
   write "Hello, World!";
end;
```

History

JADE was originally conceived by Sir Gilbert Simpson and is currently developed by the Jade Software Corporation.[1]

The first version of JADE was JADE 3, released September 1996.

The current version is JADE 7.

References

[1] Jade Software Corporation (http://www.jadeworld.com)

External links

- Official website (http://www.jadeworld.com/)

Tutorials and resources

- Examples and Tutorials for JADE Programmers (http://www.jader.co.nz/)

Media coverage

- Scoop Independent News - JADE 6.3 (http://www.scoop.co.nz/stories/BU0905/S00125.htm)
- JADE 6.1 delivers data replication to Microsoft SQL Server (http://www.enterprisenetworksandservers.com/monthly/art.php?1899)

ObjectDB

ObjectDB Object Database

Developer(s)	ObjectDB Software
Stable release	2.5.3 / September 4, 2013
Written in	Java
Operating system	Cross-platform
Type	Object database
License	Proprietary[1]
Website	www.objectdb.com [2]

ObjectDB is an object database for Java. It can be used in client-server mode and in embedded (in process) mode.

Unlike other object databases, ObjectDB does not provide its own proprietary API. Accordingly, working with ObjectDB requires using one of the two standard Java APIs - JPA or JDO. Both APIs are built-in in ObjectDB,[3][4] so an intermediate ORM software is not needed.[5][6]

Features

ObjectDB is a cross platform software and can be used on various operating systems with Java SE 5 or higher. It can be integrated into Java EE and Spring web applications and deployed on servlet containers (Tomcat, Jetty) as well as on Java EE application servers (GlassFish, JBoss).[7][8] It was tested on various JVMs, including HotSpot, JRockit and IBM J9.[9]

The maximum database size is 128 TB (131,072 GB). The number of objects in a database is unlimited (except by the database size).

All the persistable types of JPA and JDO are supported by ObjectDB, including user defined entity classes, user defined embeddable classes, standard Java collections, basic data types (primitive values, wrapper values, String, Date, Time, Timestamp) and any other serializable classes.

Every object in the database has a unique ID. ObjectDB supports both traditional object database IDs, as well as RDBMS like primary keys, including composite primary keys and auto value generation and assignment, as part of its support of JPA, which is mainly an API for RDBMS.

Two query languages are supported. The JDO Query Language (JDOQL), which is based on Java syntax, and the JPA Query Language (JPQL), which is based on SQL syntax. JPA 2 criteria queries are also supported.

ObjectDB automatic schema evolution handles most changes to classes transparently, including adding and removing of persistent fields, changing types of persistent fields, and modifying class hierarchy. Renaming persistable classes and persistent fields is also supported.

Tools and Utilities

The following tools and utilities are included in the ObjectDB distribution:[10]

- **Database Explorer** - GUI tool for querying, viewing and editing database content.
- **Database Doctor** - Diagnoses and repairs possible database problems.
- **Replication** - Master-Slave replication (clustering) with unlimited number of slave nodes.
- **Online Backup** - Database backup by a simple query on an EntityManager.
- **Class Enhancer** - Boosts performance by preparing classes for persistence.
- **Transaction Replayer** - Recorder and replayer of database transactions.
- **BIRT Reports Driver** - Adds ObjectDB as a BIRT data source and JPQL / JDOQL queries as data sets.

References

[1] http://www.objectdb.com/object/db/database/license
[2] http://www.objectdb.com
[3] http://www.objectdb.com/
[4] http://stackoverflow.com/questions/5291950/is-objectdb-production-ready
[5] http://www.javabeat.net/2011/02/create-applications-using-objectdb-and-jpa-in-netbeans
[6] http://www.jpab.org
[7] http://www.objectdb.com/tutorial
[8] http://www.developer.com/java/web/integrate-objectdb-into-your-jpa-based-java-web-app.html
[9] http://www.objectdb.com/object/db/database/features
[10] http://www.objectdb.com/java/jpa/tool

External links

- Official website (http://www.objectdb.com)

Versant Object Database

Versant Object Database

Developer(s)	Versant Corporation
Stable release	8.0.2.15 / October 1, 2012
Development status	Active
Written in	Java, C, C#, C++, Smalltalk, Python
Operating system	Cross-platform Solaris, Linux, Windows (NT thru Vista), AIX, HP-UX (both 32 and 64 bit for all platforms)
Type	Object Database
License	All rights reserved
Website	www.versant.com [1]

Versant Object Database (VOD) is an object database software product developed by Versant Corporation.

The Versant Object Database enables developers using object oriented languages to transactionally store their information by allowing the respective language to act as the Data Definition Language (DDL) for the database. In other words, the memory model is the database schema model.[1]

In general, persistence in VOD in implemented by declaring a list of classes, then providing a transaction demarcation application programming interface to use cases. Respective language integrations adhere to the constructs of that language, including syntactic and directive sugars.

Additional APIs exist, beyond simple transaction demarcation, providing for the more advanced capabilities necessary to address practical issues found when dealing with performance optimization and scalability for systems with large amounts of data, many concurrent users, network latency, disk bottlenecks, etc.

Feature highlights

- Transparent object persistence from C++, Java and .NET
- Support for standards, e.g., JDO
- Seamless database distribution
- Enterprise-class high availability
- Dynamic schema evolution
- Low administration
- Multithreading, multisession
- End-to-end object architecture
- Fine-grained concurrency control

Supported languages

Primary supported languages are Java, C# and C++. Versant also has language support for Smalltalk and Python.

Query systems

VOD supports queries via a server side indexing and query execution engine. Query support includes both a Versant-specific and a standards-based query language syntax. Versant provides this query capability in a number of forms depending on the developer's chosen language binding. For example, in Java VOD provides VQL (Versant Query Language), JDOQL, EJB QL and OQL. In C++ Versant provides VQL and OQL, with C# support for VQL, OQL and LINQ. VOD will do optimization of query execution based on available attribute indexes. Versant also has support for standard SQL queries against the Versant database using ODBC/JDBC drivers.

Versant Query Language

The native Versant Query Language (VQL) is similar to SQL92. It is a string based implementation which allows parameterized runtime binding. The difference is that instead of targeting tables and columns, it targets classes and attributes.

Other object-oriented elements apply to query processing. For example, a query targeting a super class will return all instances of concrete subclasses that satisfy the query predicate. VOD is a distributed database: a logical database can be composed of many physical database nodes, with queries are performed in parallel.

Versant query support includes most of the core concepts found in relational query languages including: pattern matching, join, set operators, orderby, existence, distinct, projections, numerical expressions, indexing, cursors, etc.

Indexing

VOD supports indexes on large collections. However it is not necessary to have a collection in order to have a queryable object with a usable index. Unlike other OODB implementations, any object in a Versant database is indexable and accessible via query. Indexes can be placed on attributes of classes and those classes can then be the target of a query operation. Indexes can be hash, b-tree, unique, compound, virtual and can be created online either using a utility, via a graphical user interface or via an API call.

Large collection support

VOD provides pagination support for large collections using a special node based implementation. These collections are designed in such a way that access is done so that only nodes needed by the client are brought resident into memory, instead of having to load the entire collection.

These large collections are created and operated on just as other persistent collection classes. The interface is also consistent with the appropriate language constructs. For example C++ Standard Template Library, Java iterators, C# enumerables, etc.

Collections of objects by default are only a collection of object identifiers. So, these can be very large, yet have a small resident memory footprint. To iterate the collection, objects are dereferenced into client memory space in either a configurable batch mode or one at a time. A query on the collection can be done using the “in” operator (or other set based operators like subset_of, superset_of, etc.) without loading the collection to the client memory space.

Data replication

There are several mechanisms for replication on VOD that depend on the motivation behind the replication. It is for high availability or for distribution or integration.

High availability

Versant does synchronous pair replication. Full replication for fault tolerance only requires installation of one configuration file specifying the buddy node names: New connections notice the existence of the replica file and on connect, check the file for a buddy pair and if it exists, connect to both buddies. This could be a distributed database so that there are many buddy pairs. Then all transactional changes are committed synchronously to the buddy database server processes.

If any one of the databases in the buddy pair should become unreachable, the in-flight transactions are handled so that there is no commit failure, instead in-flight transactions on node failure will continue to the node that is still alive in the buddy pair. On the machine where the node is still alive and processing transactions, a new process will start that monitors for the crashed database to become accessible again. Once the previously failed node is alive, the monitoring process starts replicating all changes that have occurred since the time of failure to bring the two buddies back into full synchronization. Once they are in full sync, a flag is set and on the next transaction clients will move back to full synchronous operation. All of this is handled without any user involvement.

In the case of extreme failure, like a broken disk drive, etc., the replicated node can be recreated from an online backup of the live node. Simply install a new disk drive, take an online backup of the live node, restore on the failed machine, start the monitor to sync the last few transactions and restore full replication at clients.

Distribution

Distribution is handled using Versant Asynchronous Replication (VAR), a channel driven, master-slave or peer-to-peer replication framework with rule based conflict detection and resolution.

An administrator uses a utility to define replication channels. Channels are named entities that define a scope of replication within a physical node. The “scope” can be anything from full database replication to something as fine grained as anything definable by a Versant query. Once the channels are defined, applications can register as listeners on these channels, at which point changes from those channel begin to flow to the respective clients.

These channels provide both persistence and reliable messaging. In the event that a connection is lost between a registered listener and a channel, ongoing changes will be guaranteed delivery once the connection is re-established. There are multiple transport protocols that can be configured for optimization in highly reliable LAN networks or high reliability in unreliable WAN type of environments.

In bi-directional channel replication, a set of conflict detection rules are put in place so that conflicting changes can be resolved at runtime without disrupting channel activity. There are other forms of data distribution.

Integration

Usually, integration requires some kind of custom code. Users can connect to both relational and Versant databases using ORM products. They can load objects either from a relational database or Versant and then with some minor code implementation, disconnect those objects from the source and write them to a target. This can be used for import/export in a batch processing mode for integration with other database systems.

Data distribution architecture

VOD handles distributed data processing using a distributed two-phase commit protocol across multiply connected databases. In this process, VOD uses an internal resource manager that is handling the distributed transactions. Versant also supports the XA protocol allowing external transaction monitors to control the transactional context, so for example plug into a CORBA or J2EE application server.

Versant allows object relationships to span physical resource (database) nodes. Shared information referenced from object graphs that reside in other databases and resolution of that information is transparent at runtime. For example, several physical databases may hold user information models that are partitioned by account number holding aggregations on account activities such as trades and then have some more databases holding actual trade models and these users and trades can be related. A query across all of the user databases and return a user (or set of users), then as messages are sent to user objects involving trades, the trade models will automatically be resolved across the distribution. After updates of any of those objects, at commit time Versant will ensure that all changes commit back to their respective physical nodes in a completely ACID 2phase commit process.

Object id's are guaranteed to be unique across all physical nodes. Objects could be "moved" from one physical node to another without any application code changes required.

Schema evolution

Schema evolution is handled via a normal update of the application's class models and then applying those changes to the operational database. Those schema changes can be applied to an existing database either via a utility or API. The result is a versioning of the database schema.

Existing objects in the database are lazily evolved to the latest schema version. No object is actually evolved unless it is made dirty (marked for update) and committed back to the database. In general this means an application with the new schema will not cause evolution, expect for new and updated objects.

There are utilities that can "crawl" a database slowly evolving all instanced to the latest version by grabbing sets of them, marking them dirty, committing. This is sometimes desired for embedded or real-time systems where performance and space needs to be optimized.

In most cases, older clients get patch updates with the new schema in conjunction with updates to the server. The clients schema version is in sync with the database server. Versant's loose schema mapping facility can also be used. This is enabled by a flag in the client so that it does not complain about a mismatch in schema version and instead filters the incoming objects to match the old schema. Using this facility requires some forethought to avoid any unintended side effects.

The process goes as follows:

1. class definitions are updated, i.e. add new subclasses, add attributes, rename attributes, remove attributes, etc. and recompile. When the application connects to a Versant database, a schema version mismatch will be detected and you would normally get an error unless you take some action to avoid the mismatch.
2. The schema mismatch can be avoided using a number of techniques.
    1. a utility can be used to describe the new schema to the database. The utility will show a list of incompatibilities and ask how you want them to be resolved. Your action will depend on whether you are in

development, QA, production, etc. Regardless, actions like dropping the existing class, evolving the schema version and keeping all existing objects, rename and retype, etc, are also possible.

2. the evolution process can be automated via connection options. This is normally used in development mode and allows the schema to automatically evolve any mismatches on connect and continue preserving the existing objects.
3. specific API's can be used to dynamically evolve the database schema. This is an advanced topic, involving what's called Versant runtime classes. Basically, you can create completely dynamic schema structure for the database so that new classes and attributes can be created on the fly.

3. If clients with the older schema continue to operate on the database, loose_schema_mapping in the application profile file should be set to true.
4. Optionally, a utility can be started to crawl the database and force version migration of all existing instances.

The general guidelines for schema evolution are that any schema changes can be made and existing instances preserved, without having to write custom evolution code, with the exception of two things:

1. Changes to the middle of an inheritance hierarchy. Inserting a new class into the middle of a hierarchy is impossible without losing your existing objects, unless custom code is written to do this operation in a series of steps.
2. Incompatible type changes like Array to a String.

All other forms of evolution like renaming attributes, deleting leaf classes, adding leaf classes, adding new classes, adding or removing attributes, etc. can be done online and without custom code. If actions like setting non standard default values for newly added attributes are necessary, this can be done in callback functions within the objects. There are a set of standard object lifecycle callbacks that get invoked in activities like cache load. Those callbacks can be used to check for default values and take action if necessary.

Persistent object lifecycle

The lifecycle of an object load can be controlled on a use case basis.

By default, objects are loaded only when they are sent a message. This includes the default behavior for queries which only return a collection of references to objects that satisfied the query predicate, not the actual objects. When an object is loaded, all it's non-reference attributes (primitives) are also loaded and remaining reference types follow the same pattern as the referencing object.

When a message is sent to an object VOD looks into internal structures to see if the object is already in client memory. If not, VOS does an RPC to load the object. At the time VOD loads the object, it will also look at the connections locking strategy to decide how to deal with locking the object on load. VOD supports both global locking strategies that can be applied to a connection and extremely fine grained control to override behavior for a particular use case.

Once an object is loaded and locked it stays in the client cache, with an equivalent lock in the server, until one of a number of events occurs.

The most common event, the current transaction ends with commit. In the default case, this will release the lock and object from memory. However, note that there are forms of commit that will do combinations of things like, keep the cache and the locks and start a new transaction, keep the cache, but release the locks and start a new transaction. These forms and others are used to optimize cache effectiveness when using non-default locking strategies like optimistic locking or when you have a series of transactions that form a task and operate on the same set of objects.

Another possibility is that your client cache starts to get full. In this case, VOD may decide to swap objects back to the server process to make space and do some work that will have to be done at commit anyway. VOD does this in a fully transactional way, so that even if modified objects get swapped to the server, they will still be undone if the transaction is rolled back. Also, you have the ability to "pin" objects into the client cache to prevent swapping of

important sets of objects, enabling the use of direct memory pointers without concern for memory faults.

Another possible event is a query call which has the option set to flush the cache of objects in the target class, so that changed objects currently in your cache become part of the current query execution evaluation.

Other possibilities include API calls that result in explicit release of the object, like a call to refresh or a call to release.

There are many ways to override the default behavior. Those are in fact commonly used to performance tune on a use case basis. For example, if you are going to iterate over a collection of 1000 objects, you don't want to do 1000 RPC's. Giving the collection of references to a call to groupRead will use a single RPC and load all objects. Similarly, you can make a call to getClosure which will use groupRead behavior to load all referenced objects in a graph from the starting point, down to your specified level of reachability. Further, queries have options to set a lock and load result sets rather than just references or to use cursors. There are API's to explicitly load objects into cache and set higher lock levels than the connection defaults, etc.

Achieving persistence

For users of C++, Versant requires that the uppermost class in an inheritance hierarchy inherit from a base class "PObject", which handles database activities.

Then there is a file setup, `schema.imp`, that declares which classes in the model are to be made persistent and that file is used in a pre-compilation phase where Versant's necessary magicWikipedia:Please clarify is added to the persistent classes. Finally, the resulting `schema.cxx` file is compiled and linked with the application.

The pre-compilation phase is done with a utility though note this is typically automatically set up in one's visual development environment so the process is automatic when a build is done.

When using Java or .NET, this same procedure described above with C++ is accomplished using post-processing byte code enhancement. One sets up a file that declares which classes are to be persistent and then uses a utility, or API, or IDE integration to enhance the classes before running or debugging.

Versant provides other Java APIs based on standards JDO and JPA. In those versions of the API, the system adheres to the standards defined for declaring persistence whether it be some kind of XML or annotation. Enhancement is then done using a utility (similarly with .NET) or more commonly with Eclipse plug-in or Microsoft Visual Studio integration during the build process.

Integration with relational databases

A large percentage of Versant's customers do some form of integration with relational tables. This can be accomplished in a couple of ways depending on the requirements such as: on-line/off-line, batch based, transactional, etc.

XA

Versant supports the XA protocol for distributed transactions. This allows participation in online distributed transactions with relational databases. The interaction with the relational tables can take many forms from custom code to ORM solutions to J2EE application servers (Entity Relationship Modeling) to message passing to ORBs, etc. The XA API allows the Versant database to act as a resource controlled by an external transaction monitor coordinating changes to both Versant and relational databases in the same transactional context.

ORM

Versant can interact with relational databases using Java ORM technology such as JDO (Java Data Objects) and Hibernate JPA. These standards-based implementations have the ability to detach objects from their transactional context and then attach them to another connection. There are restrictions in that Versant requires the application to use a concept known as database identity in order for replication to work with relations intact. Versant does not support the ORM form of application identity in anything other than a disconnected data form.

XML

Versant has tools that enable the import and export of XML data. For example, batch based replication of data can be accomplished by exporting objects from the Versant database as XML, if necessary applying an XSLT transform and then importing into relational tables. The opposite direction is also possible. With Java, the most common approach using XML is to dynamically replicate information using JAXB which runtime converts objects into and out of an XML form. Using JAXB, the Versant database only needs to work with objects rather than importing an XML form. In essence, XML coming from relational databases are converted to objects at runtime using JAXB and those objects are then persisted into the Versant database.

Custom code

Users of C++ are especially challenged in integrating with relational databases. Versant provides consulting to help these customers with their integration challenges, but does not make those solutions, which require customization for each application, available in a productized form.

Transactions

Versant by default is always implicitly in a transaction when connected to the database. In addition, VOD supports the XA protocol and apply that to certain standards based API' such as JDO and JPA which require explicit transaction demarcation. There is a non-implicit form of transaction where transaction begin/end must be declared.

In order to discard from memory objects that have been modified in the current transaction you can either do it globally for the current transaction by issuing a rollback which also implicitly starts another transaction or you can do it in isolation or globally using specific calls within the same transaction.

Locking and caching strategies

Versant by default uses a pessimistic locking strategy to ensure that objects in the database server are in sync with client access in an ACID way. This is done by using a combination of locks against both schema and instance objects.

The database server process maintains lock request queues at the object level to control concurrency of access to the same object. A request for update will establish a queue if there are any existing readers of an object. The request either goes through when all current readers release their locks or times-out (an exception which can be handled by client is thrown). Locks are generally released at transaction boundaries. When a queue is established by an update request, all other subsequent requests fall in queue behind the update request. Once the update request has been filled, all read requests in the queue rush in and get their read lock, return the object, and if there are no other updates, the queue disappears. In this architecture, locks are done at the object level so false waits and false deadlocks do not occur.

Other ways of keeping client caches in sync are, for example, an optimistic locking strategy, using a classic timestamp mechanism. VOD also provides forms of client cache synchronization using multi-cast. Additionally it provides an event mechanism where clients can register for triggering events within the database server to be used for synchronization or for business logic work flow.

Scalability

Storage

Versant supports, multiple file and multiple process configurations. Data storage is done in a single or multiple files, but there are supporting files for the logging subsystem (logical and physical log files). These logging files are used for high performance and scalability under concurrent user loads and for online database backup processes.

Clients

Versant is a multi-user client server database and has production applications with thousands of concurrently connected users. Thus, Versant can also run linked and embedded in the same address space as the application process (so it can be also an embedded database).

Performance

Versant uses internal performance and scalability benchmarks to monitor and measure behavior over time across releases, patches and generations of new hardware.

Versant has done other non-standard benchmarking activities in a public forum.[2] .[3]

Versant ran the 007 benchmarks in the early 90's but currently doesn't provide any comparisons because there are no industry benchmarks that make sense for object databases,

One of the candidates considered was TPC-E, which was supposed to be the new OLTP standard database benchmark with new complex models aimed at being representative of today's computing environment. The TPC-E is based on a financial trading system model. Still, comparative results could not be obtained. The reason is that the TPC specifies requirements regarding what part of the code resides in the "driver" of the benchmark and what part resides in "database" functionality. However, the driver to application logic interface is completely defined at the data level. This means that when measuring relational access you would not incur any overhead for mapping into a C++ object. The mapping of the raw data into what ever form was necessary in the driver to implement the business logic was completely outside of the benchmark measurements. When it comes to the object database, you need to now un-map the C++ objects into the driver data structures and in doing so, measure the cost of that activity as part of the benchmark timings.

But this is the opposite of a real world application where people write object oriented applications resulting in object oriented models. In a relational database, you need to map/un-map from objects to the relational data structures. The TPC-E was written in a way as to exclude the "mapping effect" from the measurements, which by the very nature of how an object database works means the TPC-E was written in a way that forces measurement of an "un-mapping effect", an activity which does not occur in a real world application. Thus with TPC-E, the true cost of computing is removed for relational and even worse added to object databases.

Add-on Modules

Versant provides add-on modules for deployment or access to its Object Database.

- V/Management Center: V/MC delivers real-time views of performance data and analytical information about the Versant Object Database. For example, it alerts administrators about potential issues before the database availability is affected. It's designed as an Eclipse-based RCP client.
- Versant Compact: Online Database Maintenance.
- Versant FTS: High Availability Database Server.
- Versant Async Server: Production Database Replication.
- Versant HA Backup: High Availability Backup Solution.
- Versant SQL: SQL Access & Reporting.

Applications

Usually the “best kind of application” to use a Versant database are those applications requiring an application specific database of an OLTP nature. There are certain application characteristics where Versant technology provides better performance and scalability than traditional relational technology: complex models, large amount of data, large number of concurrent users.

Thus, VOD is found in applications within many different vertical industries: global trading platforms for large stock exchanges, network management for large telecommunications providers, intelligence analytics for defense agencies, reservation systems for large airline/hotel companies, risk management analytics for banking and transportation organizations, Massively multiplayer online game systems, network security and fraud detection, local number portability, advanced simulations, social networking, etc..

References

[1] "TechView Product Report: Versant Object Database" (http://www.odbms.org/download\048.04 TechView Versant.pdf), *odbms.org*. Retrieved 6 October 2010.

[2] "Poleposition, the open source database benchmark" (http://polepos.sourceforge.net/results/PolePositionClientServer.pdf),*polepos.org*. Retrieved 24 Februar 2011.

[3] "Accelerating IBM WebSphere Application Server Performance with Versant enJin" (http://www.redbooks.ibm.com/abstracts/SG246561.html), *ibm.com*. Retrieved 6 October 2010.

Zope Object Database

Zope Object Database

Developer(s)	Zope Corporation
Stable release	3.10.3 / April 12, 2011[1]
Written in	Python
Operating system	Cross-platform
Type	Object Database
License	Zope Public License
Website	www.zodb.org [2]

The **Zope Object Database** (**ZODB**) is an object-oriented database for transparently and persistently storing Python objects. It is included as part of the Zope web application server, but can also be used independently of Zope.

Features of the ZODB include: transactions, history/undo, transparently pluggable storage, built-in caching, multiversion concurrency control (MVCC), and scalability across a network (using ZEO).

History

- Created by Jim Fulton of Zope Corporation in the late 90s.
- Started as simple Persistent Object System (POS) during Principia development (which later became Zope)
- ZODB 3 was renamed when a significant architecture change was landed.
- ZODB 4 was a short lived project to re-implement the entire ZODB 3 package using 100% Python.

Implementation

Basics

A ZODB storage is basically a directed graph of (Python) objects pointing at each other, with a Python dictionary at the root. Objects are accessed by starting at the root, and following pointers until the target object. In this respect, ZODB can be seen as a sophisticated Python persistence layer.

Example

For example, say we have a car described using 3 classes Car, Wheel and Screw. In Python, this could be represented that way (coding style is awful, but this is for an illustration purpose):

```
class Car: [...]
class Wheel: [...]
class Screw: [...]

myCar = Car()
myCar.wheel1 = Wheel()
myCar.wheel2 = Wheel()
for wheel in (myCar.wheel1, myCar.wheel2):
    wheel.screws = [Screw(), Screw()]
```

(Car() creates a new instance of class Car).

If the variable zodb is the root of persistence, then

```
zodb['mycar'] = mycar
```

puts all the objects (the instances of car, wheel, screws and so on) into the storage, and can be retrieved later. If for example, another program gets a connection to the database through the zodb object, performing:

```
carzz = zodb['mycar']
```

retrieves all the objects, the pointer to the car being held in the carzz variable.

The object can then be altered, for example if some later Python code reads:

```
carzz.wheel3 = Wheel()
carzz.wheel3.screws = [Screw()]
```

the storage is altered to reflect the change of data (after a commit is ordered).

There is no declaration of the data structure in Python, so there is none in ZODB, new fields can be freely added to an existing object.

Storage unit

Actually, the above oversimplifies a bit. For persistence to take place, the Python Car class must be derived from the persistence.Persistent class — this class both holds the data necessary for the persistence machinery to work, such as the internal object id, state of the object, and so on, but also defines the boundary of the persistence in the following sense: every object whose class derives from Persistent is the atomic unit of storage (the whole object is copied to the storage when a field is modified).

In the example above, if Car is the only class deriving from Persistent, when wheel3 is added to car, all the objects must be written to the storage (the Car, wheel1, wheel2, the screws and so on). In contrast, if Wheel also derives from Persistent, then when carzz.wheel3 = Wheel is performed, a new record is written to the storage to hold the new value of the Car, but the existing Wheel are kept, and the new record for the Car points to the already existing Wheel record inside the storage.

The ZODB machinery doesn't chase modification down through the graph of pointers. In the example above, carzz.wheel3 = something is a modification automatically tracked down by the ZODB machinery, because carzz is of (Persistent) class Car. The ZODB machinery does this basically by marking the record as `dirty`. However, if there is a list (for example), a change inside the list isn't noticed by the ZODB machinery, and the programmer must help by manually adding

```
carzz._p_changed = 1
```

to notify ZODB that the record actually changed. Thus the programmer must be aware to a certain point of the working of the persistence machinery.

Atomicity

The storage unit (that is, an object whose class derives from Persistent) is also the atomicity unit. In the example above, if Cars is the only Persistent class, a thread modifies a Wheel (the Car record must be notified), and another thread modifies another Wheel inside another transaction, the second commit will fail. If Wheel is also Persistent, both Wheels can be modified independently by two different threads in two different transactions.

Class persistence

The class persistence (that is, writing the class of a particular object into the storage), is obtained by writing a kind of "fully qualified" name of the class into each record on the disk. It should be noted than, in Python, the name of the class involves the hierarchy of directory the source file of the class resides in. A consequence is that the *source* file of persisting object cannot be moved. If it is, the ZODB machinery is unable to locate the class of an object when retrieving it from the storage, resulting into a broken object.

Log file

(missing)

ZEO

ZEO (Zope Enterprise Objects) is a ZODB storage implementation that allows multiple client processes to persist objects to a single ZEO server. This allows transparent scaling, but the ZEO server is still a single point of failure.

Pluggable Storages

- Network Storage (aka ZEO) - Enables multiple python processes load and store persistent instances concurrently.
- File Storage - Enables a single python process to talk to a file on disk.
- relstorage - Enables the persistence backing store to be a RDBMS.
- Directory Storage - Each persistent data is stored as a separate file on the filesystem. Similar to FSFS in Subversion.
- Demo Storage - An in-memory back end for the persistent store.
- BDBStorage - Which uses Berkeley DB back end. Now abandoned.

Failover Technologies:

- Zope Replication Services (ZRS) - A commercial add-on (open-source since May 2013) that removes the single point of failure, providing hot backup for writes and load-balancing for reads.
- zeoraid - An open source solution that provides a proxy Network Server that distributes object stores and recovery across a series of Network Servers.
- relstorage - since RDBMS technologies are used this obviates need for ZEO server.
- NEO - Distributed (fault tolerance, load-balancing) storage implementation.

References

[1] ZODB3 3.10.3 (http://pypi.python.org/pypi/ZODB3/3.10.3)
[2] http://www.zodb.org/

External links

- Introduction to the Zope Object Database (http://www.python.org/workshops/2000-01/proceedings/papers/fulton/zodb3.html)
- ZODB/ZEO Programming Guide (http://wiki.zope.org/ZODB/guide/index.html)
- ZODB Online Book (http://www.zodb.org/zodbbook/)

No-SQL Databases

Strozzi NoSQL (RDBMS)

Strozzi NoSQL (RDBMS)

Original author(s)	Carlo Strozzi
Initial release	1998
Stable release	4.1.10 / September 13, 2010
Platform	Unix-like (e.g., Cygwin)
Type	RDBMS
License	GPL
Website	www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page [1]

Strozzi NoSQL is a shell-based relational database management system initialized and developed by Carlo Strozzi that runs under Unix-like operating systems, or others with compatibility layers (e.g., Cygwin under Windows). Its file name *NoSQL* merely reflects the fact that it does not express its queries using Structured Query Language; the NoSQL RDBMS is distinct from the circa-2009 general concept of NoSQL databases, which are typically non-relational, unlike the NoSQL RDBMS. Strozzi NoSQL is released under the GNU GPL.

Construction

NoSQL uses the operator-stream paradigm, where a number of "operators" perform a unique function on the passed data. The stream used is supplied by the UNIX input/output redirection system so that over the pipe system, the result of the calculation can be passed to other operators. As UNIX pipes run in memory, it is a very efficient way of implementation.

NoSQL, with development led by Carlo Strozzi, is the latest and perhaps the most active in a line of implementations of the stream-operator database design originally described by Evan Shaffer, Rod Manis, and Robert Jorgensen in a 1991 Unix Review article and an associated paper [2]. Other implementations include the Perl-based rdb, a commercial version by the original authors called /rdb [3], and Starbase [4], a version with added astronomical data operators by John Roll of Harvard and the Smithsonian Astrophysical Observatory. Because of its strengths in dealing with pipe data, most implementations are a mixture of awk and other programming languages, usually C or Perl.

The concept was originally described in a 1991 Unix Review article, and later expanded in a paper (see reference above), as well as in the book, "Unix Relational Database Management". NoSQL (along with other similar stream-operator databases) is well-suited to a number of fast, analytical database tasks, and has the significant advantage of keeping the tables in ASCII text form, allowing many powerful text processing tools to be used as an adjunct to the database functions themselves. Popular tools for use with NoSQL include Python, Perl, awk, and shell scripts using the ubiquitous Unix text processing tools (cut, paste, grep, sort, uniq, etc.)

NoSQL is written mostly in interpretive languages, slowing actual process execution, but its ability to use ordinary pipes and filesystems means that it can be extremely fast for many applications when using RAM filesystems or heavily leveraging pipes, which are mostly memory-based in many implementations.

Philosophy

The reasons for avoiding SQL are as follows:

1. Complexity: Most commercial database products are often too costly for minor projects, and free databases are too complex. They also do not have the shell-level approach that NoSQL has.
2. Portability:
    1. Data: The data from NoSQL can be easily ported to other types of machines, like Macintoshes or Windows computers, since tables exist as simple ASCII text and can be easily read from or redirected to files at any point in processing.
    2. Software: NoSQL can run on any UNIX machine that has the Perl and the AWK programming languages installed, and perhaps even on the Cygwin UNIX-like environment for Microsoft Windows.
3. Unlimited: NoSQL has no arbitrary limits, like a data field size, column number, or file size limit, and can principally work where other products cannot. (The number of columns in a table may actually be limited to 32,768 by some implementations of the AWK1 programming language).
4. Usability: With its straight forward and logical concept, NoSQL can easily be used by non-computer people. For instance, rows of data are selected with the 'row' operator, columns with the 'column' operator.

In contrast to other RDBMS, NoSQL has the full power of UNIX during application development and usage. Its user interface uses the UNIX shell. So, it is not necessary to learn a set of new commands to administer the database. From the view of NoSQL, the database is not more than a set of files similar to any other user file. No scripting or other type of database language is used besides the UNIX shell. This shell-nature encourages casual use of this database, which makes its use familiar, resulting in formal use. In other words, NoSQL is a set of shell routines that access normal files of the operating system.

Examples

To retrieve information about a particular employee, a query in SQL might look like this:

```
select e.*, a.*, mgr.* from EMPLOYEES e, ADDRESSES a, MANAGERS mgr
WHERE .....
```

Since a document-oriented NoSQL database often retrieves a pre-connected document representing the entire employee, the query might look like this:

```
$e = doc("/employee/emp_1234")
return $e/address/zip
```

The stream-operator paradigm differs from conventional SQL, but since the NoSQL DB is relational, it is possible to map NoSQL operators to their SQL equivalents:

SQL	NoSQL or /rdb
select col1 col2 from filename	column col1 col2 < filename
where column - expression	row 'column == expression'
compute column = expression	compute 'column = expression'
group by	subtotal
having	row
order by column	sorttable column
unique	uniq
count	wc -l

outer join	jointable -al
update	delete, replace
nesting	pipes

Further reading

- Ayers, Larry (November 1998). "How Not To Re-Invent The Wheel" [5]. *Linux Gazette*. Retrieved 2011-05-03.
- Litt, Steve (April 2007). "NoSQL: The Unix Database (With awk)" [6]. *Linux Productivity Magazine*. Retrieved 2011-05-03.
- Paterno, Giuseppe (November 1, 1999). "NoSQL Tutorial" [7]. *Linux Journal*. Retrieved 2011-05-03.

External links

- NoSQL: a non-SQL RDBMS [1]

References

[1] http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page
[2] http://www.rdb.com/lib/4gl.pdf
[3] http://www.rdb.com/
[4] https://www.cfa.harvard.edu/~john/starbase/starbase.1.html
[5] http://linuxgazette.net/issue34/ayers2.html
[6] http://www.troubleshooters.com/lpm/200704/200704.htm
[7] http://www.linuxjournal.com/article/3294

NoSQL

A **NoSQL** database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability. The data structure (e.g., tree, graph, key-value) differs from the RDBMS, and therefore some operations are faster in NoSQL and some in RDBMS. There are differences though and the particular suitability of a given NoSQL DB depends on the problem to be solved (e.g., does the solution use tree algorithms?).

NoSQL databases are finding significant and growing industry use in big data and real-time web applications. NoSQL systems are also referred to as "Not only SQL" to emphasize that they may in fact allow SQL-like query languages to be used. In the context of the CAP theorem, NoSQL stores often compromise consistency in favor of availability and partition tolerance. Barriers to the greater adoption of NoSQL data stores in practice include: the lack of full ACID transaction support, the use of low-level query languages, the lack of standardized interfaces, and the huge investments already made in SQL by enterprises.

History

Carlo Strozzi used the term *NoSQL* in 1998 to name his lightweight, open-source relational database that did not expose the standard SQL interface. Strozzi suggests that, as the current NoSQL movement "departs from the relational model altogether; it should therefore have been called more appropriately 'NoREL'.

Eric Evans (then a Rackspace employee) reintroduced the term *NoSQL* in early 2009 when Johan Oskarsson of Last.fm wanted to organize an event to discuss open-source distributed databases. The name attempted to label the emergence of a growing number of non-relational, distributed data stores that often did not attempt to provide atomicity, consistency, isolation and durability guarantees that are key attributes of classic relational database systems.

Taxonomy

There have been various approaches to classify NoSQL databases, each with different categories and subcategories. Because of the variety of approaches and overlaps it is difficult to get and maintain an overview of non-relational databases. Nevertheless, the basic classification that most would agree on is based on data model. A few of these and their prototypes are:

- **Column**: HBase, Accumulo, Cassandra
- **Document**: MarkLogic, MongoDB, Couchbase
- **Key-value**: Dynamo, Riak, Redis, MemcacheDB, Project Voldemort
- **Graph**: Neo4J, OrientDB, Allegro, Virtuoso

Classification based on data model

Stephen Yen in his blog post "NoSQL is a Horseless Carriage" suggests the following:[1]

Term	**Matching Database**
KV Cache	Memcached, Repcached, Coherence, Hazelcast, Infinispan, eXtreme Scale, JBoss Cache, Velocity, Terracotta, GigaSpaces
KV Store	Keyspace, Flare, SchemaFree, RAMCloud
KV Store - Eventually consistent	Dynamo, Voldemort, Dynomite, SubRecord, MotionDb, DovetailDB
Data-structures server	Redis
KV Store - Ordered	TokyoTyrant, Lightcloud, NMDB, Luxio, MemcacheDB, Actord
Tuple Store	GigaSpaces, Coord, Apache River
Object Database	ZopeDB, DB4O, Shoal, Perst
Document Store	MarkLogic, CouchDB, MongoDB, Jackrabbit, XML-Databases, ThruDB, CloudKit, Persevere, Riak Basho, Scalaris
Wide Columnar Store	BigTable, HBase, Cassandra, Hypertable, KAI, OpenNeptune, Qbase, KDI

Classification based on feature

Ben Scofield categorized NoSQL databases based on nonfunctional categories ("(il)ities") plus a rating of their feature coverage: [*citation needed*]

Data Model	Performance	Scalability	Flexibility	Complexity	Functionality
Key–value Stores	high	high	high	low	variable (none)
Column Store	high	high	moderate	low	minimal
Document Store	high	variable (high)	high	low	variable (low)
Graph Database	variable	variable	high	high	graph theory
Relational Database	variable	variable	low	moderate	relational algebra

Examples

Document store

The central concept of a document store is the notion of a "document". While each document-oriented database implementation differs on the details of this definition, in general, they all assume that documents encapsulate and encode data (or information) in some standard formats or encodings. Encodings in use include XML, YAML, and JSON as well as binary forms like BSON, PDF and Microsoft Office documents (MS Word, Excel, and so on).

Different implementations offer different ways of organizing and/or grouping documents:

- Collections
- Tags
- Non-visible Metadata
- Directory hierarchies

Compared to relational databases, for example, collections could be considered as tables as well as documents could be considered as records. But they are different: every record in a table has the same sequence of fields, while documents in a collection may have fields that are completely different.

Documents are addressed in the database via a unique **key** that represents that document. One of the other defining characteristics of a document-oriented database is that, beyond the simple key-document (or key–value) lookup that you can use to retrieve a document, the database will offer an API or query language that will allow retrieval of documents based on their contents.

Name	Language	Notes
BaseX	Java, XQuery	XML database
Cloudant	Erlang, Java, Scala, C	JSON store (online service)
Clusterpoint	C++	XML, geared for Full text search
Couchbase Server	Erlang, C, C++	Support for JSON and binary documents
Apache CouchDB	Erlang	JSON database
djondb[2][3][4]	C++	JSON, ACID Document Store
Solr	Java	Search engine
ElasticSearch	Java	JSON, Search engine
eXist	Java, XQuery	XML database
Jackrabbit	Java	Java Content Repository implementation

IBM Lotus Notes and Lotus Domino	LotusScript, Java, IBM X Pages, others	MultiValue
MarkLogic Server	XQuery, Java, REST	XML database with support for JSON, text, and binaries
MongoDB	C++, C#, Go	BSON store (binary format JSON)
ObjectDatabase++	C++, C#, TScript	Binary Native C++ class structures
Oracle NoSQL Database	Java, C	
OrientDB	Java	JSON, SQL support
CoreFoundation Property list	C, C++, Objective-C	JSON, XML, binary
Sedna	XQuery, C++	XML database
SimpleDB	Erlang	online service
TokuMX	C++, C#, Go	MongoDB with Fractal Tree indexing
OpenLink Virtuoso	C++, C#, Java, SPARQL	middleware and database engine hybrid

Graph

This kind of database is designed for data whose relations are well represented as a graph (elements interconnected with an undetermined number of relations between them). The kind of data could be social relations, public transport links, road maps or network topologies, for example.

Name	**Language**	**Notes**
AllegroGraph	SPARQL	RDF GraphStore
IBM DB2	SPARQL	RDF GraphStore added in DB2 10
DEX/Sparksee	Java, C++, .NET, Python	High-performance graph database
FlockDB	Scala	
InfiniteGraph	Java	High-performance, scalable, distributed graph database
Neo4j	Java	
OpenLink Virtuoso	C++, C#, Java, SPARQL	middleware and database engine hybrid
OrientDB	Java	
Sones GraphDB	C#	
Sqrrl Enterprise	Java	Distributed, real-time graph database featuring cell-level security
OWLIM	Java, SPARQL 1.1	RDF graph store with reasoning

Key–value stores

Key–value stores allow the application to store its data in a schema-less way. The data could be stored in a datatype of a programming language or an object. Because of this, there is no need for a fixed data model. The following types exist:

KV - eventually consistent

- Apache Cassandra
- Dynamo
- Hibari
- OpenLink Virtuoso
- Project Voldemort
- Riak

KV - hierarchical

- GT.M
- InterSystems Caché

KV - cache in RAM

- memcached
- redis
- OpenLink Virtuoso
- Hazelcast
- Oracle Coherence

KV - solid state or rotating disk

- Aerospike
- BigTable
- CDB
- Couchbase Server
- Keyspace
- LevelDB
- MemcacheDB (using Berkeley DB)
- MongoDB
- OpenLink Virtuoso
- Tarantool
- Tokyo Cabinet
- Tuple space
- Oracle NoSQL Database

KV - ordered

- Berkeley DB
- FoundationDB
- IBM Informix C-ISAM
- InfinityDB
- MemcacheDB
- NDBM

Object database

- db4o
- GemStone/S
- InterSystems Caché
- JADE
- NeoDatis ODB
- ObjectDatabase++
- ObjectDB
- Objectivity/DB
- ObjectStore
- ODABA
- Perst
- OpenLink Virtuoso

- Versant Object Database
- WakandaDB
- ZODB

Tabular

- Apache Accumulo
- BigTable
- Apache Hbase
- Hypertable
- Mnesia
- OpenLink Virtuoso

Tuple store

- Apache River
- OpenLink Virtuoso
- Tarantool
- GigaSpaces

Triple/Quad Store (RDF) database

- SparkleDB
- Virtuoso Universal Server
- Ontotext-OWLIM
- Apache JENA
- Oracle NoSQL database
- MarkLogic

Hosted

- Freebase
- OpenLink Virtuoso
- Datastore on Google Appengine
- Amazon DynamoDB
- Cloudant Data Layer (CouchDB)

Multivalue databases

- Northgate Information Solutions Reality, the original Pick/MV Database
- Extensible Storage Engine (ESE/NT)
- OpenQM
- Revelation Software's OpenInsight
- Rocket U2
- D3 Pick database
- InterSystems Caché
- InfinityDB

NoSQL databases on the cloud

NoSQL databases can be run on-premises, but are also often run on IaaS or PaaS platforms like Amazon Web Services, RackSpace or Heroku. There are three common deployment models for NoSQL on the cloud:

- **Virtual machine image** - cloud platforms allow users to rent virtual machine instances for a limited time. It is possible to run a NoSQL database on these virtual machines. Users can upload their own machine image with a database installed on it, use ready-made machine images that already include an optimized installation of a database, or install the NoSQL database on a running machine instance.
- **Database as a service** - some cloud platforms offer options for using familiar NoSQL database products as a service, such as MongoDB, Redis and Cassandra, without physically launching a virtual machine instance for the database. The database is provided as a managed service, meaning that application owners do not have to install and maintain the database on their own, and pay according to usage. Some database as a service providers provide additional features, such as clustering or high availability, that are not available in the on-premise version of the database (see the table below for several examples).
- **Native cloud NoSQL databases** - some providers offer a NoSQL database service which is available only on the cloud. A well-known example is Amazon's SimpleDB, a simple NoSQL key-value store. SimpleDB cannot be installed on a local machine and cannot be used on any cloud platform except Amazon's.

The following table provides notable examples of NoSQL databases available on the cloud in each of these deployment models:

Deployment Model	**Database Technology**	**Provider**	**Cloud-Specific Features**	**Pricing Model**
Virtual machine image	MongoDB	MongoDB - machine images for Amazon EC2[5] and Windows Azure[6]	None	• Database and machine image - open source • Amazon/Azure instances - pay per use
Virtual Machine Image	Redis	• Redis - standard open source installation • Script for installation on Amazon EC2 [7] • Recommended installation on Windows Azure [8]	None	• Database and machine image - open source • Amazon/Azure instances - pay per use
Virtual machine image	Cassandra	Apache Cassandra - machine image for Amazon EC2[9]	None	• Database and machine image - open source • Amazon instances - pay per use
Database as a Service	MongoDB	MongoLab[10] - available on Amazon, Google, Joyent, Rackspace and Windows Azure	• Managed service • High availability • Automatic failover • Pre-configured clustering	• Free up to 500MB (on disk)[11] • Paid plans based on architecture and storage size
Database as a Service	Redis/Memcached	Amazon Web Services - ElastiCache[12]	• Managed service • Automatic healing of failed nodes • Resilient system to prevent overloaded DBs • Performance monitoring	• Free for 750 hours on micro instance[13] • Pay per use for machine utilization, no separate charge for data usage[14]

Database as a Service	Redis	RedisToGo[15] - available on Amazon EC2, RackSpace, Heroku, AppHarbor, Orchestra	• Managed service • Daily backups • API enabling creation, deletion, or download of Redis instances	• Free up to 5MB (memory) • Paid plans based on memory usage
Database as a Service	Redis	Redis Cloud (Redis Labs)[16] - available on Amazon EC2, Windows Azure, Heroku, Cloud Foundry, OpenShift, AppFog, AppHarbor	• Managed service • Automatic scaling, unlimited Redis nodes • High availability • Built-in clustering	• Free up to 25MB (memory)[17] • Pay per use
Database as a Service	Cassandra	Instaclustr[18] - available on Amazon EC2, RackSpace, Windows Azure, Joyent, Google Compute Engine	• Managed service • Performance tuning • Monitoring • Automated backups • DataStax OpsCenter for cluster administration	Paid plans based on disk storage, memory usage and CPU cores[19]
Native cloud NoSQL database	Amazon SimpleDB	Amazon Web Services	• Managed service • High availability • Unlimited scale • Data durability	• Free for 750 hours on micro instance[20] • Pay per use - separate charge for machine utilization and data usage
Native cloud NoSQL database	Google App Engine Datastore[21]	Google	• No planned downtime • Atomic transactions • High availability of reads and writes	• Free with quota system limiting instance hours, storage and throughput[22] • Pay per use based on instance hours, storage, throughput and other parameters
Native cloud NoSQL database	SalesForce Database.com[23]	SalesForce	• Unlimited scale • Access to SalesForce meta data • Social API • Support for mobile clients • Multi-tenancy	• Free up to 100K records and 50K transactions[24] • Pay per use based on users, number of records and transactions

References

[1] A Yes for a NoSQL Taxonomy (http://highscalability.com/blog/2009/11/5/a-yes-for-a-nosql-taxonomy.html). High Scalability (2009-11-05). Retrieved on 2013-09-18.

[2] The enterprise class NoSQL database (http://djondb.com). djondb. Retrieved on 2013-09-18.

[3] http://tinman.cs.gsu.edu/~raj/8711/sp13/djondb/Report.pdf

[4] Undefined Blog: Meeting with DjonDB (http://undefvoid.blogspot.com/2013/03/meeting-with-djondb.html). Undefvoid.blogspot.com. Retrieved on 2013-09-18.

[5] " Neo4J in the Cloud (http://wiki.neo4j.org/content/Neo4j_in_the_Cloud)", Neo4J Wiki (http://wiki.neo4j.org), Retrieved 2011-11-10.

[6] " MongoDB on Azure (http://www.mongodb.org/display/DOCS/MongoDB+on+Azure), MongoDB.org (http://www.mongodb.org), Retrieved 2011-11-10.

[7] " Install Redis.sh (https://gist.github.com/dstroot/2776679)", GitHub Gist (https://gist.github.com), Retrieved 2013-12-29.

[8] " Running Redis on a CentOS Linux VM in Windows Azure (http://blogs.msdn.com/b/tconte/archive/2012/06/08/running-redis-on-a-centos-linux-vm-in-windows-azure.aspx)", Thomas Conté's MSDN Weblog (http://blogs.msdn.com/b/tconte/), Retrieved 2013-12-29.

[9] " Setting up Cassandra in the Cloud (http://wiki.apache.org/cassandra/CloudConfig)", Cassandra Wiki (http://wiki.apache.org/cassandra/), Retrieved 2011-11-10.

[10] " MongoLab Product Overview (https://mongolab.com/products/)", MongoLab.com (https://mongolab.com), Retrieved 2013-12-29.

[11] " MongoLab Plans and Pricing (https://mongolab.com/products/pricing/)", MongoLab.com (https://mongolab.com), Retrieved 2013-12-29.
[12] " Amazon ElastiCache (http://aws.amazon.com/elasticache/)", Amazon Web Services (http://aws.amazon.com), Retrieved 2013-12-29.
[13] " Amazon ElastiCache Free Usage Tier (http://aws.amazon.com/elasticache/free/)", Amazon Web Services (http://aws.amazon.com), Retrieved 2013-12-29.
[14] " Amazon ElastiCache Pricing (http://aws.amazon.com/elasticache/pricing/)", Amazon Web Services (http://aws.amazon.com), Retrieved 2013-12-29.
[15] " RedisToGo Documentation (http://www.redistogo.com/documentation?language=en)", RedisToGo.com (http://www.redistogo.com), Retrieved 2013-12-29.
[16] Redis Cloud by Redis Labs (http://redis-cloud.com/), Redis-Cloud.com (http://redis-cloud.com), Retrieved 2013-12-29.
[17] " Garantia Data Pricing (http://garantiadata.com/Pricing)", GarantiaData.com (http://garantiadata.com), Retrieved 2013-12-29.
[18] " Instaclustr Managed Apache Cassandra Hosting (https://www.instaclustr.com/)", Instaclustr.com (https://www.instaclustr.com), Retrieved 2013-12-29.
[19] Instaclustr Providers & Pricing (https://www.instaclustr.com/pricing), Instaclustr.com (https://www.instaclustr.com), Retrieved 2013-12-29.
[20] Amazon SimpleDB Pricing (http://aws.amazon.com/simpledb/#pricing), Amazon Web Services (http://aws.amazon.com), Retrieved 2013-12-29.
[21] " Java Datastore API (https://developers.google.com/appengine/docs/java/datastore/?hl=en)", Google App Engine (https://developers.google.com/appengine/), Retrieved 2013-12-29.
[22] App Engine Pricing (https://cloud.google.com/products/app-engine/#pricing), Google Cloud Platform (https://cloud.google.com), Retrieved 2013-12-29.
[23] " How it works (http://www.database.com/en/howitworks)", Database.com (http://www.database.com), Retrieved 2013-12-29.
[24] " Database.com Pricing (http://www.database.com/en/pricing)", Database.com (http://www.database.com), Retrieved 2013-12-29.

Further reading

- Pramod Sadalage and Martin Fowler (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley. ISBN 0-321-82662-0.
- Christof Strauch (2012). "NoSQL Databases" (http://www.christof-strauch.de/nosqldbs.pdf).
- Moniruzzaman AB, Hossain SA (2013). "NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison" (http://arxiv.org/abs/1307.0191).
- Kai Orend (2013). *Analysis and Classification of NoSQL Databases and Evaluation of their Ability to Replace an Object-relational Persistence Layer* (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.483&rep=rep1&type=pdf).
- Ganesh Krishnan, Sarang Kulkarni, Dharmesh Kirit Dadbhawala. "Method and system for versioned sharing, consolidating and reporting information" (https://www.google.com/patents/US7383272?pg=PA1&dq=ganesh+krishnan&hl=en&sa=X).
- Sugam Sharma. "A Brief Review on Modern NoSQL Data Models, Handling Big Data" (http://www.cs.iastate.edu/~sugamsha/articles).

External links

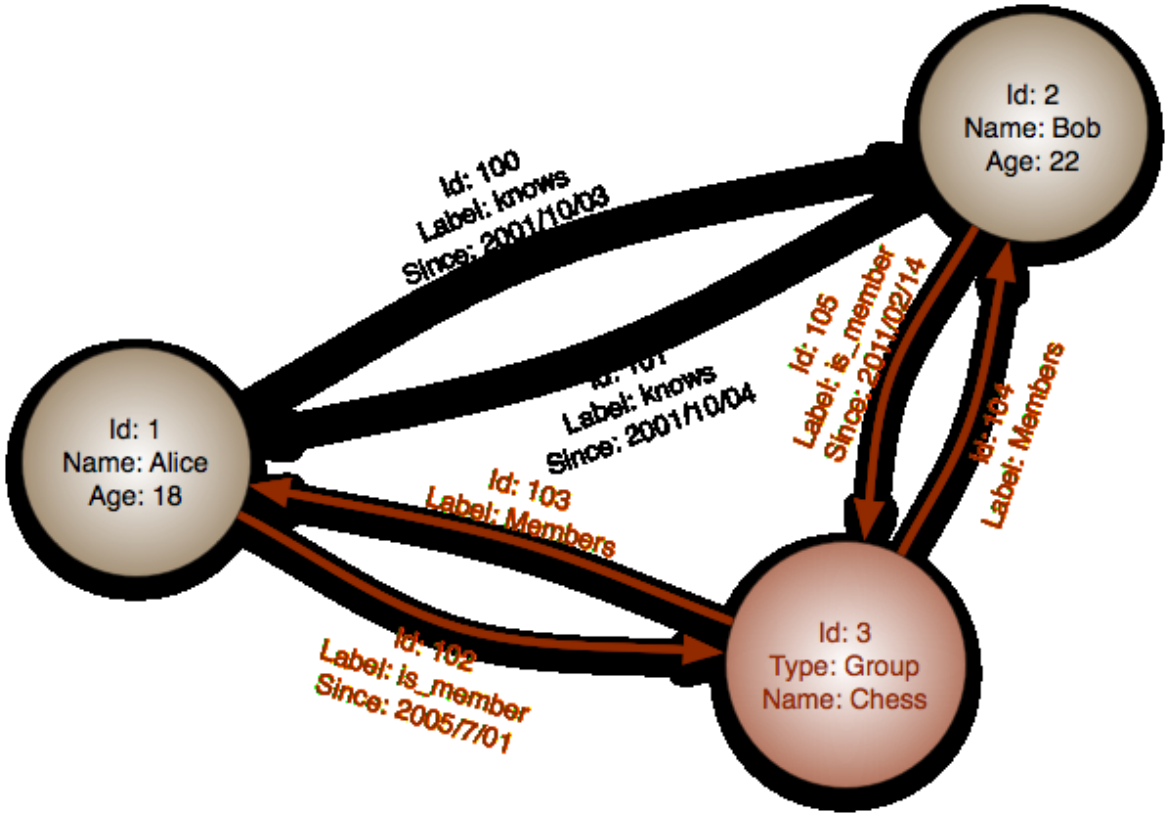
- Christoph Strauch. "NoSQL whitepaper" (http://www.christof-strauch.de/nosqldbs.pdf). Hochschule der Medien, Stuttgart.
- Stefan Edlich. "NoSQL database List" (http://nosql-database.org/).
- Peter Neubauer (2010). "Graph Databases, NOSQL and Neo4j" (http://www.infoq.com/articles/graph-nosql-neo4j).
- Sergey Bushik (2012). "A vendor-independent comparison of NoSQL databases: Cassandra, HBase, MongoDB, Riak" (http://www.networkworld.com/news/tech/2012/102212-nosql-263595.html). NetworkWorld.

Graph database

A **graph database** is a database that uses graph structures with nodes, edges, and properties to represent and store data. A graph database is any storage system that provides index-free adjacency. This means that every element contains a direct pointer to its adjacent elements and no index lookups are necessary. General graph databases that can store any graph are distinct from specialized graph databases such as triplestores and network databases.

Structure

Graph databases are based on graph theory. Graph databases employ nodes, properties, and edges.



Nodes represent entities such as people, businesses, accounts, or any other item you might want to keep track of.

Properties are pertinent information that relate to nodes. For instance, if "Wikipedia" were one of the nodes, one might have it tied to properties such as "website", "reference material", or "word that starts with the letter 'w'", depending on which aspects of "Wikipedia" are pertinent to the particular database.

Edges are the lines that connect nodes to nodes or nodes to properties and they represent the relationship between the two. Most of the important information is really stored in the edges. Meaningful patterns emerge when one examines the connections and interconnections of nodes, properties, and edges.

Properties

Compared with relational databases, graph databases are often faster for associative data sets[*citation needed*], and map more directly to the structure of object-oriented applications. They can scale more naturally to large data sets as they do not typically require expensive join operations. As they depend less on a rigid schema, they are more suitable to manage ad hoc and changing data with evolving schemas. Conversely, relational databases are typically faster at performing the same operation on large numbers of data elements.

Graph databases are a powerful tool for graph-like queries, for example computing the shortest path between two nodes in the graph. Other graph-like queries can be performed over a graph database in a natural way (for example graph's diameter computations or community detection).

Graph database projects

The following is a list of several well-known graph database projects:[1]

Name	Version	License	Language	Description
AllegroGraph	4.11 (June 2013)	Proprietary, Clients - Eclipse Public License v1	C#, C, Common Lisp, Java, Python	A RDF and graph database.
ArangoDB [2]	2.0.0 (March 2014)	Apache 2	C, C++ & Javascript	A distributed multi-model document store and graph database. Highly scalable supporting ACID and full transaction support. Including a built-in graph explorer.
Bigdata [3]		GPL	Java	A RDF/graph database capable of clustered deployment.
Bitsy [4]	1.5.0	AGPL, Enterprise license (unlimited use, annual/perpetual)	Java	A small, embeddable, durable in-memory graph database
BrightstarDB [5]		MIT License [6]	C#	An embeddable NoSQL database for the .NET platform with code-first data model generation.
DEX/Sparksee[7]	5.0.0 (2014)	evaluation, research or development use (free) / commercial use	C++	A high-performance and scalable graph database management system from Sparsity Technologies [8], a technology transition company from DAMA-UPC [9]. Its main characteristics is its query performance for the retrieval & exploration of large networks. Sparksee 5 mobile is the first graph database for mobile devices.
Filament [10]		BSD	Java	A graph persistence framework and associated toolkits based on a navigational query style.
GraphBase [11]	1.0.03a	Proprietary	Java	A customizable, distributed, small-footprint graph store with a rich tool set from FactNexus [12].
Graphd		Proprietary		The proprietary back-end of Freebase.
Horton [13]		Proprietary	C#	A graph database from Microsoft Research Extreme Computing Group (XCG) [14] based on the cloud programming infrastructure Orleans [15].
HyperGraphDB [16]	1.2 (2012)	LGPL	Java	A graph database supporting generalized hypergraphs where edges can point to other edges.
InfiniteGraph [17]	3.0 (January 2013)	Proprietary	Java	A distributed and cloud-enabled commercial product with flexible licensing.
InfoGrid [18]	2.9.5 (2011)	AGPLv3, free for small entities[19]	Java	A graph database with web front end and configurable storage engines (MySQL, PostgreSQL, Files, Hadoop).
jCoreDB Graph [20]				An extensible database engine with a graph database subproject.

Neo4j	1.9.2 [21] (July 2013)	GPLv3 Community Edition. Commercial & AGPLv3 options for Enterprise and Advanced editions[22]	Java	A highly scalable open source graph database that supports ACID, has high-availability clustering for enterprise deployments, and comes with a web-based administration tool that includes full transaction support and visual node-link graph explorer.[23] Neo4j is accessible from most programming languages using its built-in REST web API interface. Neo4j is the most popular graph database in use today.[24]
OpenLink Virtuoso				A RDF graph database server, deployable as a local embedded instance (as used in the Nepomuk Semantic Desktop), a single-instance network server, or a shared-nothing network cluster instance.
Oracle Spatial and Graph [25]	11.2 (2012)	Proprietary	Java, PL/SQL	1) RDF Semantic Graph: comprehensive W3C RDF graph management in Oracle Database with native reasoning and triple-level label security. 2) Network Data Model property graph: for physical/logical networks with persistent storage and a Java API for in-memory graph analytics.
Oracle NoSQL Database [26]	2.0.39 (2013)	Proprietary	Java	RDF Graph for Oracle NoSQL Database is a feature of Enterprise Edition providing W3C RDF graph capabilities in NoSQL Database.
OrientDB	1.6.1 (November 2013)	Apache 2	Java	A distributed Graph Database with a hybrid model taken from Document Database.
OQGRAPH [27]		GPLv2		A graph computation engine for MySQL, MariaDB and Drizzle.
Ontotext OWLIM [28]	5.3	OWLIM Lite is free OWLIM SE and Enterprise are commercially licenced	Java	A graph database engine, based entirely on Semantic Web standards from W3C: RDF, RDFS, OWL, SPARQL. OWLIM Lite is an "in memory" engine. OWLIM SE is robust standalone database engine. OWLIM Enterprise is a clustered version which offers horizontal scalability and failover support and other enterprise features.
R2DF [29]				R2DF framework for ranked path queries over weighted RDF graphs.
ROIS [30]		Freeware	Modula-2	A programmable knowledge server that supports inheritance and transitivity. Used in OpenGALEN as a Terminology Server.
sones GraphDB		AGPLv3[31]	C#	A graph database and universal access layer (funded by Deutsche Telekom).
Sqrrl Enterprise [32]	v1.1 (2013)	Proprietary	Java	Distributed, real-time graph database featuring cell-level security and massive scalability.
Teradata Aster [33]	v6 (2013)	Proprietary	Java, SQL, Python, C++, R	A high performance, multi-purpose, highly scalable and extensible MPP database incorporating patented engines supporting native SQL, MapReduce and Graph data storage and manipulation. An extensive set of analytical function libraries and data visualization capabilities are also provided.
Titan [34]	0.4.1 (2013)	Apache 2	Java	A distributed, real-time, transactional graph database developed by Aurelius [35].
Trinity [36]			C#, C, X64 Assembly	A distributed general purpose graph engine on a memory cloud.
VelocityGraph [37]		Open source with proprietary back-end	C#	High performance, scalable & flexible graph database build with VelocityDB [38] object database.
VertexDB [39]		Revised BSD	C	A graph database server that supports automatic garbage collection.

WhiteDB [40]	0.7.0 (October 2013)	GPLv3 and a free commercial licence	C	A graph/N-tuples shared memory database library.

Graph database features

The following table compares the features of the above graph databases.

Name	Graph Model	API	Query Methods	Visualizer	Consistency	Backend	Scalability
AllegroGraph	RDF	Java, Java:Sesame, JavaJena, Python, Ruby, Perl, C#, Clojure, Lisp, Scala, REST	SPARQL 1.1, Prolog, JIG, JavaScript	Gruff - View Graphs, Visual Query Builder for SPARQL and Prolog	ACID	Native Graph Storage	1 Trillion RDF triples
ArangoDB [2]	Property Graph [41]	JavaScript, Blueprints, REST	Graph Traversals via JavaScript, Gremlin	Built-in graph explorer	MVCC/ACID	native C/C++	Replication and Sharding
Bigdata [3]							
Bitsy [4]	Property Graph	Blueprints	Gremlin, Pixy [42]		ACID with optimistic concurrency control	Human-readable JSON-encoded text files with checksums and markers for recovery	
DEX/Sparksee[43]	Labeled and directed attributed multigraph	Java, C++, .NET, Python	Native Java, C#, Python and C++ APIs, Blueprints, Gremlin	Exporting functionality to visualization formats	Consistency, durability and partial isolation and atomicity	Native graph. light and independent data structures with a small memory footprint for storage	Master/Slave replication
Filament [10]							
GraphBase Enterprise(1) [44] GraphBase Agility(2) [45]	(1) mixed, (2) Framework-managed Simple Graph	Java	Bounds Language, embedded java	GraphPad, BoundsPad, Navigator	ACID, graph-based transactions	proprietary native	(1) shared nothing distributed, (2) simple replication, 100+ Billion arcs per server
Graphd							
Horton [13]	Attributed multigraph		Horton Query Language (Regular Language Expression + SQL)			C#, .Net Framework, Asynchronous communication protocols	

HyperGraphDB [16]	Object-oriented multi-relational labeled hypergraph	Custom,Java			MVCC/STM		
InfiniteGraph [17]	Labeled and directed multi-property graph	Java, Blueprints (Read Only)	Java (with parallel, distributed queries), Gremlin (Read Only)	Graph browser for developers. Plugins to allow use of external libraries.	ACID. There is also a parallel, loosely synchronized batch loader.	Objectivity/DB on standard filesystems	Distributed & Sharded. Objectivity/DB was the first DBMS to store a Petabyte of objects.
InfoGrid [18]	Dynamically typed, object-oriented graph, multigraphs, semantic models						
jCoreDB Graph [20]							
Neo4j	Property Graph	Java, Python, JPython, Ruby, JRuby, JavaScript (Node.js), PHP, .NET, Django, Clojure, Spring, Scala, or REST (any language)	Cypher (native/preferred), Native Java APIs (special cases), Traverser API, REST, Blueprints, Gremlin	Data Browser included. Supports a variety of 3rd party tools: Gephi, Linkurio.us, Cytoscape, Tom Sawyer, Keylines, etc.	ACID	Native graph storage with native graph processing engine	Horizontal read scaling via master-slave clustering with cache sharding.
OpenLink							
Oracle Spatial and Graph [25]	RDF graph: Triple & Quad (named graphs); Network Data Model property graph	Java; Apache Jena; PL/SQL	SPARQL 1.1; SPARQL web service end point; SQL	SPARQL-compliant tools; Apache Jena-based tools; XML & JSON-based tools; SQL based tools	ACID	Efficient, compressed, partitioned graph storage; Native persisted in-database inferencing; SPARQL 1.1 & SQL integration; Triple-level label security; Semantic indexing of documents	Parallel load, query, inference; Query controls; Scales from PC to Oracle Exadata; Supports Oracle Real Application Clusters and Oracle Database 8 exabytes
Oracle NoSQL Database [26]	RDF graph: Triple default graph, Triple & Quad named graphs	Java (Apache Jena)	SPARQL 1.1; SPARQL web service end point	SPARQL-compliant tools; Apache Jena-based tools; XML & JSON-based tools	ACID; Configurable consistency & durability policies	Key/value store; W3C SPARQL 1.1 & Update; In-memory RDFS/OWL inferencing	Parallel load/query; Query controls for: parallel execution, timeout, query optimization hints

OrientDB	Property Graph	Java Traverser API, Blueprints, Rexster, Javacript[46]	Own SQL-like Query Language, Gremlin		ACID, MVCC	Custom on disc or in memory	
OQGRAPH [27]							
R2DF [29]							
ROIS [30]							
sones GraphDB							
Sqrrl Enterprise [47]	Property Graph	Thrift, Blueprint	Own SQL-like query language and Java API	Integrates with 3rd party tools	Fully Consistent and ACID (transactions limited to a single node)	Accumulo	Distributed cluster with tens of trillions of edges [48]
Titan [49]	Property Graph	Java, Blueprints, REST, RexPro binary protocol, Python, Clojure (any language)	Gremlin, SPARQL	Integrates with 3rd party tools	ACID or Eventually Consistent	Cassandra, HBase, MapR M7 Tables, Berkeley DB, Persistit, Hazelcast	Distributed cluster (120 billion+ edges) or single server.
Trinity [36]	Cell Based Graph Model [50]	C#	Trinity Query Language		Cell level Atomicity [50]	Native graph store and processing engine	billion node in-memory graph
VertexDB [39]							

Distributed Graph Processing

- Angrapa [51] - graph package in Hama [52], a bulk synchronous parallel (BSP) platform
- Apache Hama [52] - a pure BSP(Bulk Synchronous Parallel) computing framework on top of HDFS (Hadoop Distributed File System) for massive scientific computations such as matrix, graph and network algorithms.
- Bigdata [3] - a RDF/graph database capable of clustered deployment.
- Faunus [53] - a Hadoop-based graph computing framework that uses Gremlin as its query language. Faunus provides connectivity to Titan, Rexster-fronted graph databases, and to text/binary graph formats stored in HDFS. Faunus is developed by Aurelius [35].
- FlockDB - an open source distributed, fault-tolerant graph database based on MySQL and the Gizzard framework for managing Twitter-like graph data (single-hop relationships) FlockDB on GitHub [54].
- Giraph [55] - a Graph processing infrastructure that runs on Hadoop (see Pregel).
- GraphBase [56] - Enterprise Edition supports embedding of callable Java Agents within the vertices of a distributed graph.
- GoldenOrb [57] - Pregel implementation built on top of Apache Hadoop
- GraphLab [58] - A framework for machine learning and data mining in the cloud

- GraphX [59] - GraphLab built on the Spark [60] cluster computing system. Dr. Joseph Gonzalez is the project lead, the creator of GraphLab.
- HipG [61] - a library for high-level parallel processing of large-scale graphs. HipG is implemented in Java and is designed for distributed-memory machine
- InfiniteGraph [17] - a commercially available distributed graph database that supports parallel load and parallel queries.
- JPregel [62] - In-memory java based Pregel implementation
- KDT [63] - An open-source distributed graph library with a Python front-end and C++/MPI backend (Combinatorial BLAS [64]).
- OpenLink Virtuoso - the shared-nothing Cluster Edition supports distributed graph data processing.
- Oracle Spatial and Graph [25] - loading, inferencing, and querying workloads are automatically and transparently distributed across the nodes in an Oracle Real Application Cluster, Oracle Exadata Database Machine, and Oracle Database Appliance.
- Phoebus [65] - Pregel implementation written in Erlang
- Pregel [66] - Google's internal graph processing platform, released details in ACM paper.
- Powergraph [67] - Distributed graph-parallel computation on natural graphs.
- PowerLyra [68] - differentiated graph computation and partitioning on skewed graphs (dynamically applying different computation and partition strategies for different vertices).
- Sedge [69] - A framework for distributed large graph processing and graph partition management (including an open source version of Google's Pregel)
- Signal/Collect [70] - a framework for parallel graph processing written in Scala
- Sqrrl Enterprise - distributed graph processing utilizing Apache Accumulo and featuring cell-level security, massive scalability, and JSON support
- Titan [49] - A distributed, disk-based graph database developed by Aurelius [35].
- Trinity [71] - Distributed in-memory graph engine under development at Microsoft Research Labs.
- Parallel Boost Graph Library (PBGL) [72] - a C++ library for graph processing on distributed machines, part of Boost framework.
- Mizan [73] - An optimized Pregel clone that can be deployed easily on Amazon EC2, local clusters, stand-alone Linux systems and supercomputers (IBM BlueGene/P). It utilizes runtime graph repartitioning between iterations to provide dynamic load balancing for better algorithm performance.[74]

APIs and Graph Query/Programming Languages

- Bounds Language [75] - terse C-style syntax which initiates concurrent traversals in GraphBase and supports interaction between them.
- Blueprints [76] - a Java API for Property Graphs from TinkerPop [77] and supported by a few graph database vendors.
- Blueprints.NET [78] - a C#/.NET API for generic Property Graphs.
- Bulbflow [79] - a Python persistence framework for Rexster, Titan, and Neo4j Server.
- Cypher Query Language [80] - a declarative graph query language for Neo4j that enables ad hoc as well as programmatic (SQL-like) access to the graph
- Gremlin [81] - an open-source graph programming language that works over various graph database systems.
- Neo4jClient [82] - a .NET client for accessing Neo4j.
- Neography [83] - a thin Ruby wrapper that provides access to Neo4j via REST.
- Neo4jPHP [84] - a PHP library wrapping the Neo4j graph database.
- NodeNeo4j [85] - a Node.js driver for Neo4j that provides access to Neo4j via REST
- Pacer [86] - a Ruby dialect/implementation of the Gremlin graph traversal language.

- Pipes [87] - a lazy dataflow framework written in Java that forms the foundation for various property graph traversal languages.
- Pixy [42] - a declarative graph query language that works on any Blueprints-compatible graph database
- PYBlueprints [88] - a Python API for Property Graphs.
- Pygr [89] - a Python API for large-scale analysis of biological sequences and genomes, with alignments represented as graphs.
- Rexster [90] - a graph database server that provides a REST or binary protocol API (RexPro). Supports Titan, Neo4j, OrientDB, Dex, and any TinkerPop/Blueprints-enabled graph.
- SPARQL - a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format.
- SPASQL [91] - an extension of the SQL standard, allowing execution of SPARQL queries within SQL statements, typically by treating them as subquery or function clauses. This also allows SPARQL queries to be issued through "traditional" data access APIs (ODBC, JDBC, OLE DB, ADO.NET, etc.)
- Spring Data Neo4j [92] - an extension to Spring Data [93] (part of the Spring Framework), providing direct/native access to Neo4j
- Oracle SQL and PL/SQL APIs [25] - have graph extensions for Oracle Spatial and Graph.
- Styx [94] (previously named Pipes.Net) - a data flow framework for C#/.NET for processing generic graphs and Property Graphs.
- Thunderdome [95] - a Titan Rexster Object-Graph Mapper for Python

References

[1] http://graph-database.org
[2] http://www.arangodb.org
[3] http://www.bigdata.com/blog
[4] http://bitbucket.org/lambdazen/bitsy
[5] http://www.brightstardb.com
[6] http://brightstardb.com/blog/2013/02/brightstardb-goes-open-source/
[7] http://sparsity-technologies.com#sparksee
[8] http://sparsity-technologies.com
[9] http://www.dama.upc.edu/technology-transfer/dex
[10] http://filament.sourceforge.net/
[11] http://graphbase.net/
[12] http://factnexus.com/
[13] http://research.microsoft.com/en-us/projects/ldg
[14] http://research.microsoft.com/en-us/labs/xcg
[15] http://research.microsoft.com/en-us/projects/orleans/default.aspx
[16] http://www.hypergraphdb.org
[17] http://infinitegraph.com
[18] http://infogrid.org/
[19] http://infogrid.org/wiki/Docs/License
[20] http://www.jcoredb.org
[21] http://www.neo4j.org/download
[22] neo4j.org (http://www.neo4j.org)
[23] Neo4j, World's Leading Graph Database (http://www.neotechnology.com/neo4j-graph-database/). Retrieved September 16, 2013.
[24] DB-Engines Ranking of Graph DBMS (http://db-engines.com/en/ranking/graph+dbms). Retrieved July 19, 2013.
[25] http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/index.html
[26] http://www.oracle.com/technetwork/products/nosqldb/overview/index.html
[27] http://openquery.com/graph
[28] http://www.ontotext.com/owlim
[29] http://dl.acm.org/citation.cfm?id=1988736/
[30] http://rois.eggbird.eu/
[31] http://sones.com/
[32] http://sqrrl.com/

[33] http://www.asterdata.com/
[34] http://titan.thinkaurelius.com/
[35] http://thinkaurelius.com
[36] http://research.microsoft.com/trinity/
[37] http://www.VelocityGraph.com
[38] http://www.VelocityDB.com
[39] http://www.dekorte.com/projects/opensource/vertexdb/
[40] http://whitedb.org
[41] https://github.com/tinkerpop/blueprints/wiki/Property-Graph-Model
[42] https://github.com/lambdazen/pixy/wiki
[43] http://sparsity-technologies.com#sparksee
[44] http://graphbase.net/Enterprise.html/
[45] http://graphbase.net/Agility.html/
[46] OrientDB Javascript API wiki on the project's Github page (https://github.com/orientechnologies/orientdb/wiki/Javascript-API)
[47] http://sqrrl.com
[48] http://www.pdl.cmu.edu/SDI/2013/slides/big_graph_nsa_rd_2013_56002v1.pdf
[49] http://thinkaurelius.github.com/titan/
[50] http://research.microsoft.com/apps/pubs/default.aspx?id=183710
[51] http://wiki.apache.org/hama/GraphPackage
[52] http://incubator.apache.org/hama/
[53] http://thinkaurelius.github.com/faunus/
[54] https://github.com/twitter/flockdb
[55] http://incubator.apache.org/giraph/
[56] http://graphbase.net/Enterprise.html
[57] http://www.goldenorbos.org
[58] http://graphlab.org
[59] http://amplab.github.io/graphx/
[60] http://spark.incubator.apache.org/
[61] http://www.cs.vu.nl/~ekr/hipg/
[62] http://kowshik.github.com/JPregel/
[63] http://kdt.sourceforge.net
[64] http://gauss.cs.ucsb.edu/~aydin/CombBLAS/html/index.html
[65] http://github.com/xslogic/phoebus
[66] http://portal.acm.org/citation.cfm?id=1582723
[67] http://graphlab.org/powergraph-presented-at-osdi/
[68] http://ipads.se.sjtu.edu.cn/projects/powerlyra.html
[69] http://grafia.cs.ucsb.edu/sedge/
[70] http://code.google.com/p/signal-collect/
[71] http://research.microsoft.com/en-us/projects/trinity/
[72] http://www.boost.org/doc/libs/1_51_0/libs/graph_parallel/doc/html/index.html
[73] http://thegraphsblog.wordpress.com/the-graph-blog/mizan/
[74] http://dl.acm.org/citation.cfm?id=2465369
[75] http://graphbase.net/JavaAPIHelp.html#BoundsLanguage
[76] http://blueprints.tinkerpop.com
[77] http://www.tinkerpop.com/
[78] https://github.com/Vanaheimr/Blueprints.NET
[79] http://bulbflow.com
[80] http://docs.neo4j.org/chunked/snapshot/cypher-query-lang.html
[81] http://gremlin.tinkerpop.com/
[82] http://hg.readify.net/neo4jclient
[83] https://github.com/maxdemarzi/neography/
[84] https://github.com/jadell/neo4jphp/wiki
[85] https://github.com/thingdom/node-neo4j
[86] http://github.com/pangloss/pacer
[87] http://pipes.tinkerpop.com
[88] http://pypi.python.org/pypi/pyblueprints/0.1
[89] http://code.google.com/p/pygr/
[90] http://rexster.tinkerpop.com
[91] http://www.w3.org/wiki/SPASQL

[92] http://www.springsource.org/spring-data/neo4j
[93] http://www.springsource.org/spring-data
[94] https://github.com/ahzf/Styx
[95] https://github.com/StartTheShift/thunderdome

External links

- NoSQL Frankfurt 2010 - The GraphDB Landscape and sones (http://www.slideshare.net/ahzf/nosql-frankfurt-2010-the-graphdb-landscape-and-sones)
- Graph Databases and the Future of Large-Scale Knowledge Management (http://highscalability.com/paper-graph-databases-and-future-large-scale-knowledge-management)
- Graphs in the database: SQL meets social networks (http://techportal.ibuildings.com/2009/09/07/graphs-in-the-database-sql-meets-social-networks/)
- Social networks in the database: using a graph database (http://blog.neo4j.org/2009/09/social-networks-in-database-using-graph.html)
- Scaling Online Social Networks without Pains (http://netdb09.cis.upenn.edu/netdb09papers/netdb09-final3.pdf)
- Large-scale Graph Computing at Google (http://googleresearch.blogspot.com/2009/06/large-scale-graph-computing-at-google.html)
- Eric Lai. (2009, July 1). No to SQL? Anti-database movement gains steam (http://www.computerworld.com/s/article/9135086/No_to_SQL_Anti_database_movement_gains_steam_)
- Renzo Angles, Claudio Gutierrez. Survey of graph database models (http://portal.acm.org/citation.cfm?id=1322433). ACM Computing Surveys, Feb. 2008.
- InfoGrid (http://infogrid.org/) - an open-source application platform including a graph database
- Rodriguez, M.A., Neubauer, P, The Graph Traversal Pattern (http://arxiv.org/abs/1004.1001) article.
- Optimizing Schema-Last Tuple-Store Queries in Graphd (http://portal.acm.org/citation.cfm?id=1807283) SIGMOD 2010

DEX (Graph database)

DEX

Developer(s)	Sparsity Technologies [1]
Initial release	2008
Stable release	v5.0 / 2013
Development status	Active
Operating system	Cross-platform
Type	Graph Database
License	Dual-licensed: personal evaluation use / commercial use
Website	Sparsity-Technologies: DEX [2]

DEX is a high-performance and scalable graph database management system written in C++.

Its development started on 2006 and its first version was available on Q3 - 2008. Fourth version is available since Q3-2010. There's a free community version, for academic or evaluation purposes, available to download (link web) limited to 1 Million nodes, no limit on edges.

DEX is a product originated by the research carried out at DAMA-UPC (Data Management group at the Polytechnic University of Catalonia). On March 2010 a spin-off called **Sparsity-Technologies** has been created at the UPC to commercialize and give services to the technologies developed at DAMA-UPC.

DEX changed name to Sparksee [3] on its 5th release on February 2014.

Graph Model [4]

DEX is based on a graph database model,[5] that is basically characterized by three properties: data structures are graphs or any other structure similar to a graph; data manipulation and queries are based on graph-oriented operations; and there are data constraints to guarantee the integrity of the data and its relationships.

A DEX graph is a Labeled Directed Attributed Multigraph. Labeled because nodes and edges in a graph belong to types. Directed because it supports directed edges as well as undirected. Attributed because both nodes and edges may have attributes and Multigraph meaning that there may be multiple edges between the same nodes even if they are from the same edge type.

One of its main characteristics is its performance storage and retrieval for large graphs (in the order of billions of nodes, edges and attributes) implemented with specialized structures.

Technical Details

- Programming Language: C++
- API: Java, .NET, C++, Python
- OS Compatibility: Windows, Linux, Mac OS, iOS, BB10
- Persistency: Disk
- Transactions: aCiD (Dex guarantees consistency and durability but only partial isolation and atomicity)
- Recovery Manager

References

[1] http://www.sparsity-technologies.com/

[2] http://www.sparsity-technologies.com/dex

[3] http://www.sparsity-technologies.com/#sparksee

[4] Martínez-Bazan, N., Muntés-Mulero, V., Gómez-Villamor, S., Nin, J., Sánchez-Martínez, M., and Larriba-Pey, J. 2007. Dex: high-performance exploration on large graphs for information retrieval. In Proceedings of the Sixteenth ACM Conference on Conference on information and Knowledge Management (Lisbon, Portugal, November 06–10, 2007). CIKM '07. ACM, New York, NY, 573-582.

[5] R. Angles and C. Gutierrez. Survey of graph database models. Technical Report TR/DCC-2005-10, Computer Science Department, Universidad de Chile, October 2005.

Also

- D. Domínguez-Sal, P. Urbón-Bayes, A.Giménez-Vañó, S. Gómez-Villamor, N.Martínez-Bazán, J.L. Larriba-Pey. Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark. International Workshop on Graph Databases. July 2010. (http://www.icst.pku.edu.cn/IWGD2010/index.html)

External links

- DEX homepage at Sparsity-Technologies (http://www.sparsity-technologies.com/dex)

Neo4j

Neo4j

Developer(s)	Neo Technology
Initial release	2007
Written in	Java
Operating system	Cross-platform
Type	Graph database
License	Dual-licensed: GPLv3 and AGPLv3 / commercial
Website	neo4j.org [1]

Neo4j is an open-source graph database, implemented in Java. The developers describe Neo4j as "embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables". Neo4j is the most popular graph database.

Neo4j version 1.0 was released in February, 2010. The community edition of the database is licensed under the free GNU General Public License (GPL) v3. The additional modules, such as online backup and high availability, are licensed under the free Affero General Public License (AGPL) v3. The database, with the additional modules, is also available under a commercial license, in a dual license model.

Neo4j version 2.0 was released in December, 2013.

Neo4j was developed by Neo Technology, Inc., based in the San Francisco Bay Area, US and Malmö, Sweden. Neo Technology board of directors consists of Rod Johnson, (founder of the Spring Framework), Magnus Christerson (Vice President of Intentional Software Corp), Nikolaj Nyholm (CEO of Polar Rose), Sami Ahvenniemi (Partner at Conor Venture Partners) and Johan Svensson (CTO of Neo Technology).

References

[1] http://neo4j.org

External links

- Official website (http://neo4j.org)
- Neo Technology website (http://neotechnology.com/)

Sones GraphDB

GraphDB

Developer(s)	sones GmbH
Stable release	2.0 / May 11, 2011
Operating system	Cross-platform
Available in	English, German
Type	Database
License	Dual licensing: Community Edition is AGPLv3, Enterprise Edition is commercial & proprietary
Website	sones.de [1]

Sones GraphDB was declared bankrupt in Germany on January 1, 2012.

Sones GraphDB was developed by the company *sones* in Erfurt and Leipzig. *GraphDB* is a new type of database with its design based on weighted graphs. The open source edition has been available since July 2010.[2] The commercially available enterprise version offers a wider variety of functions.

GraphDB was entirely developed in C# and runs on Microsoft's .NET platform as well as on the open source reimplementation Mono.[3][4]

GraphDB is available as SaaS on the Microsoft cloud Azure Services Platform.[5]

The company *sones* is a member of Lisog; *GraphDB* is a component of an open source solution stack.

Functionality

Index-free adjacency

Graph databases like the sones GraphDB address the paradigm of the index-free adjacency. That means that is not necessary to manage a global index for relationships between nodes/entities. The linked objects contain direct reference to their adjacent neighboring nodes.

Handling semi-structured data

The semi-structured data approach was established in the mid-90s. This approach is based on the fact that many application areas rarely allow a structured table structure due to complex information characteristics. An example of such problem domains can be found in bioinformatics or the Semantic Web. The sones graph database is able to store and retrieve unstructured properties in any node of the graph. The idea is also to transfer unstructured data to structured data and vice versa.

Dynamic type extension

Another advantage is that structured data can be dynamically extended with high performance in nodes and edges during runtime. Additional properties can easily be entered or deleted from vertex types in a short amount of time. The number of nodes is irrelevant here.

Graph query language

The sones GraphQL is a user-friendly domain-specific language and can be thought of as an "SQL for graphs." The similarity to SQL is intentional and makes the transition much easier for developers/consultants. It enables queries to the sones graph database property hypergraph and can be dynamically extended during runtime using plugins such as functions or aggregates.

Solving the object-relational depiction problem

sones GraphDB solves this problem by using an object-oriented concept. This results in better integration into object-oriented languages, since no O/R mapper is required.

Interfaces

REST API

In addition to providing a number of interfaces (e.g., Java, C#, WebShell, WebDAV) the *sones* graph database also offers a REST API. This enables uncomplicated interaction with state-of-the-art web technologies. A REST-query is all that is required to execute CRUD operations directly on the database.

Traverser API

The Traverser API makes it possible to analyze local data. Based on a number of nodes (local), neighboring nodes can be searched recursively (breadth/depth first).

Architecture

The *sones GraphDB* has a modular structure consisting of 4 application layers. The storage engines act as the interface to different storage media. The *GraphFS* serializes and deserializes database objects (nodes and edges) and operates the available storage engines. The actual graph-oriented database logic as well as all functionalities specific to the database are implemented in the *GraphDB*. The *GraphDS* provides the interface for using the database. The interfaces between the application layers are generic, which makes it possible to update components separately.

References

[1] http://www.sones.de/static-en/
[2] Announcement of open source version (German) (http://www.heise.de/open/meldung/sones-gibt-Quellcode-der-GraphDB-frei-1032398.html)
[3] Mono Release Notes 2.8 mit Verweis auf GraphDB (http://www.mono-project.com/Release_Notes_Mono_2.8)
[4] .NET versus Mono Benchmark (http://www.linuxtoday.com/news_story.php3?ltsn=2010-09-11-004-35-NW-SW)
[5] Azure SaaS Announcement (German) (http://www.heise.de/developer/meldung/GraphDB-als-SaaS-fuer-Windows-Azure-1049816.html)

External links

- Official website (dead link!) (http://www.sones.de/static-en/)Wikipedia:Link rot
- Download of open source edition (dead link!) (http://developers.sones.de/wiki/doku.php?id=trysones)Wikipedia:Link rot

- Download of community edition 2.1 (https://github.com/sones/sones)
- German Interview with Alexander Oelling on RadioTUX (http://blog.radiotux.de/2010/12/13/sendung-graphdb/)
- Presentation on the GraphDB at the 2010 NoSQL conference in Frankfurt (http://www.slideshare.net/ahzf/nosql-frankfurt-2010-the-graphdb-landscape-and-sones?from=ss_embed)
- Sones at TechCrunch (http://techcrunch.com/2011/01/17/nosql-graphdb-maker-sones-raises-millions-to-expand-cloud-computing-business)

Apache Cassandra

Apache Cassandra

Original author(s)	Avinash Lakshman, Prashant Malik
Developer(s)	Apache Software Foundation, DataStax
Initial release	2008
Stable release	2.0.6 / March 10, 2014
Development status	Active
Written in	Java
Operating system	Cross-platform
Available in	English
Type	Database
License	Apache License 2.0
Website	cassandra.apache.org [1]

Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients.

Cassandra also places a high value on performance. University of Toronto researchers studying NoSQL systems concluded that "In terms of scalability, there is a clear winner throughout our experiments. Cassandra achieves the highest throughput for the maximum number of nodes in all experiments."

Cassandra's data model is a partitioned row store with tunable consistency. Rows are organized into tables; the first component of a table's primary key is the partition key; within a partition, rows are clustered by the remaining columns of the key. Other columns may be indexed separately from the primary key.

Tables may be created, dropped, and altered at runtime without blocking updates and queries.

Cassandra does not support joins or subqueries, except for batch analysis via Hadoop. Rather, Cassandra emphasizes denormalization through features like collections.

History

Apache Cassandra was developed at Facebook to power their Inbox Search feature by Avinash Lakshman (one of the authors of Amazon's Dynamo) and Prashant Malik. It was released as an open source project on Google code in July 2008. In March 2009, it became an Apache Incubator project. On February 17, 2010 it graduated to a top-level project.

Releases after graduation include

- 0.6, released Apr 12 2010, added support for integrated caching, and Apache Hadoop MapReduce[2]
- 0.7, released Jan 08 2011, added secondary indexes and online schema changes[3]
- 0.8, released Jun 2 2011, added the Cassandra Query Language (CQL), self-tuning memtables, and support for zero-downtime upgrades[4]
- 1.0, released Oct 17 2011, added integrated compression, leveled compaction, and improved read performance[5]
- 1.1, released Apr 23 2012, added self-tuning caches, row-level isolation, and support for mixed ssd/spinning disk deployments[6]
- 1.2, released Jan 2 2013, added clustering across virtual nodes, inter-node communication, atomic batches, and request tracing[7]
- 2.0, released Sep 4 2013, added lightweight transactions, triggers, improved compactions
- 2.0.4, released Dec 30 2013, added allowing specifying datacenters to participate in a repair, client encryption support to sstableloader, allow removing snapshots of no-longer-existing CFs[8]

Licensing and support

Apache Cassandra is an Apache Software Foundation project, so it has an Apache License (version 2.0).

Main features

Decentralized

> Every node in the cluster has the same role. There is **no single point of failure**. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request.

Supports replication and multi data center replication

> Replication strategies are configurable.[9] Cassandra is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers. Key features of Cassandra's distributed architecture are specifically tailored for multiple-data center deployment, for redundancy, for failover and disaster recovery.

Scalability

> Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.

Fault-tolerant

> Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.

Tunable consistency

> Writes and reads offer a tunable level of consistency, all the way from "writes never fail" to "block for all replicas to be readable", with the quorum level in the middle.

MapReduce support

> Cassandra has Hadoop integration, with MapReduce support. There is support also for Apache Pig and Apache Hive.[10]

Query language

CQL (Cassandra Query Language) was introduced, a SQL-like alternative to the traditional RPC interface. Language drivers are available for **Java** (JDBC), **Python** (DBAPI2) and **Node.JS** (Helenus).

Data model

Cassandra is essentially a hybrid between a key-value and a column-oriented (or tabular) database.

A column family resembles a table in an RDBMS. Column families contain rows and columns. Each row is uniquely identified by a row key. Each row has multiple columns, each of which has a name, value, and a timestamp. Unlike a table in an RDBMS, different rows in the same column family do not have to share the same set of columns, and a column may be added to one or multiple rows at any time.

Each key in Cassandra corresponds to a value which is an object. Each key has values as columns, and columns are grouped together into sets called column families.

Thus, each key identifies a row of a variable number of elements. These column families could be considered then as tables. A table in Cassandra is a distributed multi dimensional map indexed by a key.

Furthermore, applications can specify the sort order of columns within a Super Column or Simple Column family.

Clustering

When the cluster for Apache Cassandra is designed, an important point is to select the right partitioner. Two partitioners exist:

1. RandomPartitioner (RP): This partitioner randomly distributes the key-value pairs over the network, resulting in a good load balancing. Compared to OPP, more nodes have to be accessed to get a number of keys.
2. OrderPreservingPartitioner (OPP): This partitioner distributes the key-value pairs in a natural way so that similar keys are not far away. The advantage is that fewer nodes have to be accessed. The drawback is the uneven distribution of the key-value pairs.

Prominent users

- Apixio uses Cassandra to store its Patient Object Model and extracted features about patients and patient populations
- AppScale uses Cassandra as a back-end for Google App Engine applications
- Cisco's WebEx uses Cassandra to store user feed and activity in near real time.
- The CERN ATLAS experiment uses Cassandra to archive its online DAQ system's monitoring information
- Cloudkick uses Cassandra to store the server metrics of their users.[11]
- Constant Contact uses Cassandra in their email and social media marketing applications. Over 200 nodes are deployed.
- Digg, a large social news website, announced on Sep 9th, 2009 that it is rolling out its use of Cassandra and confirmed this on March 8, 2010. TechCrunch has since linked Cassandra to Digg v4 reliability criticisms and recent company struggles. Lead engineers at Digg later rebuked these criticisms as red herring and blamed a lack of load testing.
- Facebook used Cassandra to power Inbox Search, with over 200 nodes deployed. This was abandoned in late 2010 when they built Facebook Messaging platform on HBase.
- IBM has done research in building a scalable email system based on Cassandra.
- InWorldz has researched and developed a scalable high-performance storage system for user inventory items Cassandra.
- Netflix uses Cassandra as their back-end database for their streaming services[12]
- Formspring uses Cassandra to count responses, as well as store Social Graph data (followers, following, blockers, blocking) for 26 Million accounts with 10 million responses a day

- Mahalo.com uses Cassandra to record user activity logs and topics for their Q&A website[13]
- Ooyala Built a scalable, flexible, real-time analytics engine using Cassandra[14]
- At Openwave, Cassandra acts as a distributed database and serves as a distributed storage mechanism for Openwave's next generation messaging platform[15]
- OpenX is running over 130 nodes on Cassandra for their OpenX Enterprise product to store and replicate advertisements and targeting data for ad delivery[16]
- Plaxo has "reviewed 3 billion contacts in [their] database, compared them with publicly available data sources, and identified approximately 600 million unique people with contact info."
- PostRank uses Cassandra as their backend database
- Rackspace is known to use Cassandra internally.
- Reddit switched to Cassandra from memcacheDB on March 12, 2010 and experienced some problems in May due to insufficient nodes in their cluster.
- RockYou uses Cassandra to record every single click for 50 million Monthly Active Users in real-time for their online games
- SoundCloud uses Cassandra to store the dashboard of their users
- Talentica Software uses Cassandra as a back-end for Analytics Application with Cassandra cluster of 30 nodes and inserting around 200GB data on daily basis.[17]
- Twitter announced it is planning to use Cassandra because it can be run on large server clusters and is capable of taking in very large amounts of data at a time. Twitter continues to use it but not for Tweets themselves.
- Urban Airship uses Cassandra with the mobile service hosting for over 160 million application installs across 80 million unique devices
- @WalmartLabs[18] (previously Kosmix) uses Cassandra with SSD
- Yakaz uses Cassandra on a five-node cluster to store millions of images as well as its social data.
- ZangBeZang uses Cassandra as the datastore for its carrier grade recommendation and marketing platform.
- Zoho uses Cassandra for generating the inbox preview in their Zoho#Zoho Mail service

Ironically, Facebook moved off its pre-Apache Cassandra deployment in late 2010 when they replaced Inbox Search with the Facebook Messaging platform. In 2012, Facebook began using Apache Cassandra in its Instagram unit.

Cassandra is the most popular wide column store.

References

[1] http://cassandra.apache.org/

[2] The Apache Software Foundation Announces Apache Cassandra Release 0.6 : The Apache Software Foundation Blog (https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces3)

[3] The Apache Software Foundation Announces Apache Cassandra 0.7 : The Apache Software Foundation Blog (https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces9)

[4] [Cassandra-user] [RELEASE] 0.8.0 - Grokbase (http://grokbase.com/t/cassandra/user/1162fkpwx2/release-0-8-0)

[5] Cassandra 1.0.0. Is Ready for the Enterprise (http://www.infoq.com/news/2011/10/Cassandra-1)

[6] The Apache Software Foundation Announces Apache Cassandra™ v1.1 : The Apache Software Foundation Blog (https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces26)

[7] The Apache Software Foundation Announces Apache Cassandra™ v1.2 (https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces38)

[8] [Cassandra-user] [RELEASE] Apache Cassandra 2.0.4 (http://qnalist.com/questions/4662083/release-apache-cassandra-2-0-4)

[9] "Deploying Cassandra across Multiple Data Centers" article on Datastax Cassandra Developer Center (http://www.datastax.com/dev/blog/deploying-cassandra-across-multiple-data-centers)

[10] "Hadoop Support" (http://wiki.apache.org/cassandra/HadoopSupport) article on Cassandra's wiki

[11] 4 Months with Cassandra, a love story | Cloudkick, manage servers better (https://www.cloudkick.com/blog/2010/mar/02/4_months_with_cassandra/)

[12] cite web|url=http://www.slideshare.net/adrianco/migrating-netflix-from-oracle-to-global-cassandra

[13] Watch Cassandra at Mahalo.com | DataStax Episodes | Blip (http://blip.tv/datastax/cassandra-at-mahalo-com-4030941)

[14] http://www.datastax.com/wp-content/uploads/2011/04/WP-Ooyala.pdf

[15] http://www.datastax.com/wp-content/uploads/2011/05/DataStax-CaseStudy-Openwave.pdf

[16] Ad Serving Technology - Advanced Optimization, Forecasting, & Targeting | OpenX (http://openx.com/publisher/technology)
[17] cite web|url=http://www.talentica.com
[18] Walmart Labs (http://www.walmartlabs.com)

Bibliography

- Hewitt, Eben (December 15, 2010). *Cassandra: The Definitive Guide* (http://oreilly.com/catalog/0636920010852) (1st ed.). O'Reilly Media. p. 300. ISBN 978-1-4493-9041-9.
- Capriolo, Edward (July 15, 2011). *Cassandra High Performance Cookbook* (http://www.packtpub.com/cassandra-apache-high-performance-cookbook/book) (1st ed.). Packt Publishing. p. 324. ISBN 1-84951-512-3.

External links

- Avinash Lakshman (25 August 2008). "Cassandra - A structured storage system on a P2P Network" (http://www.facebook.com/note.php?note_id=24413138919&id=9445547199&index=9). Engineering @ Facebook's Notes. Retrieved 2009-06-04.
- Project Website (http://cassandra.apache.org/)
- Project Wiki (http://wiki.apache.org/cassandra/)
- Adopting Apache Cassandra (http://www.infoq.com/presentations/Adopting-Apache-Cassandra) presented by Eben Hewitt on December 1, 2010
- LADIS 2009 WhitePaper by the original contributors Avinash Lakshman & Prashant Malik (http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf)
- Presentation on RDBMS vs. Dynamo, BigTable, and Cassandra (http://www.slideshare.net/jbellis/what-every-developer-should-know-about-database-scalability)
- RPM build for the apache cassandra project (http://code.google.com/p/cassandra-rpm/)
- Cassandra by example - the path of read and write requests (http://de.slideshare.net/grro/cassandra-by-example-the-path-of-read-and-write-requests)
- A vendor-independent comparison of NoSQL databases: Cassandra, HBase, MongoDB, Riak (http://www.networkworld.com/news/tech/2012/102212-nosql-263595.html) (NetworkWorld)

Triplestore

A **triplestore** is a purpose-built database for the storage and retrieval of triples,[1] a triple being a data entity composed of subject-predicate-object, like "Bob is 35" or "Bob knows Fred".

Much like a relational database, one stores information in a triplestore and retrieves it via a query language. Unlike a relational database, a triplestore is optimized for the storage and retrieval of triples. In addition to queries, triples can usually be imported/exported using Resource Description Framework (RDF) and other formats.

Some triplestores can store billions of triples.

Implementation

Some triplestores have been built as database engines from scratch, while others have been built on top of existing commercial relational database engines (i.e. SQL-based).[2] Like the early development of online analytical processing (OLAP) databases, this intermediate approach allowed large and powerful database engines to be constructed for little programming effort in the initial phases of triplestore development. Long-term though it seems likely that native triplestores will have the advantage for performance. A difficulty with implementing triplestores over SQL is that although *triples* may thus be *stored*, implementing efficient querying of a graph-based RDF model (i.e. mapping from SPARQL) onto SQL queries is difficult.

List of implementations

Name	Developed in language	Homepage	Licence
3store	C	www.aktors.org/technologies/3store [3], sourceforge.net/projects/threestore/ [4]	GPL
4store	C	www.4store.org [5]	GPL v3
5store	C	4store.org/trac/wiki/5store [6]	
Algebraix Data	C++	algebraixdata.com [7]	
AllegroGraph	Common Lisp	www.franz.com/agraph/allegrograph [8]	
ARC2	PHP	github.com/semsol/arc2/wiki [5]	W3C Software License or GPL
Bigdata	Java	www.bigdata.com [9]	GPL v2 ***or*** Commercial ***or*** Research
BigOWLIM	Java	www.ontotext.com/owlim [28]	
BrightstarDB	C#	brightstardb.com [10]	
ClioPatria	SWI-Prolog, C	cliopatria.swi-prolog.org [11]	GPL v2
Dydra	Common Lisp, C++	dydra.com [10]	Commercial
IBM DB2	Java, SQL	pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.swg.im.dbclient.rdf.doc/doc/c0059661.html [12]	
Apache Jena	Java	jena.apache.org [13]	Apache 2

KiWi (Apache Marmotta)	Java	marmotta.apache.org/kiwi [14]	Apache 2
MarkLogic	C++	www.marklogic.com [14]	Commercial
Mulgara	Java	www.mulgara.org [15]	OSL, moving to Apache 2
OntoBroker	Java	www.semafora-systems.com/en/products/ontobroker/ [16]	
OntoQuad RDF Server	C++	www.ontos.com/products/ontoquad/ [17]	
OpenAnzo	Java	www.openanzo.org [18]	EPL
OpenLink Virtuoso	C	virtuoso.openlinksw.com [19]	GPL v2 ***or*** Commercial
Oracle	Java, PL/SQL, SQL	www.oracle.com/technetwork/database-options/spatialandgraph/overview/rdfsemantic-graph-1902016.html [17]	
OWLIM	Java	www.ontotext.com/owlim [28]	
SparkleDB	C++	sparkledb.net [20]	
Parliament	Java, C++	parliament.semwebcentral.org [21]	BSD license
Pointrel System	Java, Python	sourceforge.net/projects/pointrel [22]	LGPLv3[23]
Profium Sense	Java	www.profium.com/technologies/profium-sense [24]	
RAP	PHP	www4.wiwiss.fu-berlin.de/bizer/rdfapi [25]	
RDF::Core	Perl	metacpan.org/module/RDF::Core [26]	
RDF::Trine	Perl	www.perlrdf.org [27]	Same as Perl
RDF-3X	C++	www.mpi-inf.mpg.de/~neumann/rdf3x [28]	CC-BY-NC-SA 3.0
RDFBroker	Java	rdfbroker.opendfki.de [29]	
RDFLib	Python	github.com/RDFLib/rdflib [30]	BSD
Redland	C	librdf.org [31]	Apache or LGPL or GPL[32]
RedStore	C	www.aelius.com/njh/redstore [33]	
Saffron Memory Base	Java	www.saffrontech.com [34]	
Semantics Platform	C#	www.intellidimension.com [35]	
SemWeb-DotNet	C#	razor.occams.info/code/semweb [36]	
Sesame	Java	www.openrdf.org [37]	BSD-style license
Soprano	C++	soprano.sourceforge.net [38]	LGPLv2 [39]
Stardog	Java	stardog.com [28]	

StrixDB	C++, Lua	www.strixdb.com [40]	
YARS	Java	sw.deri.org/2004/06/yars [41]	
Smart-M3	Python, Java, C, C#	sourceforge.net/projects/smart-m3 [42]	BSD License [43]

Technical overview

The following table is an overview of available triplestores, their technical implementation, support for the SPARQL World Wide Web Consortium (W3C) recommendations, and available application programming interfaces (API).

Solution Name	Native storage	Native SPARQL support	Native SPARQL/Update support	Native SPARQL Protocol Endpoint	Native APIs
4store	Triplestore	√	√	√	Command line only
AllegroGraph	Graph	√	√	√	For most modern programming languages
ARC2	3rd party	√	√	√	PHP
ARQ	3rd party	√	√		Java
BigData	Triplestore	√	√	√	Java
BrightstarDB	Graph data model in Heap file	√			.NET Framework or Web Service
Corese	3rd party	√			Java
D2R Server	3rd party	√	√	√	Java
Dydra	Graph database in the cloud SaaS	√	√	√	REST API
Hercules	Stored in web browser	√			JavaScript
IBM DB2	Object-relational				Java
Intellidimension Semantics Platform 2.0	3rd party	√			.NET Framework
Jena	Tuple store	√	√	√	Java
KAON2	3rd party	√			Java
SparkleDB	Triplestore / Quadstore	√	√	√	For most modern programming languages
MarkLogic	Triplestore / Quadstore	√		√	REST API, SPARQL Endpoint, Graph Protocol Endpoint, Java API, XQuery, SQL/ODBC
Mulgara	3rd party	√			Java or REST API
OntoBroker	Triplestore	√	√	√	Java
OntoQuad RDF Server	Triplestore / Quadstore	√	√	√	Java, SPARQL Endpoint or REST API
Ontotext OWLIM	3rd party	√	√	√	Java
Open Anzo	3rd party	√		√	Java, JavaScript, .NET Framework

OpenLink Virtuoso	Hybrid (Relational Tables and Relational Property Graphs)	√	√	√	ODBC, JDBC, ADO.NET, OLE DB, XMLA, HTTP, etc., serving most modern programming languages including C, PHP, Perl, Python, Ruby, Java, JavaScript, .NET Framework, etc.
Oracle DB Enterprise Ed.	Object-relational				For most modern programming languages
Parliament	3rd party	√	√	√	Java or C++
Pellet	3rd party	√			Java
Pointrel	Triplestore				Python
Profium Sense	In-memory triplestore	√		√	Java
RAP	In-memory triplestore or heap file	√			PHP
RDF API for PHP	3rd party	√			PHP
RDF::Query	3rd party	√	√	√	Perl
RDF-3X	Triplestore	√			Command line only
RDFBroker	3rd party				Java
Redland, Redstore	3rd party	√	√	√	C
SemWeb.NET	3rd party	√		√	.NET Framework
Sesame	3rd party	√	√	√	Java
Soprano	3rd party				C++
SPARQL Engine	3rd party	√			Java
Stardog	Triplestore	√	√	√	Java, Groovy
StrixDB	Triplestore	√	√	√	Lua
Twinql	3rd party	√			Lisp
YARS	3rd party				HTTP, JDBC

References

[1] TripleStore (http://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/rusher.html), Jack Rusher, Semantic Web Advanced Development for Europe (SWAD-Europe), Workshop on Semantic Web Storage and Retrieval - Position Papers

[2] Storage and Management of Semi-structured Data (Use of SQL relational databases as an RDF triple store), 2003

[3] http://www.aktors.org/technologies/3store

[4] http://sourceforge.net/projects/threestore/

[5] http://www.4store.org

[6] http://4store.org/trac/wiki/5store

[7] http://algebraixdata.com

[8] http://www.franz.com/agraph/allegrograph

[9] http://www.bigdata.com

[10] http://brightstardb.com

[11] http://cliopatria.swi-prolog.org

[12] http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.swg.im.dbclient.rdf.doc/doc/c0059661.html

[13] http://jena.apache.org/

[14] http://marmotta.apache.org/kiwi

[15] http://www.mulgara.org

[16] http://www.semafora-systems.com/en/products/ontobroker/

[17] http://www.ontos.com/products/ontoquad/

[18] http://www.openanzo.org

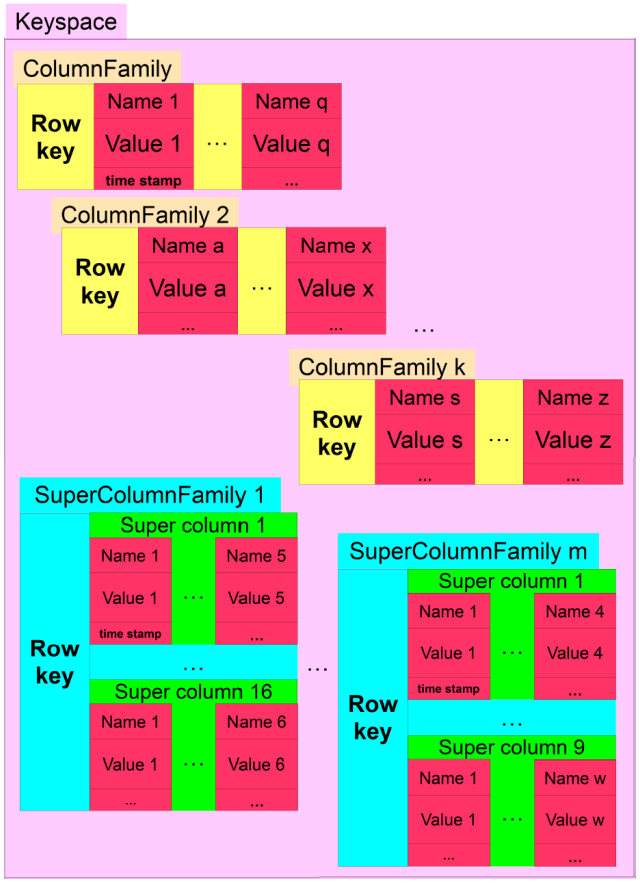
[19] http://virtuoso.openlinksw.com
[20] http://www.sparkledb.net
[21] http://parliament.semwebcentral.org
[22] http://sourceforge.net/projects/pointrel
[23] sourceforge.net/projects/smart-m3
[24] http://www.profium.com/technologies/profium-sense
[25] http://www4.wiwiss.fu-berlin.de/bizer/rdfapi
[26] http://metacpan.org/module/RDF::Core
[27] http://www.perlrdf.org
[28] http://www.mpi-inf.mpg.de/~neumann/rdf3x
[29] http://rdfbroker.opendfki.de
[30] http://github.com/RDFLib/rdflib
[31] http://librdf.org
[32] http://librdf.org/LICENSE.html
[33] http://www.aelius.com/njh/redstore
[34] http://www.saffrontech.com
[35] http://www.intellidimension.com
[36] http://razor.occams.info/code/semweb
[37] http://www.openrdf.org
[38] http://soprano.sourceforge.net
[39] http://sourceforge.net/projects/soprano/
[40] http://www.strixdb.com
[41] http://sw.deri.org/2004/06/yars
[42] http://sourceforge.net/projects/smart-m3
[43] sourceforge.net/projects/smart-m3

External links

- A list of large triplestores (http://esw.w3.org/topic/LargeTripleStores)
- Lehigh University Benchmark (LUBM) (http://swat.cse.lehigh.edu/projects/lubm/)
- How RDF Databases Differ from Other NoSQL Solutions (http://blog.datagraph.org/2010/04/rdf-nosql-diff)
- W3C SPARQL Working Group (http://www.w3.org/2001/sw/DataAccess/), was RDF Data Access Working Group
- SPARQL Query language (http://www.w3.org/TR/rdf-sparql-query/)
- SPARQL Protocol (http://www.w3.org/TR/rdf-sparql-protocol/)
- SPARQL 1.1 Update (http://www.w3.org/TR/sparql11-update/) W3C Recommendation 21 March 2013

Keyspace (distributed data store)

A **key space** (or **keyspace**) in a NoSQL data store is an object that holds together all column families of a design. It is the outer most grouping of the data in the data store. It resembles to the schema concept in Relational database management systems. Generally, there is one keyspace per application.



A keyspace example with a number of column families.

Structure

A keyspace may contain column families or super columns. Each super column contains one or more column family, each column family at least one column. The keyspace is the highest abstraction in a distributed data store.

Comparison with relational database systems

The keyspace has similar importance like a schema has in a database. In contrast to the schema, however, it does not stipulate any concrete structure, like it is known in the entity-relationship model used widely in the relational data models. For instance, the contents of the keyspace can be column families, each having different number of columns, or even different columns. So, the column families that somehow relate to the row concept in relational databases do not stipulate any fixed structure. The only point that is the same with a schema is that it also contains a number of "objects", which are tables in RDBMS systems and here column families or super columns.

So, in distributed data stores, the whole burden to handle rows that may even change from data-store update to update lies on the shoulders of the programmers.

Examples

As an example, we show a number of column families in a keyspace. The `CompareWith` keyword defines how the column comparison is made. In the example, the UTF8 standard has been selected. Other ways of comparison exist, such as `AsciiType`, `BytesType`, `LongType`, `TimeUUIDType`.

```
<Keyspace Name="DeliciousClone">
  <KeysCachedFraction>0.01</KeysCachedFraction>
  <ColumnFamily CompareWith="UTF8Type" Name="Users"/>
  <ColumnFamily CompareWith="UTF8Type" Name="Bookmarks"/>
  <ColumnFamily CompareWith="UTF8Type" Name="Tags"/>
  <ColumnFamily CompareWith="UTF8Type" Name="UserTags"/>
  <ColumnFamily CompareWith="UTF8Type" CompareSubcolumnsWith="TimeUUIDType" ColumnType="Super" Name="UserBookmarks"/>
</Keyspace>
```

Another example shows a simplified Twitter clone data model:

```
<Keyspace Name="TwitterClone">
  <KeysCachedFraction>0.01</KeysCachedFraction>
  <ColumnFamily CompareWith="UTF8Type" Name="Users" />
  <ColumnFamily CompareWith="UTF8Type" Name="UserAudits" />
  <ColumnFamily CompareWith="UTF8Type" CompareSubcolumnsWith="TimeUUIDType" ColumnType="Super" Name="UserRelationships" />
  <ColumnFamily CompareWith="UTF8Type" Name="Usernames" />
  <ColumnFamily CompareWith="UTF8Type" Name="Statuses" />
  <ColumnFamily CompareWith="UTF8Type" Name="StatusAudits" />
  <ColumnFamily CompareWith="UTF8Type" CompareSubcolumnsWith="TimeUUIDType" ColumnType="Super" Name="StatusRelationships" />
</Keyspace>
```

References

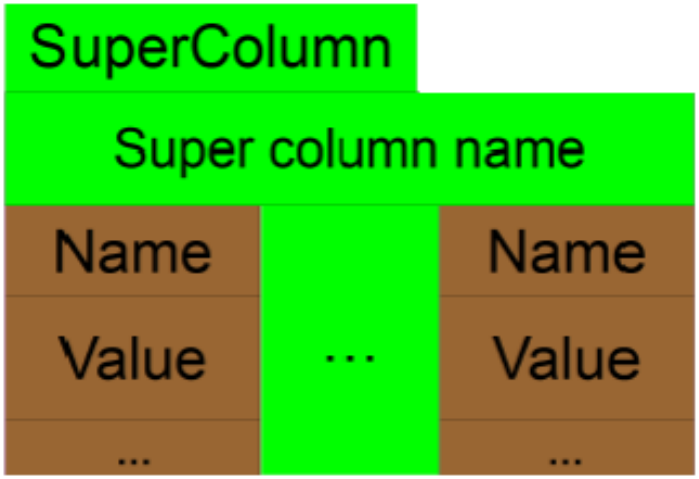
External links

- Cassandra – Getting Started (Java) (http://schabby.de/cassandra-getting-started/)

Super column

A **super column** is a tuple (a pair) with a binary super column name and a value that maps it to many columns. They consist of a key-value pairs, where the values are columns. Theoretically speaking, super columns are (sorted) associative array of columns. Similar to a regular column family where a row is a sorted map of column names and column values, a row in a super column family is a sorted map of super column names that maps to column names and column values.

A super column is part of a keyspace (data model) together with other super columns and column families, and columns.



The super column consists of a (unique) super column name, and a number of columns.

Code example

Written in the JSON-like syntax, a super column definition can be like this:

```
 {
   "mccv": {
     "Tags": {
       "cassandra": {
         "incubator": {"url":
"http://incubator.apache.org/cassandra/"},
```

```
          "jira": {"url":
"http://issues.apache.org/jira/browse/CASSANDRA"}
        },
        "thrift": {
          "jira": {"url": "http://issues.apache.org/jira/browse/THRIFT"}
        }
      }
    }
 }
```

References

External links

- The Apache Cassandra data model (http://wiki.apache.org/cassandra/DataModel)

BigTable

BigTable is a compressed, high performance, and proprietary data storage system built on Google File System, Chubby Lock Service, SSTable (log-structured storage like LevelDB) and a few other Google technologies. It is not distributed outside Google, although Google offers access to it as part of its Google App Engine.

History

BigTable development began in 2004 and is now used by a number of Google applications, such as web indexing, MapReduce, which is often used for generating and modifying data stored in BigTable,[1] Google Maps, Google Book Search, "My Search History", Google Earth, Blogger.com, Google Code hosting, Orkut, YouTube, and Gmail. Google's reasons for developing its own database include scalability and better control of performance characteristics.[2]

Google's Spanner RDBMS is layered on an implementation of BigTable with a Paxos group for two-phase commits to each tablet. Google F1 was built using Spanner to replace an implementation based on MySQL.

Design

BigTable maps two arbitrary string values (row key and column key) and timestamp (hence three-dimensional mapping) into an associated arbitrary byte array. It is not a relational database and can be better defined as a sparse, distributed multi-dimensional sorted map.[3]Wikipedia:Citing sources BigTable is designed to scale into the petabyte range across "hundreds or thousands of machines, and to make it easy to add more machines [to] the system and automatically start taking advantage of those resources without any reconfiguration".

Each table has multiple dimensions (one of which is a field for time, allowing for versioning and garbage collection). Tables are optimized for Google File System (GFS) by being split into multiple *tablets* – segments of the table are split along a row chosen such that the tablet will be ~200 megabytes in size. When sizes threaten to grow beyond a specified limit, the tablets are compressed using the algorithm BMDiff and the Zippy compression algorithm publicly known and open-sourced as Snappy, which is a less space-optimal variation of LZ77 but more efficient in terms of computing time. The locations in the GFS of tablets are recorded as database entries in multiple special tablets, which are called "META1" tablets. META1 tablets are found by querying the single "META0" tablet, which typically resides on a server of its own since it is often queried by clients as to the location of the "META1" tablet

which itself has the answer to the question of where the actual data is located. Like GFS's master server, the META0 server is not generally a bottleneck since the processor time and bandwidth necessary to discover and transmit META1 locations is minimal and clients aggressively cache locations to minimize queries.

Other similar software

- Apache Accumulo — built on top of Hadoop, ZooKeeper, and Thrift. Has cell-level access labels and a server-side programming mechanism. Written in Java.
- Apache Cassandra — brings together Dynamo's fully distributed design and BigTable's data model. Written in Java.
- Apache HBase — Provides BigTable-like support on the Hadoop Core. Written in Java.
- Hypertable — Hypertable is designed to manage the storage and processing of information on a large cluster of commodity servers. Written in C++.
- "KDI" [4], *Bluefish*, GitHub — Kosmix attempt to make a BigTable clone. Written in C++.
- LevelDB — Google's embedded key/value store that uses similar design concepts as the BigTable Tablet.

References

[1] Chang et al. 2006, p. 3: 'Bigtable can be used with MapReduce, a framework for running large-scale parallel computations developed at Google. We have written a set of wrappers that allow a Bigtable to be used both as an input source and as an output target for MapReduce jobs'

[2] Chang et al. 2006, Conclusion: 'We have described Bigtable, a distributed system for storing structured data at Google... Our users like the performance and high availability provided by the Bigtable implementation, and that they can scale the capacity of their clusters by simply adding more machines to the system as their resource demands change over time... Finally, we have found that there are significant advantages to building our own storage solution at Google. We have gotten a substantial amount of flexibility from designing our own data model for Bigtable.'

[3] Chang et al. 2006.

[4] http://github.com/bluefish/kdi

Bibliography

- Chang, Fay; Dean, Jeffrey; Ghemawat, Sanjay; Hsieh, Wilson C; Wallach, Deborah A; Burrows, Michael 'Mike'; Chandra, Tushar; Fikes, Andrew; Gruber, Robert E (2006), "Bigtable: A Distributed Storage System for Structured Data" (http://research.google.com/archive/bigtable-osdi06.pdf) (PDF), *Research*, Google.

External links

- *BigTable: A Distributed Structured Storage System* (http://www.cs.washington.edu/htbin-post/mvis/mvis?ID=437), Washington. *Video* (http://video.google.com/videoplay?docid=7278544055668715642), Google.
  - *UWTV* (http://www.uwtv.org/programs/displayevent.asp?rid=2787) (video).
  - Witchcock, Andrew, *Google's BigTable* (http://andrewhitchcock.org/?post=214) (notes on the official presentation).
- Carr, David F (2006-07-06), "How Google Works" (http://www.baselinemag.com/article2/0,1540,1985050,00.asp), *Baseline*.
- "Is the Relational Database Doomed?" (http://readwrite.com/2009/02/12/is-the-relational-database-doomed), *Read-write web*.

Flat file database

A **flat file database** describes any of various means to encode a database model (most commonly a table) as a single file.

A flat file can be a plain text file or a binary file. There are usually no structural relationships between the records.

Flat File Model

	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

Example of a flat file model[1]

Overview

Plain text files usually contain one record per line, There are different conventions for depicting data. In comma-separated values and delimiter-separated values files, fields can be separated by delimiters such as comma or tab characters. In other cases, each field may have a fixed length; short values may be padded with space characters. Extra formatting may be needed to avoid delimiter collision. More complex solutions are markup languages and programming languages.

Using delimiters incurs some overhead in locating them every time they are processed (unlike fixed-width formatting), which may have performance implications. However, use of character delimiters (especially commas) is also a crude form of data compression which may assist overall performance by reducing data volumes — especially for data transmission purposes. Use of character delimiters which include a length component (Declarative notation) is comparatively rare but vastly reduces the overhead associated with locating the extent of each field.

Typical examples of flat files are `/etc/passwd` and `/etc/group` on Unix-like operating systems. Another example of a flat file is a name-and-address list with the fields *Name*, *Address*, and *Phone Number*.

A list of names, addresses, and phone numbers written by hand on a sheet of paper is a flat file database. This can also be done with any typewriter or word processor. A spreadsheet or text editor program may be used to implement a flat file database, which may then be printed or used online for improved search capabilities.

History

The first uses of computing machines were implementations of simple databases. Herman Hollerith conceived the idea that census data could be represented by holes punched in paper cards and tabulated by machine. He sold his concept to the US Census Bureau; thus, the 1890 United States Census was the first computerized database—consisting, in essence, of thousands of boxes full of punched cards.

Hollerith's enterprise grew into computer giant IBM, which dominated the data processing market for most of the 20th century. IBM's fixed-length field, 80-column punch cards became the ubiquitous means of inputting electronic data until the 1970s.

In the 1980s, configurable flat-file database computer applications were popular on DOS and the Macintosh. These programs were designed to make it easy for individuals to design and use their own databases, and were almost on par with word processors and spreadsheets in popularity.[*citation needed*] Examples of flat-file database products were early versions of FileMaker and the shareware PC-File. Some of these, like dBase II, offered limited relational capabilities, allowing some data to be shared between files.

Contemporary implementations

FairCom's c-tree is an example of a modern enterprise-level solution, and spreadsheet software and text editors can be used for this purpose. WebDNA is a scripting language designed for the World Wide Web, with a hybrid flat file in-memory database system making it easy to build resilient database-driven websites. With the in-memory concept, WebDNA searches and database updates are almost realtime while the data is stored as text files within the website itself. Otherwise, flat file database is implemented in Microsoft Works and Apple Works. Over time, products like Borland's Paradox, and Microsoft's Access started offering some relational capabilities, as well as built-in programming languages. Database Management Systems (DBMS) like MySQL or Oracle generally require programmers to build applications.

Faceless flat file database engines are used internally by Mac OS X, Firefox, and other computer software to store configuration data. Programs to manage collections of books or appointments and address book are essentially single-purpose flat file database applications, allowing users to store and retrieve information from flat files using a predefined set of fields. As of 2011[2], one of the most popular flat file database engines is SQLite, which is the engine used by Firefox and Android and is part of the PHP5 standard distribution.

Data transfer operations

Flat files are used not only as data storage tools in DB and CMS systems, but also as data transfer tools to remote servers (in which case they become known as information streams).

In recent years, this latter implementation has been replaced with XML files, which not only contain but also describe the data. Those still using flat files to transfer information are mainframes employing specific procedures which are too expensive to modify.

One criticism often raised against the XML format as a way to perform mass data transfer operations is that file size is significantly larger than that of flat files, which is generally reduced to the bare minimum. The solution to this problem consists in XML file compression (a solution that applies equally well to flat files), which has nowadays gained EXI standards (i.e., Efficient XML Interchange, which is often used by mobile devices).

It is advisable that transfer data be performed via EXI rather than flat files because defining the compression method is not required, because libraries reading the file contents are readily available, and because there is no need for the two communicating systems to preliminarily establish a protocol describing data properties such as position, alignment, type, and format. However, in those circumstances where the sheer mass of data and/or the inadequacy of legacy systems becomes a problem, the only viable solution remains the use of flat files. In order to successfully handle those problems connected with data communication, format, validation, control and much else (be it a flat file or an XML file data source), it is advisable to adopt a Data Quality Firewall.

Terminology

"Flat file database" may be defined very narrowly, or more broadly. The narrower interpretation is correct in database theory; the broader covers the term as generally used.

Strictly, a flat file database should consist of nothing but data and, if records vary in length, delimiters. More broadly, the term refers to any database which exists in a single file in the form of rows and columns, with no relationships or links between records and fields except the table structure.

Terms used to describe different aspects of a database and its tools differ from one implementation to the next, but the concepts remain the same. FileMaker uses the term "Find", while MySQL uses the term "Query"; but the concept is the same. FileMaker "files", in version 7 and above, are equivalent to MySQL "databases", and so forth. To avoid confusing the reader, one consistent set of terms is used throughout this article.

However, the basic terms "record" and "field" are used in nearly every flat file database implementation.

Example database

The following example illustrates the basic elements of a flat-file database. The data arrangement consists of a series of columns and rows organized into a tabular format. This specific example uses only one table.

The columns include: *name* (a person's name, second column); *team* (the name of an athletic team supported by the person, third column); and a numeric *unique ID*, (used to uniquely identify records, first column).

Here is an example textual representation of the described data:

```
id    name    team
1     Amy     Blues
2     Bob     Reds
3     Chuck   Blues
4     Dick    Blues
5     Ethel   Reds
6     Fred    Blues
7     Gilly   Blues
8     Hank    Reds
```

This type of data representation is quite standard for a flat-file database, although there are some additional considerations that are not readily apparent from the text:

- **Data types:** each column in a database table such as the one above is ordinarily restricted to a specific data type. Such restrictions are usually established by convention, but not formally indicated unless the data is transferred to a relational database system.
- **Separated columns:** In the above example, individual columns are separated using whitespace characters. This is also called indentation or "fixed-width" data formatting. Another common convention is to separate columns using one or more delimiter characters. More complex solutions are markup and programming languages.
- **Relational algebra:** Each row or record in the above table meets the standard definition of a tuple under relational algebra (the above example depicts a series of 3-tuples). Additionally, the first row specifies the field names that are associated with the values of each row.
- **Database management system:** Since the formal operations possible with a text file are usually more limited than desired, the text in the above example would ordinarily represent an intermediary state of the data prior to being transferred into a database management system.

References

[1] Data Integration Glossary (http://knowledge.fhwa.dot.gov/tam/aashto.nsf/All+Documents/4825476B2B5C687285256B1F00544258/$FILE/DIGloss.pdf), U.S. Department of Transportation, August 2001.

[2] http://en.wikipedia.org/w/index.php?title=Flat_file_database&action=edit

Terminfo

Terminfo is a library and database that enables programs to use display terminals in a device-independent manner. This library has its origins in the UNIX System III operating system. Mark Horton implemented the first terminfo library in 1981-1982 as an improvement over termcap. The improvements include

- faster access to stored terminal descriptions,
- longer, more understandable names for terminal capabilities and
- general expression evaluation for strings sent to the terminal.

Terminfo soon became the preferred form of terminal descriptions in UNIX, rather than termcap. This was imitated in **pcurses** in 1982-1984 by Pavel Curtis, and was available on other UNIX implementations, adapting or incorporating fixes from Mark Horton. For more information, refer to the posting on the `comp.sources.unix` newsgroup from December 1986.

A terminfo database can describe the capabilities of hundreds of different display terminals. This allows external programs to be able to have character-based display output, independent of the type of terminal.

Some configurations are:

- Number of lines on the screen
- Mono mode; suppress color
- Use visible bell instead of beep

Data model

Terminfo databases consist of one or more descriptions of terminals.

Indices

Each description must contain the canonical name of the terminal. It may also contain one or more aliases for the name of the terminal. The canonical name or aliases are the keys by which the library searches the terminfo database.

Data values

The description contains one or more capabilities, which have conventional names. The capabilities are typed: *boolean*, *numeric* and *string*. The terminfo library has predetermined types for each capability name. It checks the types of each capability by the syntax:

- *string* capabilities have an "=" between the capability name and its value,
- *numeric* capabilities have a "#" between the capability name and its value, and
- *boolean* capabilities have no associated value (they are always *true* if specified).

Applications which use terminfo know the types for the respective capabilities, and obtain the values of capabilities from the terminfo database using library calls that return successfully only when the capability name corresponds to one of the predefined typed capabilities.

Like termcap, some of the *string* capabilities represent escape sequences which may be sent to the host by pressing special keys on the keyboard. Other capabilities represent strings that may be sent by an application to the terminal. In the latter case, the terminfo library functions (as does a termcap library) for substituting application parameters into the string which is sent. These functions provide a stack-based expression parser, which is primarily used to help minimize the number of characters sent for control sequences which have optional parameters such as SGR (Select Graphic Rendition). In contrast, termcap libraries provide a limited set of operations which are useful for most terminals.

Hierarchy

Terminfo descriptions can be constructed by including the contents of one description in another, suppressing capabilities from the included description or overriding or adding capabilities. No matter what storage model is used, the terminfo library returns the terminal description from the requested description, using data which is compiled using a standalone tool (e.g., **tic**).

Storage model

Terminfo data is stored as a binary file, making it less simple to modify than termcap. The data can be retrieved by the terminfo library from the files where it is stored. The data itself is organized as tables for the boolean, numeric and string capabilities, respectively. This is the scheme devised by Mark Horton, and except for some differences regarding the available names is used in most terminfo implementations. X/Open does not specify the format of the compiled terminal description. In fact, it does not even mention the common **tic** or **infocmp** utilities. Because the compiled terminfo entries do not contain metadata identifying the indices within the tables to which each capability is assigned, they are not necessarily compatible between implementations. However, since most implementations use the same overall table structure (including sizes of header and data items), it is possible to automatically construct customized terminfo libraries which can read data for a given implementation. For example, ncurses can be built to match the terminfo data for several other implementations.

Directory tree

The original (and most common) implementation of the terminfo library retrieves data from a directory hierarchy. By using the first character of the name of the terminal description as one component of the pathname, and the name of the terminal description as the name of the file to retrieve, the terminfo library usually outperforms searching a large termcap file.

Hashed database

Some implementations of terminfo store the terminal description in a hashed database (e.g., something like Berkeley DB version 1.85). These store two types of records: aliases which point to the canonical entry, and the canonical entry itself, which contains the data for the terminal capabilities.

Limitations and extensions

The Open Group documents the limits for terminfo (minimum guaranteed values), which apply only to the source file.[1] Two of these are of special interest:

- 14 character maximum for terminal aliases
- 32,767 maximum for numeric quantities

The 14-character limit addresses very old filesystems which could represent filenames no longer than that. While those filesystems are generally obsolete, these limits were as documented from the late 1980s, and unreviewed since then.

The 32,767 limit is for positive values in a signed two's complement 16-bit value. A terminfo entry may use negative numbers to represent cancelled or absent values.

Unlike termcap, terminfo has both a source and compiled representation. The limits for the compiled representation are unspecified. However, most implementations note in their documentation for **tic** (terminal information compiler) that compiled entries cannot exceed 4,096 bytes in size.

References

[1] Most of this was done before X/Open merged with Open Software Foundation to form The Open Group, consequently there are many sources that say *X/Open*.

External links

- Current terminfo data (http://invisible-island.net/ncurses/ncurses.faq.html#which_terminfo)
- Termcap/Terminfo Resources Page (http://www.catb.org/~esr/terminfo/) at Eric S. Raymond's website
- man terminfo(5) (http://invisible-island.net/ncurses/man/terminfo.5.html)

Termcap

Termcap (***term****inal* ***cap****ability*) is a software library and database used on Unix-like computers. It enables programs to use display computer terminals in a device-independent manner, which greatly simplifies the process of writing portable text mode applications. Bill Joy wrote the first termcap library in 1978[1][2] for the Berkeley Unix operating system; it has since been ported to most Unix and Unix-like environments. Joy's design was reportedly influenced by the design of the terminal data store in the earlier Incompatible Timesharing System.[3]

A termcap database can describe the capabilities of hundreds of different display terminals. This allows programs to have character-based display output, independent of the type of terminal. On-screen text editors such as vi and emacs are examples of programs that may use termcap. Other programs are listed in the Termcap category.

Examples of what the database describes:

- how many columns wide the display is
- what string to send to move the cursor to an arbitrary position (including how to encode the row and column numbers)
- how to scroll the screen up one or several lines
- how much padding is needed for such a scrolling operation.

Data model

Termcap databases consist of one or more descriptions of terminals.

Indices

Each description must contain the canonical name of the terminal. It may also contain one or more aliases for the name of the terminal. The canonical name or aliases are the keys by which the library searches the termcap database.

Data values

The description contains one or more capabilities, which have conventional names. The capabilities are typed: *boolean*, *numeric* and *string*. The termcap library has no predetermined type for each capability name. It determines the types of each capability by the syntax:

- *string* capabilities have an "=" between the capability name and its value,
- *numeric* capabilities have a "#" between the capability name and its value, and
- *boolean* capabilities have no associated value (they are always *true* if specified).

Applications which use termcap do expect specific types for the commonly used capabilities, and obtain the values of capabilities from the termcap database using library calls that return successfully only when the database contents matches the assumed type.

Hierarchy

Termcap descriptions can be constructed by including the contents of one description in another, suppressing capabilities from the included description or overriding or adding capabilities. No matter what storage model is used, the termcap library constructs the terminal description from the requested description, including, suppressing or overriding at the time of the request.

Storage model

Termcap data is stored as text, making it simple to modify. The text can be retrieved by the termcap library from files or environment variables.

Environment variable

The **TERMCAP** environment variable may contain a termcap database. It is most often used to store a single termcap description, set by a terminal emulator to provide the terminal's characteristics to the shell and dependent programs.

Flat file

The original (and most common) implementation of the termcap library retrieves data from a flat text file. Searching a large termcap file, e.g., 500Kb, can be slow. To aid performance, a utility such as **reorder** is used to put the most frequently used entries near the beginning of the file.

Hashed database

Newer implementations of termcap store the terminal description in a hashed database (e.g., something like Berkeley DB version 1.85). These store two types of records: aliases which point to the canonical entry, and the canonical entry itself. The text of the termcap entry is stored literally.

Limitations and extensions

The original termcap implementation was designed to use little memory:

- the first name is two characters, to fit in 16 bits
- capability names are two characters
- descriptions are limited to 1023 characters.
- only one description can be included, and must be at the end.

Newer implementations of the termcap interface generally do not require the two-character name at the beginning of the entry.

Capability names are still two characters in all implementations.

The **tgetent** function used to read the terminal description uses a buffer whose size must be large enough for the data, and is assumed to be 1024 characters. Newer implementations of the termcap interface may relax this constraint by allowing a null pointer in place of the fixed buffer,[4] or by hiding the data which would not fit, e.g., via the **ZZ** capability in NetBSD termcap.[5] The terminfo library interface also emulates the termcap interface, and does not actually use the fixed-size buffer.

The terminfo library's emulation of termcap allows multiple descriptions to be included without restricting the position. A few other newer implementations of the termcap library may also provide this ability, though it is not well documented.[6]

References

[1] Peter H. Salus, "The history of Unix is as much about collaboration as it is about technology", Byte, October 1994. (http://landley.net/history/mirror/unix/art3.htm)
[2] Kenneth C. R. C. Arnold and Elan Amir, "Screen Updating and Cursor Movement Optimization: A Library Package" (http://www.mirbsd.org/cman/manPSD/19.curses.htm)
[3] alt.sys.pdp10 posting (http://www.inwap.com/pdp10/usenet/its)
[4] The GNU Termcap Library (http://www.gnu.org/software/termutils/manual/termcap-1.3/termcap.html)
[5] NetBSD termcap file format (http://www.daemon-systems.org/man/termcap.5.html)
[6] Discussion of termcap in vi (http://freshmeat.net/projects/vi/)

External links

- Current termcap data (http://invisible-island.net/ncurses/ncurses.faq.html#which_terminfo)
- Termcap/Terminfo Resources Page (http://www.catb.org/~esr/terminfo/) at Eric S. Raymond's website

MultiValue

MultiValue is a type of NoSQL and multidimensional database, typically considered synonymous with PICK, a database originally developed as the Pick operating system.

MultiValue databases include commercial products from Rocket Software, TigerLogic, jBASE, Revelation, Ladybridge, InterSystems, Northgate Information Solutions and other companies. These databases differ from a relational database in that they have features that support and encourage the use of attributes which can take a list of values, rather than all attributes being single-valued. They are often categorized with MUMPS within the category of post-relational databases, although the data model actually pre-dates the relational model. Unlike SQL-DBMS tools, most MultiValue databases can be accessed both with or without SQL.

History

Don Nelson designed the MultiValue data model in the early to mid-1960s. Dick Pick, a developer at TRW, worked on the first implementation of this model for the US Army in 1965. Pick considered the software to be in the public domain because it was written for the military. This was but the first dispute regarding MultiValue databases that was addressed by the courts.[1]

Ken Simms wrote DataBASIC, sometimes known as S-BASIC, in the mid-70's. It was based on Dartmouth BASIC, but had enhanced features for data management. Simms played a lot of Star Trek while developing the language, in order to have the language function to his satisfaction.

Three of the implementations of MultiValue: PICK version R77, Microdata Reality [2] 3.x, and Prime Information 1.0, were very similar. In spite of attempts to standardize, particularly by International Spectrum and the Spectrum Manufacturers Association, who designed a logo for all to use, there are no standards across MultiValue implementations. Subsequently, these flavors diverged, although with some cross-over. These streams of MultiValue database development could be classified as one stemming from PICK R83, one from Microdata Reality [2], and one from Prime Information. Because of the differences, some implementations have provisions for supporting several flavors of the languages. An attempt to document the similarities and differences can be found at the PRDB

Marketing groups and others in the industry over the years have classified MultiValue databases as pre-relational, post-relational, relational, and embedded, with detractors classifying it as legacy. It could now be classified as NoSQL. With a data model that aligns well with XML and that permits access with or without the use of SQL.

One reasonable hypothesis for this data model lasting more than 40 years, with new database implementations of the model even in the 21st century is that it provides database solutions in a big bang for the buck fashion. Historically,

with industry benchmarks tied to SQL transactions, this has been a difficult hypothesis to test, although there are considerable anecdotes of failed attempts to get the functionality of a MultiValue application into a relational database framework.

In spite of a history of more than 40 years of implementations, starting with TRW, many in the MultiValue industry have remained current so that various MultiValue implementations now employ object-oriented versions of Data BASIC, support AJAX frameworks, and because no one needs to use SQL (but some can) they fit under the NoSQL umbrella. In fact, MultiValue developers were the first to acquire nosql domain names, likely prior to other database products classifying their offerings as NoSQL as well. MultiValue is a seasoned data model, but with so many vendors competing in this space, it has been constantly enhanced over the years.

Data model example

In a MultiValue database system:

- a Database is called an "Account"
- a Table is called a "File"
- a Column is an "Attribute" composed of "Multivalue attribute" and "Subvalue attribute" to store multiple values in the same attribute.

Data is stored using two separate files. A "File" to store raw data and a "Dictionary" to store the format for displaying the raw data.

For example, assume there's a file (table) called "PERSON". And in this file there is a attribute called "eMailAddress". The eMailAddress field can store a variable number of email address values in the single record.

So the list [joe@example.com, jdb@example.net, joe_bacde@example.org] can be stored and accessed via a single query / disk read when accessing the associated record.

To achieve the same (1-to-many) relationship within a traditional Relational Database System one would be required to create an additional table to store the variable number of email Addresses associated to a single "PERSON" record. However, modern Relational Database Systems do support this data model too. For example, in PostgreSQL one can set up a column of type Array of baseType (baseType being any PostgreSQL data type).

MultiValue DataBASIC

Like the Java programming language, the typical DataBASIC compiler compiles to P-code and runs in a P-machine. It has as many different implementations (compilers) as there are MultiValue databases.

Like PHP programming language, the DataBASIC language does all the typecasting for the programmer.

MultiValue Query Language

Known as ENGLISH, ACCESS, AQL, UniQuery, Retrieve, CMQL, and by many other names over the years, corresponding to the different MultiValue implementations, the MultiValue query language differs from SQL in several respects. Each query is issued against a single dictionary within the schema, which could be understood as a virtual file or a portal to the database through which to view the data.

```
LIST PEOPLE LAST_NAME FIRST_NAME EMAIL_ADDRESSES WITH LAST_NAME LIKE "Van..."
```

The above statement would list all e-mail addresses for each person whose last name starts with "Van". A single entry would be output for each person, with multiple lines showing the multiple e-mail addresses (without repeating other data about the person).

References

[1] http://www.microdata-alumni.org/historical.htm#history_of_pick
[2] http://www.northgate-is.com/reality

OpenInsight

OpenInsight is a database application development tool from Revelation Software. It was first released in 1992; version 9.2 was released in 2010.[1]

OpenInsight is a windows-based development tool. It contains its own database which is a type of post-relational database known as a MultiValue database. OpenInsight contains tools for creating database applications that can run on Windows workstations, Windows, Linux, and Novell networks, and browser-based applications. The tools include, but are not limited to, a Table Builder, a Database Manager, editor, debugger and programming language, a forms designer for creating data entry forms, User Interface tools, reporting tools, and deployment tools.[2]

OpenInsight's programming language is called Basic+, and is an extension or dialect of the BASIC programming language. OpenInsight applications can use data from Revelation Software's built-in database, or they can use data from SQL Databases, from Rocket Software's Rocket U2 databases,[3] as well as Tiger Logic's D3 database.

Related software

- Rocket U2
- Reality

References

[1] Revelation Website: http://www.revelation.com/Revelation.nsf/byTitle/E7DB7F0ED01DF91085256DD00050F08A?OpenDocument
[2] Revelation Website: http://www.revelation.com/Revelation.nsf/1f484fef89cd63e1852569c900787d0e/be0c2876a6f1056b85256dc500587dd5?OpenDocument
[3] IBM developerWorks http://www.ibm.com/developerworks/wikis/display/im/OpenInsight+for+U2

External links

- Revelation Software (http://www.revelation.com)

Document-oriented database

A **document-oriented database** is a computer program designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data. Document-oriented databases are one of the main categories of so-called NoSQL databases and the popularity of the term "document-oriented database" (or "document store") has grown[1] with the use of the term NoSQL itself. In contrast to relational databases and their notions of "Relations" (or "Tables"), these systems are designed around an abstract notion of a "Document".

Documents

The central concept of a document-oriented database is the notion of a *Document*. While each document-oriented database implementation differs on the details of this definition, in general, they all assume documents encapsulate and encode data (or information) in some standard formats or encodings. Encodings in use include XML, YAML, JSON, and BSON, as well as binary forms like PDF and Microsoft Office documents (MS Word, Excel, and so on).

Documents inside a document-oriented database are similar, in some ways, to records or rows in relational databases, but they are less rigid. They are not required to adhere to a standard schema, nor will they have all the same sections, slots, parts, or keys. For example, the following is a document:

```
{
   FirstName: "Bob",
   Address: "5 Oak St.",
   Hobby: "sailing"
}
```

A second document might be:

```
{
  FirstName: "Jonathan",
  Address: "15 Wanamassa Point Road",
  Children: [
        {Name: "Michael", Age: 10},
        {Name: "Jennifer", Age: 8},
        {Name: "Samantha", Age: 5},
        {Name: "Elena", Age: 2}
  ]
 }
```

These two documents share some structural elements with one another, but each also has unique elements. Unlike a relational database where every record contains the same fields, leaving unused fields empty; there are no empty 'fields' in either document (record) in the above example. This approach allows new information to be added to some records without requiring that every other record in the database share the same structure.

Keys

Documents are addressed in the database via a unique *key* that represents that document. This key is often a simple string, a URI, or a path. The key can be used to retrieve the document from the database. Typically, the database retains an index on the key to speed up document retrieval.

Retrieval

Another defining characteristic of a document-oriented database is that, beyond the simple key-document (or key-value) lookup that can be used to retrieve a document, the database offers an API or query language that allows the user to retrieve documents based on their content. For example, you may want a query that retrieves all the documents with a certain field set to a certain value. The set of query APIs or query language features available, as well as the expected performance of the queries, varies significantly from one implementation to the next.

Organization

Implementations offer a variety of ways of organizing documents, including notions of

- Collections
- Tags
- Non-visible Metadata
- Directory hierarchies
- Buckets

Implementations

Name	Publisher	License	Language	Notes	RESTful API
ArangoDB [2]	triAGENS [3]	Apache 2 License	C, C++ & Javascript	A distributed multi model, high-performance document store and graph database.	Yes [4]
BaseX	BaseX Team [5]	BSD License	Java, XQuery	Support for XML, JSON and binary formats; client-/server based architecture; concurrent structural and full-text searches and updates; REST APIs.	Yes
Cassandra	Apache Software Foundation	Apache License	Java	JSON over HTTP	Yes
Cloudant	Cloudant, Inc. [6]	Proprietary	Erlang, Java, Scala, and C	Distributed database service based on BigCouch, the company's open source fork of the Apache-backed CouchDB project.	Yes
Clusterpoint	Clusterpoint Ltd. [7]	Free community license / Commercial[8]	C++	Schema-free, document-oriented database management system platform with server based data storage, full text search engine functionality, information ranking for search relevance and clustering.	Yes
Couchbase Server	Couchbase, Inc.	Apache License	Erlang and C	Distributed NoSQL Document Database.	Yes [9]
CouchDB	Apache Software Foundation	Apache License	Erlang	JSON over REST/HTTP with Multi-Version Concurrency Control and limited ACID properties. Uses map and reduce for views and queries.[10]	Yes [11]

eXist	eXist, [12]	GPL	XQuery, Java	XML over REST/HTTP, WebDAV, Lucene Fulltext search, validation, versioning, clustering, triggers, URL rewriting, collections, ACLS, XQuery Update	Yes [13]
FleetDB	FleetDB [14]	MIT License	Clojure	A JSON-based [15] schema-free database optimized for agile development.	(unknown)
Jackrabbit	Apache Software Foundation	Apache License	Java		(unknown)
Informix	IBM	Proprietary	Various (Compatible with MongoDB API)	RDBMS with JSON, replication, sharding and ACID compliance	(unknown)
Inquire	Infodata Systems, Inc.	Proprietary	unknown	In the mid-80's this was the dominant document-oriented commercial database, widely successful. The company seems to have gone out of business in 2005.	(unknown)
Lotus Notes	IBM	Proprietary	LotusScript, Java, Lotus @Formula		(unknown)
MarkLogic	MarkLogic Corporation	Free Developer license [16] or Commercial [16]	REST, Java, XQuery, XSLT, C++	Distributed document-oriented database with Multi-Version Concurrency Control, integrated Full text search and ACID-compliant transaction semantics	Yes
MongoDB	MongoDB, Inc	GNU AGPL v3.0[17]	C++	Document database with replication and sharding	Optional [18]
MUMPS Database[19]		Proprietary and Affero GPL[20]	MUMPS	Commonly used in health applications.	(unknown)
OrientDB	Orient Technologies [21]	Apache License	Java	JSON over HTTP	Yes
RavenDB	Hibernating Rhinos LTD [22]	Proprietary and modified Affero GPL[23]	C#, JavaScript		Yes
Redis		BSD License	ANSI C	Key-value store supporting lists and sets with binary-safe protocol	(unknown)
RethinkDB		GNU APGL for the DBMS, Apache 2 License for the client drivers	C++		(unknown)
Rocket U2	Rocket Software	Proprietary		UniData, UniVerse	Yes (Beta)
Sqrrl Enterprise [32]	sqrrl	Proprietary	Java	Distributed, real-time database featuring cell-level security and massive scalability.	Yes

XML database implementations

Most XML databases are document-oriented databases.

References

[1] DB-Engines Ranking per database model category (http://db-engines.com/en/ranking_categories)
[2] http://www.arangodb.org/
[3] http://www.triagens.com/
[4] ArangoDB REST API (http://www.arangodb.org/manuals/current/ImplementorManual.html)
[5] http://basex.org/
[6] https://cloudant.com/
[7] http://www.clusterpoint.com
[8] Clusterpoint DBMS Licensing Options (http://www.clusterpoint.com/licensing/)
[9] Documentation (http://www.couchbase.com/docs/). Couchbase. Retrieved on 2013-09-18.
[10] CouchDB Overview (http://couchdb.apache.org/docs/overview.html)
[11] CouchDB Document API (http://wiki.apache.org/couchdb/HTTP_Document_API)
[12] http://exist-db.org
[13] eXist-db Open Source Native XML Database (http://exist-db.org). Exist-db.org. Retrieved on 2013-09-18.
[14] http://fleetdb.org/
[15] http://fleetdb.org/docs/protocol.html
[16] http://developer.marklogic.com/licensing
[17] MongoDB License (http://www.mongodb.org/display/DOCS/Licensing)
[18] MongoDB REST Interfaces (http://www.mongodb.org/display/DOCS/Http+Interface#HttpInterface-RESTInterfaces)
[19] Extreme Database programming with MUMPS Globals (http://gradvs1.mgateway.com/download/extreme1.pdf)
[20] GTM MUMPS FOSS on SourceForge (http://sourceforge.net/projects/fis-gtm/)
[21] http://www.orientechnologies.com/
[22] http://hibernatingrhinos.com
[23] Ravendb Licensing (http://ravendb.net/licensing)

Further reading

- Assaf Arkin. (2007, September 20). Read Consistency: Dumb Databases, Smart Services. (http://blog.labnotes.org/2007/09/20/read-consistency-dumb-databases-smart-services/) Labnotes:Don't let the bubble go to your head!

MongoDB

MongoDB

Developer(s)	MongoDB Inc.
Initial release	2009
Stable release	2.4.9 / 10 January 2014
Preview release	2.5.4 / 18 November 2013
Development status	Active
Written in	C++
Operating system	Cross-platform
Available in	English
Type	Document-oriented database
License	GNU AGPL v3.0 (drivers: Apache license)
Website	www.mongodb.org [1]

MongoDB (from "hu**mongo**us") is a cross-platform document-oriented database system. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License, MongoDB is free and open source software.

First developed by the software company 10gen (now MongoDB Inc.) in October 2007 as a component of a planned platform as a service product, the company shifted to an open source development model in 2009, with 10gen offering commercial support and other services. Since then, MongoDB has been adopted as backend software by a number of major websites and services, including Craigslist, eBay, Foursquare, SourceForge, and The New York Times, among others. MongoDB is the most popular NoSQL database system.

History

Development of MongoDB began in 2007, when the company (then named 10gen) was building a platform as a service similar to Windows Azure or Google App Engine.[2] In 2009, MongoDB was open sourced as a stand-alone product[3] with an AGPL license.

From version 1.4 (March 2010), MongoDB has been considered production ready.[4]

The latest stable version, 2.4.9, was released on January 10, 2014.

Licensing and support

MongoDB is available for free under the GNU Affero General Public License. The language drivers are available under an Apache License. In addition, MongoDB Inc. offers commercial licenses for MongoDB.[5]

Main features

The following is a brief summary of some of the main features:[6]

Ad hoc queries

MongoDB supports search by field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.

Indexing

Any field in a MongoDB document can be indexed (indices in MongoDB are conceptually similar to those in RDBMSes). Secondary indices are also available.

Replication

MongoDB provides high availability and increased throughput with replica sets. A replica set consists of two or more copies of the data. Each replica may act in the role of primary or secondary replica at any time. The primary replica performs all writes and reads by default. Secondary replicas maintain a copy of the data on the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can also perform read operations, but the data is eventually consistent by default.

Load balancing

MongoDB scales horizontally using sharding. The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more slaves.)

MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure. Automatic configuration is easy to deploy, and new machines can be added to a running database.

File storage

MongoDB can be used as a file system, taking advantage of load balancing and data replication features over multiple machines for storing files.

This function, called GridFS,[7] is included with MongoDB drivers and available with no difficulty for development languages (see "Language Support" for a list of supported languages). MongoDB exposes functions for file manipulation and content to developers. GridFS is used, for example, in plugins for NGINX[8] and lighttpd.[9] Instead of storing a file in a single document, GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.[10]

In a multi-machine MongoDB system, files can be distributed and copied multiple times between machines transparently, thus effectively creating a load-balanced and fault-tolerant system.

Aggregation

MapReduce can be used for batch processing of data and aggregation operations. The aggregation framework enables users to obtain the kind of results for which the SQL GROUP BY clause is used.

Server-side JavaScript execution

JavaScript can be used in queries, aggregation functions (such as MapReduce), and sent directly to the database to be executed.

Capped collections

MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a circular queue.

Criticisms

MongoDB uses a readers-writer lock that allows concurrent read access to a database but exclusive write access to a single write operation.[11] Before version 2.2, this lock was implemented on a per-mongod basis. Since version 2.2, the lock has been implemented at the database level.[12] One approach to increase concurrency is to use sharding.[13] In some situations, reads and writes will yield their locks. If MongoDB predicts a page is unlikely to be in memory, operations will yield their lock while the pages load. The use of lock yielding expanded greatly in 2.2.[14]

Another criticism related to the limitations of MongoDB when used on 32-bit systems.[15] In some cases, this was due to inherent memory limitations.[16] MongoDB recommends 64-bit systems and that users provide sufficient RAM for their working set.[17] Some users encounter issues when their working set exceeds available RAM and the system encounters page faults. MongoHQ, a provider of managed MongoDB infrastructure, recommends a scaling checklist for large systems.[18]

Additionally, MongoDB provides very limited support for sorting on utf-8 encoded string data,[19] which poses a substantial problem when storing non-english text. Specifically, MongoDB currently uses the memcmp [20] function when sorting strings, which doesn't correctly handle utf-8 data under different locales.

Language support

MongoDB has official drivers for a variety of popular programming languages and development environments. Web programming language Opa also has built-in support for MongoDB, which is tightly integrated in the language and offers a type-safety layer on top of MongoDB. There are also a large number of unofficial or community-supported drivers for other programming languages and frameworks.

Management and graphical front-ends

MongoDB tools

In a MongoDB installation the following commands are available:

mongo

MongoDB offers an interactive shell called **`mongo`**,[21] which lets developers view, insert, remove, and update data in their databases, as well as get replication information, set up sharding, shut down servers, execute JavaScript, and more.

Administrative information can also be accessed through a **web interface**,[22] a simple webpage that serves information about the current server status. By default, this interface is 1000 ports above the database port (28017).

mongostat

`mongostat`[23] is a command-line tool that displays a summary list of status statistics for a currently running MongoDB instance: how many inserts, updates, removes, queries, and commands were performed, as well as what percentage of the time the database was locked and how much memory it is using. This tool is similar to the UNIX/Linux vmstat utility.

mongotop

`mongotop`[24] is a command-line tool providing a method to track the amount of time a MongoDB instance spends reading and writing data. mongotop provides statistics on the per-collection level. By default, mongotop returns values every second. This tool is similar to the UNIX/Linux top utility.

mongosniff

`mongosniff`[25] is a command-line tool providing a low-level tracing/sniffing view into database activity by monitoring (or "sniffing") network traffic going to and from MongoDB. `mongosniff` requires the Libpcap network library and is only available for Unix-like systems. A cross-platform alternative is the open source Wireshark packet analyzer which has full support for the MongoDB wire protocol.

mongoimport, mongoexport

`mongoimport`[26] is a command-line utility to import content from a JSON, CSV, or TSV export created by `mongoexport`[27] or potentially other third-party data exports.

mongodump, mongorestore

`mongodump`[28] is a command-line utility for creating a binary export of the contents of a Mongo database; `mongorestore`[29] can be used to reload a database dump.

Production deployments

Some of the prominent users of MongoDB include:[30]

- MetLife uses MongoDB for “The Wall," a customer service application providing a "360-degree view" of MetLife customers.[31]
- Craigslist stores over 2 billion records in MongoDB.[32]
- SAP uses MongoDB in the SAP PaaS.[33]
- Forbes stores articles and companies data in MongoDB.[34]
- The New York Times uses MongoDB in its form-building application for photo submissions.[35]
- Sourceforge uses MongoDB for its back-end storage pages.[36]
- Codecademy uses MongoDB as the datastore for its online learning system.[37]
- Shutterfly uses MongoDB for its photo platform. As of 2013, the photo platform stores 18 billion photos uploaded by Shutterfly's 7 million users.[38][39]
- The Guardian uses MongoDB for its identity system.[40]
- CERN uses MongoDB as the primary back-end for the *Data Aggregation System* for the Large Hadron Collider.[41]
- Foursquare deploys MongoDB on Amazon AWS to store venues and user check-ins into venues.[42]
- eBay uses MongoDB in the search suggestion and the internal Cloud Manager *State Hub*.[43]

References

[1] http://www.mongodb.org/
[2] MongoDB daddy: My baby beats Google BigTable (http://www.theregister.co.uk/2011/05/25/the_once_and_future_mongodb/)
[3] The MongoDB NoSQL Database Blog, The AGPL (http://blog.mongodb.org/post/103832439/the-agpl)
[4] The MongoDB NoSQL Database Blog, MongoDB 1.4 Ready for Production (http://blog.mongodb.org/post/472835820/mongodb-1-4-ready-for-production)
[5] MongoDB Support by 10gen (http://www.10gen.com/subscription)
[6] MongoDB Developer Manual (http://www.mongodb.org/display/DOCS/Manual)
[7] GridFS article on MongoDB Developer's Manual (http://www.mongodb.org/display/DOCS/GridFS)
[8] NGINX plugin for MongoDB source code (http://github.com/mdirolf/nginx-gridfs)
[9] lighttpd plugin for MongoDB source code (http://bitbucket.org/bwmcadams/lighttpd-gridfs/src/)
[10] Expertstown - MongoDB overview (http://www.expertstown.com/mongodb-overview/)
[11] FAQ: Concurrency (http://docs.mongodb.org/manual/faq/concurrency/)
[12] FAQ Concurrency - How Granular Are Locks (http://docs.mongodb.org/manual/faq/concurrency/#how-granular-are-locks-in-mongodb)
[13] FAQ Concurrency - How Does Sharding Affect Concurrency (http://docs.mongodb.org/manual/faq/concurrency/#how-does-sharding-affect-concurrency)
[14] FAQ Concurrency - Do Operations Ever Yield the Lock (http://docs.mongodb.org/manual/faq/concurrency/#does-a-read-or-write-operation-ever-yield-the-lock)

[15] 32-bit Limitations (http://blog.mongodb.org/post/137788967/32-bit-limitations)
[16] Does Everybody Hate MongoDB (https://blog.serverdensity.com/does-everyone-hate-mongodb/)
[17] What is the Working Set (http://docs.mongodb.org/manual/faq/storage/#what-is-the-working-set)
[18] Optimizing Your MongoDB Dataset (http://blog.mongohq.com/mongodb-scaling-to-100gb-and-beyond/)
[19] MongoDB Jira ticket 1920 (https://jira.mongodb.org/browse/SERVER-1920)
[20] http://en.cppreference.com/w/c/string/byte/memcmp
[21] mongo - The Interactive Shell (http://www.mongodb.org/display/DOCS/mongo+-+The+Interactive+Shell)
[22] HTTP Console (http://www.mongodb.org/display/DOCS/Http+Interface#HttpInterface-HTTPConsole)
[23] mongostat Manual (http://docs.mongodb.org/manual/reference/mongostat/)
[24] mongotop Manual (http://docs.mongodb.org/manual/reference/mongotop/)
[25] mongosniff Manual (http://docs.mongodb.org/manual/reference/mongosniff/)
[26] mongoimport Manual (http://docs.mongodb.org/manual/reference/mongoimport/)
[27] mongoexport Manual (http://docs.mongodb.org/manual/reference/mongoexport/)
[28] mongodump Manual (http://docs.mongodb.org/manual/reference/mongodump/)
[29] mongorestore Manual (http://docs.mongodb.org/manual/reference/mongorestore/)
[30] Production Deployments (http://www.mongodb.org/about/production-deployments/)
[31] MetLife Uses NoSQL For Customer Service Breakthrough (http://www.informationweek.com/software/information-management/metlife-uses-nosql-for-customer-service/240154741)
[32] Lessons Learned from Migrating 2+ Billion Documents at Craigslist (http://www.10gen.com/presentations/lessons-learned-migrating-2-billion-documents-craigslist)
[33] The Quest to Understand the Use of MongoDB in the SAP PaaS (http://scn.sap.com/people/richard.hirsch/blog/2011/09/30/the-quest-to-understand-the-use-of-mongodb-in-the-sap-paas)
[34] Supporting Distributed Global Workforce of Contributors with MongoDB (http://www.10gen.com/presentations/supporting-distributed-global-workforce-contributors-mongodb)
[35] NYT + MongoDB in Production (http://www.10gen.com/presentations/nyt-mongodb-production)
[36] Scaling SourceForge with MongoDB (http://www.oscon.com/oscon2010/public/schedule/detail/13669)
[37] How Codecademy is Using MongoDB (http://www.10gen.com/presentations/how-codecademy-using-mongodb)
[38] Real World NoSQL: MongoDB at Shutterfly (http://gigaom.com/2011/01/28/real-world-nosql-mongodb-at-shutterfly/)
[39] Here's How We Think Of Shutterfly's Stock Value (http://seekingalpha.com/article/1241641-heres-how-we-think-of-shutterflys-stock-value)
[40] MongoDB at The Guardian (http://www.10gen.com/presentations/mongodb-guardian)
[41] Holy Large Hadron Collider, Batman! (http://blog.mongodb.org/post/660037122/holy-large-hadron-collider-batman)
[42] Experiences Deploying MongoDB on AWS (http://www.10gen.com/presentations/experiences-deploying-mongodb-aws)
[43] MongoDB at eBay (http://www.slideshare.net/mongodb/mongodb-at-ebay)

Bibliography

- Banker, Kyle (March 28, 2011), *MongoDB in Action* (1st ed.), Manning, p. 375, ISBN 978-1-935182-87-0
- Chodorow, Kristina; Dirolf, Michael (September 23, 2010), *MongoDB: The Definitive Guide* (1st ed.), O'Reilly Media, p. 216, ISBN 978-1-4493-8156-1
- Pirtle, Mitch (March 3, 2011), *MongoDB for Web Development* (1st ed.), Addison-Wesley Professional, p. 360, ISBN 978-0-321-70533-4
- Hawkins, Tim; Plugge, Eelco; Membrey, Peter (September 26, 2010), *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing* (1st ed.), Apress, p. 350, ISBN 978-1-4302-3051-9

External links

- Official website (http://www.mongodb.org/)
- MongoDB Manual (http://docs.mongodb.org/manual/)
- Designing for the Cloud (http://www.technologyreview.com/video/?vid=356) at MIT Technology Review
- A vendor-independent comparison of NoSQL databases: Cassandra, HBase, MongoDB, Riak (http://www.networkworld.com/news/tech/2012/102212-nosql-263595.html) (NetworkWorld)

Spatiotemporal Databases

Temporal database

A **temporal database** is a database with built-in support for handling data involving time, for example a temporal data model and a temporal version of Structured Query Language (SQL).

More specifically the temporal aspects usually include valid time and transaction time. These attributes can be combined to form bitemporal data.

- **Valid time** is the time period during which a fact is true with respect to the real world.
- **Transaction time** is the time period during which a fact stored in the database is considered to be true.
- **Bitemporal data** combines both Valid and Transaction Time.

It is possible to have timelines other than Valid Time and Transaction Time, such as Decision Time, in the database. In that case the database is called a multitemporal database as opposed to a bitemporal database. However, this approach introduces additional complexities such as dealing with the validity of (foreign) keys.

Temporal databases are in contrast to current databases, which store only facts which are believed to be true at the current time.

Features

Temporal databases support managing and accessing temporal data by providing one or more of the following features:

- A time period datatype, including the ability to represent time periods with no end (infinity or forever)
- The ability to define valid and transaction time period attributes and bitemporal relations
- System-maintained transaction time
- Temporal primary keys, including non-overlapping period constraints
- Temporal constraints, including non-overlapping uniqueness and referential integrity
- Update and deletion of temporal records with automatic splitting and coalescing of time periods
- Temporal queries at current time, time points in the past or future, or over durations
- Predicates for querying time periods, often based on Allen's interval relations

History

With the development of SQL and its attendant use in real-life applications, database users realized that when they added date columns to key fields, some issues arose. For example, if a table has a primary key and some attributes, adding a date to the primary key to track historical changes can lead to creation of more rows than intended. Deletes must also be handled differently when rows are tracked in this way. In 1992, this issue was recognized but standard database theory was not yet up to resolving this issue, and neither was the then-newly-formalized SQL-92 standard.

Richard Snodgrass proposed in 1992 that temporal extensions to SQL be developed by the temporal database community. In response to this proposal, a committee was formed to design extensions to the 1992 edition of the SQL standard (ANSI X3.135.-1992 and ISO/IEC 9075:1992); those extensions, known as TSQL2, were developed during 1993 by this committee.[1] In late 1993, Snodgrass presented this work to the group responsible for the American National Standard for Database Language SQL, ANSI Technical Committee X3H2 (now known as NCITS H2). The preliminary language specification appeared in the March 1994 ACM SIGMOD Record. Based on responses to that specification, changes were made to the language, and the definitive version of the TSQL2

Language Specification was published in September, 1994

An attempt was made to incorporate parts of TSQL2 into the new SQL standard SQL:1999, called SQL3. Parts of TSQL2 were included in a new substandard of SQL3, ISO/IEC 9075-7, called SQL/Temporal. The TSQL2 approach was heavily criticized by Chris Date and Hugh Darwen.[2] The ISO project responsible for temporal support was canceled near the end of 2001.

As of December 2011, ISO/IEC 9075, Database Language SQL:2011 Part 2: SQL/Foundation included clauses in table definitions to define "application-time period tables" (valid time tables), "system-versioned tables" (transaction time tables) and "system-versioned application-time period tables" (bitemporal tables). A substantive difference between the TSQL2 proposal and what was adopted in SQL:2011 is that there are no hidden columns in the SQL:2011 treatment, nor does it have a new data type for intervals; instead two date or timestamp columns can be bound together using a `PERIOD FOR` declaration. Another difference is replacement of the controversial (prefix) statement modifiers from TSQL2 with a set of temporal predicates.

Example

For illustration, consider the following short biography of a fictional man, John Doe:

> John Doe was born on April 3, 1975 in the Kids Hospital of Medicine County, as son of Jack Doe and Jane Doe who lived in Smallville. Jack Doe proudly registered the birth of his first-born on April 4, 1975 at the Smallville City Hall. John grew up as a joyful boy, turned out to be a brilliant student and graduated with honors in 1993. After graduation he went to live on his own in Bigtown. Although he moved out on August 26, 1994, he forgot to register the change of address officially. It was only at the turn of the seasons that his mother reminded him that he had to register, which he did a few days later on December 27, 1994. Although John had a promising future, his story ends tragically. John Doe was accidentally hit by a truck on April 1, 2001. The coroner reported his date of death on the very same day.

Using a current database

To store the life of John Doe in a current (non-temporal) database we use a table *Person (Name, Address)*. (In order to simplify *Name* is defined as the primary key of *Person*.)

John's father officially reported his birth on April 4, 1975. On this date a Smallville official inserted the following entry in the database: `Person(John Doe, Smallville)`. Note that the date itself is not stored in the database.

After graduation John moves out, but forgets to register his new address. John's entry in the database is not changed until December 27, 1994, when he finally reports it. A Bigtown official updates his address in the database. The *Person* table now contains `Person(John Doe, Bigtown)`. Note that the information of John living in Smallville has been overwritten, so it is no longer possible to retrieve that information from the database. An official accessing the database on December 28, 1994 would be told that John lives in Bigtown. More technically: if a database administrator ran the query `SELECT ADDRESS FROM PERSON WHERE NAME='John Doe'` on December 26, 1994, the result would be `Smallville`. Running the same query 2 days later would result in `Bigtown`.

Until his death the database would state that he lived in Bigtown. On April 1, 2001 the coroner deletes the John Doe entry from the database. After this, running the above query would return no result at all.

Date	Real world event	Database Action	What the database shows
April 3, 1975	John is born	Nothing	There is no person called John Doe
April 4, 1975	John's father officially reports John's birth	Inserted:Person(John Doe, Smallville)	John Doe lives in Smallville
August 26, 1994	After graduation, John moves to Bigtown, but forgets to register his new address	Nothing	John Doe lives in Smallville
December 26, 1994	Nothing	Nothing	John Doe lives in Smallville
December 27, 1994	John registers his new address	Updated:Person(John Doe, Bigtown)	John Doe lives in Bigtown
April 1, 2001	John dies	Deleted:Person(John Doe)	There is no person called John Doe

Using Valid time

Valid time is the time for which a fact is true in the real world. A valid time period may be in the past, span the current time, or occur in the future.

For the example above, to record valid time the *Person* table has two fields added, *Valid-From* and *Valid-To*. These specify the period when a person's address is valid in the real world. On April 4, 1975 John's father registered his son's birth. An official then inserts a new entry into the database stating that John lives in Smallville from April 3. Note that although the data was inserted on the 4th, the database states that the information is valid since the 3rd. The official does not yet know if or when John will move to another place, so the *Valid-To* field is set to infinity (∞). The entry in the database is:

```
Person(John Doe, Smallville, 3-Apr-1975, ∞).
```

On December 27, 1994 John reports his new address in Bigtown where he has been living since August 26, 1994. A new database entry is made to record this fact:

```
Person(John Doe, Bigtown, 26-Aug-1994, ∞).
```

The original entry `Person (John Doe, Smallville, 3-Apr-1975, ∞)` is not deleted, but has the *Valid-To* attribute updated to reflect that it is now known that John stopped living in Smallville on August 26, 1994. The database now contains two entries for John Doe

```
Person(John Doe, Smallville, 3-Apr-1975, 26-Aug-1994).
Person(John Doe, Bigtown, 26-Aug-1994, ∞).
```

When John dies his current entry in the database is updated stating that John does not live in Bigtown any longer. The database now looks like this

```
Person(John Doe, Smallville, 3-Apr-1975, 26-Aug-1994).
Person(John Doe, Bigtown, 26-Aug-1994, 1-Apr-2001).
```

Using Transaction time

Transaction time records the time period during which a database entry is accepted as correct. This enables queries that show the state of the database at a given time. Transaction time periods can only occur in the past or up to the current time. In a transaction time table, records are never deleted. Only new records can be inserted, and existing ones updated by setting their transaction end time to show that they are no longer current.

To enable transaction time in the example above, two more fields are added to the Person table: *Transaction-From* and *Transaction-To*. *Transaction-From* is the time a transaction was made, and *Transaction-To* is the time that the transaction was superseded (which may be infinity if it has not yet been superseded). This makes the table into a bitemporal table.

What happens if the person's address as stored in the database is incorrect? Suppose an official accidentally entered the wrong address or date? Or, suppose the person lied about their address for some reason. Upon discovery of the error, the officials update the database to correct the information recorded.

For example, from 1-Jun-1995 to 3-Sep-2000 John Doe moved to Beachy. But to avoid paying Beachy's exorbitant residence tax, he never reported it to the authorities. Later during a tax investigation it is discovered on 2-Feb-2001 that he was in fact in Beachy during those dates. To record this fact the existing entry about John living in Bigtown must be split into two separate records, and a new record inserted recording his residence in Beachy. The database would then appear as follows:

```
Person(John Doe, Smallville, 3-Apr-1975, 26-Aug-1994).
Person(John Doe, Bigtown, 26-Aug-1994, 1-Jun-1995).
Person(John Doe, Beachy, 1-Jun-1995, 3-Sep-2000).
Person(John Doe, Bigtown, 3-Sep-2000, 1-Apr-2001).
```

However, this leaves no record that the database ever claimed that he lived in Bigtown during 1-Jun-1995 to 3-Sep-2000. This might be important to know for auditing reasons, or to use as evidence in the official's tax investigation. Transaction time allows capturing this changing knowledge in the database, since entries are never directly modified or deleted. Instead, each entry records when it was entered and when it was superseded (or logically deleted). The database contents then look like this:

```
Person(John Doe, Smallville, 3-Apr-1975,  ∞,           4-Apr-1975,  27-Dec-1994).
Person(John Doe, Smallville, 3-Apr-1975,  26-Aug-1994, 27-Dec-1994, ∞     ).
Person(John Doe, Bigtown,    26-Aug-1994, ∞,           27-Dec-1994, 2-Feb-2001 ).
Person(John Doe, Bigtown,    26-Aug-1994, 1-Jun-1995,  2-Feb-2001,  ∞     ).
Person(John Doe, Beachy,     1-Jun-1995,  3-Sep-2000,  2-Feb-2001,  ∞     ).
Person(John Doe, Bigtown,    3-Sep-2000,  ∞,           2-Feb-2001,  1-Apr-2001 ).
Person(John Doe, Bigtown,    3-Sep-2000,  1-Apr-2001,  1-Apr-2001,  ∞     ).
```

The database records not only what happened in the real world, but also what was officially recorded at different times.

Bitemporal relations

A bitemporal relation contains both valid and transaction time. This provides both **historical** and **rollback** information. Historical information (e.g.: "Where did John live in 1992?") is provided by the valid time. Rollback (e.g.: "In 1992, where did the database believe John lived?") is provided by the transaction time. The answers to these example questions may not be the same - the database may have been altered since 1992, causing the queries to produce different results.

The valid time and transaction time do not have to be the same for a single fact. For example, consider a temporal database storing data about the 18th century. The valid time of these facts is somewhere between 1701 and 1800.

The transaction time records when the facts are inserted into the database (for example, January 21, 1998).

Schema Evolution

A challenging issue is the support of temporal queries in a transaction time database under evolving schema. In order to achieve perfect archival quality it is of key importance to store the data under the schema version under which they firstly appeared. However even the most simple temporal query rewriting the history of an attribute value would be required to be manually rewritten under each of the schema versions, potentially hundreds as in the case of MediaWiki [1] This process would be particularly taxing for users. A proposed solution is to provide automatic query rewriting, although this is not part of SQL or similar standards.

Implementations in databases

The following implementations provide temporal features in a relational database management system (RDBMS).

- Oracle Workspace Manager [3] is a feature of Oracle Database which enables application developers and DBAs to manage current, proposed and historical versions of data in the same database.
- TimeDB [4] is a free temporal relational DBMS by TimeConsult. It runs as a frontend to Oracle that accepts TSQL2 statements and generates SQL92 statements.
- PostgreSQL [5] has an open-source contributed package that can be installed in the database to manage temporal data. The function reference is here [6].
- Teradata version 13.10 [7] and Teradata version 14 has temporal features based on TSQL2[8] built into the database.
- Anchor Modeling [9] emulates temporal features and automates the implementation in databases that lack support.
- IBM DB2 version 10 added a feature called "time travel query"[] which is based on the temporal capabilities of the SQL:2011 standard.[]

References

[1] Snodgrass, 1999, p. 9

[2] Hugh Darwen, C.J. Date, " An overview and Analysis of Proposals Based on the TSQL2 Approach (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.116.7598)", In *Date on Database: Writings 2000-2006*, C.J. Date, Apress, 2006, pp. 481-514

[3] http://www.oracle.com/technology/products/database/workspace_manager/index.html

[4] http://www.timeconsult.com/Software/Software.html

[5] http://pgfoundry.org/projects/temporal/

[6] http://temporal.projects.postgresql.org/reference.html

[7] http://www.teradata.com/t/database/Teradata-Temporal/

[8] Al-Kateb, Mohammed et al. " Temporal Query Processing in Teradata (http://www.edbt.org/Proceedings/2013-Genova/papers/edbt/a51-al-kateb.pdf)". EDBT/ICDT '13 March 18–22, 2013, Genoa, Italy

[9] http://www.anchormodeling.com/

- C.J. Date, Hugh Darwen, Nikos Lorentzos (2002). *Temporal Data & the Relational Model, First Edition* (The Morgan Kaufmann Series in Data Management Systems); Morgan Kaufmann; 1st edition; 422 pages. ISBN 1-55860-855-9.
- Joe Celko (2005). *Joe Celko's SQL for Smarties: Advanced SQL Programming* (The Morgan Kaufmann Series in Data Management); Morgan Kaufmann; 3rd edition; 808 pages. ISBN 0-12-369379-9.—Chapters 4 and 29 in particular discuss temporal issues.
- Snodgrass, Richard T. (1999). *Developing Time-Oriented Database Applications in SQL* (http://www.cs.arizona.edu/people/rts/tdbbook.pdf) PDF (4.77 MiB) (Morgan Kaufmann Series in Data Management Systems); Morgan Kaufmann; 504 pages; ISBN 1-55860-436-7

Further reading

- Bitemporal Database Basics (http://www.xosoftware.co.uk/Articles/BitemporalDatabaseBasics/)

External links

- TimeCenter (http://www.cs.arizona.edu/projects/timecenter)
- TimeConsult (http://www.timeconsult.com)
- Temporal Relations in RDF (http://blogs.sun.com/roller/page/bblfish/20060320)
- Temporal Scope for RDF Triples (http://www.jenitennison.com/blog/node/101)
- com.ervacon.bitemporal (https://svn.ervacon.com/public/projects/bitemporal/trunk/readme.txt) Simple bitemporal framework in Java
- DAOFusion (http://opensource.anasoft.com/daofusion-site/reference/bitemporal-pattern.html) open source project with temporal data support
- Developing Time-Oriented Database Applications in SQL (http://www.cs.arizona.edu/people/rts/tdbbook.pdf) by Richard Snodgrass
- IBM DB2 10 for z/OS (http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z10.doc/db2z_10_prodhome.htm)
- Time and Time Again series of articles by Randy Weis and Tom Johnston (http://www.inbaseinc.com/dm-series.htm)
- Temporal Patterns (http://martinfowler.com/eaaDev/timeNarrative.html) by Martin Fowler

RRDtool

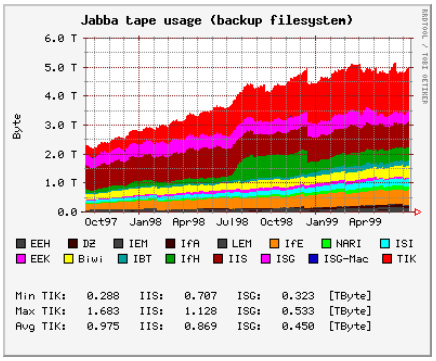
RRDtool

Original author(s)	Tobi Oetiker
Stable release	1.4.8 / May 23, 2013
Written in	C
License	GNU General Public License
Website	oss.oetiker.ch/rrdtool/ [1]

RRDtool (acronym for **r**ound-**r**obin **d**atabase **tool**) aims to handle time-series data like network bandwidth, temperatures, CPU load, etc. The data are stored in a round-robin database (circular buffer), thus the system storage footprint remains constant over time.

It also includes tools to extract **RRD** data in a graphical format, for which it was originally intended.

Bindings exist for Perl, Python, Ruby, Tcl, PHP and Lua. And there is an independent full Java implementation, rrd4j.



RRDtool has a graph function, which presents data from an RRD in a customizable graphical format

General data storage

RRDtool assumes time-variable data in intervals of a certain length. This interval, usually named **step**, is specified upon creation of an RRD file and cannot be changed afterwards. Because data may not always be available at just the right time, RRDtool will automatically interpolate any submitted data to fit its internal time-steps.

The value for a specific step, that has been interpolated, is named a primary data point (**PDP**). Multiple PDPs may be consolidated according to a consolidation function (**CF**) to form a consolidated data point (**CDP**). Typical consolidation functions are average, minimum, maximum.

After the data have been consolidated, the resulting CDP is stored in a round-robin archive (**RRA**). A round-robin archive stores a fixed number of CDPs and specifies how many PDPs should be consolidated into one CDP and which CF to use. The total time covered by an RRA can be calculated as follows:

```
time covered = (#CDPs stored) * (#PDPs per CDP) * steps
```

After this time the archive will "wrap around": the next insertion will overwrite the oldest entry. This behavior is sometimes referred to as "round-robin" and is the reason for the program's name.

To cover several timespans and/or use several consolidation functions, an RRD file may contain multiple RRAs. The data retrieval function of RRDtool automatically selects the archive with the highest resolution that still covers the requested timespan. This mechanism is also used by RRDtool's graphing subsystem.

Release history

Colour	Meaning
Red	Release no longer supported
Green	Release still supported
Blue	Future release

RRDTool is sponsored since 1.2, each release comes with a list of sponsors.

The following table contains the **release history of RRDtool**, showing its major releases.

Version number	Date	Links	Notable changes
1.0	July 16, 1999	Full release notes [2], Announce [3]	First release. Basically MRTG "done right".
1.1	April 25, 2005	Full release notes [4], Announce [5]	libart; output EPS, PDF & SVG; VDEF; trends; percentiles; updatev; Holt-Winters Forecasting; COMPUTE; .rrd format change.
1.3	June 11, 2008	Full release notes [6], Announce [7]	Safer & faster file access; cairo/pango; anti-aliasing; TEXTALIGN; dashed lines; new HWPREDICT; libxml; i18n; XML dump;
1.4	October 27, 2009	Full release notes [8], Announce [9]	Caching daemon; VDEF PERCENTNAN; CDEF PREDICT & PREDICTSIGMA; libDBI; graph legends positioning; Lua bindings; 3D border width; and more ...

Other tools that use RRDtool as a DBMS and/or graphing subsystem

- BackupPC
- Cacti
- Cherokee
- collectd
- Ganglia – system monitor clusters and grids
- Lighttpd
- Monitorix
- MRTG
- Munin
- N2rrd
- Nagios
- Nmon
- NMIS[10]
- ntop
- Observium
- OpenNMS
- ServersCheck
- SNM [11] - System and Network Monitor
- Zenoss

External links

- Official website [12]
- RRDtool screenshot gallery [13]
- RRDtool tutorial [14]
- RRDtool frontends [15]

References

[1] http://oss.oetiker.ch/rrdtool/
[2] http://oss.oetiker.ch/rrdtool-trac/browser/tags/1.0/program/NEWS
[3] https://lists.oetiker.ch/pipermail/rrd-announce/1999-July/000007.html
[4] http://oss.oetiker.ch/rrdtool-trac/browser/tags/1.2.0/NEWS
[5] https://lists.oetiker.ch/pipermail/rrd-announce/2005-April/000071.html
[6] http://oss.oetiker.ch/rrdtool-trac/browser/tags/1.3.0/NEWS
[7] https://lists.oetiker.ch/pipermail/rrd-announce/2008-June/000109.html
[8] http://oss.oetiker.ch/rrdtool-trac/browser/tags/1.4.0/NEWS
[9] https://lists.oetiker.ch/pipermail/rrd-announce/2009-October/000134.html
[10] http://nmis.sourceforge.net/
[11] http://snm.sf.net
[12] http://oss.oetiker.ch/rrdtool
[13] http://oss.oetiker.ch/rrdtool/gallery/
[14] http://oss.oetiker.ch/rrdtool/tut/
[15] http://web.archive.org/web/20020204003747/http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/rrdworld/

Spatial database

A **spatial database** is a database that is optimized to store and query data that represents objects defined in a geometric space. Most spatial databases allow representing simple geometric objects such as points, lines and polygons. Some spatial databases handle more complex structures such as 3D objects, topological coverages, linear networks, and TINs. While typical databases are designed to manage various numeric and character types of data, additional functionality needs to be added for databases to process spatial data types efficiently. These are typically called *geometry* or *feature*. The Open Geospatial Consortium created the Simple Features specification and sets standards for adding spatial functionality to database systems.[1]

Features of spatial databases

Database systems use indexes to quickly look up values and the way that most databases index data is not optimal for spatial queries. Instead, spatial databases use a spatial index to speed up database operations.

In addition to typical SQL queries such as SELECT statements, spatial databases can perform a wide variety of spatial operations. The following operations and many more are specified by the Open Geospatial Consortium standard:

- Spatial Measurements: Computes line length, polygon area, the distance between geometries, etc.
- Spatial Functions: Modify existing features to create new ones, for example by providing a buffer around them, intersecting features, etc.
- Spatial Predicates: Allows true/false queries about spatial relationships between geometries. Examples include "do two polygons overlap" or 'is there a residence located within a mile of the area we are planning to build the landfill?' (see DE-9IM)
- Geometry Constructors: Creates new geometries, usually by specifying the vertices (points or nodes) which define the shape.

- Observer Functions: Queries which return specific information about a feature such as the location of the center of a circle

Some databases support only simplified or modified sets of these operations, especially in cases of NoSQL systems like MongoDB and CouchDB.

Spatial index

Spatial indices are used by spatial databases (databases which store information related to objects in space) to optimize spatial queries. Conventional index types do not efficiently handle spatial queries such as how far two points differ, or whether points fall within a spatial area of interest. Common spatial index methods include:

- Grid (spatial index)
- Z-order (curve)
- Quadtree
- Octree
- UB-tree
- R-tree: Typically the preferred method for indexing spatial data.[*citation needed*] Objects (shapes, lines and points) are grouped using the minimum bounding rectangle (MBR). Objects are added to an MBR within the index that will lead to the smallest increase in its size.
- R+ tree
- R* tree
- Hilbert R-tree
- X-tree
- kd-tree
- m-tree - an m-tree index can be used for the efficient resolution of similarity queries on complex objects as compared using an arbitrary metric.

Spatial database systems

- All OpenGIS Specifications compliant products[2]
- Open source spatial databases and APIs, some of which are OpenGIS compliant[3]
- Boeing's Spatial Query Server spatially enables Sybase ASE.
- Smallworld VMDS, the native GE Smallworld GIS database
- SpatiaLite extends Sqlite with spatial datatypes, functions, and utilities.
- IBM DB2 Spatial Extender can be used to enable any edition of DB2, including the free DB2 Express-C, with support for spatial types
- Oracle Spatial
- Microsoft SQL Server has support for spatial types since version 2008
- PostgreSQL DBMS (database management system) uses the spatial extension PostGIS to implement the standardized datatype *geometry* and corresponding functions.
- Linter SQL Server supports spatial types and spatial functions according to the OpenGIS specifications.
- MySQL DBMS implements the datatype *geometry* plus some spatial functions that have been implemented according to the OpenGIS specifications.[4] However, in MySQL version 5.5 and earlier, functions that test spatial relationships are limited to working with minimum bounding rectangles rather than the actual geometries. MySQL versions earlier than 5.0.16 only supported spatial data in MyISAM tables. As of MySQL 5.0.16, InnoDB, NDB, BDB, and ARCHIVE also support spatial features.
- Neo4j - Graph database that can build 1D and 2D indexes as Btree, Quadtree and Hilbert curve directly in the graph

- AllegroGraph - a Graph database provides a novel mechanism for efficient storage and retrieval of two-dimensional geospatial coordinates for Resource Description Framework data. It includes an extension syntax for SPARQL queries
- MongoDB and RavenDB [5] support geospatial indexes in 2D
- Esri has a number of both single-user and multiuser geodatabases.
- SpaceBase [6] is a real-time spatial database.[7]
- CouchDB a document based database system that can be spatially enabled by a plugin called Geocouch
- CartoDB [8] is a cloud based geospatial database on top of PostgreSQL with PostGIS.
- StormDB [9] is an upcoming cloud based database on top of PostgreSQL with geospatial capabilities.
- SpatialDB [10] by MineRP is the world's first open standards (OGC) spatial database with spatial type extensions for the Mining Industry.[11]
- H2 supports geometry types[12] and spatial indices[13] as of version 1.3.173 (2013-07-28). An extension called H2GIS [14] available on Maven Central gives full OGC Simple Features support.

References

[1] OGC Homepage (http://www.opengeospatial.org)
[2] All Registered Products at opengeospatial.org (http://www.opengeospatial.org/resources/?page=products)
[3] Open Source GIS website (http://opensourcegis.org/)
[4] http://dev.mysql.com/doc/refman/5.5/en/gis-introduction.html
[5] http://ravendb.net
[6] http://paralleluniverse.co
[7] SpaceBase product page on the Parallel Universe website (http://paralleluniverse.co/spacebase)
[8] http://cartodb.com/
[9] http://www.stormdb.com
[10] http://www.minerpsolutions.com/en/software/enterprise/spatialDB
[11] SpatialDB product page on the MineRP website (http://www.minerpsolutions.com/en/software/enterprise/spatialDB)
[12] H2 geometry type documentation (http://www.h2database.com/html/datatypes.html#geometry_type)
[13] H2 create spatial index documentation (http://www.h2database.com/html/grammar.html#create_index)
[14] http://www.h2gis.org

Further reading

- Spatial Databases: A Tour (http://www.spatial.cs.umn.edu/Book/), Shashi Shekhar and Sanjay Chawla, Prentice Hall, 2003 (ISBN 0-13-017480-7)
- ESRI Press (http://gis.esri.com/esripress/). ESRI Press titles include **Modeling Our World: The ESRI Guide to Geodatabase Design**, and **Designing Geodatabases: Case Studies in GIS Data Modeling** , 2005 Ben Franklin Award (http://www.pma-online.org/benfrank2005_winnerfinalist.cfm) winner, PMA, The Independent Book Publishers Association.
- Spatial Databases - With Application to GIS (http://mkp.com/books/data-management) Philippe Rigaux, Michel Scholl and Agnes Voisard. Morgan-Kauffman Publishers. 2002 (ISBN 1-55860-588-6)

External links

- An introduction to PostgreSQL PostGIS (http://www.mapbender.org/presentations/Spatial_Data_Management_Arnulf_Christl/html/)
- PostgreSQL PostGIS as components in a Service Oriented Architecture (http://www.gisdevelopment.net/magazine/years/2006/jan/18_1.htm) SOA
- A Trigger Based Security Alarming Scheme for Moving Objects on Road Networks (http://www.springerlink.com/content/vn7446g28924jv5v/) Sajimon Abraham, P. Sojan Lal, Published by Springer Berlin / Heidelberg-2008.

Spatial query

A **spatial query** is a special type of database query supported by geodatabases and spatial databases. The queries differ from SQL queries in several important ways. Two of the most important are that they allow for the use of geometry data types such as points, lines and polygons and that these queries consider the spatial relationship between these geometries.

Types of queries

The function names for queries differ across geodatabases. The following list contains commonly used functions built into PostGIS, a free geodatabase which is a PostgreSQL extension (the term 'geometry' refers to a point, line, box or other two or three dimensional shape):

Function prototype: *functionName (parameter(s)) : return type*

- Distance(geometry, geometry) : number
- Equals(geometry, geometry) : boolean
- Disjoint(geometry, geometry) : boolean
- Intersects(geometry, geometry) : boolean
- Touches(geometry, geometry) : boolean
- Crosses(geometry, geometry) : boolean
- Overlaps(geometry, geometry) : boolean
- Contains(geometry, geometry) : boolean
- Length(geometry) : number
- Area(geometry) : number
- Centroid(geometry) : geometry

Spatiotemporal database

A **spatiotemporal database** is a database that manages both space and time information. Common examples include:

- Tracking of moving objects, which typically can occupy only a single position at a given time.
- A database of wireless communication networks, which may exist only for a short timespan within a geographic region.
- An index of species in a given geographic region, where over time additional species may be introduced or existing species migrate or die out.
- Historical tracking of plate tectonic activity.

At first glance, spatiotemporal databases are an extension of spatial databases. A spatiotemporal database embodies spatial, temporal, and spatiotemporal database concepts, and captures spatial and temporal aspects of data and deals with

- geometry changing over time[1] and/or
- location of objects moving over invariant geometry (known variously as *moving objects databases* or real-time locating systems)

However, although there exist numerous relational databases with spatial extensions, the spatiotemporal databases are not based on the relational model for practical reasons, chiefly among them that the data is multi-dimensional and capturing complex structures and behaviours. As of 2008, there are no RDBMS products with spatiotemporal extensions. There are some products such as the open-source TerraLib which use a middleware approach storing their data in a relational database. Unlike in the pure spatial domain, there are however no official or *de facto* standards for spatio-temporal data models and their querying. In general, the theory of this area is also less well-developed. Another approach is the constraint database system such as MPLQ (Management of Linear Programming Queries).[2]

Organizations

- http://vldb.org (Very Large Databases)
- http://www.dexa.org (Database and Expert Systems Applications)

References

[1] Data Models and Query Languages of Spatio-Temporal Information (http://webcache.googleusercontent.com/search?q=cache:9qs8QhcwW74J:wis.cs.ucla.edu/wis/theses/thesis_cchen.ps+&cd=1&hl=en&ct=clnk&gl=us)

[2] http://www.cse.unl.edu/~revesz/MLPQ/mlpq.htm

Object-based spatial database

An **object-based spatial database** is a spatial database that stores the location as objects. The object-based spatial model treats the world as surface littered with recognizable objects (e.g. cities, rivers), which exist independent of their locations.

Objects can be simple as polygons and lines, or be more complex to represent cities.

While a field-based data model sees the world as a continuous surface over which features (e.g. elevation) vary, using an object-based spatial database, it is easier to store additional attributes with the objects, such as direction, speed, etc. Using these attributes can make it easier to answer queries like "find all tanks whose speed is 10 km and oriented to north". Or "find all enemy tanks in a certain region".

Storing attributes with objects can provide better result presentation and improved manipulation capabilities in a more efficient way. In a field-based data model, this information is usually stored at different layers and it is harder to extract different information from various layers. This data model can be applied above the ER as in GERM model and GISER.

S.Shekhar introduces direction as a spatial object and presents a solution to object-direction-based queries.

Data model representation

The most common representations for the data model follow.

PostGIS

An open-source software program that adds support for geographic objects to the PostgreSQL object-relational database. PostGIS follows the Simple Features for SQL specification from the Open Geospatial Consortium.

OMT-G

Provides a UML representation for geographic applications, it can represent the concept of field, object and provides a way to differentiate between spatial relation and simple association.

GraphDB

Represents a framework of objects as classes that are partitioned into three kinds of classes: simple classes, link classes, and path classes. Objects of a simple class are on the one hand just like objects in other models. They have an object type and an object identity and can have attributes whose values are either of a data type (e.g. integer, string) or of an object type (that is, an attribute may contain a reference to another object). So the structure of an object is basically that of a tuple or record. On the other hand, objects of a simple class are nodes of the database graph – the whole database can also be viewed as a single graph. Objects of a link class are like objects of a simple class but additionally contain two distinguished references to source and target objects (belonging to simple classes), which makes them edges of the database graph. Finally, an object of a path class is like an object of a simple class, but contains additionally a list of references to node and edge objects which form a path over the database graph.

GEIS

Represent a data model to store geographic information on top of EER model, GEIS define the input data model and provide the following for data model Geometry. In the GISER model, geometry is an entity that is related to a spatial object by the relationship determines shape of. Additional entities represent the primitives such as points, lines, and polygons as proposed in related models. Topology. Topology is a property belonging to a spatial object and that property remains unaltered even when the object deforms. An example is a road network. The two nodes in the

network thus remain connected even if the path between the nodes is changed by road construction. In order to represent the topology, the basic primitives such as networks (i.e., graphs) and partitions are provided. Additional primitives can be added on lines of the Worboy model, This system support representation for stored data.

Oracle spatial

Oracle spatial is a component of enterprise Oracle 10g and provides support to stores object such as road on top of the current implentend construction but it used network data model to store geographic data as nodes and links (a graph representation) with each node or links it has a set of attributes. For example a route object can be added to the database.

GRASS GIS

It supports raster and some set of vector representation.

References

- Borges, K. A., Davis, C. A., and Laender, A. H. 2001. OMT-G: "An Object-Oriented Data Model for Geographic Applications." [1] *Geoinformatica 5*, 3 (Sep. 2001), 221-260.
- Flick, S. 1996. "An object-oriented framework for the realization of 3D geographic information systems." In *Proceedings of the Second Joint European Conference &Amp; Exhibition on Geographical information (Vol. 1) : From Research To Application Through Cooperation: From Research To Application Through Cooperation* (Barcelona, Spain). M. Rumor, R. McMillan, and H. F. Ottens, Eds. IOS Press, Amsterdam, The Netherlands, 187-196.
- Shekhar, S., Coyle, M., Goyal, B., Liu, D., and Sarkar, S. 1997. "Data models in geographic information systems." *Commun. ACM* 40, 4 (Apr. 1997), 103-111. [2]

Oracle spatial documentation

- Medeiros, C. B. and Pires, F. 1994. "Databases for GIS." *SIGMOD Rec.* 23 [3], 1 (Mar. 1994), 107-115.
- Orenstein, J. A. 1986. "Spatial query processing in an object-oriented database system." [4] In *Proceedings of the 1986 ACM SIGMOD international Conference on Management of Data* (Washington, D.C., United States, May 28–30, 1986). C. Zaniolo, Ed. SIGMOD '86. ACM Press, New York, NY, 326-336.

References

[1] http://dx.doi.org/10.1023/A:1011482030093
[2] http://doi.acm.org/10.1145/248448.248465
[3] http://doi.acm.org/10.1145/181550.181566
[4] http://doi.acm.org/10.1145/16894.16886

Tuple-versioning

Tuple-versioning (also called **point-in-time**) is a mechanism used in a relational database management system to store past states of a relation. Normally, only the current state is captured.

Using tuple-versioning techniques, typically two values for time are stored along with each tuple: a start time and an end time. These two values indicate the validity of the rest of the values in the tuple.

Typically when tuple-versioning techniques are used, the current tuple has a valid start time, but a null value for end time. Therefore, it is easy and efficient to obtain the current values for all tuples by querying for the null end time.

A single query that searches for tuples with start time less than, and end time greater than, a given time (where null end time is treated as a value greater than the given time) will give as a result the valid tuples at the given time.

For example, if a person's job changes from Engineer to Manager, there would be two tuples in an Employee table, one with the value Engineer for job and the other with the value Manager for job. The end time for the Engineer tuple would be equal to the start time for the Manager tuple.

The pattern known as log trigger uses this technique to automatically store historical information of a table in a database.

References

- Comparison of Access Methods for Time-Evolving Data [1], by Betty Salzberg and Vassilis J. Tsotras, ACM Computing Surveys, Vol. 31, No. 2, June 1999.

References

[1] http://portal.acm.org/citation.cfm?doid=319806.319816

Valid time

Valid time (VT), a concept originated by Richard T. Snodgrass and his doctoral student, is used in temporal databases.[1] It denotes the time period during which a database fact was, is, or will be valid in the modeled reality. As of December 2011, ISO/IEC 9075, Database Language SQL:2011 Part 2: SQL/Foundation included clauses in table definitions to define "application-time period tables" (that is, valid-time tables).

In a database table valid time is often represented by two extra table-columns *StartVT* and *EndVT*. The time interval is closed at its lower bound and open at its upper bound.

Example:

Date	What happened in the real world	Database Action	What the database shows
April 3, 1975	John is born	Nothing	There is no person called John Doe
April 4, 1975	John's father officially reports John's birth	Inserted:Person(John Doe, Smallville)	John Doe lives in Smallville
August 26, 1994	After graduation, John moves to Bigtown, but forgets to register his new address	Nothing	John Doe lives in Smallville
December 26, 1994	Nothing	Nothing	John Doe lives in Smallville
December 27, 1994	John registers his new address	Updated:Person(John Doe, Bigtown)	John Doe lives in Bigtown
April 1, 2001	John dies	Deleted:Person(John Doe)	There is no person called John Doe

Valid time is the time for which a fact is true in the real world. In the example above, the Person table gets two extra fields, Valid-From and Valid-To, specifying when a person's address was valid in the real world. On April 4, 1975 John's father proudly registered his son's birth. An official will then insert a new entry to the database stating that John lives in Smallville from April, 3rd. Notice that although the data was inserted on the 4th, the database states that the information is valid since the 3rd. The official does not yet know if or when John will ever move to a better place so in the database the Valid-To is filled with infinity (∞). Resulting in this entry in the database:

```
Person(John Doe, Smallville, 3-Apr-1975, ∞).
```

December 27, 1994 John reports his new address in Bigtown where he has been living since August 26, 1994. The Bigtown official does not change the address of the current entry of John Doe in the database. He adds a new one:

```
Person (John Doe, Big Town, 26-Aug-1994, ∞).
```

The original entry Person (John Doe, Smallville, 3-Apr-1975, ∞) is then updated (not removed!). Since it is now known that John stopped living in Smallville on August 26, 1994 the Valid-To entry can be filled in. The database now contains two entries for John Doe

```
Person(John Doe, Smallville, 3-Apr-1975, 26-Aug-1994).
```

```
Person(John Doe, Bigtown, 26-Aug-1994, ∞).
```

When John dies the database is once more updated. The current entry will be updated stating that John does not live in the Bigtown any longer. No new entry is being added because officials never report heaven as a new address. The database now looks like this

```
Person(John Doe, Smallville, 3-Apr-1975, 26-Aug-1994).
```

Person(John Doe, Bigtown, 26-Aug-1994, 1-Apr-2001).

References

[1] Richard T. Snodgrass and Ilsoo Ahn, "Temporal Databases," IEEE Computer 19(9), September, 1986, pp. 35–42.

Transaction time

Transaction time (TT), a concept originated by Richard T. Snodgrass and his doctoral student, is used in temporal databases. It denotes the time period during which a fact stored in the database is considered to be true; example. As of December 2011, ISO/IEC 9075, Database Language SQL:2011 Part 2: SQL/Foundation included clauses in table definitions to define "system-versioned tables" (that is, transaction-time tables).

In a database table transaction time is often represented by two extra table-columns *StartTT* and *EndTT*. The time interval is closed at its lower bound and open at its upper bound.

When the ending transaction time is unknown, it may be considered as "Until Closed". Academic researchers and some RDBMS have represented "Until Closed" with the largest timestamp supported or the keyword "forever". This convention is not technically precise.

References

Geospatial metadata

Geospatial metadata (also **geographic metadata**, or simply **metadata** when used in a geographic context) is a type of metadata that is applicable to objects that have an explicit or implicit geographic extent, in other words, are associated with some position on the surface of the Globe. Such objects may be stored in a geographic information system (GIS) or may simply be documents, datasets, images or other objects, services, or related items that exist in some other native environment but whose features may be appropriate to describe in a (geographic) metadata catalogue (may also be known as a data directory, data inventory, etc.).

Definition

ISO 19115 "**Geographic Information - Metadata**"[1] from ISO/TC 211, the current "best practice" standard for geospatial metadata, does not in fact provide a definition of geospatial (or geographic) metadata; however, uses the following wording in its "scope" section:

"*This International Standard provides information about the identification, the extent, the quality, the spatial and temporal schema, spatial reference, and distribution of digital geographic data.*"

A little further, it is stated: "*Though this International Standard is applicable to digital data, its principles can be extended to many other forms of geographic data such as maps, charts, and textual documents as well as non-geographic data.*"

The U.S. FGDC (Federal Geographic Data Committee) describes (geospatial) metadata as follows:

"*A metadata record is a file of information, usually presented as an XML document, which captures the basic characteristics of a data or information resource. It represents the who, what, when, where, why and how of the resource. Geospatial metadata are used to document geographic digital resources such as Geographic Information System (GIS) files, geospatial databases, and earth imagery. A geospatial metadata record includes core library catalog elements such as Title, Abstract, and Publication Data; geographic elements such as Geographic Extent and*

Projection Information; and database elements such as Attribute Label Definitions and Attribute Domain Values."[2]

History

The growing appreciation of the value of geospatial metadata through the 1980s and 1990s led to the development of a number of initiatives to collect metadata according to a variety of formats either within agencies, communities of practice, or countries/groups of countries. For example, NASA's "DIF" metadata format was developed during an Earth Science and Applications Data Systems Workshop in 1987,[3] and formally approved for adoption in 1988. Similarly, the U.S. FGDC developed its geospatial metadata standard over the period 1992-1994.[4] The Spatial Information Council of Australia and New Zealand (ANZLIC), a combined body representing spatial data interests in Australia and New Zealand, released version 1 of its "metadata guidelines" in 1996.[5] ISO/TC 211 undertook the task of harmonizing the range of formal and *de facto* standards over the approximate period 1999-2002, resulting in the release of ISO 19115 "**Geographic Information - Metadata**" in 2003. As of 2011[6] individual countries, communities of practice, agencies, etc. have started re-casting their previously-used metadata standards as "profiles" or recommended subsets of ISO 19115, optionally with the inclusion of additional metadata elements as formal extensions to the ISO standard. The growth in popularity of Internet technologies and data formats, such as Extensible Markup Language (XML), during the 1990s led to the development of mechanisms for exchanging geographic metadata on the Web. In 2004, the Open Geospatial Consortium released the current version (3.1) of Geography Markup Language (GML), an XML grammar for expressing geospatial features and corresponding metadata. With the growth of the Semantic Web in the 2000s, the geospatial community has begun to develop ontologies for representing semantic geospatial metadata. Some examples include the Hydrology and Administrative ontologies [7] developed by the Ordnance Survey in the United Kingdom.

ISO 19115: Geographic information - Metadata

ISO 19115 is a standard of the International Organization for Standardization (ISO).[8] The standard is part of the ISO geographic information suite of standards (19100 series). ISO 19115 and its parts defines how to describe geographical information and associated services, including contents, spatial-temporal purchases, data quality, access and rights to use.

The objective of this International Standard is to provide a clear procedure for the description of digital geographic datasets so that users will be able to determine whether the data in a holding will be of use to them and how to access the data. By establishing a common set of metadata terminology, definitions and extension procedures, this standard will promote the proper use and effective retrieval of geographic data.

ISO 19139 Geographic information Metadata XML schema implementation

ISO 19139[9] provides the XML implementation schema for ISO 19115 specifying the metadata record format and may be used to describe, validate, and exchange geospatial metadata prepared in XML.[10]

The standard is part of the ISO geographic information suite of standards (19100 series), and provides a spatial metadata XML (spatial metadata eXtensible Mark-up Language (smXML)) encoding, an XML schema implementation derived from ISO 19115, Geographic information – Metadata. The metadata includes information about the identification, constraint, extent, quality, spatial and temporal reference, distribution, lineage, and maintenance of the digital geographic dataset.

Metadata directories

Also known as metadata catalogues or data directories.

(need discussion of, and subsections on GCMD, FGDC metadata gateway, ASDD, European and Canadian initiatives, etc. etc.)

- GCMD [11] - Global Change Master Directory's goal is to enable users to locate and obtain access to Earth science data sets and services relevant to global change and Earth science research. The GCMD database holds more than 20,000 descriptions of Earth science data sets and services covering all aspects of Earth and environmental sciences.
- ECHO [12] - The EOS Clearing House (ECHO) is a spatial and temporal metadata registry, service registry, and order broker. It allows users to more efficiently search and access data and services through the Reverb Client [13] or Application Programmer Interfaces (APIs). ECHO stores metadata from a variety of science disciplines and domains, totalling over 3400 Earth science data sets and over 118 million granule records.
- GoGeo [14] - GoGeo is a service run by EDINA (University of Edinburgh) and is supported by Jisc. GoGeo allows users to conduct geographically targeted searches to discover geospatial datasets. GoGeo searches many data portals from the HE and FE community and beyond. GoGeo also allows users to create standards compliant metadata through its Geodoc metadata editor.

Geospatial metadata tools

There are many commercial GIS or geospatial products that support metadata viewing and editing on GIS resources. For example, ESRI's ArcGIS Desktop, SOCET GXP, Autodesk's AutoCAD Map 3D 2008 and Intergraph's GeoMedia support geospatial metadata extensively.

GeoNetwork opensource [15] is a comprehensive Free and Open Source Software solution to manage and publish geospatial metadata and services based on international metadata and catalog standards. The software is part of the Open Source Geospatial Foundation's software stack.

GeoCat Bridge [16] allows to edit, validate and directly publish metadata from ArcGIS Desktop to GeoNetwork [15] (and generic CSW catalogs) and publishes data as map services on GeoServer [17]. Several metadata profiles are supported.

CATMDEdit [18] terraCatalog ArcCatalog ArcGIS Server Portal GeoNetwork opensource [15] IME [19] M3CAT MetaD [20] MetaGenie [21] Parcs Canada Metadata Editor Mapit/CADit NOKIS Editor

References

[1] ISO 19115 Geographic Information - Metadata. International Organization for Standardization (ISO), Geneva, 2003

[2] "Geospatial Metadata" on FGDC website, visited 16 October 2006 (http://www.fgdc.gov/metadata)

[3] Gene Major and Lola Olsen: "A short history of the DIF". On GCMD website, visited 16 October 2006 (http://gcmd.nasa.gov/User/difguide/whatisadif.html)

[4] MIT Libraries Guide: "Federal Geographic Data Committee (FGDC) Metadata". On MIT Libraries website, visited 16 October 2006 (http://libraries.mit.edu/guides/subjects/metadata/standards/fgdc.html)

[5] ANZLIC Metadata Guidelines: Core metadata elements for geographic data in Australia and New Zealand, Version 2 (February 2001) (http://www.anzlic.org.au/get/2358011755)

[6] http://en.wikipedia.org/w/index.php?title=Geospatial_metadata&action=edit

[7] http://www.ordnancesurvey.co.uk/oswebsite/ontology/

[8] ISO 19115 Geographic Information - Metadata. International Organization for Standardization (ISO), Geneva, 2003

[9] ISO 19139 Geographic information Metadata XML schema implementation. International Organization for Standardization (ISO), Geneva, 2003

[10] "ISO 19139 Geographic information Metadata XML schema implementation" (http://marinemetadata.org/references/iso19139), Marine Metadata Interoperability Project

[11] http://gcmd.nasa.gov

[12] http://earthdata.nasa.gov/echo

[13] http://reverb.earthdata.nasa.gov/echo
[14] http://www.gogeo.ac.uk/gogeo/
[15] http://geonetwork-opensource.org
[16] http://geocat.net/bridge
[17] http://geoserver.org
[18] http://catmdedit.sourceforge.net/
[19] http://www.conterra.de/en/products/sdi/terracatalog/index.shtm
[20] http://www.intelec.ca/html/en/technologies/m3cat.html
[21] http://www.gigateway.org.uk/metadata/metagenie.html

ANZLIC Metadata Profile Version 1.0 (viewed 17 Sept. 2007) (http://www.anzlic.org.au/get/2440920572.PDF)

External links

- FGDC metadata page (http://www.fgdc.gov/metadata)
- Global Change Master Directory(GCMD) (http://gcmd.nasa.gov/)
- Geospatial Exploitation of Motion Imagery (http://wiki.milcord.com/wiki/Geospatial_Exploitation_of_Motion_Imagery) is a geospatially aware and integrated Intelligent Video Surveillance (IVS) software system targeted at real-time and forensic video analytic and mining applications that require low-resolution detection, tracking, and classification of moving objects (people and vehicles) in outdoor, wide-area scenes.
- ISO 19115:2003 Geographic information -- Metadata (http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=26020)
- Geographic information -- Metadata -- XML schema implementation (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32557)
- EarthDataModels design for Metadata (http://www.earthdatamodels.org/designs/metadata_BGS.html) is a logical data model and physical implementation of a Spatial Metadata Database, based on ISO19115 and is INSPIRE compliant.

Geographical database

A **geographical database** is a database of geographic data, such as countries, administrative divisions, cities, and related information. Such databases can be useful for websites that wish to identify the locations of their visitors for customization purposes.

External links

- GeoNames [1]
- Gazetiki [2]
- GeoDataSource [3]

References

[1] http://geonames.org
[2] http://georama-project.labs.exalead.com/gazetiki.htm
[3] http://geodatasource.com

Time series database

A **time series database server (TSDS)** is a software system that is optimized for handling time series data, arrays of numbers indexed by time (a datetime or a datetime range). In some fields these *time series* are called profiles, curves, or traces. A time series of stock prices might be called a price curve. A time series of energy consumption might be called a load profile. A log of temperature values over time might be called a temperature trace. Despite the disparate names, many of the same mathematical operations, queries, or database transactions are useful for analysing all of them. And the implementation of a database that can correctly, reliably, and efficiently implement these operations must be specialized for time-series data. Despite this specialized treatment, TSDSs are nonetheless subject to Brewer's Theorem[1] which states that CAP (Consistency, Availability, Partitionability) is a spectrum of capabilities and not a single achievable requirement, at least for large transaction bandwidth and large data sets.[2]

TSDSs are databases that are optimized for time series data. Software with complex logic or business rules and high transaction volume for time series data are not practical with the alternative to TSDS, relational database management systems. Flat file databases are not a viable option either, if the data and transaction volume reaches a maximum threshold determined by the capacity of individual servers (processing power and storage capacity). Queries for historical data, replete with time ranges and roll ups and arbitrary time zone conversions are difficult in a relational database. Compositions of those rules are even more difficult. This is a problem compounded by the free nature of relational systems themselves. Many relational systems are often not modelled correctly with respect to time series data. TSDS on the other hand impose a model and this allows them to provide more features for doing so.

Ideally, these repositories are often natively implemented using specialized database algorithms. However, it is possible to store time series as binary large objects (BLOBs) in a *relational database* or by using a *VLDB* approach coupled with a pure *star schema*. Efficiency is often improved if time is treated as a discrete quantity rather than as a continuous mathematical dimension. Database joins across multiple time series data sets is only practical when the time tag associated with each data entry spans the same set of discrete times for all data sets across which the join is performed.[3]

Overview

The TSDS allows users to create, enumerate, update and destroy various time series and organize them in some fashion. These series may be organized hierarchically and optionally have companion metadata available with them. The server often supports a number of basic calculations that work on a series as a whole, such as multiplying, adding, or otherwise combining various time series into a new time series. They can also filter on arbitrary patterns defined by the day of the week, low value filters, high value filters, or even have the values of one series filter another. Some TSDSs also build in a wealth of statistical functions.

For example, consider the following hypothetical "time series" or "profile" expression:

```
select nymex/gold_price * nymex/gold_volume
```

To analyze this, the TSDS would join the two series nymex/gold_price and nymex/gold_volume based on the overlapping areas of time for each, multiply the values where they intersect, and then output a single composite time series.

Obviously, more complex expressions are allowed. TSDSs often allow users to manage a repository of filters or masks that specify in some way a pattern based on the day of a week and a set of holidays. In this way, one can readily assemble time series data. Assuming such a filter exists, one might hypothetically write

```
select onpeak( cellphoneusage )
```

which would extract out the time series of cellphoneusage that only intersects that of 'onpeak'. Some systems might generalize the filter to be a time series itself.

This syntactical simplicity drives the appeal of the TSDS. For example, a simple utility bill might be implemented using a query such as:

```
select max( onpeak( powerusagekw ) ) * demand_charge;

select sum( onpeak( powerusagekwh ) ) * energy_charge;
```

TSDS also generally have conversions to and from specific time zones implemented at the server level.

Example

A workable implementation of a time series database can be easily deployed in a conventional SQL-based relational database provided that the database software supports both binary large objects (BLOBs) and user-defined functions. SQL statements that operate on one or more time series quantities on the same row of a table or join can easily be written, as the user-defined time series functions operate comfortably inside of a SELECT statement. However, time series functionality such as a SUM function operating in the context of a GROUP BY clause cannot be easily achieved.

Specialized database systems designed for TSDS are often dubbed *NoSQL*, because of their break from SQL, the language of relational database management system queries. Some example database software packages optimized for dealing with large volumes of time series data include:

- Apache Accumulo
- Apache Cassandra
- CouchDB
- Couchbase Server
- FAME
- HBase (Hadoop)
- InfluxDB [4]

- IBM Informix
- OpenTSDB [5]
- SiteWhere [6]
- Riak
- Redis
- tsdb[7]
- TempoDB [8]
- Treasure Data [9]

References

[1] http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf
[2] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.1495&rep=rep1&type=pdf
[3] http://www.julianbrowne.com/article/viewer/brewers-cap-theorem
[4] http://influxdb.org/
[5] http://www.opentsdb.net/
[6] http://www.sitewhere.org/
[7] tsdb: A Compressed Database for Time Series (http://link.springer.com/chapter/10.1007/978-3-642-28534-9_16) Luca Deri, Simone Mainardi, Francesco Fusco: *Proceedings of the 4th International Workshop on Traffic Monitoring and Analysis (TMA)*, Wien, 2012
[8] http://tempo-db.com/
[9] http://www.treasure-data.com/

Operational historian

Operational Historian refers to a database software application that logs or historizes time-based process data. Historian software is used to record trends and historical information about industrial processes for future reference. It captures plant management information about production status, performance monitoring, quality assurance, tracking and genealogy, and product delivery with enhanced data capture, data compression, and data presentation capabilities.

Operational Historians are like Enterprise Historians but differ in that they are used by engineers on the plant floor rather than by business processes. They are typically cheaper, lighter weight, and easier to use and reconfigure than enterprise historians. Having an operational historian enables "at the source" analysis of the historical data that is not typically possible with enterprise historians.

Typically, these applications offer two layers of data access: through a dedicated SDK (sometimes in two different flavours: full administration API and high-speed read/write API), as well as user front-end tools (for instance, administration panels, engineering consoles or portal-like web clients).

Due to the fact that these applications are designed to fulfil specific operation time requirements, their marketing materials often indicate that these are real-time database systems.[1] However, since such performance measurements are often executed for atomic operations (especially write operations), not necessarily whole transactions, not all of the operational historians must be in fact real-time databases.

Usual challenges the operational historians must address are as follows:

- data collection from real-time external systems,
- storage and archiving of very large volumes of data,
- tag organisation (typically time series, where a single sample contains the information about the time stamp, the value and the sample quality),
- basic data limit monitoring (alarms) and user prompts (messages),
- performance of read/write operations.

Data Access

As opposed to enterprise historians, the data access layer in the operational historians is designed to offer sophisticated data fetching modes without complex information analysis facilities. The following settings are typically available for data access operations:

- Data scope (single point, history based on time range, history based on sample count),
- Request modes (raw data, last-known value, aggregation, interpolation),
- Sampling (single point, all points without sampling, all points with interval sampling),
- Data omission (based on the sample quality, based on the sample value, based on the count).

Even though the operational historians are rarely relational database management systems, they often offer SQL-based interface to query the database. In most of such implementations, the dialect does not follow SQL standard in order to provide syntax for specifying data access operations parameters.

Implementations

Such systems are available from e.g.:

- Aspen Technology InfoPlus.21
- Automsoft rapidHistorian
- Canary Enterprise Historian
- Capstone Technology dataPARC
- Datacenter Dynamysk [2]
- GE Proficy Historian
- Honeywell PHD
- Inductive Automation Ignition SCADA
- Inductive Automation FactorySQL (depricated January 2010)
- Instep eDNA
- m:pro IT Consult
- Yokogawa Exaquantum Historian[3]
- National Instruments LabView DSC Citadel
- OSIsoft - PI
- Parsec Automation - HISTORITrak
- Polyhedra DBMS
- Prediktor OPC Historian
- Wonderware Historian [4]
- DotSystems PulsarTSDB
- OpenTSDB
- Iconics Hyper Historian [5]

References

[1] Wonderware Historian - Example of naming the operational historian the real-time database (http://global.wonderware.com/EN/Pages/WonderwareHistorian.aspx)
[2] http://www.Dynamysk.com
[3] http://www.yokogawa.com/eu/pims/eu-exaquantum_ser.htm
[4] http://software.invensys.com/products/wonderware/production-information-management/historian/
[5] http://www.iconics.com/Home/Products/Historians/Hyper-Historian.aspx

External links

- AutomationWorld.com (http://www.automationworld.com/primers-3388)
- Automsoft rapidHistorian (http://www.automsoft.com/solutions/rapidhistorian)
- Nimbits Temporal Database Server (http://www.nimbits.com)
- Honeywell PHD (http://hpsweb.honeywell.com/Cultures/en-US/Products/BusinessApplications/InformationManagement/UniformancePHD/default.htm)
- Capstone Technology (http://www.capstonetechnology.com)
- m:pro IT Consult (http://www.mpro-it.com)
- DotSystems PulsarTSDB (http://www.dotsystems.pl/products/pulsartsdbsm.html)
- OpenTSDB (http://opentsdb.net/)
- Wonderware Production Information Management Solutions (http://software.invensys.com/products/wonderware/production-information-management)

Special Databases

Real-time database

A **real-time database** is a database system which uses real-time processing to handle workloads whose state is constantly changing.[1] This differs from traditional databases containing persistent data, mostly unaffected by time. For example, a stock market changes very rapidly and is dynamic. The graphs of the different markets appear to be very unstable and yet a database has to keep track of current values for all of the markets of the New York Stock Exchange.[2] Real-time processing means that a transaction is processed fast enough for the result to come back and be acted on right away.[3] Real-time databases are useful for accounting, banking, law, medical records, multi-media, process control, reservation systems, and scientific data analysis.[4]

Overview

Real-time databases are traditional databases that use an extension to give the additional power to yield reliable responses. They use timing constraints that represent a certain range of values for which the data are valid. This range is called temporal validity. A conventional database cannot work under these circumstances because the inconsistencies between the real world objects and the data that represents them are too severe for simple modifications. An effective system needs to be able to handle time-sensitive queries, return only temporally valid data, and support priority scheduling. To enter the data in the records, often a sensor or an input device monitors the state of the physical system and updates the database with new information to reflect the physical system more accurately. [5] When designing a real-time database system, one should consider how to represent valid time, how facts are associated with real-time system. Also, consider how to represent attribute values in the database so that process transactions and data consistency have no violations.

When designing a system, it is important to consider what the system should do when deadlines are not met. For example, an air-traffic control system constantly monitors hundreds of aircraft and makes decisions about incoming flight paths and determines the order in which aircraft should land based on data such as fuel, altitude, and speed. If any of this information is late, the result could be devastating. To address issues of obsolete data, the timestamp can support transactions by providing clear time references.

Preserving data consistency

Although the real-time database system may seem like a simple system, problems arise during overload when two or more database transactions require access to the same portion of the database. A transaction is usually the result of an execution of a program that accesses or changes the contents of a database. [6] A transaction is different from a stream because a stream only allows read-only operations, and transactions can do both read and write operations. This means in a stream, multiple users can read from the same piece of data, but they cannot both modify it. [7] A database must let only one transaction operate at a time to preserve data consistency. For example, if two students demand to take the remaining spot for a section of a class and they hit submit at the same time, only one student should be able to register for it. [8]

Real-time databases can process these requests utilizing scheduling algorithms for concurrency control, prioritizing both students' requests in some way. Throughout this article, we assume that the system has a single processor, a disk based database, and a main memory pool.[9]

In real-time databases, deadlines are formed and different kinds of systems respond differently to data that does not meet its deadline. In a real-time system, each transaction uses a timestamp to schedule the transactions. [10] A priority mapper unit assigns a level of importance to each transaction upon its arrival in the database system that is dependent on how the system views times and other priorities. The timestamp method on relies on the arrival time in the system. Researchers indicate that for most studies, transactions are sporadic with unpredictable arrival times. For example, the system gives an earlier request deadline to a higher priority and a later deadline to a lower priority.[11] Below is a comparison of different scheduling algorithms.

Earliest Deadline

PT = DT — The value of a transaction is not important. An example is a group of people calling to order a product.

Highest Value

PT = 1/VT — The deadline is not important. Some transactions should get to CPU based on criticalness, not fairness. This is an example of least slack that can wait the least amount of time. If the telephone switchboards were overloaded, people who call 911 should get priority.[12]

Value inflated deadline

PT = DT/VT — Gives equal weight to deadline and values based on scheduling. An example is registering for classes where the student selects a block of classes that he wishes to take and presses submit. In this scenario, higher priorities often take up precedence. A school registration system probably uses this technique when the server receives two registration transactions. If one student had 22 credits and the other had 100 credits, the person with 100 credits would take priority (Value based scheduling).

Timing constraints and deadlines

A system that correctly perceives the serialization and timing constraints associated with transactions with soft or firm deadlines, takes advantage of absolute consistency.[13] Another way of making sure that data is absolute is using relative constraints. Relative constraints ensure transactions enter into the system at the same time as the rest of the group that the data transaction is associated with. Using the mechanisms of absolute and relative constraints greatly ensures the accuracy of data.

An additional way of dealing with conflict resolution in a real-time database system besides deadlines is a wait policy method. This process helps ensure the latest information in time critical systems. The policy avoids conflict by asking all non-requesting blocks to wait until the most essential block of data is processed.[14] While studies in labs have found that data-deadline based policies do not improve performance significantly, the forced wait policy can improve performance by 50 percent.[15] The forced wait policy may involve waiting for higher priority transactions to process in order to prevent deadlock. Another example of when data can be delayed is when a block of data is about to expire. The forced wait policy delays processing until the data is updated using new input data. The latter method helps increase the accuracy of the system and can cut down on the number of necessary processes that are aborted. [16] Generally relying on wait policies is a not optimal.[17]

It is necessary to discuss the formation of deadlines. Deadlines are the constraints for soon-to-be replaced data accessed by the transaction. Deadlines can be either observant or predictive.[18] In an observant deadline system, all unfinished transactions are examined and the processor determines whether any had met its deadline.[19] Problems arise in this method because of variations caused by seek time variations, buffer management and page faults (An Overview of Real-Time Database Systems). A more stable way of organizing deadlines is the predictive method. It builds a candidate schedule and determines if a transaction would miss its deadline under the schedule.[20]

The type of response to a missed deadline depends on whether the deadline is hard, soft, or firm. Hard deadlines require that each data packet reach its destination before the packet has expired and if not, the process could be lost, causing a possible problem. Problems like these are not very common because omnipotence of the system is required

before assigning deadlines to determine worst case. This is very hard to do and if something unexpected happens to the system such as a minute hardware glitch, it could throw the data off. For soft or firm deadlines, missing a deadline can lead to a degraded performance but not a catastrophe.[21] A soft deadline meets as many deadlines as possible. However, no guarantee exists that the system can meet all deadlines. Should a transaction miss its deadline, the system has more flexibility and the transaction may increase in importance. Below is a description of these responses:

Hard deadline: If not meeting deadlines creates problems, a hard deadline is best. It is periodic, meaning that it enters the database on a regular rhythmic pattern. An example is data gathered by a sensor. These are often used in life critical systems.[22]

Firm deadline: Firm deadlines appear to be similar to hard deadlines yet they differ from hard deadlines because firm deadlines measure how important it is to complete the transaction at some point after the transaction arrives. Sometimes completing a transaction after its deadline has expired may be harmful or not helpful, and both the firm and hard deadlines consider this. An example of a firm deadline is an autopilot system.[23]

Soft deadline: If meeting time constrains is desirable but missing deadlines do not cause serious damage, a soft deadline may be best. It operates on an aperiodic or irregular schedule. In fact, the arrival of each time for each task is unknown. An example is an operator switchboard for a telephone.[24]

Hard deadline processes abort transactions that have passed the deadline, improving the system by cleaning out clutter that needs to be processed. Processes can clear out not only the transactions with expired deadlines but also transactions with the longest deadlines, assuming that once they reach the processor they would be obsolete. This means other transactions should be of higher priority. In addition, a system can remove the least critical transactions. When I was pre-selecting classes on during a high traffic period, a field in the database can become so busy with registration requests that it was unavailable for a while and the result of my transaction was a display of the SQL query sent and a message that said that the data is currently unavailable. This error is caused by the checker, a mechanism that checks the condition of the rules, and the rule that occurred before it. [25]

The goal of scheduling periods and deadlines is to update transactions guaranteed to complete before their deadline in such a way that the workload is minimal. With large real-time databases, buffering functions can help improve performance tremendously. A buffer is part of the database that is stored in main memory to reduce transaction response time. In order to reduce disk input and output transactions, a certain number of buffers should be allocated. [26] Sometimes multiversions are stored in buffers when the data block the transaction needs is currently in use. Later, the database has the data appended to it. Different strategies allocate buffers and must balance between taking an excessive amount of memory and having everything in one buffer that it has to search for. The goal is to eliminate search time and distribute the resources between buffer frames in order to access data quickly. A buffer manager is capable of allocating more memory, if necessary, to improve response time. The buffer manager can even determine whether a transaction that it has should advance. Buffering can improve speed in real-time systems.[27]

Examples

Following is a list of real time database applications and architecture schemes produced and used by both universities and corporations. [28]

Eaglespeed-RTDB

Clustra

Timesten

Empress

eXtremeDB

SolidDB

RODAIN (Real-Time Object-Oriented Database Architecture for Intelligent Networks)

M2RTSS

RT-CARAT
REACH (Real-time Active and Heterogeneous Mediator System Project)
STRIP (Stanford Real-time Information Processor)

Future database systems

Traditional databases are persistent but are incapable of dealing with dynamic data that constantly changes. Therefore, another system is needed. Real-time databases may be modified to improve accuracy and efficiency and to avoid conflict, by providing deadlines and wait periods to insure temporal consistency. Real-time database systems offer a way of monitoring a physical system and representing it in data streams to a database. A data stream, like memory, fades over time. In order to guarantee that the freshest and most accurate information is recorded there are a number of ways of checking transactions to make sure they are executed in the proper order. An online auction house provides an example of a rapidly changing database.

Now database systems are faster than they were in the past. In the future, we can look forward to even faster database systems. Although we have faster systems now, an effort to reduce misses and tardy times will still be beneficial. The ability to process results in a timely and predictable manner will always be more important than fast processing. Fast processing that is misapplied is not helpful for real-time database systems. Transactions that run faster still sometimes block in such a way that they have to be aborted and restarted. In fact, faster processing hurts some real-time applications because increased speed brings more complexity and more of a chance for problems caused by a variance of speed. Faster processing makes it harder to determine which deadlines have been met successfully. [29] With future database systems running even faster than ever, there is a need to do more studies so we can continue to have efficient systems. [30]

The amount of research studying real-time database systems will increase because of commercial applications such as web based auction houses like e-bay. More developing countries are expanding their phone systems, and the number of people with cell phones in the United States as well as other places in the world continues to grow. Also likely to spur real-time research is the exponentially increasing speed of the microprocessor. This also enables new technologies such as web-video conferencing and instant messenger conversations in sound and high-resolution video, which are reliant on real-time database systems. Studies of temporal consistency result in new protocols and timing constraints with the goal of handling real-time transactions more effectively.[31]

References

[1] (Buchmann)
[2] (Kanitkar)
[3] (Capron)
[4] (Snodgrass)
[5] (Abbot)
[6] (Singhal)
[7] (Abbot)
[8] (Abbot)
[9] (Haritsa)
[10] (Abbot)
[11] (Haritsa)
[12] (Snodgrass)
[13] (Lee)
[14] (Abbot)
[15] (Porkka)
[16] (Kang)
[17] (Kang)
[18] (Kang)
[19] (Abbot)
[20] (Abbot)

[21] (Haritsa)
[22] (Stankovic)
[23] (Snodgrass)
[24] (Stankovic)
[25] (Ramamritham)
[26] (O'Neil)
[27] (O'Neil)
[28] (Lindstrom)
[29] (Lam)
[30] (Lam)
[31] (Haritsa)

- Abbot, Robert K., and Hector Garcia-Molina. Scheduling Real-Time Transactions: a Performance Evaluation. Stanford University and Digital Equipment Corp. ACM, 1992. 13 Dec. 2006 <http://delivery.acm.org/10.1145/140000/132276/p513-abbott.pdf?key1=132276&key2=5193406611&coll=portal&dl=ACM&CF
- Buchmann, A. "Real Time Database Systems." Encyclopedia of Database Technologies and Applications. Ed. Laura C. Rivero, Jorge H. Doorn, and Viviana E. Ferraggine. Idea Group, 2005.
- Carpron, H.L., J. A. Johnson. Computers: Tools for the Information Age. Prentice Hall, 1998. 5th ed.
- Haritsa, J., J. Stankovic, and M Xiong. A State-Conscious Concurrency Control Protocol for Replicated Real-Time Databases. University of Virginia. IEEE Real-Time Applications Symposium. 13 Dec. 2006 <http://www.cs.virginia.edu/~stankovic/mmdb.html>.
- Kang, K D., S Son, and J Stankovic. Specifying and Managing Quality of Real-Time Data Services. University of Virginia. IEEE TKDE, 2004.
- Kanitkar, Vinay, and Alex Delis. A Case for Real-Time Client-Server Databases. Polytechnic University. Brooklyn, New York, 1997. 13 Dec. 2006 <http://citeseer.ist.psu.edu/kanitkar97case.html>.
- Kao, Ben, and Hector Garcia-Molina. An Overview of Real-Time Database Systems. NATO Advanced Study Institute on Real-Time Computing, 9 Oct. 1992, NATO. 13 Dec. 2006 <http://dbpubs.stanford.edu/pub/1993-6>.
- Lam, Kam-Yiu, and Tei-Wei Kuo. Real-Time Database Systems: Architecture and Techniques. Springer, 2001.
- Lee, Juhnyoung. Concurrency Control Algorithms for Real-Time Database Systems. Diss. Univ. of Virginia, 1994. 13 Dec. 2006 <http://citeseer.ist.psu.edu/lee94concurrency.html>.
- Lindstrom, Jan. Real Time Database Systems. Solid, 2008. March 25, 2008 <http://www.cs.helsinki.fi/u/jplindst/papers/rtds.pdf>
- Ozsoyoglu, Gultekin, and Richard T. Snodgrass. Temporal and Real-Time Databases: a Survey. Knowledge and Data Engineering, 1995. 13 Dec. 2006 <http://citeseer.ist.psu.edu/ozsoyoglu95temporal.html>.
- Singhal, Mukesh. Approaches to Design of Real-Time Database Systems, SIGMOD Record, volume 17, no. 1, March 1988
- Sivasankaran, Rajendran M., John A. Stankovic, Don Towsley, Bhaskar Purimetla, and Krithi Ramamaritham. Priority Assignment in Real-Time Active Databases. University of Massachusetts. Amaherst, NY, 1996. 13 Dec. 2006 <http://citeseer.ist.psu.edu/317150.html>.
- Stankovic, John A., Marco Spuri, Krithi Ramamritham, and Giorgio C. Buttazzo. Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms. Springer, 1998.
- Stonebraker, Michael, et al. HStore: A High Performance, Distributed Main Memory Transaction Processing System, 2008. <http://cs-www.cs.yale.edu/homes/dna/papers/hstore-demo.pdf>.

In-memory database

An **in-memory database** (**IMDB**; also **main memory database system** or **MMDB** or **memory resident database**) is a database management system that primarily relies on main memory for computer data storage. It is contrasted with database management systems that employ a disk storage mechanism. Main memory databases are faster than disk-optimized databases since the internal optimization algorithms are simpler and execute fewer CPU instructions. Accessing data in memory eliminates seek time when querying the data, which provides faster and more predictable performance than disk.

In applications where response time is critical, such as telecommunications network equipment and mobile advertising networks, main memory databases are often used. IMDBs have gained a lot of traction, especially in the data analytics space, starting mid-2000s mainly due to cheaper RAM.

With the introduction of NVDIMM technology, in-memory databases will now be able to run at full speed and maintain data in the event of power failure.[1][2][3]

Storage memory

Another variation is to have large amounts of nonvolatile memory in the server. For example Flash memory chips as addressable memory rather than structured as disk arrays. A database in this form of memory combines very fast access speed with persistence over reboots and power losses.[4]

Products

Name	Developer	License	Description/Notes
ActiveSpaces	TIBCO Software	Proprietary with developer download	For Java/.Net./C, distributed, hybrid, event enabled, NewSQL
ALTIBASE HDB	Altibase Corporation	Proprietary	"Hybrid DBMS" that combines an in-memory database with a conventional disk-resident database
BigMemory	Terracotta, Inc. (Software AG)	Proprietary (free editions)	
Datablitz (formerly Dali)	Bell Labs (Alcatel-Lucent)	Proprietary	Dali prototype was developed for in-house Bell Labs needs
DB2 BLU	IBM	Proprietary	IBM DB2 for Linux, UNIX and Windows supports Dynamic In-memory (in-memory columnar processing) Parallel Vector Processing, Actionable Compression, and Data Skipping technologies, collectively called IBM BLU Acceleration by IBM. Released in June 2013.
Ehcache	Terracotta, Inc. (Software AG)	Open source (Apache License)	For Java, distributed
EXASolution	EXASOL	Proprietary	Relational (SQL over ODBC, JDBC, or ADO.NET), multi-dimensional (MDX over ODBO or XMLA); EXASOL leads the well-respected international TPC-H benchmark,[5] since 2008, in the datavolume-based categories 100GB, 300GB, 1TB, 3TB, and 10TB; EXASolution architecture: shared-nothing, column-store, in-memory database
eXtremeDB	McObject	Proprietary	Cross-platform, including Linux, Windows, RTOS and server platforms. Interfaces include type-safe, native C/C++; native Java & .NET; SQL/ODBC/JDBC. Specialized editions for clustering, high availability, 64-bit support, hybrid (in-memory and persistent) storage, and more. eXtremeDB Financial Edition implements columnar data handling, vector-based statistical function library, integrated performance monitoring.

H2 (DBMS)	H2	Mozilla Public License or Eclipse Public License	For Java
Hazelcast	Hazelcast Team	Open source (Apache License 2.0)	For Java, NoSQL, distributed
Kognitio Analytical Platform	Kognitio, Limited	Proprietary	Development of an in-memory database, specialized for analytical workloads was started at White Cross Systems, Limited in 1988. The first beta release of that system was in 1989. It was based on the INMOS Transputer. The first full production release was offered in 1992. White Cross merged in 2005 with Kognitio, Limited in the United Kingdom and is currently marketing version 8 of the same code base as the "Kognitio Analytical Platform".
MemSQL	MemSQL, Inc.	Proprietary	SQL relational
Microsoft COM+ IMDB	Microsoft Corporation	Proprietary	Defunct
Microsoft SQL Server	Microsoft	Proprietary	SQL Server 2012 contains an in-memory technology called xVelocity column store indexes targeted for data warehouse workload. The recently announced SQL Server 2014 will contain a in-memory technology with the code name Hekaton targeted for OLTP type workloads.
Oracle Coherence	Oracle Corporation	Proprietary with developer download	For Java, relational, distributed
Oracle Exalytics	Oracle Corporation	Proprietary	Appliance
Polyhedra	ENEA AB (previously Perihelion Software)	Proprietary, with a free to use edition (Polyhedra Lite)	Relational (SQL, ODBC, JDBC) in-memory database system originally developed for use in SCADA and embedded systems, but used in a variety of other applications including financial systems. Supports data durability via snapshots and journal logging, and high availability via a hot-standby. First release was in 1993; 8.7 released in March 2013. Polyhedra Lite was released under a free-to-use license in 2012.
SAP HANA	SAP AG	Proprietary	SAP HANA, short for 'High Performance Analytic Appliance' is an in-memory, column-oriented, relational database management system written in C, C++
solidDB	IBM	Proprietary	Relational
SQLFire	VMware	Proprietary	Relational, distributed, NewSQL
TimesTen	Oracle Corporation	Proprietary	Standalone database or in-memory cache for Oracle Database
UnQLite Embedded Database	Symisc Systems	BSD, SPL	UnQLite has support for in-memory databases as well on-disk databases using the same API with pluggable run-time interchangeable storage engines (B+tree, Hash, etc.)
VoltDB	VoltDB Inc.	Open source (GPL) / Proprietary	Relational; implements H-Store design
WebDNA	WebDNA Software Corporation	Freeware	Robust hybrid in-memory database system and scripting language designed for the World Wide Web. Resilience is provided by both in-memory and on-disk tables in a single database.
Xeround	Xeround Inc.	Proprietary/Not for sale, service only	Cloud database

Many DBMS support in-memory-only storage engines, including:

- MySQL
- Adaptive Server Enterprise
- Raima

References

[1] Whole-system Persistence with Non-volatile Memories http://research.microsoft.com/apps/pubs/default.aspx?id=160853
[2] The Bleak Future of NAND Flash Memory http://research.microsoft.com/apps/pubs/default.aspx?id=162804
[3] Using NVDIMM as Storage, In-Memory Database Gains Durability & Keeps High Performance http://low-latency.com/article/using-nvdimm-storage-memory-database-gains-durability-keeps-high-performance
[4] "Truly these are the GOLDEN YEARS of Storage." (http://www.theregister.co.uk/2013/01/30/storage_glory_days/)
[5] TPC-H - Homepage (http://www.tpc.org/tpch/default.asp). Tpc.org (2010-04-21). Retrieved on 2013-09-18.

- Jack Belzer. Encyclopedia of Computer Science and Technology - *Volume 14: Very Large Data Base Systems to Zero-Memory and Markov Information Source*. Marcel Dekker Inc. ISBN 0-8247-2214-0.

External links

- In-Memory Database Systems Questions and Answers (http://www.mcobject.com/in_memory_database)
- The Rebirth of the In-Memory Database (http://www.benstopford.com/2011/08/14/distributed-storage-phase-change-memory-and-the-rebirth-of-the-in-memory-database/)

Probabilistic database

Most real databases contain data whose correctness is uncertain. In order to work with such data, there is a need to quantify the integrity of the data. This is achieved by using probabilistic databases.

A **probabilistic database** is an uncertain database in which the possible worlds have associated probabilities. Probabilistic database management systems are currently an active area of research. "While there are currently no commercial probabilistic database systems, several research prototypes exist..."[1]

Probabilistic databases distinguish between the logical data model and the physical representation of the data much like relational databases do in the ANSI-SPARC Architecture. In probabilistic databases this is even more crucial since such databases have to represent very large numbers of possible worlds, often exponential in the size of one world (a classical database), succinctly.[2][3]

Terminology

In a probabilistic database, each data item - relation, tuple and value that an attribute can take - is associated with a probability $\in (0,1]$, with 0 representing that the data is certainly incorrect, and 1 representing that it is certainly correct.

Possible Worlds

A probabilistic database could exist in multiple states. For example, if we are uncertain about the existence of a tuple in the database, then the database could be in two different states with respect to that tuple - the first state contains the tuple, while the second one does not. Similarly, if an attribute can take one of the values x, y or z, then the database can be in three different states with respect to that attribute.

Each of these *states* is called a possible world.

Consider the following database:

An Incomplete Database

A	B
a1	b1
a2	b2
a3	{b3,b3′,b3″}

(Here *{b3,b3′,b3″}* denotes that the attribute can take any of the values *b3*,*b3′* or *b3″*)

- Let us assume that we are uncertain about the first tuple, certain about the second tuple and uncertain about the value of attribute **B** in the third tuple.

Then the actual state of the database may or may not contain the first tuple (depending on whether it is correct or not). Similarly, the value of the attribute **B** may be *b3*,*b3′* or *b3″*.

Consequently, the possible worlds corresponding to the database are as follows:

World 1

A	B
a1	b1
a2	b2
a3	b3

World 2

A	B
a1	b1
a2	b2
a3	b3′

World 3

A	B
a1	b1
a2	b2
a3	b3″

World 4

A	B
a2	b2
a3	b3

World 5

A	B
a2	b2
a3	b3′

World 6

A	B
a2	b2
a3	b3″

Types of Uncertainties

There are essentially two kinds of uncertainties that could exist in a probabilistic database, as described in the table below:

Types of Uncertainties

Tuple-level uncertainty	Attribute-level uncertainty
Here, we are not sure if a tuple is correct or not, that is, whether it should exist in the database or not.	Here, we are not sure about the values that an attribute of a tuple can take, that is, it could take one of the several possible values.
Corresponding to each uncertain tuple, there are two possible worlds: one which includes the tuple while the other which does not.	Corresponding to each uncertain attribute which can take one of the values $a_1,...,a_n$, there are n possible worlds.
Tuple-level uncertainty can be seen as a boolean random variable associated with each uncertain tuple.	Attribute-level uncertainty can be seen as a random variable associated with each uncertain attribute which can take values $a_1,...,a_n$.

By assigning values to random variables associated with the data items, we can represent different possible worlds.

References

[1] Vinod Muthusamy, Haifeng Liu, Hans-Arno Jacobsen: Predictive Publish/Subscribe Matching. (http://www.eecg.toronto.edu/~jacobsen/ptopss.pdf) University of Toronto.

[2] Nilesh N. Dalvi, Dan Suciu: Efficient query evaluation on probabilistic databases. VLDB J. 16(4): 523-544 (2007)

[3] Lyublena Antova, Christoph Koch, Dan Olteanu: 10^(10^6) Worlds and Beyond: Efficient Representation and Processing of Incomplete Information. ICDE 2007: 606-615

External links

- The MayBMS (http://www.cs.cornell.edu/database-OLD/maybms/) project at Cornell University (sourceforge.net project site (http://maybms.sourceforge.net))
- The MystiQ (http://www.cs.washington.edu/homes/suciu/project-mystiq.html) project at the University of Washington

- The Orion (http://orion.cs.purdue.edu/) project at Purdue University
- The Trio (http://infolab.stanford.edu/trio/) project at Stanford University
- The BayesStore (http://www.eecs.berkeley.edu/Research/Projects/Data/102060.html/) project at the University of California, Berkeley
- The PrDB (http://www.cs.umd.edu/~amol/PrDB/) project at the University of Maryland, College Park

Deductive database

A **Deductive database** is a database system that can make deductions (i.e., conclude additional facts) based on rules and facts stored in the (deductive) database. Datalog is the language typically used to specify facts, rules and queries in deductive databases. Deductive databases have grown out of the desire to combine logic programming with relational databases to construct systems that support a powerful formalism and are still fast and able to deal with very large datasets. Deductive databases are more expressive than relational databases but less expressive than logic programming systems. In recent years, deductive databases such as Datalog have found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.[1]

Deductive databases and logic programming

Deductive databases reuse a large number of concepts from logic programming; rules and facts specified in the deductive database language Datalog look very similar to those in Prolog. However important differences between deductive databases and logic programming:

- Order sensitivity and procedurality: In Prolog, program execution depends on the order of rules in the program and on the order of parts of rules; these properties are used by programmers to build efficient programs. In database languages (like SQL or Datalog), however, program execution is independent of the order of rules and facts.
- Special predicates: In Prolog, programmers can directly influence the procedural evaluation of the program with special predicates such as the cut, this has no correspondence in deductive databases.
- Function symbols: Logic Programming languages allow function symbols to build up complex symbols. This is not allowed in deductive databases.
- Tuple-oriented processing: Deductive databases use set-oriented processing while logic programming languages concentrate on one tuple at a time.

References

[1] Datalog and Emerging applications (http://www.cs.ucdavis.edu/~green/papers/sigmod906t-huang.pdf)

Further reading

- Author: Stefano Ceri, Georg Gottlob, Letizia Tanca: *Logic Programming and Databases*. Publisher: Springer-Verlag. ISBN 978-0-387-51728-5
- Author: Ramez Elmasri and Shamkant Navathe: *Fundamentals of Database Systems* (3rd edition). Publisher: Addison-Wesley Longman. ISBN 0-201-54263-3

Deductive language

A **deductive language** is a computer programming language in which the program is a collection of predicates ('facts') and rules that connect them. Such a language is used to create knowledge based systems or expert systems which can deduce answers to problems set them by applying the rules to the facts they have been given. An example of a deductive language is Prolog, or it's database-query cousin, Datalog.

Mobile database

--*mobile Devices database Management commonly called as Mobile Database*' is either a stationary database that can be connected to by a mobile computing device - such as smart phones or PDAs - over a mobile network, or a database which is actually carried by the mobile device. This could be a list of contacts, price information, distance travelled, or any other information.[1]

Many applications require the ability to download information from an information repository and operate on this information even when out of range or disconnected. An example of this is your contacts and calendar on the phone. In this scenario, a user would require access to update information from files in the home directories on a server or customer records from a database. This type of access and work load generated by such users is different from the traditional workloads seen in client–server systems of today.[*citation needed*]

Mobile databases are not just update of company contacts and calendars, but used in a number of industries.

Considerations

- Mobile users must be able to work without a network connection due to poor or even non-existent connections. A cache could be maintained to hold recently accessed data and transactions so that they are not lost due to connection failure. Users might not require access to truly live data, only recently modified data, and uploading of changing might be deferred until reconnected.
- Bandwidth must be conserved (a common requirement on wireless networks that charge per megabyte or data transferred).
- Mobile computing devices tend to have slower CPUs and limited battery life.
- Users with multiple devices (e.g. smartphone and tablet) need to synchronize their devices to a centralized data store. This may require application-specific automation features.[2]

This is in database theory known as "replication", and good mobile database system should provide tools for automatic replication that takes into account that others may have modified the same data as you while you were away, and not just the last update is kept, but also supports "merge" of variants.

- Users may change location geographically and on the network. Usually dealing with this is left to the operating system, which is responsible for maintaining the wireless network connection.

Products

Commercially available mobile databases include those shown on this comparison chart.

Name	Developer	Type	Description
SQL Anywhere	Sybase iAnywhere	Relational	Embedded/portable database, can synchronize with stationary database
DB2 Everyplace	IBM	Relational	Portable, can synchronize with stationary database
IBM Mobile Database	IBM	Relational	Portable/embedded small-footprint version of solidDB server
SQL Server Compact	Microsoft	Relational	Small-footprint embedded/portable database for Microsoft Windows mobile devices and desktops, supports synchronization with Microsoft SQL Server
SQL Server Express	Microsoft	Relational	Embedded database, free download
Oracle Database Lite	Oracle Corporation		Portable, can synchronize with stationary database
SQLite	D. Richard Hipp	C programming library	Public domain
SQLBase	Gupta Technologies LLC of Redwood Shores, California		
Sparksee 5 mobile [3]	Sparsity Technologies [4]	Graph database	Small-footprint embedded/portable database for Android, iOS and BB10. It allows traversals of nodes, minimum distance computations, community search, etc.

SQL Anywhere hasWikipedia:Manual of Style/Dates and numbers#Chronological items about 68% of the mobile database market.[*citation needed*]

References

[1] Organize your business with a mobile database, Kevin Ebi, Microsoft.com, retrieved 14/12/08 (http://www.microsoft.com/windowsmobile/en-us/business/solutions/small-business-database.mspx)

[2] The 5 Traits of Great Cloud-Syncing Apps (http://smallmobile.infogenium.com/2012/11/06/the-5-traits-of-great-cloud-syncing-apps/)

[3] http://www.sparsity-technologies.com#sparksee

[4] http://www.sparsity-technologies.com

External links

- http://www.informit.com/articles/article.aspx?p=26223&seqNum=3
- http://www.informit.com/articles/article.aspx?p=25328
- http://www.asc.di.fct.unl.pt/dagora/docs/papers/epcm99-np.pdf

Well Known Databases

Standard data model

A **standard data model** or industry standard data model (ISDM) is a data model that is widely applied in some industry, and shared amongst competitors to some degree. They are often defined by standards bodies, database vendors or operating system vendors.

When in use, they enable easier and faster information sharing because heterogeneous organizations have a standard vocabulary and pre-negotiated semantics, format, and quality standards for exchanged data. The standardization has an impact on software architecture as solutions that vary from the standard may cause data sharing issues and problems if data is out of compliance with the standard.

The more effective standard models have developed in the banking, insurance, pharmaceutical and automotive industries, to reflect the stringent standards applied to customer information gathering, customer privacy, consumer safety, or just in time manufacturing.

Typically these use the popular relational model of database management, but some use the hierarchical model, especially those used in manufacturing or mandated by governments, e.g., the DIN codes specified by Germany. While the format of the standard may have implementation trade-offs, the underlying goal of these standards is to make sharing of data easier.

The most complex data models known are in military use, and consortia such as NATO tend to require strict standards of their members' equipment and supply databases. However, they typically do not share these with non-NATO competitors, and so calling these 'standard' in the same sense as commercial software is probably not very appropriate.

An emerging area of standard data model is in the identity card arena, where a vast number of security engineering solutions for public spaces, e.g., airports, other public transport, hospitals, are expected soon to rely on a standard data model for identifying the card holder/user of the facility. This may contain biometric information or other data that would be standardized across an entire trade bloc, e.g., the European Union or the North American Free Trade Agreement (NAFTA). This raises many privacy and carceral state concerns. These are discussed more deeply in an article on standard user models.

Example Standard Data Models

- ISO 10303 CAE Data Exchange Standard - includes its own data modelling language, EXPRESS
- ISO 15926 Process Plants including Oil and Gas facilities Life-Cycle data
- IDEAS Group Foundation Ontology agreed by defence departments of Australia, Canada, France, Sweden, UK and USA

External links

- Professional Petroleum Data Management Association Lite Data Model [1]
- Energistics Energy Standards Resource Center [2]
- Pipeline Open Data Standard [3]

References

[1] http://www.ppdm.org/ppdm-standards/ppdm-lite-1-1-data-model
[2] http://www.energistics.org/energistics-standards-directory
[3] http://www.pods.org/4/The%20PODS%20Data%20Model/

Suppliers and Parts database

The Suppliers and Parts database is an example relational database that is referred to extensively in the literature[*citation needed*] and described in detail in C. J. Date's "Introduction" 8ed. It is a simple database comprising three tables: Supplier, Part and Shipment, and is often used as a minimal exemplar of the interrelationships found in a database.

1. The Supplier relation[1] holds information about suppliers. The SID attribute identifies the supplier, while the other attributes each hold one piece of information about the supplier.
2. The Part relation holds information about parts. Likewise, the PID attribute identifies the part, while the other attributes hold information about the part.
3. The Shipment relation holds information about shipments. The SID and PID attributes identify the supplier of the shipment and the part shipped, respectively. The remaining attribute indicates how many parts where shipped.
    - Referential constraints known as Foreign keys ensure that these attributes can only hold values that are also found in the corresponding attributes in the Supplier and Parts relations.
    - It is assumed that only one shipment exists for each supplier/part pairing, which isn't realistic for real world scenarios. This is intentionally oversimplified for pedagogical purposes, as is the entire database.

SQL

The following SQL schema is one possible expression of the Suppliers-and-Parts database.

```
CREATE TABLE Supplier (
  SID     int          primary key,
  SName   varchar(10)  NOT NULL,
  Status  int          NOT NULL,
  City    varchar(10)  NOT NULL
)

CREATE TABLE Part (
  PID     int          primary key,
  PName   varchar(10)  NOT NULL,
  Color   int          NOT NULL,
  Weight  real         NOT NULL,
  City    varchar(10)  NOT NULL
)

CREATE TABLE Shipment (
  SID     int          NOT NULL FOREIGN KEY REFERENCES Supplier(SID),
  PID     int          NOT NULL FOREIGN KEY REFERENCES Part(PID),
  Qty     int          NOT NULL,
  PRIMARY KEY (SID, PID)
)
```

Notes:

1. The ID attributes are simple integers, but they could be (among other things) UUIDs or a system-defined identifier type that holds system-generated values.
2. The choice of VARCHAR(10) is arbitrary and would be too small for real-world use.
3. The application of the NOT NULL constraint to all attributes is a design decision based on the view that NULLs are to be avoided. It is not, strictly speaking, a requirement of the schema.

References

[1] Relations and SQL tables are roughly synonymous.

Internet Movie Database

Internet Movie Database (IMDb)

Web address	imdb.com [1]
Commercial?	Yes
Type of site	Online database for movies, television, and video games
Registration	Registration is optional for members to participate in discussions, comments, ratings, and voting.
Available language(s)	English
Owner	Amazon.com
Created by	Col Needham (CEO)
Launched	October 17, 1990
Alexa rank	▼ 44 (March 2014[2])
Current status	Active

Internet Movie Database (abbreviated **IMDb**) is an online database of information related to films, television programs and video games, taking in actors, production crew, fictional characters, biographies, plot summaries and trivia. Actors and crew can post their own résumé and upload photos of themselves for a yearly fee. U.S. users can also view over 6,000 movies and television shows from CBS, Sony and various independent film makers.

Launched in 1990 by professional computer programmer Col Needham, the company was incorporated in the UK as Internet Movie Database Ltd in 1996, with revenue generated through advertising, licensing and partnerships. In 1998, it became a subsidiary of Amazon.com, who were then able to use it as an advertising resource for selling DVDs and videotapes.

As of February 27, 2014, IMDb had 2,798,497 titles (includes episodes) and 5,749,077 personalities in its database, as well as 51 million registered users [*citation needed*] and is an Alexa Top 50 site.

The site enables any user to submit new material and request edits to existing entries. Although all data is checked before going live, the system has been open to abuse, and occasional errors are acknowledged. Users are also invited to rate any film on a scale of 1 to 10, and the totals are converted into a weighted mean-rating that is displayed beside each title, with online filters employed to deter ballot-stuffing. The site also features Message Boards, which stimulate regular debates among authenticated users.

History

History before website

IMDb originated with a Usenet posting by British film fan and professional computer programmer Col Needham entitled "Those Eyes", about actresses with beautiful eyes. Others with similar interests soon responded with additions or different lists of their own. Needham subsequently started a (male) "Actors List", while Dave Knight began a "Directors List", and Andy Krieg took over "THE LIST" from Hank Driskill, which would later be renamed the "Actress List". Both lists had been restricted to people who were alive and working, but soon retired people were

added so Needham started what was then (but did not remain) a separate "Dead Actors/Actresses List". The goal of the participants now was to make the lists as inclusive as possible. By late 1990, the lists included almost 10,000 movies and television series correlated with actors and actresses appearing therein. On October 17, 1990, Needham developed and posted a collection of Unix shell scripts which could be used to search the four lists, and thus the database that would become the IMDb was born. At the time, it was known as the "rec.arts.movies movie database", but by 1993 had been moved out of the Usenet group as an independent website underwritten and controlled by Needham and personal followers. Other website users were invited to contribute data which they may have collected and verified, on a volunteer basis, which greatly increased the amount and types of data to be stored. Entire new sections were added. As the site grew hugely, full production crews, uncredited performers and other demographic data were added. Needham's group allowed some advertising to support ongoing operations of the site, including the hiring of full-time paid data managers. All the primary staff came (and still come) from the burgeoning computer industry and/or training schools and did not have extensive expertise in the visual media.[*citation needed*] In 1998, unable to secure sufficient funding from limited advertising, contributions and unable to raise support from the visual media industries or academia, Needham sold the IMDb to Amazon.com, on condition that its operation would remain in the hands of Needham and his small cadre of managers, who soon were able to move into full-time paid staff positions.

On the web

The database had been expanded to include additional categories of filmmakers and other demographic material, as well as trivia, biographies, and plot summaries. The movie ratings had been properly integrated with the list data and a centralized email interface for querying the database had been created by Alan Jay. Later in the yearWikipedia:Manual of Style/Dates and numbers#Chronological items it moved onto the World Wide Web (a network in its infancy at that time) under the name of *Cardiff Internet Movie Database*. The database resided on the servers of the computer science department of Cardiff University in Wales. Rob Hartill was the original web interface author. In 1994 the email interface was revised to accept the submission of all information meaning that people no longer had to email the specific list maintainer with their updates. However, the structure remained that information received on a single film was divided among multiple section managers, the sections being defined and determined by categories of film personnel and the individual filmographies contained therein. Over the next few years, the database was run on a network of mirrors across the world with donated bandwidth.[*citation needed*]

The website is Perl-based.[3] As of May 2011, the site has been filtered in China for more than one year, although many users address it through proxy server or by VPN.

On October 17, 2010, IMDb launched original video (www.imdb.com/20) in celebration of its 20th anniversary.

As an independent company

In 1996 IMDb was incorporated in the United Kingdom, becoming the Internet Movie Database Ltd. Founder Col Needham became the primary owner as well as the figurehead. General revenue for site operations was generated through advertising, licensing and partnerships.

As Amazon.com subsidiary

In 1998, Jeff Bezos, founder, owner and CEO of Amazon.com, struck a deal with Col Needham and other principal shareholders to buy IMDb outright and attach it to Amazon as a subsidiary, private company. This gave IMDb the ability to pay the shareholders salaries for their work, while Amazon.com would be able to use the IMDb as an advertising resource for selling DVDs and videotapes.

IMDb continued to expand its functionality. On January 15, 2002, it added a subscription service known as IMDbPro, aimed at entertainment professionals. IMDbPro was announced and launched at the 2002 Sundance Film Festival. It provides a variety of services including film production and box office details, as well as a company

directory.

As an additional incentive for users, as of 2003, users identified as one of "the top 100 contributors" of hard data received complimentary free access to IMDbPro for the following calendar year; for 2006 this was increased to the top 150 contributors, and for 2010 to the top 250. In 2008 IMDb launched their first official foreign language version with the German IMDb.de. Also in 2008, IMDb acquired two other companies, Withoutabox and Box Office Mojo.

Television episodes

On January 26, 2006, "Full Episode Support" came online, allowing the database to support separate cast and crew listings for each episode of every television series. This was described by Col Needham as "the largest change we've ever made to our data model",[*citation needed*] and increased the number of titles in the database from 485,000 to nearly 755,000.[*citation needed*]

Characters' filmography

On October 2, 2007,[*citation needed*] the characters' filmography was added. Character entries are created from character listings in the main filmography database, and as such do not need any additional verification by IMDb staff. They have already been verified when they are added to the main filmography.

Instant viewing

On September 15, 2008, a feature was added that enables instant viewing of over 6,000 movies and television shows from CBS, Sony and a number of independent film makers, with direct links from their profiles. Due to licensing restrictions, this feature is only available to viewers in the United States.

Content and format

Data provided by subjects

In 2006, IMDb introduced its "Résumé subscription service", where actors and crew can post their own résumé and upload photos of themselves[4] for a yearly fee.[5] The base annual charge for including a photo with an account was $39.95 until 2010, when it was increased to $54.95. IMDb résumé pages are kept on a sub-page of the regular entry about that person, with a regular entry automatically created for each résumé subscriber who does not already have one.[6]

As of 2012, Resume Services is now included as part of an IMDbPro subscription, and is no longer offered as a separate subscription service.

Copyright, vandalism, and error issues

All volunteers who contribute content to the database technically retain copyright on their contributions but the compilation of the content becomes the exclusive property of IMDb with the full right to copy, modify, and sublicense it and they are verified before posting.[7] Credit is not given on specific title or filmography pages to the contributor(s) who have provided information. Conversely, a credited text entry, such as a plot summary, may be "corrected" for content, grammar, sentence structure, perceived omission or error, by other contributors without having to add their names as co-authors. Due to the process of having the submitted data or text reviewed by a section manager, IMDb is different from database projects like Wikipedia, Discogs, or OpenStreetMap in that contributors cannot add, delete, or modify the data or text on impulse, and the manipulation of data is controlled by IMDb technology and salaried staff.[8] Nevertheless, although it is generally assumed to be reliable,[9] IMDb has been subject to deliberate additions of false information, as acknowledged by a spokesperson in 2012: "We make it easy for users and professionals to update much of our content, which is why we have an 'edit page.' The data that is

submitted goes through a series of consistency checks before it goes live. Given the sheer volume of the information, occasional mistakes are inevitable, and, when reported, they are promptly fixed. We always welcome corrections."

The Java Movie Database (JMDB) is reportedly creating an IMDb_Error.log file that lists all the errors found while processing the IMDb plain text files. A Wiki alternative to IMDb is Open Media Database [10] whose content is also contributed by users but licensed under CC-by and the GFDL. Since 2007, IMDb has been experimenting with wiki-programmed sections for complete film synopses, parental guides, and FAQs about titles as determined by (and answered by) individual contributors.

Data format and access

IMDb does not provide an API for automated queries. However most of the data can be downloaded as compressed plain text files and the information can be extracted using the command-line interface tools provided. Beside that there is the Java-based graphical user interface (GUI) application available which is able to process the compressed plain text files and allow to search and display the information. This GUI application supports different languages but the movie related data is of course English as made available by IMDb. A Python package called IMDbPY can also be used to process the compressed plain text files into a number of different SQL databases, enabling easier access to the entire dataset for searching or data mining.

Film titles

The IMDb has sites in English as well as versions translated completely or in part into other languages (Finnish, French, German, Hungarian, Italian, Polish, Portuguese and Romanian). The non-English language sites display film titles in the specified language. While originally the IMDb's English-language sites displayed titles according to their original country-of-origin language, in 2010 the IMDb began allowing individual users in the UK and USA to choose primary title display by either the original-language titles, or the US or UK release title (normally, in English).

Ancillary features

User ratings of films

As one adjunct to data, the IMDb offers a rating scale that allows users to rate films on a scale of one to ten. The rating system is recognized as being severely flawed for several reasons.

IMDb indicates that submitted ratings are filtered and weighted in various ways in order to produce a weighted mean that is displayed for each film, series, and so on. It states that filters are used to avoid ballot stuffing; the method is not described in detail to avoid attempts to circumvent it. In fact, it sometimes produces an extreme difference between the weighted average and the arithmetic mean. For example, *Jonas Brothers: The 3D Concert Experience* is considered to be the worst film with a weighted average of 2.1 as of 2014, but has a rather ordinary arithmetic mean of 3.9.[11][12]

Film rankings (IMDb Top 250)

The IMDb Top 250[13] is intended to be a listing of the top 'rated' 250 films, based on ratings by the registered users of the website using the methods described. Only non-documentary theatrical releases running at least forty-five minutes with over 25,000 ratings are considered; all other products are ineligible. Also, the 'top 250' rating is based on only the ratings of "regular voters". The exact number of votes a registered user would have to make to be considered to be a user who votes regularly has been kept secret. IMDb has stated that to maintain the effectiveness of the top 250 list they *"deliberately do not disclose the criteria used for a person to be counted as a regular voter"*.[14] In addition to other weightings, the top 250 films are also based on a weighted rating formula referred to in actuarial science as a *credibility formula*.[15] This label arises because a statistic is taken to be more credible the

greater the number of individual pieces of information; in this case from eligible users who submit ratings. IMDb uses the following formula to calculate the weighted rating:

$$W = \frac{Rv + Cm}{v + m}$$

where:

W = weighted rating

R = average for the movie as a number from 0 to 10 (mean) = (Rating)

v = number of votes for the movie = (votes)

m = minimum votes required to be listed in the Top 250 (currently 25,000)

C = the mean vote across the whole report (currently 7.0)

The W in this formula is equivalent to a Bayesian posterior mean (See Bayesian statistics).

The IMDb also has a Bottom 100 feature which is assembled through a similar process although only 1500 votes must be received to qualify for the list.

The top 250 list comprises a wide range of films, including major releases, cult films, independent films, critically acclaimed films, silent films and non-English language films.

Fan activity

One of the most used features of the Internet Movie Database is the message boards that coincide with every title (excepting, as of 2013, TV episodes[16]) and name entry, along with over 140 main boards. This section is one of the more recent features of IMDb, having its beginnings in 2001. In order to post on the message boards a user needs to "authenticate" their account via cell phone, credit card, or by having been a recent customer of the parent company Amazon.com. Message boards have expanded in recent years. The Soapbox started in 1999 is a general message board meant for debates on any subject. The Politics board started in 2007 is a message board to discuss politics, news events and current affairs as well as history and economics. Both these message boards have become the most popular message boards in IMDb, more popular on a long term basis than any individual movie message board.

Litigation

In 2011, in the case of *Hoang v. Amazon.com*, IMDb was sued by an anonymous actress for more than US$1 million due to IMDb revealing her age (40, at the time). The actress claimed that revealing her age could cause her to lose acting opportunities. Judge Marsha J. Pechman, a U.S. district judge in Seattle, dismissed the lawsuit, saying the actress had no grounds to proceed with an anonymous complaint. She re-filed and so revealed that the complainant is a Huong Hoang of Texas, who uses the stage name Junie Hoang. In 2013, Pechman dismissed all causes of action except for a breach of contract claim against IMDb; a jury then sided with IMDb on that claim.

Also in 2011, in the case of United Video Properties Inc., et al. v. Amazon.Com Inc. et al., IMDb and Amazon were sued by Rovi Corporation and others for patent infringement over their various program listing offerings. The patent claims were ultimately construed in a way favorable to IMDb and Rovi/United Video Properties lost the case, though it is currently on appeal.

Notes

[1] http://www.imdb.com/
[2] http://en.wikipedia.org/w/index.php?title=Internet_Movie_Database&action=edit
[3] What software/hardware are you using to run the site? (http://www.imdb.com/help/show_leaf?techinfo) imdb.com
[4] Lycos Europe and IMDb sign sales agreement for 9 European markets (https://web.archive.org/web/20061023141910/http://www.lycos-europe.com/Index-Eng/G-English-Files/PR-20060710-IMDb.html). Lycos Europe press release, July 10, 2006.
[5] IMDb Resume FAQ: Can I subscribe only for one month or one year? (http://resume.imdb.com/help/show_leaf?resumenotrecurring). Retrieved January 22, 2008.
[6] IMDb Resume FAQ: Is there any difference between a regular IMDb name page and an IMDb name page created via IMDb Resume? (http://resume.imdb.com/help/show_leaf?resumenamepagediff). Retrieved January 22, 2008.
[7] IMDb Copyright and Conditions of Use (http://www.imdb.com/help/show_article?conditions). imdb.com
[8] The Plain Text Data Files (http://www.imdb.com/interfaces#plain) IMDb – Alternate Interfaces
[9] It may be assumed to be generally reliable but the IMDb does not claim that it is 100% accurate.
[10] http://www.omdb.org
[11] IMDb Charts: IMDb Bottom 100 (http://www.imdb.com/chart/bottom?tt1229827). imdb.com
[12] Jonas Brothers: The 3D Concert Experience (2009) – User ratings (http://www.imdb.com/title/tt1229827/ratings). imdb.com
[13] http://www.imdb.com/chart/top
[14] The user votes average on film X is 9.4, so it should appear in your top 250 films listing, yet it doesn't. Why? (http://www.imdb.com/help/search?domain=helpdesk_faq&index=1&file=notintop250)
[15] mirror (http://isfaserveur.univ-lyon1.fr/~norberg/links/papers/CRED-eas.pdf)
[16] Each TV episode uses the same message board for the whole series

References

External links

- Official website (http://www.imdb.com/)
- Official Android app (http://apinik.com/apps/com.imdb.mobile/?lang=en)
- Official IOS app (https://itunes.apple.com/en/app/imdb-movies-tv/id342792525?mt=8)

YAGO (ontology)

YAGO

Developer(s)	Max-Planck-Institute Saarbrücken
Initial release	2008
Stable release	YAGO2s / 16 October 2012
Type	Semantic Web, Linked Data
License	Creative Commons CC-BY 3.0
Website	www.mpi-inf.mpg.de/yago-naga/yago/ [1]

YAGO is a knowledge base developed at the Max Planck Institute for Computer Science in Saarbrücken. It is automatically extracted from Wikipedia and other sources.

As of 2012, YAGO2s has knowledge of more than 10 million entities and contains more than 120 million facts about these entities. The information in YAGO is extracted from Wikipedia (e.g., categories, redirects, infoboxes), WordNet (e.g., synsets. hyponymy) and GeoNames.[2] The accuracy of YAGO was manually evaluated to be above 95% on a sample of facts. To integrate it to the linked data cloud, YAGO has been linked to the DBpedia ontology and to the SUMO ontology.

YAGO2s is provided in Turtle and tsv formats. Dumps of the whole database are available, as well as thematic and specialized dumps. It can also be queried through various online browsers [3] and through a SPARQL endpoint hosted by OpenLink Software. YAGO has been used in the Watson artificial intelligence system.[4]

References

[1] http://www.mpi-inf.mpg.de/yago-naga/yago/

[2] Fabian M. Suchanek, Gjergji Kasneci and Gerhard Weikum. "Yago - A Core of Semantic Knowledge". 16th international World Wide Web conference (WWW 2007) (http://suchanek.name/work/publications/www2007.pdf)

[3] http://www.mpi-inf.mpg.de/yago-naga/yago/demo.html

[4] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, Chris Welty. Building Watson: An Overview of the DeepQA Project. AI Magazine 31(3): 59-79 (2010)

External links

- YAGO Homepage (http://www.mpi-inf.mpg.de/yago-naga/yago)
- Text-based browser (https://d5gate.ag5.mpi-sb.mpg.de/webyagospo/Browser)
- SVG-based browser (https://d5gate.ag5.mpi-sb.mpg.de/webyagospo/SvgBrowser)

World Wide Molecular Matrix

The **World Wide Molecular Matrix** (WWMM) is an electronic repository for unpublished chemical data. First proposed in 2002 by Peter Murray-Rust and his colleagues in the chemistry department at the University of Cambridge in the United Kingdom, WWMM provides a free, easily searchable database for information about thousands of complicated molecules, data that would otherwise remain inaccessible to scientists.

Murray-Rust, a chemical informatics specialist, has estimated that 80% of the results produced by chemists around the world is never published in scientific journals. Most of this data is not ground-breaking, yet it could conceivably be of use to scientists doing related projects—if they could access it. The WWMM was proposed as a solution to this problem. It would house the results of experiments on over 100,000 molecules in physical chemistry, organic chemistry, biochemistry and medicinal chemistry.

In other scientific fields, the need for a similar depository to house inaccessible information could be more acute. In a presentation at the "CERN Workshop on Innovations in Scholarly Communications (OAI4)", Murray-Rust said that chemistry actually leads other fields in published data. He estimated that as much as 99% of the data in some scientific fields never reaches publication.[*citation needed*]

Although scientific in nature, the WWMM is part of the broader open archives and open source movements, pushes to make more and more information freely available to any user via the Internet or World Wide Web. In his CERN presentation, Murray-Rust stated that the WWMM was a "response to the expense of [scientific] journals," and he asked the rhetorical question, "Can we win the war to make data open, or will it be absorbed into the publishing and pseudo-publishing world?" Murray-Rust and his colleagues are also responsible for the development of the Chemical Mark-up Language (CML), a variant of XML intended for chemists.

External links

- The home page of Dr. Peter Murray-Rust at the University of Cambridge [1]
- The Cambridge Center for molecular informatics [2]
- An outline of the WWMM [3]
- CERN Workshop on Innovations in Scholarly Communication (OAI4) [4]Wikipedia:Verifiability

References

[1] http://www.ch.cam.ac.uk/person/pm286
[2] http://www.escience.cam.ac.uk/projects/mi/mi_call.html
[3] http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/157.pdf
[4] http://oai4.web.cern.ch/OAI4/

Voter database

A **voter database** is a database containing information on voters for the purpose of assisting a political party or an individual politician, in their Get out the vote (GOTV) efforts and other areas of the campaign.

In most countries, the election agency makes the electoral roll available to all campaigns soon after the election campaign has begun. Campaigns can then merge this information with the other data they have collected on voters over the years to create their database. Often basic information such as phone numbers and postal codes are not included on the voters list, and the campaign will have to procure this data as well.

The United States has no state or federal election agency, and thus no central lists. In 2002, the United States Congress passed the Help America Vote Act (HAVA). HAVA required that each state compile an official state voter database by January 2006. Most states complied with HAVA by gathering the voter files available from each individual county. States decided what information to include, what restrictions to place on the use of their voter database, and how much the database would cost. In the United States, several companies have merged state voter information with commercially obtained data to create comprehensive voter databases that include a plethora of personal details on each voter. These companies often provide United States Voter Files to statutorily permitted or otherwise non-restricted users.

Uses

The voter database is central to many parts of a campaign:

- Fundraising: The database can determine who should receive fundraising direct mail or telephone calls. These letters and calls can be tailored to reflect the issues and concerns of each potential donor. Past donor history, support for related advocacy groups, magazine subscriptions, and consumer behaviour can all be used to find likely donors and maximize the returns of any fundraising efforts.
- Recruitment: As with fundraising, databases, especially those with detailed past election behaviour, are essential to recruiting volunteers and also finding locations for lawn signs.
- Issue tracking: A campaign can track how a certain issues are perceived across geographic and demographic lines and can show how to adjust the campaign's message for different audiences. By databasing all incoming telephone calls and e-mails as well as entering petitions and supporter lists from advocacy groups and NGOs (Non-Government Organizations) one can closely track how issues are followed by the electorate.
- Get out the vote: One of the most important parts of a modern campaign is the campaign to ensure one's own supporters go to the polls on election day, and databases are central to this. A successful voter identification campaign requires connecting with a significant portion of the electorate and recording how they are going to vote in a database. On election day this information needs to be given out with accurate contact information for each voter so that they can be pulled to the polls.

Voter information

Personal data frequently included in a voter database:

- Name
- Physical address
- Mailing address
- Phone number
- Party membership or affiliation
- Voting history (including federal, sub-national, primary, municipal, or special election voting history)
- Absentee or military voter designations
- Source of voter registration, i.e., DMV/MVA, Public Assistance Office, etc.

- Ethnicity or emerges race hypothesis
- Gender
- Birth date or age range

Data that may be added via commercial routes:

- "Extreme voters" status (voters who vote very frequently)
- Homeownership
- Hunting or fishing license holders
- Boat owners
- Concealed Weapon Permit Holders
- Occupation such as physical therapist, teacher, etc.
- Charitable or political contributions
- Magazine subscription status

Voter database management software

The use of voter databases has been established in political campaigns from local school board to national elections:

In the United States

In the 2004 presidential election in the United States, the Republican Party used the Voter Vault platform and the Democratic Party used DataMart. Currently, the Republicans use GOP Data Center and the Democrats use Votebuilder from the Voter Activation Network (VAN). There are non partisan firms that offer registered voter data in the United Sates too: eMerges.com [1] and Labels and Lists [2].

In the United Kingdom

In 2005, the British Conservative Party used the Voter Vault software to assist them in the 2005 General Election.

Currently,Wikipedia:Manual of Style/Dates and numbers#Chronological items the British Conservative Party uses MERLIN (Managing Elector Relationships through Local Information Networks) which is a national database accessible by every local Conservative Association in the country. This was commissioned in 2005,[3] and went live in 2008.[4]

Britain's Labour Party has used a variety of voter databases through the past two decades. Its most recent incarnation is the Labour Contact Creator system, an online tool accessible from anywhere with an internet connection. Party members and activists are provided with a username and password and voter contact details, preferences, interests, past voting behaviour, and demographic/socio-economic information are available. The system allows voters to be selected on the basis of a MOSAIC grouping, which attempts to determine the sort of interests and activities a voter or a household might display. Maps of where key voters live and information can be cross-referenced so users can find where target voters live, how often they are contacted, how they prefer to be contacted, and what responses have been provided upon making contact.

The Contact Creator system is also linked with the Labour Party's other new media tools, Print Creator and Email Creator. Print Creator allows key voters identified through Contact Creator to be targeted with direct mails and leaflets about Labour Party activity. Email Creator allows the user to collect a list of email addresses on the Contact Creator system, e-mail thousands of voters, and monitor the response rate from targets.

The Labour Party also operates 'Membersnet', which allows party members to update their registration details, inform other members and Party HQ of rival campaign activities, create events and invite others to attend, email members, create blogs, and share best practice campaign material.

The Labour Party introduced a new system to Membersnet and Contact Creator in 2013,[5] focusing on a new user-friendly interface, which allows Party Members anywhere in the UK to contact target voters and identify their

voting preferences. They will then be able to enter the information gained directly into the Contact Creator system.

In late 2011 Liberal Democrats began to adopt the use of a variant of NGP VAN's Voter Activation Network (VAN) software named "Connect".[6] This firm also supplies "Votebuilder" and VAN to the Democrats in the USA and to the Liberal Party of Canada as "Liberal List".

In Canada

In Canada, the Conservative Party of Canada uses the in-house developed C-VOTE database, and used to use Constituent Information Management System (CIMS) [7], originally developed by the same company which produced Trackright, which the Progressive Conservative Party of Ontario, uses to manage voter information. It is similar to the Voter Vault software. The New Democratic Party uses their own custom database system called NDP Vote. The Liberal Party has recently introduced "Liberalist [8]" based on the US's Democrats' Voter Activation Network (VAN) Previously the Liberals used a system called ManagElect[9][10] .

Availability

The availability of voter files sometimes creates a need for voter list management software as opposed to, for example, using Excel spreadsheets. Political campaigns generally have three options:

- Write-their-own software: A political campaign may choose to build their own database management system to handle their voter registration files. This may be accomplished in a spreadsheet program like Microsoft Excel. This method can be time-consuming and require a certain level of technology know-how, but may be the only option for certain campaigns.
- Desktop-based software: A variety of companies provide voter list management software for a desktop or laptop computer. Generally, this type of software is distributed on a per-computer basis and can therefore be expensive for larger campaigns. However, desktop-based software is easier to use than create-your-own and can provide many integrated, helpful solutions for political campaigns such as walking list generation, computer-based phone banking, and more.
- Internet-based software: Many companies provide online voter list management software. Internet-based software has many of the same benefits of desktop-based software, and has the additional advantage of being accessible from any computer (Windows, Mac, or Linux) with an Internet connection. Internet-based software helps campaigns eliminate the need for large campaign offices with central computer and phone banks.

References

[1] http://www.eMerges.com
[2] http://www.labelsandlists.com/
[3] "David Cameron's battle to connect" (http://www.wired.co.uk/magazine/archive/2010/04/features/david-camerons-battle-to-connect)
[4] "Tory activists uneasy over poll software" (http://www.ft.com/cms/s/0/dd091fb6-404a-11df-8d23-00144feabdc0.html)
[5] http://labourlist.org/2013/01/5-key-things-to-take-away-from-labours-target-seat-list-and-election-strategy/
[6] http://www.libdemvoice.org/thank-you-ears-but-the-van-is-coming-24486.html
[7] http://cpccims.ca
[8] http://liberalist.ca/
[9] "The ManagElect "merge" step". Angry in the Great White North. October 22, 2007. (http://stevejanke.com/archives/244292.php)
[10] "The Canadian general election of 2000". By Christopher Dornan, Jon H. Pammett (http://books.google.ca/books?id=7lFIdomfb_IC&lpg=PA46&ots=juteLTQD2Q&dq=managelect&pg=PA46#v=onepage&q=managelect&f=false)

External links

- Official site of the Liberalist voting database of the Canadian Liberal party. (http://liberalist.ca)
- Official site of the Constituent Information Management System of the Conservative Party of Canada. (http://cpccims.ca)

VIOLIN vaccine database

The **Vaccine Investigation and OnLine Information Network** (**VIOLIN**) is the largest web-based vaccine database and analysis system. VIOLIN currently contains over 2,700 vaccines or vaccine candidates for over 170 pathogens. The vaccine information in the database is collected by manual curation from over 1,600 peer-reviewed papers. Different from most existing vaccine databases, VIOLIN focuses on vaccine research data. Different types of information is curated, including vaccine name, license status, antigens used, vaccine adjuvants, vaccine vectors, vaccination procedure, host immune response, challenge procedure, vaccine efficacy, adverse events, etc. All vaccine information in the VIOLIN vaccine database is supported by quoted references. The data generated by a curator is published only after a careful review and approval by a vaccine domain expert.

In addition, VIOLIN includes many vaccine analysis programs. For example, VIOLIN includes Vaxign (http://www. violinet. org/ vaxign), the first web-based vaccine design program based on the strategy of reverse vaccinology. Vaxign has been tested in different pathogen models, including uropathogenic E. coli and Brucella spp.

VIOLIN also maintains the official web page for the development of community-based Vaccine Ontology (VO) (http://www. violinet. org/ vaccineontology). VO is a formal biomedical ontology in the domain of vaccine and vaccination. VO is targeted for vaccine data standardization and integration, and supporting automated reasoning. VO has been shown to enhance vaccine literature mining.

References

External links

- Official site (http://www.violinet.org)

Census of Governments

The **Census of Governments** is a census performed by the United States Census Bureau. It is mandated by 13 U.S.C. § 161 [1]. The Census of Governments occurs twice-per-decade, in years ending with 2 and 7. The survey identifies the scope of American government at the federal, state, and municipal level by measuring public finance, public employment, classifies municipal government structures and activities, and identifies municipal authorities and Special-purpose districts. The Census of Governments in its current form started in year 1957.

References

[1] http://www.law.cornell.edu/uscode/13/161.html

External links

- Census of Governments (http://www.census.gov/govs/cog/) at United States Census Bureau

Management information base

A **management information base** (**MIB**) is a database used for managing the entities in a communications network. Most often associated with the Simple Network Management Protocol (SNMP), the term is also used more generically in contexts such as in OSI/ISO Network management model. While intended to refer to the complete collection of management information available on an entity, it is often used to refer to a particular subset, more correctly referred to as MIB-module.

Objects in the MIB are defined using a subset of Abstract Syntax Notation One (ASN.1) called "Structure of Management Information Version 2 (SMIv2)" RFC 2578.The software that performs the parsing is a MIB compiler.

The database is hierarchical (tree-structured) and each entry is addressed through an object identifier (OID). Internet documentation RFCs discuss MIBs, notably RFC 1155, "Structure and Identification of Management Information for TCP/IP based internets", and its two companions, RFC 1213, "Management Information Base for Network Management of TCP/IP-based internets", and RFC 1157, "A Simple Network Management Protocol".

Abstract Syntax Notation One (ASN.1)

In telecommunications and computer networking, Abstract Syntax Notation One (ASN.1) is a standard and flexible notation that describes data structures for representing, encoding, transmitting, and decoding data. It provides a set of formal rules for describing the structure of objects that are independent of machine-specific encoding techniques and is a precise, formal notation that removes ambiguities.

ASN.1 is a joint ISO and ITU-T standard, originally defined in 1984 as part of CCITT X.409:1984. ASN.1 moved to its own standard, X.208, in 1988 due to wide applicability. The substantially revised 1995 version is covered by the X.680 series.

An adapted subset of ASN.1, Structure of Management Information (SMI), is specified in SNMP to define sets of related MIB objects; these sets are termed MIB modules.

MIB hierarchy

The MIB hierarchy can be depicted as a tree with a nameless root, the levels of which are assigned by different organizations. The top-level MIB OIDs belong to different standards organizations, while lower-level object IDs are allocated by associated organizations. This model permits management across all layers of the OSI reference model, extending into applications such as databases, email, and the Java reference model, as MIBs can be defined for all such area-specific information and operations.

A managed object (sometimes called a MIB object, an object, or a MIB) is one of any number of specific characteristics of a managed device. Managed objects are made up of one or more object instances (identified by their OIDs), which are essentially variables.

Two types of managed objects exist:

- Scalar objects define a single object instance.
- Tabular objects define multiple related object instances that are grouped in MIB tables.

An example of a managed object is `atInput`, which is a scalar object that contains a single object instance, the integer value that indicates the total number of input AppleTalk packets on a router interface.

An object identifier (or object ID or OID) uniquely identifies a managed object in the MIB hierarchy.

SNMPv1 and SMI-specific data types

The first version of the SMI (SMIv1) specifies the use of a number of SMI-specific data types, which are divided into two categories:

- Simple data types
- Application-wide data types

Simple data types

Three simple data types are defined in the SNMPv1 SMI:

- The integer data type is a signed integer in the range of -2^{31} to $2^{31}-1$.
- Octet strings are ordered sequences of 0 to 65,535 octets.
- Object IDs come from the set of all object identifiers allocated according to the rules specified in ASN.1.

Application-wide data types

The following application-wide data types exist in the SNMPv1 SMI:

- *Network addresses* represent addresses from a particular protocol family. SMIv1 supports only 32-bit (IPv4) addresses (SMIv2 uses Octet Strings to represent addresses generically, and thus are usable in SMIv1 too. SMIv1 had an explicit IPv4 address datatype.)
- *Counters* are non-negative integers that increase until they reach a maximum value and then roll over to zero. SNMPv1 specifies a counter size of 32 bits.
- *Gauges* are non-negative integers that can increase or decrease between specified minimum and maximum values. Whenever the system property represented by the gauge is outside of that range, the value of the gauge itself will vary no further than the respective maximum or minimum, as specified in RFC 2578.
- *Time ticks* represent time since some event, measured in hundredths of a second.
- *Opaques* represent an arbitrary encoding that is used to pass arbitrary information strings that do not conform to the strict data typing used by the SMI.
- *Integers* represent signed integer-valued information. This data type redefines the integer data type, which has arbitrary precision in ASN.1 but bounded precision in the SMI.
- *Unsigned integers* represent unsigned integer-valued information, which is useful when values are always non-negative. This data type redefines the integer data type, which has arbitrary precision in ASN.1 but bounded

precision in the SMI.

SNMPv1 MIB tables

The SNMPv1 SMI defines highly structured tables that are used to group the instances of a tabular object (that is, an object that contains multiple variables). Tables are composed of zero or more rows, which are indexed in a way that allows SNMP to retrieve or alter an entire row with a single `Get`, `GetNext`, or `Set` command.

SMIv2 and structure of management information

The second version of the SMI (SMIv2) is described in RFC 2578 and RFC 2579. It enhances and adds to the SMIv1-specific data types, such as including bit strings, network addresses, and counters. Bit strings are defined only in SMIv2 and comprise zero or more named bits that specify a value. Network addresses represent an address from a particular protocol family. Counters are non-negative integers that increase until they reach a maximum value and then return to zero. In SMIv1, a 32-bit counter size is specified. In SMIv2, 32-bit and 64-bit counters are defined.

SMIv2 also specifies information modules, which specify a group of related definitions. Three types of SMI information modules exist: MIB modules, compliance statements, and capability statements.

- MIB modules contain definitions of interrelated managed objects.
- Compliance statements provide a systematic way to describe a group of managed objects that must be implemented for conformance to a standard.
- Capability statements are used to indicate the precise level of support that an agent claims with respect to a MIB group. An NMS can adjust its behavior toward agents according to the capabilities statements associated with each agent.

Updating MIB Modules

MIB modules are occasionally updated to add new functionality, remove ambiguities and to fix defects. These changes are made in conformance to section 10 of RFC 2578 and section 5 of RFC 2579. An example of a MIB module that has been updated many times is the important set of objects that was originally defined in RFC 1213, also known as "MIB-II". This MIB module has since been split up and can be found in MIB modules such as RFC 4293 "Management Information Base for the Internet Protocol (IP)", RFC 4022 "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4113 "Management Information Base for the User Datagram Protocol (UDP)", RFC 2863 "The Interfaces Group MIB" and RFC 3418 "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)".

MIBs index

There are a large number of MIBs defined by both standards organizations like the IETF, private enterprises and other entities.

IETF maintained

There are 318 RFCs in the first 5000 RFCs from the IETF that contain MIBs. This list is merely a fraction of the MIBs that have been written:

- **SNMP - SMI**: RFC 1155 — Defines the Structure of Management Information (SMI)
- **MIB-I**: RFC 1156 — Historically used with CMOT, not to be used with SNMP
- **SNMPv2-SMI**: RFC 2578 — Structure of Management Information Version 2 (SMIv2)
- **MIB-II**: RFC 1213 — Management Information Base for Network Management of TCP/IP-based internets

- **SNMPv2-MIB**: RFC 3418 — Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)
- **TCP-MIB**: RFC 4022 — Management Information Base for the Transmission Control Protocol (TCP)
- **UDP-MIB**: RFC 4113 — Management Information Base for the User Datagram Protocol (UDP)
- **IP-MIB**: RFC 4293 — Management Information Base for the Internet Protocol (IP)
- **IF-MIB**: RFC 2863 — The Interfaces Group MIB
- **ENTITY-MIB**: RFC 4133 — Entity MIB (Version 3)
- **ENTITY-STATE-MIB**: RFC 4268 — Entity State MIB
- **ALARM-MIB**: RFC 3877 — Alarm Management Information Base (MIB)
- Fibre Channel
  - **FC-MGMT-MIB**: RFC 4044 Fibre Channel Management MIB
  - **FIBRE-CHANNEL-FE-MIB**: RFC 2837 Definitions of Managed Objects for the Fabric Element in Fibre Channel Standard
- **HPR-IP-MIB**: RFC 2584 — Definitions of Managed Objects for APPN/HPR in IP Networks

IEEE maintained

The IETF and IEEE have agreed to move MIBs relating to IEEE work (for example Ethernet and bridging) to their respective IEEE workgroup. This is in process and a few items are complete.

- Network bridge
  - IEEE 802.1ap-2008 consolidated the IEEE and IETF RFCs related to bridging networks into eight related MIBs.

References

External links

- ByteSphere's MIB Database (http://www.oidview.com/mibs/), a free online MIB repository for thousands of SNMP MIBs.
- MIBSearch (http://www.mibsearch.com/), a free search engine for SNMP MIB files.
- SimpleWeb MIBs (http://www.simpleweb.org/ietf/mibs/)
- *MIB index* (http://www.icir.org/fenner/mibs/mib-index.html), ICIR.
- *MIB Compilers and Loading MIBs* (http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a00800b4cee.shtml), Cisco.
- ipMonitor's SNMP Center (http://support.ipmonitor.com/snmp_center.aspx)
- MIB Depot (http://www.mibdepot.com/) — extensive list of MIBs
- PEN (Private Enterprise Number) registry (http://www.iana.org/assignments/enterprise-numbers)
- PEN request authority (http://pen.iana.org/pen/PenApplication.page)

MIB Browsers

- SnmpB: A graphical open source MIB browser (http://sourceforge.net/projects/snmpb/) for Windows, MacOSX and Linux.
- mbrowse: A graphical SNMP MIB browser for Linux (http://sourceforge.net/projects/mbrowse/), based upon GTK+ and Net-SNMP.
- BlackOwl MIB Browser: A graphical MIB browser for Windows and Linux (http://sourceforge.net/projects/blackowl/) which can extract MIBs from RFCs and display graphs.
- SMI-Mib Browser: A graphical MIB browser (http://sourceforge.net/projects/mibview/) — as of 2010-05-18, this project is no longer under active development.
- MBJ: A graphical MIB browser, written in Java (http://sourceforge.net/projects/mbj/)
- JMibBrowser: A graphical MIB browser, written in Java (http://sourceforge.net/projects/jmibbrowser/). It can send SNMP requests and dynamically load MIB data.
- qtmib: An open source graphical MIB browser (http://qtmib.sourceforge.net/) written in C++. It is build as a front-end for Net-SNMP.
- NetDecision MIB Browser: A graphical MIB browser, written in C++ (http://www.netmechanica.com/products/?prod_id=1009). It can send SNMP requests and dynamically load MIB files.

Planetary Data System

The **Planetary Data System** (PDS) is a distributed data system that NASA uses to archive data collected by Solar System robotic missions and ground-based support data associated with those missions. PDS is managed by NASA Headquarters' Planetary Sciences Division. The PDS is an active archive that makes available well documented, peer reviewed data to the research community. The archive and data within are held to high quality standards established by the PDS. The PDS is divided into a number of science discipline "nodes" which are individually curated by planetary scientists. The Solar System Exploration Data Services Office at the Goddard Space Flight Center handles PDS Project Management.

PDS archiving philosophy

The main objective of the PDS is to maintain a planetary data archive that will withstand the test of time such that future generations of scientists can access, understand and use preexisting planetary data. The PDS tries to ensure compatibility of the archive by adhering to strict standards of storage media, archiving formats, and required documentation.

Storage media

One critical component of the PDS archive is the storage media. The data must be stored effectively and efficiently with no degradation of the data over the archive's lifespan. Therefore, the physical media must have large capacity and must remain readable over many years. PDS is migrating toward electronic storage as its "standard" media.

Archiving formats

The format of the data is also important. In general, transparent, non-proprietary formats are best. When a proprietary format is submitted to the archive (such as a Microsoft Word document) an accompanying plain text file is also required. It is assumed that the scientists of the future will at least be able to make sense of regular ASCII bytes even if the proprietary software and support ceases to exist. PDS allows figures and illustrations to be included in the archive as individual images. PDS adheres to many other standards including, but not limited to, special directory and file naming conventions and label requirements. Each file in the PDS archive is accompanied by a

searchable label (attached or detached) that describes the file content.

Archiving documents

The archive must be complete and be able to stand alone. There is no guarantee that the people who originally worked with and submitted the data to the archive will be available in the future to field questions regarding the data, its calibration or the mission. Therefore, the archive must include good descriptive documentation of how the spacecraft and its instruments worked, how the data were collected and calibrated, and what the data mean. The quality of the documentation is examined during a mission independent PDS peer review.

Nodes

The PDS [1] is composed of 8 nodes, 6 science discipline nodes and 2 support nodes. In addition, there are several subnodes and data nodes whose exact status tends to change over time.

Science Discipline Nodes

- Atmospheres Node [2] - handles non-imaging atmospheric data (New Mexico State University)
- Geosciences Node [3] - handles data of the surfaces and interiors of terrestrial planetary bodies (Washington University)
- Imaging Node [4] - archives many of the larger planetary image data collections (Astrogeology Research Program of the United States Geological Survey, and Jet Propulsion Laboratory)
- Planetary Plasma Interaction (PPI) Node [5] - handles data consisting of the interaction between the solar wind and planetary winds with planetary magnetospheres, ionospheres and surfaces (University of California, Los Angeles)
- Planetary Rings Node [6] - handles planetary ring system data (SETI Institute)
- Small Bodies Node (SBN) [7] - handles asteroid, comet and planetary dust data (University of Maryland, College Park)
  - Comet Subnode (University of Maryland, College Park)
  - Asteroid/Interplanetary Dust Subnode (Planetary Science Institute)

Support Nodes

- Engineering Node [8] - provides systems engineering support to the PDS (Jet Propulsion Laboratory)
- Navigation and Ancillary Information Facility (NAIF) Node [9] - maintains the SPICE information system (Jet Propulsion Laboratory)

External links

- Official NASA PDS site [10]

References

[1] http://pds.jpl.nasa.gov
[2] http://pds-atmospheres.nmsu.edu/
[3] http://pds-geosciences.wustl.edu/
[4] http://img.pds.nasa.gov/
[5] http://pds-ppi.igpp.ucla.edu
[6] http://pds-rings.seti.org/
[7] http://pds-smallbodies.astro.umd.edu/
[8] http://pds-engnode.jpl.nasa.gov/
[9] http://naif.jpl.nasa.gov/naif/
[10] http://pds.nasa.gov/

Parameter Value Language

In computer programming, **Parameter Value Language (PVL)** is a markup language similar to XML. It is commonly employed for entries in the Planetary Database System used by NASA to store mission data, among other uses.

External links

- Specification [1]

References

[1] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=26095

National Data Repository

A **National Data Repository** (**NDR**) is a data bank that seeks to preserve and promote a country's natural resources data, particularly data related to the petroleum exploration and production (E&P) sector.

A National Data Repository is normally established by an entity that governs, controls and supports the exchange, capture, transference and distribution of E&P information, with the final target to provide the State with the tools and information to assure the growth, govern-ability, control, independence and sovereignty of the industry.

The two fundamental reasons for a country to establish an NDR are to **preserve** data generated inside the country by the industry, and to **promote** investments in the country by utilizing data to reduce the exploration, production, and transportation business risks.

Countries take different approaches towards preserving and promoting their natural resources data. The approach varies according to a country's natural resources policies, level of openness, and its attitude towards foreign investment.

Data types

NDRs store a vast array of data related to a country's natural resources. This includes wells, well log data, well reports, core samples, seismic surveys, post-stack seismic, field data/tapes, seismic (acquisition/processing) reports, production data, geological maps and reports, license data and geological models.

Funding models

Some NDRs are financed entirely by a country's government. Others are industry-funded. Still some are hybrid systems, funded in part by industry and government. NDRs typically charge fees for data requests and for data loading. The cost differs significantly between countries. In some cases an annual membership is charged to oil companies to store and access the data in the NDR.

Standards body

Energistics is the global energy standards resource center for the upstream oil and gas industry.

Energistics National Data Repository Work Group: The standards body is Energistics.[1]

Energistics-standards-directory

Global regulators of upstream oil and natural gas information, including seismic, drilling, production and reservoir data, formed the National Data Repository (NDR) Work Group in 2008 to collaborate on the development of data management standards and to assist emerging nations with hydrocarbon reserves to better collect, maintain and deliver oil and gas data to the public and to the industry.

Ten countries, led by the Netherlands, Norway and the United Kingdom, formed NDR to share best practices and to formalize the development and deployment of data management standards for regulatory agencies. The other countries involved in the NDR Work Group's formation are Australia, Canada, India, Kenya, New Zealand, South Africa and the United States.

Annual NDR Conference: Approximately every 18 months Energistics organizes a National Data Repository Conference. The purpose is to provide government and regulatory agencies from around the world an opportunity to attend a series of workshops dedicated to developing data exchange standards, improving communications with the oil and gas industry and learning data management techniques for natural resources information.[2]

NDRs around the world

Click here to open this map in Google Maps [3]

Country	Name	Agency	Scope	Status	Purposes	Data types/volumes	Standards used	Funding	Website					
Algeria	BDN	ALNAFT	Onshore and Offshore Algeria	Ongoing project - agency created by new law in 2005	Custodian of all E&P data of the country	Cultural, Seismic, Wells, Logs, Production, Facilities, Economical and Fiscality, Interpretation, Physical assets index	SEGY, UKOOA, LAS, DLIS	Government funding/Agency revenue						
Colombia	EPIS	Agencia Nacional de Hidrocarburos (ANH)	Onshore and offshore Colombia	Created originally for Ecopetrol and transferred to ANH when it was established in 2003. New system launched December 2009	Promote and preserve all the technical E&P information assets of the country	wells, surveys, licenses, seismic sections, well reports, maps	REST Web services	Government funding	http://www.epis.com.co					
Canada	CNSOPB	Nova Scotia Offshore Petroleum Board – Geoscience Research Centre- Digital Data Management Centre (DMC)	Offshore Nova Scotia, Canada	Operational since 2007	To provide an effective & efficient system for the management of digital petroleum data, assist explorers in easily obtaining access to large volumes of data via the web, Data Preservation and Data Distribution	Wells, well log curves, well reports, cores and samples, field data/tapes, seismic (acquisition/processing) reports, production data, interpretative maps and reports	LAS, DLIS, SEGY	Funded 50/50 by the Federal and Provincial Governments with some funds from industry through cost recovery	http://www.cnsopb.ns.ca/ [4]					
Australia	PIMS	Geoscience Australia		Active	Various online and web based systems exist for E &P, geosciences	Wells, well log curves, well reports, cores and samples, field data/tapes, seismic (acquisition/processing) reports, production data, interpretative maps and reports			[5] [6] [7] https://vdr.ga.gov.au/					
Western Australia	WAPIMS	Government of Western Australia		Active	WAPIMS is a petroleum, geothermal and minerals exploration database	Contains data on titles, wells, geophysical surveys and other petroleum exploration and production data submitted to DMP by the petroleum industry.			http://dmp.wa.gov.au [8]					
New South Wales		Government of New South Wales		Active	Various online geoscience databases to assist New South Wales including DIGS				[9] [10]					

Northern Territory		Government of Northern Territory		Active	Various online geoscience databases to assist Northern Territories		Wells, well log curves, well reports, cores and samples, field data/tapes, seismic (acquisition/processing) reports, production data, interpretative maps and reports		[11] [12] [13] [14]				
Queensland		Government of Queensland		Active	Various online geoscience databases to assist Queensland including Q-DEX		Wells, well log curves, well reports, cores and samples, field data/tapes, seismic (acquisition/processing) reports, production data, interpretative maps and reports		[15] [16] [17]				
South Australia	PIRSA	Government of South Australia		Active	Various online geoscience databases to assist South Australia such as PEP-SA		Wells, well log curves, well reports, cores and samples, field data/tapes, seismic (acquisition/processing) reports, production data, interpretative maps and reports		[18] [19] [20]				
Tasmania					Various online geoscience databases to assist Tasmania		Active		[21] [22]	China	CNPC	Chinese National Petroleum Corporation	Various oil companies in China with CNPC the largest and parent of Petrochina	http://www.cnpc.com.cn/en/ [23] http://www.cnooc.com.cn/ [24]
Russia		Sakhalin, DIGC RDC		Various oil companies in Russia the largest being Rosneft which is state owned					http://www.rosneft.com http://www.lukoil.com http://www.tnk-bp.com/en/ http://www.surgutneftegas.ru/ http://www.gazprom-neft.com/ [25]					

Indonesia	Indonesia's National Data Centre (NDC) for petroleum, energy and minerals data	Agency for Research and Development in the Ministry of Energy and Mineral Resources of the Republic of Indonesia	Onshore & Offshore	In 1997 Indonesia established Migas Data Management (MDM) operated by PT. Patra Nusa Data (PND)	PND manages and promotes petroleum investment opportunities by compiling and value adding available petroleum data and information.				http://www.patranusa.com/				
New Zealand	New Zealand Online Exploration Database	New Zealand Petroleum & Minerals, Ministry of Business Innovation & Employment	New Zealand onshore and offshore out to the outer continental shelf.	Opened to public in April 2007.	Data preservation, Investment facilitation, aid in monitoring regulatory compliance, maximise the return to the nation by informing public policy and business strategy.	Wells, well log curves, petroleum reports (includes wells and surveys), mineral reports, coal reports, cores and samples, seismic surveys, post-stack seismic, field data/tapes, seismic acquisition/processing reports, geophysical and geochemical data acquired in mineral and coal exploration (incorporated as enclosures to reports), VSP (incorporated as enclosures to reports), Seismic survey observer logs. GIS data and projects (minerals and coal). Estimated total NDR Size: 2.5 TB loaded, 3.0 TB staged for loading, 40 TB field data offline.	Closely follow Australian digital reporting standards. No naming standards for wells and surveys.	50% Government funding, 50% third party permit (license) fees paid by exploration companies.	https://data.nzpam.govt.nz				
Jordan	NRA	Jordan Natural Resources Authority (NRA)	Onshore	Active	Online data room allows users to browse and select large data set quickly in a controlled and secure environment	Reserves land records, field data, maps, engineering, seismic data, geological studies and well files.			http://www.nra.gov.jo/				
Angola		Sonangol	Offshore Angola	Active	Promotion, Organisation & Management of all Exploration & Production (E&P) Data of Angola	Wells, surveys, licenses, seismic sections, well reports, maps		Norad/OfD and NPD assistance	http://www.sonangol.co.ao				

France	BEPH		French Territory		Interactive maps of French territory of oil data are available to Internet users which includes: Permits for petroleum exploration, seismic exploration, oil drilling (data, documents available)	Wells, Surveys, Licenses, Seismic Sections, Well Reports, Maps			http://www.beph.net/					
São Tomé and Príncipe	ANP-STP	National Petroleum Agency of São Tomé & Principe (ANP-STP)	Offshore					Norad/OfD and NPD assistance	http://www.anp-stp.gov.st/pt/					
Tanzania	TPDC	Tanzania Petroleum Development Corp		Began in the early 1990s with Norwegian assistance	An E & P data archive centre	Geophysical survey data, Geological studies, Well drilling and completion reports, Cores and drill steam data		Norad/OfD and NPD assistance	http://www.tpdc-tz.com					
Oman	OGDR	Department of Petroleum Concession, Ministry of Oil and Gas		Department of Petroleum Concession is running an effort to establish Oman Oil&Gas Data Repository (OGDR) project.					[26]					
Netherlands	DINO	The Geological Survey of the Netherlands, a division of TNO	The Netherlands including offshore waters	Started in 2004. Currently BRO is being planned to succeed DINO.	To archive subsurface data of the Netherlands in one repository and provide easy access to the data to encourage multiple use of data.		WMS web services. DINO uses own naming conventions	100% Government funding	[27]					

India	DGH	Directorate General of Hydrocarbons (DGH)		Active - scheduled operation by April 2015	Establishing national data archival, improving data quality and access for quality exploration covering large area under exploration and providing basis for long term energy policy formulation as well as support OALP	Wells, Well Logs, Cores, Scanned core images, Seismic, Reports, production, Technical Reports		Government of India	[28]				
Sri Lanka	PRDS	Ministry of Petroleum and Petroleum Resources Development		Active since 2009	The PRDS developed a website to disseminate petroleum data and information to public and to investors to assist promotion of offshore areas to attract investors for petroleum exploration	Wells, surveys, licenses, seismic sections, well reports, maps. Data historic and current, archived on different media (paper, mylar, magnetic tape)			[29]				
Argentina	ENARSA	Energia Argentina SA		Established in 2006					http://www.enarsa.com.ar [30]				
Peru	PeruPetro			Active					http://www.perupetro.com.pe				
Kazakhstan		Ministry of Energy and Mineral Resources of the Republic of Kazakhstan (MEMR)		Active					http://www.petrodata.kz				
Pakistan	PPEPDR	Directorate General Petroleum Concessions (DGPC)		Active since 2001		Repository contains more than 10 terabytes of secure petrotechnical data			http://www.ppepdr.net/				

Nigeria	Department of Petroleum Resources			Active since December 2003.	Preserve, maintain the integrity and promote the National E&P data assets with improved quality, efficiency and accessibility in the most rapid, secure and reliable manner		International and PetroBank data management standards	Funded by Establishment Costs - one-off funding by Government and Running Costs - Subscription & Transaction Fees by Operators	http://ndr.dprnigeria.com/					
Turkey	PetroBank MDS	Turkish Petroleum Corporation (TPAO). It is NOC of Turkey.	36˚-42˚ northern parallel and the 26-45˚ eastern meridian.	Operational since 2007	Data assets preservation, easy access to assets, assets access controlling and auditing, consolidation of assets, national archive, central management of all assets, standardization of assets according to international standards and naming conventions, working with the most convenient assets.	Wells, Well log curves, well reports, cores and samples, seismic surveys, post-stack seismic, field data/tapes and seismic acquisition/processing reports.	International and PetroBank data management standards	Funded fully by the Turkish Petroleum Corporation. Service usage is free of charge.	http://www.tpao.gov.tr					

Norway	DISKOS-Norwegian National Data Repository	Norwegian Petroleum Directorate (NPD) and DISKOS Group of oil companies	Norwegian continental shelf	Started in 1995	To ensure compliance with NPD reporting regulations for digital E&P data. To reduce data redundancy. To ensure that data is made generally available to the oil and gas industry and to society as a whole Long term preservation of data.	Wells, Well Log Curves, Seismic Surveys, field, pre-stack & post-stack seismic, seismic reports, production data (monthly allocated).Size of NDR estimated at 310 Terabytes.	SEG-D for seismic field data, SEG-Y for pre-stack and post-stack seismic data (currently only limited amounts of field and pre-stack data) All relevant well data standards such as LIS, DLIS, LAS, SPLA, SCAL etc. PDF and TIF are also used.	Costs are shared equally between all participating oil companies (58) in the Diskos consortium, including the NPD. In addition reporting companies pay to submit and download data. All Norwegian universities have free access to public data in Diskos. Non-oil companies can apply for Associated Membership, there are currently 15 such members.	http://www.diskos.com http://www.npd.no					
United Kingdom	CDA	CDA Common Data Access Ltd	UK Offshore Waters	Well DataStore went live in 1995. DEAL started operations in 2000. The Seismic datastore went live in 2009. Estimated NDR size: 6 Terabytes	Save costs for licenses,Improve access to data,Comply with regulations	Well log curves, Well reports, Post-stack seismic, Seismic reports, VSP, deviation and test data. Estimated NDR size: 6 Terabytes	CDA has adopted DECC's naming standards for wells and surveys and continues to work closely with DECC and industry to identify a range of standards (see the CDA and DECC websites for more on this)	Owned by the UK oil and gas industry	[31] [32] [33] http://www.cdal.com [34]					
United Kingdom	UKOGL	UK Onshore Geophysical Library	UK onshore	In operation since 1994. Managed and operated by Lynx Information Systems Ltd on behalf of UKOGL.	Custodian of all UK onshore seismic data	Seismic, well tops, logs, cultural. Current archive size approx 6TB	SEGY, UKOOA, LAS, DLIS	Self-funded through data sales	http://www.ukogl.org.uk [35]					
Brazil	ANP	Agência Nacional do Petróleo (ANP)		BDEP formed in May 2000		Stores seismic, well log, post stack and pre-stack seismic data and potential field data(Grav/Mag)	ANP standards in place	Funded by Members	http://www.bdep.gov.br					

Mexico	Ditep	Pemex		Established in 2002		Promotes and preserve all the technical E&P information assets of the country			[36]				
Israel		The Ministry of National Infrastructures		Exploratory					[37]				
Cyprus	MCIT	Ministry of Commerce, Industry and Tourism-Energy Service	Offshore	Promotional	Responsible for granting licences for prospecting, exploration and exploitation of hydrocarbons				[38]				
South Africa		Petroleum Agency of South Africa		Active	Seismic data, Well data, Samples, reports and diagrams		Standards: Formats – SEGD, SEGY, LIS, LAS, PDF and TIFF, Media – 3480, 3590, DLT, 8mm Exabyte, DAT	From 2010 funded by Government	[39]				
Kenya	National Data Center (NDC)	National Oil Corporation of Kenya	Offshore and Onshore	Began in 2007, system implemented in 2010.	Digital data preservation, National archive, to implement integrated data management systems, provide easy access to quality-controlled data for internal and external customers, attract oil and gas exploration investment and to reduce data management costs.	Wells, well log curves, well reports, post-stack seismic, field data/tapes, seismic acquisition/processing reports, interpretive maps and reports.	Seismic data- SEGY. 3590 or 3592 data cartridges.	100% Government Funded	http://www.nockenya.co.ke/				
United States	BOEMRE	Bureau of Ocean Energy, Management, Regulation and Enforcement (BOEMRE)	Gulf of Mexico	Has replaced the former Minerals Management Service (MMS)					[40]				
United States	NGRDS	National Geoscience Data Repository System (NGDRS)			NGRDS is a system of geoscience data repositories, providing information about their respective holdings accessible through a web-based supercatalog.		geologic, geophysical, and engineering data, maps, well logs, and samples	DOE has provided funds for the NGDRS since 1993	[41] http://www.energy.gov/ [42] List of Repositories in US listed also as directory [43]				

Cambodia	CNPA	Cambodia National Petroleum Authority	Onshore & Offshore		Promotion and preservation of technical E&P information assets of the country			Norad/OfD and NPD assistance	http://www.cnpa-cambodia.com/				
Afghanistan	MOM	Ministry of Mines of the Islamic Republic of Afghanistan (MoM)	Onshore		Promotion and preservation of technical E&P information assets of the country			Norad/OfD and NPD assistance	http://www.mom.gov.af [44]				
Bangladesh	MOEMR	Hydrocarbon Unit, Ministry of Power, Energy and Mineral Resources (MOEMR)	Onshore & Offshore	Active and ongoing via HCU unit since 2005	A mini-data bank has established in the HCU to handle Production data, Resource data by using Database & GIS Software 2005 and promotion of technical E&P information assets of the country		Funded assistance	Norad/OfD and NPD assistance	http://www.hcu.org.bd/ http://www.petrobangla.org.bd http://www.bapex.com.bd				
Ethiopia	MOME	Ministry of Mines and Energy Ethiopia		Active and ongoing	Promotion of technical E&P information assets of the country				[45]				
Cameroon	SNH	SNH Cameroon		Active & Ongoing	Preservation and promotion of technical E&P information assets of the country				http://www.snh.cm				
Malaysia	PIRI	Petronas		Yet to establish full NDR	Promotion and preservation of technical E&P information assets of the country				http://www.petronas.com.my				
Spain	ATH				Online GIS databases to geophyscial information SIGEOF and ATH (Archivo de Hydrocarbures)				[46] [47] [48]				

Morocco	ONHYM	Office National des Hydrocarbures et des Mines			Promotion and preservation of technical E&P information assets of the country				http://www.onhym.com				
Madagascar	OMNIS				Promotion and preservation of technical E&P information assets of the country			Norad OfD and NPD assistance					
Sudan								Norad OfD and NPD assistance					
Morocco	ONHYM	Office National des Hydrocarbures et des Mines			Promotion and preservation of technical E&P information assets of the country				http://www.onhym.com				
Nicaragua	MEM			Active & ongoing				Norad OfD and NPD assistance	http://www.ine.gob.ni http://www.mem.gob.ni				
Iraq	MoO	Ministry of Oil Republic of Iraq		Active and Ongoing since 2005	MoO establishing a centralized data base and NDR for Iraqi petroleum data and to ensure that data & information from petroleum activities is made available and attract more investors by promoting the petroleum activities	Well logs, Maps, Magnetic tapes, Core & cutting samples, Other geological and geophysical information		Norad/OfD and NPD assistance	http://www.oil.gov.iq				
Latvia	LEGMC	Latvian Environment, Geology and Meteorology Centre	Offshore & Onshore		An E & P data archive centre which provides data available for purchase	Geological (well and seismic data, maps, reports etc.)			[49]				
Albania	AKBN	National Agency of Natural Resources			Generates and promotes exploration opportunities in Albania, maintains archive of E & P data.				[50]				

Uganda	PEPD	Petroleum Exploration & Production Dept (PEPD)	Onshore			Technical E & P data archive and information		Norad/Ofd assistance	[51] [52]				
Zambia		Ministry of Mines and Minerals Development, Geological Survey Department (GSD)	Onshore	Active and ongoing		Technical E & P data archive and information - Technical Records Unit		Norad/Ofd & NPD assistance	[53]				
Ivory Coast	MME	Ministry of Mines & Energy	Onshore & Offshore					Norad/Ofd and NPD assistance	http://www.cotedivoirepr.ci/ [54]				
Romania		National Agency for Mineral Resources			Promotion and preservation of technical E&P information assets of the country				http://www.namr.ro/main_en.htm				
Fiji	SOPAC	Mineral Resources Dept Fiji		Created as SOPAC Petroleum Data Bank a cooperative effort with Geoscience Australia	SOPAC acts as custodian and primary point for E & P data & information preserved on behalf of Pacific Island member nations	Well logs, Maps, Magnetic tapes, Core & cutting samples, Other geological and geophysical information		In part externally managed	http://www.mrd.gov.fj/gfiji/ [55] [56] [57]				
Papua New Guinea	SOPAC	Department of Petroleum and Energy		Created as SOPAC Petroleum Data Bank a cooperative effort with Geoscience Australia	SOPAC acts as custodian and primary point for E & P data & information preserved on behalf of Pacific Island member nations	Well logs, Maps, Magnetic tapes, Core & cutting samples, Other geological and geophysical information		Externally managed	http://www.petroleum.gov.pg [58] [56] [59]				
Solomon Islands	SOPAC			Created as SOPAC Petroleum Data Bank a cooperative effort with Geoscience Australia	SOPAC acts as custodian and primary point for E & P data & information preserved on behalf of Pacific Island member nations	Well logs, Maps, Magnetic tapes, Core & cutting samples, Other geological and geophysical information		Externally managed	[56] [60]				

Tonga	SOPAC			Created as SOPAC Petroleum Data Bank a cooperative effort with Geoscience Australia	SOPAC acts as custodian and primary point for E & P data & information preserved on behalf of Pacific Island member nations	Well logs, Maps, Magnetic tapes, Core & cutting samples, Other geological and geophysical information		Externally managed	[56] [61]					
Vanuatu	SOPAC			Created as SOPAC Petroleum Data Bank a cooperative effort with Geoscience Australia	SOPAC acts as custodian and primary point for E & P data & information preserved on behalf of Pacific Island member nations	Well logs, Maps, Magnetic tapes, Core & cutting samples, Other geological and geophysical information		Externally managed	[56] [62]					
Guyana	GGMC	Guyana Geology and Mines Commission			Promotion and preservation of technical E&P information assets of the country				http://www.ggmc.gov.gy					
Syria	SPC	Syrian Petroleum Company							[63]					
Liberia	NOCAL	National Oil Company of Liberia			Promotion and preservation of technical E&P information assets of the country				http://www.nocal-lr.com/					
Chile	ENAP	National Oil Company of Chile			Promotion and preservation of technical E&P information assets of the country				http://www.enap.cl					
Thailand	PTTEP	PTT Exploration and Production Public Company Ltd			Promotion and preservation of technical E&P information assets of the country				http://www.pttep.com/ [64]					
Venezuela	PDVSA	Petroleos de Venezuela	Onshore & Offshore		Promotion and preservation of technical E&P information assets of the country				http://www.pdvsa.com/					

Trinidad and Tobago		Trinidad Ministry of Energy and Energy Affairs			Promotion and preservation of technical E&P information assets of the country				[65] [66]					
Mozambique	NAPD			Established in 1999 under NORAD support	To ensure that data & information from petroleum activities is made available and attract more investors by promoting the petroleum activities	Well logs, Maps, Magnetic tapes, Core & cutting samples, Other geological and geophysical information		National Budget and INP funds	http://www.inp.gov.mz					
Denmark		Danish Energy Agency			Online GIS service for wells and license data				[67]					
Dominican Republic		Directorate of Hydrocarbons							[68]					
Equatorial Guinea					Exploration databank for Equatorial Guinea				http://www.equatorialoil.com [69]					
Faroe Islands	Jardfeingi	Jardfeingi Faorese Earth and Energy Directorate			Promotion of exploration and licensing rounds				http://www.jardfeingi.fo					
Philippines	PNOC	Philippine National Oil Company			Promotion of exploration and licensing rounds				http://www.pnoc.com.ph [70]					
Greenland	NunaGIS	BMP- Bureau of Minerals and Petroleum		Online GIS with licensing round promotion for hydrocarbons.					http://www.bmp.gl http://licence-map.bmp.gl/					
Iceland	Iceland Continental Shelf Portal (ICSP)	Orkustofnunn - National Energy Authority	Offshore	The Iceland Continental Shelf Portal (ICSP)	Provides access to information about data pertaining to the Icelandic Continental Shelf, in particular initially to the northern Dreki Area to assist with licensing round promotion				http://www.os.is [71]					
Myanmar	MOGE	Myanmar Oil & Gas Enterprise							[72]					

Yemen	PEPA	Petroleum Exploration and Production Authority (PEPA)							http://www.pepa.com.ye/					
Tunisia	ETAP	Enterprise Tunisienne D'Activities Petrolieres			Promotion and preservation of technical E&P information assets of the country				http://www.etap.com.tn					
Gabon	DGH	Direction Generale des Hydrocarbures (DGH)							[73]					
Congo	SNPC	Société Nationale des pétroles du Congo												
Mali	Aurep	Autorite pour la Promotion de la Recherce des Petroliere au Mali			Databank service managing the geological and geophysical data relative to petroleum research.				http://www.aurep.org [74]					
Guatemala	MEM	Dirección General de Hidrocarbures			Online maps and images of wells, seismic, licenses, protected areas, exploration and production				[75] [76]					
Iran	NIOC	National Oil Company of Iran							http://www.nioc.ir					
Libya	NOC	NOC Libya			Virtual data room in place for promotion of exploration and exploitation of hydrocarbons									
United Arab Emirates	ADNOC	Abu Dhabi National Oil Company							http://www.adnoc.ae					
Qatar		Qatar Petroleum							http://www.qp.com.qa					
South Korea	KNOC	Korea National Petroleum Corporation							http://www.knoc.co.kr					
Seychelles	SNOC	Seychelles National Oil Company												

Saudi Arabia		Saudi Aramco							http://www.saudiaramco.com					
Belarus									[77] [78]					
Timor-Leste	LAFAEK	Autoridade Nacional do Petróleo			Online GIS with wells and licences			Norad/OfD assistance	[79]					

Notes

[1] Energistics (http://energistics.org/energistics-standards-directory)
[2] NDR Conference page on the Energistics website (http://www.energistics.org/ndr10)
[3] http://maps.google.com/maps/ms?ie=UTF&msa=0&msid=116086457121594113105.0004962de0b54da84ae44
[4] http://ww1.cnsopbdmc.ca/dp/controller/PLEASE_LOGIN_PAGE
[5] http://en.wikipedia.org/wiki/Geoscience_Australia
[6] http://dbforms.ga.gov.au/pls/www/npm.pims_web.search
[7] http://dbforms.ga.gov.au/www/npm.well.search
[8] https://wapims.doir.wa.gov.au/dp/index.jsp
[9] http://www.dpi.nsw.gov.au/minerals
[10] http://digsopen.minerals.nsw.gov.au/
[11] http://www.nt.gov.au/d/Minerals_Energy/index.cfm?header=Petroleum
[12] http://apps.minerals.nt.gov.au/irmspet/
[13] http://www.ga.gov.au/oracle/npd/
[14] http://apps.minerals.nt.gov.au/strike/CLIENT/display/GeoSamba.asphttp://www.dpi.nsw.gov.au/minerals
[15] http://www.dme.qld.gov.au/mines/about_us.cfm
[16] http://www.dme.qld.gov.au/mines/petroleum_gas.cfm
[17] http://www.dme.qld.gov.au/mines/interactive_resource_data.cfm
[18] http://www.pir.sa.gov.au/petroleum
[19] http://www.pir.sa.gov.au/petroleum/access_to_data/well_completion_reports
[20] http://www.pir.sa.gov.au/petroleum/access_to_data/peps-sa_database
[21] http://www.mrt.tas.gov.au/portal/page?_pageid=35,1&_dad=portal&_schema=PORTAL
[22] http://www.mrt.tas.gov.au/portal/page?_pageid=35,856405&_dad=portal&_schema=PORTAL|-
[23] http://www.petrochina.com.cn/ptr/
[24] http://english.sinopec.com/index.shtml
[25] http://www.tatneft.ru/wps/wcm/connect/tatneft/portal_rus/homepage/
[26] http://www.mog.gov.om/tabid/54/Default.aspx
[27] http://www.nlog.nl/en/home/NLOGPortal.html
[28] http://www.dghindia.org/DataManagement.aspx#
[29] http://www.prds-srilanka.com/data/onlineData.faces
[30] http://energia.mecon.gov.ar/upstream/US_Pterminados.asp
[31] https://www.cdadatastore.com/dp/jsp/PleaseLogin.jsp
[32] https://www.ukdeal.co.uk/dp/jsp/PleaseLoginDeal.jsp
[33] https://www.og.decc.gov.uk/regulation/pons/index.htm
[34] https://www.ukdeal.co.uk/dp/pages/deal/CDA%20Seismic%20DataStore%20Price%20List.pdf
[35] http://maps.lynxinfo.co.uk/UKOGL_LIVE/map.html
[36] http://www.pep.pemex.com/index.html
[37] http://www.mni.gov.il/mni/en-US/NaturalResources/OilandgasExploration/OilMaps/
[38] http://www.mcit.gov.cy/mcit/mcit.nsf/dmlhexploration_en/dmlhexploration_en?OpenDocument
[39] http://www.petroleumagencysa.com
[40] http://www.gomr.boemre.gov/homepg/data_center.html
[41] http://www.agiweb.org/ngdrs/index.html
[42] http://www.agiweb.org/index.html
[43] http://www.agiweb.org/ngdrs/overview/datadirectory.html
[44] http://www.afghanistanpetroleum.com
[45] http://www.mome.gov.et/petroleum.html
[46] http://www.mityc.es/energia/petroleo/Exploracion/Paginas/Estadisticas.aspx
[47] http://hidrocarburos.mityc.es/ath/

[48] http://www.igme.es/internet/sigeof/INICIOsiGEOF.htm
[49] http://mapx.map.vgd.gov.lv/geo3/VGD_OIL_PAGE/index.htm
[50] http://www.akbn.gov.al/index.php?ak=details&cid=5&lng=en
[51] http://www.statehouse.go.ug/government.php?catId=10
[52] http://www.energyandminerals.go.ug
[53] http://www.zambiageosurvey.gov.zm/
[54] http://www.petroci.ci/index.php?numlien=31
[55] http://www.mrd.gov.fj/gfiji/petroleum/petroleum.html
[56] http://www.ga.gov.au/energy/projects/pacific-islands-applied-geoscience-commission.html
[57] http://www.sopac.org/index.php/member-countries/fiji-islands
[58] http://www.petrominpng.com.pg/about.html
[59] http://www.sopac.org/index.php/member-countries/papua-new-guinea
[60] http://www.sopac.org/index.php/member-countries/solomon-islands
[61] http://www.sopac.org/index.php/member-countries/tonga
[62] http://www.sopac.org/index.php/member-countries/vanuatu
[63] http://www.spc-sy.com/en/aboutus/aboutus1_en.php
[64] http://www.pttep.com/en/index.aspx
[65] http://www.energy.gov.tt/energy_industry.php?mid=31
[66] http://www.petrotrin.com/Petrotrin2007/UpstreamBusiness.htm
[67] http://www.ens.dk/EN-US/OILANDGAS/Sider/Oilandgas.aspx
[68] http://www.dgm.gov.do/sdhidrocarburo/index.html
[69] http://www.equatorialoil.com/database.html
[70] http://www.pnoc-ec.com.ph/business.php?id=2
[71] http://www.nea.is/oil-and-gas-exploration/
[72] http://www.energy.gov.mm/upstreampetroleumsubsector.htm
[73] http://www.gabon-industriel.com/les-actions/energie/petrole
[74] http://www.aurep.org/htmlpages/mali.html
[75] http://www.mem.gob.gt/Portal/home.aspx
[76] http://www.mem.gob.gt/Portal/Home.aspx?secid=25
[77] http://geologiya.org/index.php?categoryid=14
[78] http://minpriroda.by/ru/napravlenia/minsyrbaza
[79] http://www.anp-tl.org/webs/anptlweb.nsf/pgMaps

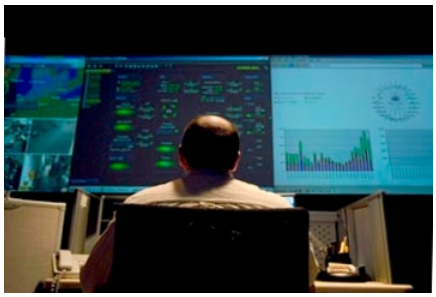
External links

- Energistics: National Data Repository Group (http://www.energistics.org/national-data-repository-ndr)
- LinkedIN: National Data Repository Group (http://www.linkedin.com/groups?home=&gid=1824856)
- National Data Repositories: the case for open data in the oil and gas industry (http://www.kadme.com/wp-content/uploads/KADME-Oil-and-Gas-Technology-Jan2011.pdf)

Data Centers

Data center

A **data center** is a facility used to house computer systems and associated components, such as telecommunications and storage systems. It generally includes redundant or backup power supplies, redundant data communications connections, environmental controls (e.g., air conditioning, fire suppression) and various security devices. Large data centers are industrial scale operations using as much electricity as a small town and sometimes are a significant source of air pollution in the form of diesel exhaust.



An operation engineer overseeing a network operations control room of a data center

History

Data centers have their roots in the huge computer rooms of the early ages of the computing industry. Early computer systems were complex to operate and maintain, and required a special environment in which to operate. Many cables were necessary to connect all the components, and methods to accommodate and organize these were devised, such as standard racks to mount equipment, raised floors, and cable trays (installed overhead or under the elevated floor). Also, a single mainframe required a great deal of power, and had to be cooled to avoid overheating. Security was important – computers were expensive, and were often used for military purposes. Basic design guidelines for controlling access to the computer room were therefore devised.

Indiana University Data Center. Bloomington, Indiana

During the boom of the microcomputer industry, and especially during the 1980s, computers started to be deployed everywhere, in many cases with little or no care about operating requirements. However, as information technology (IT) operations started to grow in complexity, companies grew aware of the need to control IT resources. With the advent of Linux and the subsequent proliferation of freely available Unix compatible PC operating systems during the 1990s, as well as MS-DOS finally giving way to a multi-tasking capable Windows OS, PCs started to find their places in the old computer rooms. These were called "servers" as timesharing operating systems like Unix rely heavily on the client-server model to facilitate sharing unique resources between multiple users. The availability of inexpensive networking equipment, coupled with new standards for network structured cabling, made it possible to use a hierarchical design that put the servers in a specific room inside the company. The use of the term "data center," as applied to specially designed computer rooms, started to gain popular recognition about this time.

The boom of data centers came during the dot-com bubble. Companies needed fast Internet connectivity and nonstop operation to deploy systems and establish a presence on the Internet. Installing such equipment was not viable for many smaller companies. Many companies started building very large facilities, called Internet data centers (IDCs),

which provide businesses with a range of solutions for systems deployment and operation. New technologies and practices were designed to handle the scale and the operational requirements of such large-scale operations. These practices eventually migrated toward the private data centers, and were adopted largely because of their practical results. Data centers for cloud computing are called cloud data centers (CDCs). But nowadays, the division of these terms has almost disappeared and they are being integrated into a term "data center."

With an increase in the uptake of cloud computing, business and government organizations are scrutinizing data centers to a higher degree in areas such as security, availability, environmental impact and adherence to standards. Standard Documents from accredited professional groups, such as the Telecommunications Industry Association, specify the requirements for data center design. Well-known operational metrics for data center availability can be used to evaluate the business impact of a disruption. There is still a lot of development being done in operation practice, and also in environmentally friendly data center design. Data centers are typically very expensive to build and maintain.

Requirements for modern data centers

Racks of telecommunications equipment in part of a data center

IT operations are a crucial aspect of most organizational operations around the world. One of the main concerns is **business continuity**; companies rely on their information systems to run their operations. If a system becomes unavailable, company operations may be impaired or stopped completely. It is necessary to provide a reliable infrastructure for IT operations, in order to minimize any chance of disruption. Information security is also a concern, and for this reason a data center has to offer a secure environment which minimizes the chances of a security breach. A data center must therefore keep high standards for assuring the integrity and functionality of its hosted computer environment. This is accomplished through redundancy of both fiber optic cables and power, which includes emergency backup power generation.

The Telecommunications Industry Association's TIA-942 Telecommunications Infrastructure Standard for Data Centers [1], specifies the minimum requirements for telecommunications infrastructure of data centers and computer rooms including single tenant enterprise data centers and multi-tenant Internet hosting data centers. The topology proposed in this document is intended to be applicable to any size data center.[2]

Telcordia GR-3160, *NEBS Requirements for Telecommunications Data Center Equipment and Spaces* [3], provides guidelines for data center spaces within telecommunications networks, and environmental requirements for the equipment intended for installation in those spaces. These criteria were developed jointly by Telcordia and industry representatives. They may be applied to data center spaces housing data processing or Information Technology (IT) equipment. The equipment may be used to:

- Operate and manage a carrier's telecommunication network
- Provide data center based applications directly to the carrier's customers
- Provide hosted applications for a third party to provide services to their customers
- Provide a combination of these and similar data center applications

Effective data center operation requires a balanced investment in both the facility and the housed equipment. The first step is to establish a baseline facility environment suitable for equipment installation. Standardization and modularity can yield savings and efficiencies in the design and construction of telecommunications data centers.

Standardization means integrated building and equipment engineering. Modularity has the benefits of scalability and easier growth, even when planning forecasts are less than optimal. For these reasons, telecommunications data centers should be planned in repetitive building blocks of equipment, and associated power and support

(conditioning) equipment when practical. The use of dedicated centralized systems requires more accurate forecasts of future needs to prevent expensive over construction, or perhaps worse — under construction that fails to meet future needs.

The "lights-out" data center, also known as a darkened or a dark data center, is a data center that, ideally, has all but eliminated the need for direct access by personnel, except under extraordinary circumstances. Because of the lack of need for staff to enter the data center, it can be operated without lighting. All of the devices are accessed and managed by remote systems, with automation programs used to perform unattended operations. In addition to the energy savings, reduction in staffing costs and the ability to locate the site further from population centers, implementing a lights-out data center reduces the threat of malicious attacks upon the infrastructure.

There is a trend to modernize data centers in order to take advantage of the performance and energy efficiency increases of newer IT equipment and capabilities, such as cloud computing. This process is also known as data center transformation.[4]

Organizations are experiencing rapid IT growth but their data centers are aging. Industry research company International Data Corporation (IDC) puts the average age of a data center at nine years old. Gartner, another research company says data centers older than seven years are obsolete.

In May 2011, data center research organization Uptime Institute, reported that 36 percent of the large companies it surveyed expect to exhaust IT capacity within the next 18 months.[5]

Data center transformation takes a step-by-step approach through integrated projects carried out over time. This differs from a traditional method of data center upgrades that takes a serial and siloed approach.[6] The typical projects within a data center transformation initiative include standardization/consolidation, virtualization, automation and security.

- Standardization/consolidation: The purpose of this project is to reduce the number of data centers a large organization may have. This project also helps to reduce the number of hardware, software platforms, tools and processes within a data center. Organizations replace aging data center equipment with newer ones that provide increased capacity and performance. Computing, networking and management platforms are standardized so they are easier to manage.[7]
- Virtualize: There is a trend to use IT virtualization technologies to replace or consolidate multiple data center equipment, such as servers. Virtualization helps to lower capital and operational expenses,[8] and reduce energy consumption.[9] Virtualization technologies are also used to create virtual desktops, which can then be hosted in data centers and rented out on a subscription basis. Data released by investment bank Lazard Capital Markets reports that 48 percent of enterprise operations will be virtualized by 2012. Gartner views virtualization as a catalyst for modernization.[10]
- Automating: Data center automation involves automating tasks such as provisioning, configuration, patching, release management and compliance. As enterprises suffer from few skilled IT workers, automating tasks make data centers run more efficiently.
- Securing: In modern data centers, the security of data on virtual systems is integrated with existing security of physical infrastructures.[11] The security of a modern data center must take into account physical security, network security, and data and user security.

Carrier neutrality

Today many data centers are run by Internet service providers solely for the purpose of hosting their own and third party servers.

However traditionally data centers were either built for the sole use of one large company, or as carrier hotels or Network-neutral data centers.

These facilities enable interconnection of carriers and act as regional fiber hubs serving local business in addition to hosting content servers.

Data center tiers

The Telecommunications Industry Association is a trade association accredited by ANSI (American National Standards Institute). In 2005 it published ANSI/TIA-942 [12], Telecommunications Infrastructure Standard for Data Centers, which defined four levels (called tiers) of data centers in a thorough, quantifiable manner. TIA-942 was amended in 2008 and again in 2010. *TIA-942:Data Center Standards Overview* describes the requirements for the data center infrastructure. The simplest is a Tier 1 data center, which is basically a server room, following basic guidelines for the installation of computer systems. The most stringent level is a Tier 4 data center, which is designed to host mission critical computer systems, with fully redundant subsystems and compartmentalized security zones controlled by biometric access controls methods. Another consideration is the placement of the data center in a subterranean context, for data security as well as environmental considerations such as cooling requirements.[13]

The German Datacenter star audit program uses an auditing process to certify 5 levels of "gratification" that affect Data Center criticality.

Independent from the ANSI/TIA-942 standard, the Uptime Institute, a think tank and professional-services organization based in Santa Fe, New Mexico, has defined its own four levels. The levels describe the availability of data from the hardware at a location. The higher the tier, the greater the availability. The levels are: [14] [15]

Tier Level	Requirements
1	• Single non-redundant distribution path serving the IT equipment • Non-redundant capacity components • Basic site infrastructure with expected availability of 99.671%
2	• Meets or exceeds all Tier 1 requirements • Redundant site infrastructure capacity components with expected availability of 99.741%
3	• Meets or exceeds all Tier 2 requirements • Multiple independent distribution paths serving the IT equipment • All IT equipment must be dual-powered and fully compatible with the topology of a site's architecture • Concurrently maintainable site infrastructure with expected availability of 99.982%
4	• Meets or exceeds all Tier 3 requirements • All cooling equipment is independently dual-powered, including chillers and heating, ventilating and air-conditioning (HVAC) systems • Fault-tolerant site infrastructure with electrical power storage and distribution facilities with expected availability of 99.995%

The difference between 99.671%, 99.741%, 99.982%, and 99.995%, while seemingly nominal, could be significant depending on the application.

Whilst no down-time is ideal, the tier system allows the below durations for services to be unavailable within one year (525,600 minutes):

- Tier 1 (99.671%) status would allow 1729.224 minutes
- Tier 2 (99.741%) status would allow 1361.304 minutes
- Tier 3 (99.982%) status would allow 94.608 minutes

- Tier 4 (99.995%) status would allow 26.28 minutes

Design considerations

A data center can occupy one room of a building, one or more floors, or an entire building. Most of the equipment is often in the form of servers mounted in 19 inch rack cabinets, which are usually placed in single rows forming corridors (so-called aisles) between them. This allows people access to the front and rear of each cabinet. Servers differ greatly in size from 1U servers to large freestanding storage silos which occupy many square feet of floor space. Some equipment such as mainframe computers and storage devices are often as big as the racks themselves, and are placed alongside them. Very large data centers may use shipping containers packed with 1,000 or more servers each; when repairs or upgrades are needed, whole containers are replaced (rather than repairing individual servers).

A typical server rack, commonly seen in colocation

Local building codes may govern the minimum ceiling heights.

Design programming

Design programming, also known as architectural programming, is the process of researching and making decisions to identify the scope of a design project.[16] Other than the architecture of the building itself there are three elements to design programming for data centers: facility topology design (space planning), engineering infrastructure design (mechanical systems such as cooling and electrical systems including power) and technology infrastructure design (cable plant). Each will be influenced by performance assessments and modelling to identify gaps pertaining to the owner's performance wishes of the facility over time.

Various vendors who provide data center design services define the steps of data center design slightly differently, but all address the same basic aspects as given below.

Modeling criteria

Modeling criteria are used to develop future-state scenarios for space, power, cooling, and costs.[17] The aim is to create a master plan with parameters such as number, size, location, topology, IT floor system layouts, and power and cooling technology and configurations.

Design recommendations

Design recommendations/plans generally follow the modelling criteria phase. The optimal technology infrastructure is identified and planning criteria is developed, such as critical power capacities, overall data center power requirements using an agreed upon PUE (power utilization efficiency), mechanical cooling capacities, kilowatts per cabinet, raised floor space, and the resiliency level for the facility.

Conceptual design

Conceptual designs embody the design recommendations or plans and should take into account “what-if” scenarios to ensure all operational outcomes are met in order to future-proof the facility. Conceptual floor layouts should be driven by IT performance requirements as well as lifecycle costs associated with IT demand, energy efficiency, cost efficiency and availability. Future-proofing will also include expansion capabilities, often provided in modern data centers through modularity.

Detail design

Detail design is undertaken once the appropriate conceptual design is determined, typically including a proof of concept. The detail design phase should include the development of facility schematics and construction documents as well as schematic of technology infrastructure, detailed IT infrastructure design and IT infrastructure documentation.

Mechanical engineering infrastructure design

Mechanical engineering infrastructure design addresses mechanical systems involved in maintaining the interior environment of a data center, such as heating, ventilation and air conditioning (HVAC); humidification and dehumidification equipment; pressurization; and so on.[18] This stage of the design process should be aimed at saving space and costs, while ensuring business and reliability objectives are met as well as achieving PUE and green requirements.[19] Modern designs include modularizing and scaling IT loads, and making sure capital spending on the building construction is optimized.

CRAC Air Handler

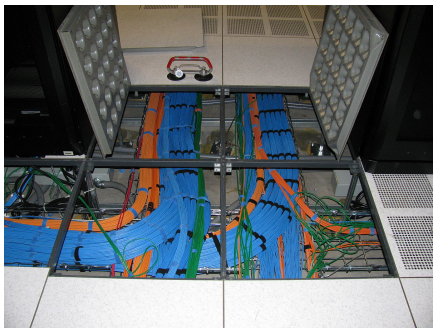
Electrical engineering infrastructure design

Electrical Engineering infrastructure design is focused on designing electrical configurations that accommodate various reliability requirements and data center sizes. Aspects may include utility service planning; distribution, switching and bypass from power sources; uninterruptable power source (UPS) systems; and more.

These designs should dovetail to energy standards and best practices while also meeting business objectives. Electrical configurations should be optimized and operationally compatible with the data center user’s capabilities. Modern electrical design is modular and scalable,[20] and is available for low and medium voltage requirements as well as DC (direct current).

Technology infrastructure design

Technology infrastructure design addresses the telecommunications cabling systems that run throughout data centers. There are cabling systems for all data center environments, including horizontal cabling, voice, modem, and facsimile telecommunications services, premises switching equipment, computer and telecommunications management connections, keyboard/video/mouse connections and data communications.[21] Wide area, local area, and storage area networks should link with other building signaling systems (e.g. fire, security, power, HVAC, EMS).



Under Floor Cable Runs

Availability expectations

The higher the availability needs of a data center, the higher the capital and operational costs of building and managing it. Business needs should dictate the level of availability required and should be evaluated based on characterization of the criticality of IT systems estimated cost analyses from modeled scenarios. In other words, how can an appropriate level of availability best be met by design criteria to avoid financial and operational risks as a result of downtime? If the estimated cost of downtime within a specified time unit exceeds the amortized capital costs and operational expenses, a higher level of availability should be factored into the data center design. If the cost of avoiding downtime greatly exceeds the cost of downtime itself, a lower level of availability should be factored into the design.[22]

Site selection

Aspects such as proximity to available power grids, telecommunications infrastructure, networking services, transportation lines and emergency services can affect costs, risk, security and other factors to be taken into consideration for data center design. Location affects data center design also because the climatic conditions dictate what cooling technologies should be deployed. In turn this impacts uptime and the costs associated with cooling.[23] For example, the topology and the cost of managing a data center in a warm, humid climate will vary greatly from managing one in a cool, dry climate.

Modularity and flexibility

Modularity and flexibility are key elements in allowing for a data center to grow and change over time. Data center modules are pre-engineered, standardized building blocks that can be easily configured and moved as needed.[24]

A modular data center may consist of data center equipment contained within shipping containers or similar portable containers.[25] But it can also be described as a design style in which components of the data center are prefabricated and standardized so that they can be constructed, moved or added to quickly as needs change.[26]

Cabinet Aisle in a Data Center

Environmental control

The physical environment of a data center is rigorously controlled. Air conditioning is used to control the temperature and humidity in the data center. ASHRAE's "Thermal Guidelines for Data Processing Environments" recommends a temperature range of 18–27 °C (64–81 °F), a dew point range of 5–15 °C (41–59 °F), and a maximum relative humidity of 60% for data center environments. The temperature in a data center will naturally rise because the electrical power used heats the air. Unless the heat is removed, the ambient temperature will rise, resulting in electronic equipment malfunction. By controlling the air temperature, the server components at the board level are kept within the manufacturer's specified temperature/humidity range. Air conditioning systems help control humidity by cooling the return space air below the dew point. Too much humidity, and water may begin to condense on internal components. In case of a dry atmosphere, ancillary humidification systems may add water vapor if the humidity is too low, which can result in static electricity discharge problems which may damage components. Subterranean data centers may keep computer equipment cool while expending less energy than conventional designs.

Modern data centers try to use economizer cooling, where they use outside air to keep the data center cool. At least one data center (located in Upstate New York) will cool servers using outside air during the winter. They do not use chillers/air conditioners, which creates potential energy savings in the millions.

Telcordia GR-2930, *NEBS: Raised Floor Generic Requirements for Network and Data Centers* [27], presents generic engineering requirements for raised floors that fall within the strict NEBS guidelines.

There are many types of commercially available floors that offer a wide range of structural strength and loading capabilities, depending on component construction and the materials used. The general types of raised floors include stringerless, stringered, and structural platforms, all of which are discussed in detail in GR-2930 and summarized below.

- ***Stringerless raised floors*** - One non-earthquake type of raised floor generally consists of an array of pedestals that provide the necessary height for routing cables and also serve to support each corner of the floor panels. With this type of floor, there may or may not be provisioning to mechanically fasten the floor panels to the pedestals. This stringerless type of system (having no mechanical attachments between the pedestal heads) provides maximum accessibility to the space under the floor. However, stringerless floors are significantly weaker than stringered raised floors in supporting lateral loads and are not recommended.
- ***Stringered raised floors*** - This type of raised floor generally consists of a vertical array of steel pedestal assemblies (each assembly is made up of a steel base plate, tubular upright, and a head) uniformly spaced on two-foot centers and mechanically fastened to the concrete floor. The steel pedestal head has a stud that is

inserted into the pedestal upright and the overall height is adjustable with a leveling nut on the welded stud of the pedestal head.

- ***Structural platforms*** - One type of structural platform consists of members constructed of steel angles or channels that are welded or bolted together to form an integrated platform for supporting equipment. This design permits equipment to be fastened directly to the platform without the need for toggle bars or supplemental bracing. Structural platforms may or may not contain panels or stringers.

Data centers typically have raised flooring made up of 60 cm (2 ft) removable square tiles. The trend is towards 80–100 cm (31–39 in) void to cater for better and uniform air distribution. These provide a plenum for air to circulate below the floor, as part of the air conditioning system, as well as providing space for power cabling.

Metal whiskers

Raised floors and other metal structures such as cable trays and ventilation ducts have caused many problems with zinc whiskers in the past, and likely are still present in many data centers. This happens when microscopic metallic filaments form on metals such as zinc or tin that protect many metal structures and electronic components from corrosion. Maintenance on a raised floor or installing of cable etc. can dislodge the whiskers, which enter the airflow and may short circuit server components or power supplies, sometimes through a high current metal vapor plasma arc. This phenomenon is not unique to data centers, and has also caused catastrophic failures of satellites and military hardware.

Electrical power

Backup power consists of one or more uninterruptible power supplies, battery banks, and/or diesel / gas turbine generators.[28]

To prevent single points of failure, all elements of the electrical systems, including backup systems, are typically fully duplicated, and critical servers are connected to both the "A-side" and "B-side" power feeds. This arrangement is often made to achieve N+1 redundancy in the systems. Static transfer switches are sometimes used to ensure instantaneous switchover from one supply to the other in the event of a power failure.

A bank of batteries in a large data center, used to provide power until diesel generators can start

Low-voltage cable routing

Data cabling is typically routed through overhead cable trays in modern data centers. But someWikipedia:Avoid weasel words are still recommending under raised floor cabling for security reasons and to consider the addition of cooling systems above the racks in case this enhancement is necessary. Smaller/less expensive data centers without raised flooring may use anti-static tiles for a flooring surface. Computer cabinets are often organized into a hot aisle arrangement to maximize airflow efficiency.

Fire protection

FM200 Fire Suppression Tanks

Data centers feature fire protection systems, including passive and active design elements, as well as implementation of fire prevention programs in operations. Smoke detectors are usually installed to provide early warning of a fire at its incipient stage. This allows investigation, interruption of power, and manual fire suppression using hand held fire extinguishers before the fire grows to a large size. An active fire protection system, such as a fire sprinkler system or a clean agent fire suppression gaseous system, is often provided to control a full scale fire if it develops. High sensitivity smoke detectors, such as Aspirating smoke detectors, activating clean agent fire suppression gaseous systems activate earlier than fire sprinklers. However, as gaseous systems have a limited fire suppression agent storage quantity, the provision of a clean agent system and a sprinkler system protects the building should the fire reignite after the gaseous agent has dispersed. Passive fire protection elements include the installation of fire walls around the data center, so a fire can be restricted to a portion of the facility for a limited time in the event of the failure of the active fire protection systems. Fire wall penetrations into the server room, such as cable penetrations, coolant line penetrations and air ducts, must be provided with fire rated penetration assemblies, such as fire stoping.

Security

Physical security also plays a large role with data centers. Physical access to the site is usually restricted to selected personnel, with controls including bollards and mantraps. Video camera surveillance and permanent security guards are almost always present if the data center is large or contains sensitive information on any of the systems within. The use of finger print recognition mantraps is starting to be commonplace.

Energy use


Google Data Center, The Dalles, Oregon

Energy use is a central issue for data centers. Power draw for data centers ranges from a few kW for a rack of servers in a closet to several tens of MW for large facilities. Some facilities have power densities more than 100 times that of a typical office building. For higher power density facilities, electricity costs are a dominant operating expense and account for over 10% of the total cost of ownership (TCO) of a data center.[29] By 2012 the cost of power for the data center is expected to exceed the cost of the original capital investment.

Greenhouse gas emissions

In 2007 the entire information and communication technologies or ICT sector was estimated to be responsible for roughly 2% of global carbon emissions with data centers accounting for 14% of the ICT footprint. The US EPA estimates that servers and data centers are responsible for up to 1.5% of the total US electricity consumption, or roughly .5% of US GHG emissions,[30] for 2007. Given a business as usual scenario greenhouse gas emissions from data centers is projected to more than double from 2007 levels by 2020.

Siting is one of the factors that affect the energy consumption and environmental effects of a datacenter. In areas where climate favors cooling and lots of renewable electricity is available the environmental effects will be more

moderate. Thus countries with favorable conditions, such as: Canada,[31] Finland,[32] Sweden[33] and Switzerland,[34] are trying to attract cloud computing data centers.

In an 18-month investigation by scholars at Rice University's Baker Institute for Public Policy in Houston and the Institute for Sustainable and Applied Infodynamics in Singapore, data center-related emissions will more than triple by 2020.

Energy efficiency

The most commonly used metric to determine the energy efficiency of a data center is power usage effectiveness, or PUE. This simple ratio is the total power entering the data center divided by the power used by the IT equipment.

$$\mathrm{PUE} = \frac{\text{Total Facility Power}}{\text{IT Equipment Power}}$$

Power used by support equipment, often referred to as overhead load, mainly consists of cooling systems, power delivery, and other facility infrastructure like lighting. The average data center in the US has a PUE of 2.0, meaning that the facility uses one watt of overhead power for every watt delivered to IT equipment. State-of-the-art data center energy efficiency is estimated to be roughly 1.2. Some large data center operators like Microsoft and Yahoo! have published projections of PUE for facilities in development; Google publishes quarterly actual efficiency performance from data centers in operation.

The U.S. Environmental Protection Agency has an Energy Star rating for standalone or large data centers. To qualify for the ecolabel, a data center must be within the top quartile of energy efficiency of all reported facilities.[35]

European Union also has a similar initiative: EU Code of Conduct for Data Centres

Energy use analysis

Often, the first step toward curbing energy use in a data center is to understand how energy is being used in the data center. Multiple types of analysis exist to measure data center energy use. Aspects measured include not just energy used by IT equipment itself, but also by the data center facility equipment, such as chillers and fans.[36]

Power and cooling analysis

Power is the largest recurring cost to the user of a data center. A power and cooling analysis, also referred to as a thermal assessment, measures the relative temperatures in specific areas as well as the capacity of the cooling systems to handle specific ambient temperatures.[37] A power and cooling analysis can help to identify hot spots, over-cooled areas that can handle greater power use density, the breakpoint of equipment loading, the effectiveness of a raised-floor strategy, and optimal equipment positioning (such as AC units) to balance temperatures across the data center. Power cooling density is a measure of how much square footage the center can cool at maximum capacity.

Energy efficiency analysis

An energy efficiency analysis measures the energy use of data center IT and facilities equipment. A typical energy efficiency analysis measures factors such as a data center's power use effectiveness (PUE) against industry standards, identifies mechanical and electrical sources of inefficiency, and identifies air-management metrics.[38]

Computational fluid dynamics (CFD) analysis

This type of analysis uses sophisticated tools and techniques to understand the unique thermal conditions present in each data center—predicting the temperature, airflow, and pressure behavior of a data center to assess performance and energy consumption, using numerical modeling.[39] By predicting the effects of these environmental conditions, CFD analysis in the data center can be used to predict the impact of high-density racks mixed with low-density

racks[40] and the onward impact on cooling resources, poor infrastructure management practices and AC failure of AC shutdown for scheduled maintenance.

Thermal zone mapping

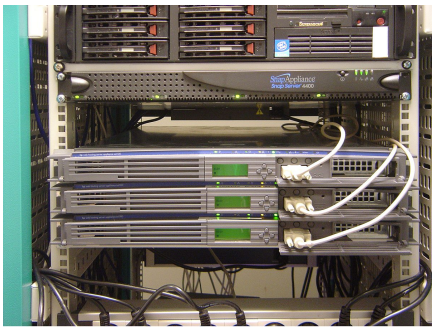
Thermal zone mapping uses sensors and computer modeling to create a three-dimensional image of the hot and cool zones in a data center.[41]

This information can help to identify optimal positioning of data center equipment. For example, critical servers might be placed in a cool zone that is serviced by redundant AC units.

Green datacenters

Datacenters use a lot of power, consumed by two main usages: the power required to run the actual equipment and then the power required to cool the equipment. The first category is addressed by designing computers and storage systems that are more and more power-efficient. And to bring down the cooling costs datacenter designers try to use natural ways to cool the equipment. Many datacenters have to be located near people-concentrations to manage the equipment, but there are also many circumstances where the datacenter can be miles away from the users and don't need a lot of local management. Examples of this are the 'mass' datacenters like Google or Facebook: these DC's are built around many standarised servers and storage-arrays and the actual users of the systems are located all around the world. After the initial build of a datacenter there is not much staff required to keep it running: especially datacenters that provide mass-storage or computing power don't need to be near population centers. Datacenters in arctic locations where outside air provides all cooling are getting more popular as cooling and electricity are the two main variable cost components.[42]

Network infrastructure



An example of "rack mounted" servers

Communications in data centers today are most often based on networks running the IP protocol suite. Data centers contain a set of routers and switches that transport traffic between the servers and to the outside world. Redundancy of the Internet connection is often provided by using two or more upstream service providers (see Multihoming).

Some of the servers at the data center are used for running the basic Internet and intranet services needed by internal users in the organization, e.g., e-mail servers, proxy servers, and DNS servers.

Network security elements are also usually deployed: firewalls, VPN gateways, intrusion detection systems, etc. Also common are monitoring systems for the network and some of the applications. Additional off site monitoring systems are also typical, in case of a failure of communications inside the data center.

Data center infrastructure management

Data center infrastructure management (DCIM) is the integration of information technology (IT) and facility management disciplines to centralize monitoring, management and intelligent capacity planning of a data center's critical systems. Achieved through the implementation of specialized software, hardware and sensors, DCIM enables common, real-time monitoring and management platform for all interdependent systems across IT and facility infrastructures.

Depending on the type of implementation, DCIM products can help data center managers identify and eliminate sources of risk to increase availability of critical IT systems. DCIM products also can be used to identify interdependencies between facility and IT infrastructures to alert the facility manager to gaps in system redundancy, and provide dynamic, holistic benchmarks on power consumption and efficiency to measure the effectiveness of "green IT" initiatives.

Measuring and understanding important data center efficiency metrics. A lot of the discussion in this area has focused on energy issues, but other metrics beyond the PUE can give a more detailed picture of the data center operations. Server, storage, and staff utilization metrics can contribute to a more complete view of an enterprise data center. In many cases, disc capacity goes unused and in many instances the organizations run their servers at 20% utilization or less. More effective automation tools can also improve the number of servers or virtual machines that a single admin can handle.

Applications


A 40-foot Portable Modular Data Center

The main purpose of a data center is running the applications that handle the core business and operational data of the organization. Such systems may be proprietary and developed internally by the organization, or bought from enterprise software vendors. Such common applications are ERP and CRM systems.

A data center may be concerned with just operations architecture or it may provide other services as well.

Often these applications will be composed of multiple hosts, each running a single component. Common components of such applications are databases, file servers, application servers, middleware, and various others.

Data centers are also used for off site backups. Companies may subscribe to backup services provided by a data center. This is often used in conjunction with backup tapes. Backups can be taken off servers locally on to tapes. However, tapes stored on site pose a security threat and are also susceptible to fire and flooding. Larger companies may also send their backups off site for added security. This can be done by backing up to a data center. Encrypted backups can be sent over the Internet to another data center where they can be stored securely.

For quick deployment or disaster recovery, several large hardware vendors have developed mobile solutions that can be installed and made operational in very short time. Companies such as Cisco Systems, Sun Microsystems (Sun Modular Datacenter),[43] Bull, IBM (Portable Modular Data Center), HP (Performance Optimized Datacenter), and Google (Google Modular Data Center) have developed systems that could be used for this purpose.

US Retail Colocation Providers

- Contains estimates from Synergy Research Group.[44]

Rank	Company Name	US Market Share
1	Equinix	18%
2	CenturyLink	8%
3	SunGard	5%
4	AT&T	5%
5	Verizon	5%
6	Telx	4%
7	CyrusOne	4%
8	Level 3 Communications	3%
9	Internap	2%

References

[1] http://global.ihs.com/doc_detail.cfm?currency_code=USD&customer_id=2125442B200A&oshid=2125442B200A&shopping_cart_id=292558332D4B404849594D5B260A&country_code=US&lang_code=ENGL&item_s_key=00414811&item_key_date=940819&input_doc_number=TIA-942&input_doc_title=

[2] http://www.tiaonline.org/standards/

[3] http://telecom-info.telcordia.com/site-cgi/ido/docs.cgi?ID=SEARCH&DOCUMENT=GR-3160&

[4] Mukhar, Nicholas. "HP Updates Data Center Transformation Solutions," August 17, 2011 (http://www.mspmentor.net/2011/08/17/hp-updates-data-transformation-solutions/)

[5] Niccolai, James. "Data Centers Turn to Outsourcing to Meet Capacity Needs," CIO.com, May 10, 2011 (http://www.cio.com/article/681897/Data_Centers_Turn_to_Outsourcing_to_Meet_Capacity_Needs)

[6] Tang, Helen. "Three Signs it's time to transform your data center," August 3, 2010, Data Center Knowledge (http://www.datacenterknowledge.com/archives/2010/08/03/three-signs-itâ□□s-time-to-transform-your-data-center/)

[7] Miller, Rich. "Complexity: Growing Data Center Challenge," Data Center Knowledge, May 16, 2007 (http://www.datacenterknowledge.com/archives/2007/05/16/complexity-growing-data-center-challenge/)

[8] Sims, David. "Carousel's Expert Walks Through Major Benefits of Virtualization," TMC Net, July 6, 2010 (http://virtualization.tmcnet.com/topics/virtualization/articles/193652-carousels-expert-walks-through-major-benefits-virtualization.htm)

[9] Delahunty, Stephen. "The New urgency for Server Virtualization," InformationWeek, August 15, 2011. (http://www.informationweek.com/news/government/enterprise-architecture/231300585)

[10] Miller, Rich. "Gartner: Virtualization Disrupts Server Vendors," Data Center Knowledge, December 2, 2008 (http://www.datacenterknowledge.com/archives/2008/12/02/gartner-virtualization-disrupts-server-vendors/)

[11] Ritter, Ted. Nemertes Research, "Securing the Data-Center Transformation Aligning Security and Data-Center Dynamics," (http://lippisreport.com/2011/05/securing-the-data-center-transformation-aligning-security-and-data-center-dynamics/)

[12] http://global.ihs.com/doc_detail.cfm?currency_code=USD&customer_id=2125452B2C0A&oshid=2125452B2C0A&shopping_cart_id=292558332C4A2020495A4D3B200A&country_code=US&lang_code=ENGL&item_s_key=00414811&item_key_date=940819&input_doc_number=TIA-942&input_doc_title=

[13] A ConnectKentucky article mentioning Stone Mountain Data Center Complex

[14] A document from the Uptime Institute describing the different tiers (click through the download page)

[15] The rating guidelines from the Uptime Institute

[16] Cherry, Edith. “Architectural Programming: Introduction”, Whole Building Design Guide, Sept. 2, 2009

[17] Mullins, Robert. “Romonet Offers Predictive Modelling Tool For Data Center Planning”, Network Computing, June 29, 2011 (http://www.networkcomputing.com/data-center/231000669)

[18] Jew, Jonathan. “BICSI Data Center Standard: A Resource for Today’s Data Center Operators and Designers,” BICSI News Magazine, May/June 2010, page 28. (http://www.nxtbook.com/nxtbooks/bicsi/news_20100506/#/26)

[19] Data Center Energy Management: Best Practices Checklist: Mechanical, Lawrence Berkeley National Laboratory (http://hightech.lbl.gov/dctraining/strategies/mam.html)

[20] Clark, Jeff. “Hedging Your Data Center Power”, The Data Center Journal, Oct. 5, 2011. (http://www.datacenterjournal.com/design/hedging-your-data-center-power/)

[21] Jew, Jonathan. “BICSI Data Center Standard: A Resource for Today’s Data Center Operators and Designers,” BICSI News Magazine, May/June 2010, page 30. (http://www.nxtbook.com/nxtbooks/bicsi/news_20100506/#/26)

[22] Clark, Jeffrey. “The Price of Data Center Availability—How much availability do you need?”, Oct. 12, 2011, The Data Center Journal (http://www.datacenterjournal.com/home/news/languages/item/2792-the-price-of-data-center-availability)

[23] Tucci, Linda. “Five tips on selecting a data center location”, May 7, 2008, SearchCIO.com (http://searchcio.techtarget.com/news/1312614/Five-tips-on-selecting-a-data-center-location)
[24] Niles, Susan. “Standardization and Modularity in Data Center Physical Infrastructure,” 2011, Schneider Electric, page 4. (http://www.apcmedia.com/salestools/VAVR-626VPD_R1_EN.pdf)
[25] Pitchaikani, Bala. “Strategies for the Containerized Data Center,” DataCenterKnowledge.com, Sept. 8, 2011. (http://www.datacenterknowledge.com/archives/2011/09/08/strategies-for-the-containerized-data-center/)
[26] Niccolai, James. “HP says prefab data center cuts costs in half,” InfoWorld, July 27, 2010. (http://www.infoworld.com/d/green-it/hp-says-prefab-data-center-cuts-costs-in-half-837?page=0,0)
[27] http://telecom-info.telcordia.com/site-cgi/ido/docs.cgi?ID=SEARCH&DOCUMENT=GR-2930&
[28] Detailed explanation of UPS topologies
[29] J Koomey, C. Belady, M. Patterson, A. Santos, K.D. Lange. Assessing Trends Over Time in Performance, Costs, and Energy Use for Servers (http://www.intel.com/assets/pdf/general/servertrendsreleasecomplete-v25.pdf) Released on the web August 17th, 2009.
[30] A calculation of data center electricity burden cited in the Report to Congress on Server and Data Center Energy Efficiency (http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf) and electricity generation contributions to green house gas emissions published by the EPA in the Greenhouse Gas Emissions Inventory Report (http://epa.gov/climatechange/emissions/downloads10/US-GHG-Inventory-2010_ExecutiveSummary.pdf). Retrieved 2010-06-08.
[31] Canada Called Prime Real Estate for Massive Data Computers - Globe & Mail (http://www.theglobeandmail.com/report-on-business/canada-called-prime-real-estate-for-massive-data-computers/article2071677/) Retrieved June 29, 2011.
[32] Finland - First Choice for Siting Your Cloud Computing Data Center. (http://www.fincloud.freehostingcloud.com/). Retrieved 4 August 2010.
[33] Stockholm sets sights on data center customers. (http://www.stockholmbusinessregion.se/templates/page____41724.aspx?epslanguage=EN) Accessed 4 August 2010.
[34] Swiss Carbon-Neutral Servers Hit the Cloud. (http://www.greenbiz.com/news/2010/06/30/swiss-carbon-neutral-servers-hit-cloud). Retrieved 4 August 2010.
[35] Commentary on introduction of Energy Star for Data Centers
[36] Sweeney, Jim. "Reducing Data Center Power and Energy Consumption: Saving Money and 'Going Green,' " GTSI Solutions, pages 2–3. (http://www.gtsi.com/cms/documents/white-papers/green-it.pdf)
[37] Needle, David. “HP's Green Data Center Portfolio Keeps Growing,” InternetNews, July 25, 2007. (http://www.internetnews.com/xSP/article.php/3690651/HPs+Green+Data+Center+Portfolio+Keeps+Growing.htm)
[38] Siranosian, Kathryn. “HP Shows Companies How to Integrate Energy Management and Carbon Reduction,” TriplePundit, April 5, 2011. (http://www.triplepundit.com/2011/04/hp-launches-program-companies-integrate-manage-energy-carbon-reduction-strategies/)
[39] Bullock, Michael. “Computation Fluid Dynamics - Hot topic at Data Center World,” Transitional Data Services,” March 18, 2010. (http://blog.transitionaldata.com/aggregate/bid/37840/Seeing-the-Invisible-Data-Center-with-CFD-Modeling-Software)
[40] Bouley, Dennis (editor). “Impact of Virtualization on Data Center Physical Infrastructure,” The Green grid, 2010. (http://www.thegreengrid.org/~/media/WhitePapers/White_Paper_27_Impact_of_Virtualization_Data_On_Center_Physical_Infrastructure_020210.pdf?lang=en)
[41] Fontecchio, Mark. “HP Thermal Zone Mapping plots data center hot spots,” SearchDataCenter, July 25, 2007. (http://searchdatacenter.techtarget.com/news/1265634/HP-Thermal-Zone-Mapping-plots-data-center-hot-spots)
[42] Gizmag Fjord-cooled DC in Norway claims to be greenest (http://www.gizmag.com/fjord-cooled-data-center/20938/), 23 December 2011. Visited: 1 April 2012
[43] And English Wiki article on Sun's modular datacentre
[44] https://www.srgresearch.com/articles/mature-us-colocation-market-led-equinix-and-centurylink-savvis

External links

- Lawrence Berkeley Lab (http://hightech.lbl.gov/datacenters.html) - Research, development, demonstration, and deployment of energy-efficient technologies and practices for data centers
- DC Power For Data Centers Of The Future (http://hightech.lbl.gov/dc-powering/faq.html) - FAQ: 380VDC testing and demonstration at a Sun data center.
- DC Compendium (http://www.dccompendium.com/) - Repository and compendium of data centers globally.
- White Paper (http://media.wix.com/ugd/fb8983_e929404b24874e4fa7a8279f1cda58f8.pdf) - Property Taxes: The New Challenge for Data Centers

Virtual data room

A **virtual data room** (sometimes called a **VDR**) is an online repository of information that is used for the storing and distribution of documents. In many cases, a virtual data room is used to facilitate the due diligence process during an M&A transaction, loan syndication, or private equity and venture capital transactions. This due diligence process has traditionally used a physical data room to accomplish the disclosure of documents. For reasons of cost, efficiency and security, virtual data rooms have widely replaced the more traditional physical data room.

Introduction

An alternative to the physical data room involves the setting up of a virtual data room in the form of an extranet to which the bidders and their advisers are given access via the internet. An extranet is essentially an Internet site with limited controlled access, using a secure log-on supplied by the vendor, which can be disabled at any time, by the vendor, if a bidder withdraws. Much of the information released is confidential and restrictions are applied to the viewer's ability to release this to third parties (by means of forwarding, copying or printing). This can be effectively applied to protect the data using digital rights management.

In the process of mergers & acquisitions the data room is set up as part of the central repository of data relating to companies or divisions being acquired or sold. The data room enables the interested parties to view information relating to the business in a controlled environment. Confidentiality is paramount and strict controls for viewing, copying and printing are imposed. Conventionally this is achieved by establishing a supervised, physical data room in secure premises with controlled access. In most cases, with a physical data room, only one bidder team can access the room at a time. This becomes time consuming.

A virtual data room has exactly the same strengths as a conventional data room: controlling access, viewing, copying and printing as well as setting time limits on viewing and logging. It has none of the disadvantages of being in a standard, physical location, needing couriers to move documents or transporting of key staff and personnel back and forth. It is also accessible 24/7 over the allowed period. With a virtual data room, documents reach the regulators and investors in a more efficient and timely manner. Due to improvements in efficiency and speed, a virtual data room typically pays for itself in a single M&A transaction.

A virtual data room is quick to set up. Scanned data and existing electronic files can be mixed, information can be added or eliminated at any time (the changes could be logged if required) and any or all information can be restricted to any or all registered viewers at any time.

Disadvantages of a physical data room

- Time consuming
- Narrow bandwidth
- Expensive
- Cost of travel
- Paper intensive

Benefits of a virtual data room

The largest financial benefits accrue to the seller although buyers also benefit. For the former, advantages include:

- Improvement in the number of bidders.
- Increased bid throughout (and time zone access) if the virtual data room is accessible 24/7 over the allowed period. [*citation needed*]
- Increased control and understanding of bidders.
- Resulting 20%-30% higher bid values. [*citation needed*]
- Increased speed of transactions owing to improved accessibility
- Enhanced information secures more deals at higher prices. [*citation needed*]
- Conventional physical data rooms restrict the bidder or buyers' ability to get the correct people to the room simply due to the physical location. However, Virtual Data Room opens up global markets for M&A, takeovers and property deals compared with purely face-to-face and hardcopy document transactions (i.e. business letters).
- Information cannot be downloaded and taken away in a true Virtual Data Room - only viewed by a user with the correct permissions

External links

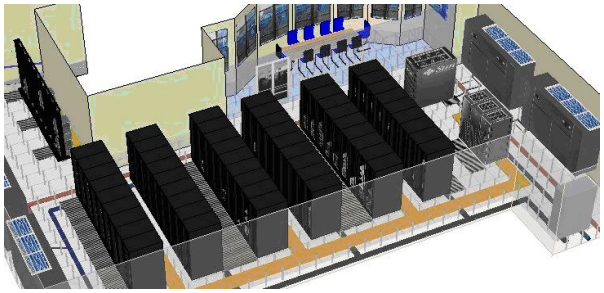
- Study on Virtual Data Rooms for M&A [1] Manda (2007)

References

[1] http://www.manda-institute.org/docs/kummer-sliskovic_do%20virtual%20data%20rooms%20add%20value%20to%20the%20mergers%20and%20acquisitions%20process.pdf

Virtual facility

A **Virtual Facility** (VF) is a highly realistic digital representation of a data center (primarily). The term virtual in Virtual Facility refers to the use of the word as in Virtual Reality rather than the abstraction of computer resources as in platform virtualization. The VF mirrors the characteristics of the physical facility over time and allows modeling all relevant characteristics of a physical data center with a high degree of precision.



A Virtual Facility snapshot created with 6SigmaDC software.

VF Model includes

- Three-dimensional physical facility layout
- Network connectivity of facility equipment
- Full inventory of facility equipment, including electronics and electrical systems such as Power Distribution Units (PDU's) and Uninterruptible Power Supplies (UPS's)
- Full air conditioning system (ACU's) and controls within the room

The term Virtual Facility was introduced by Future Facilities, a data centre design consultancy focused on delivering Design and Operational solutions to address the emerging environmental problems facing the modern Mission Critical Facility (MCF). The concept is in essence a convergence of the fields of Virtual Reality (VR), Computer Simulation and Expert Systems, applied to the specific domain of facilities.

The VF type of computer simulation allows detailed analysis and prototyping of air flow in the data center by making use of Computational Fluid Dynamics (CFD) techniques. This in turn allows the air flow and temperatures of the facility to be analyzed visually (Scientific Visualisation) and numerically to study and predict what will happen in the real facility. The importance of scientific methods in design of mission critical facilities has become a necessity, since the performance gains predicted by Moore's Law go hand in hand with a rise in power and heat dissipated by equipment. Rules of thumb have proven to be no longer adequate.

VF design purposes

- Green field design
- Asset management
- Troubleshooting existing data centers
- Making existing data centers more resilient
- Making existing data centers more energy efficient
- Cost prediction
- Staff training
- Capacity planning
- Load growth management

The VF is now being employed by many large organizations as a way of virtually assessing a situation before having to spend huge sums of money trying to solve a problem in the real facility.

It is essential to know whether adding new equipment or changing equipment will cause a logistical or thermal problem. The VF allows the designer or operator to assess the best course of action and gives in depth understanding on unintuive behaviours.

References

- Seymour, Mark, *Virtual Data Centre Design. A blueprint for success* [1], retrieved 2007-09-26

References

[1] http://www.futurefacilities.com/newsarticles/articles/commerzbankarticlesummerZDTjournal.pdf

Network-neutral data center

A **network-neutral data center** (or **carrier-neutral data center**) is a data center (or carrier hotel) which allows interconnection between multiple telecommunication carriers and/or colocation providers. Network-neutral data centers exist all over the world and vary in size and power.

While some data centers are owned and operated by a telecommunications or Internet service provider, *network-neutral* data centers are operated by a third party who has little or no part in providing Internet service to the end-user. This encourages competition and diversity as a server in a colocation centre can have one provider, multiple providers or only connect back to the headquarters of the company who owns the server.

One benefit of hosting in a network-neutral data center is the ability to switch providers without physically moving the server to another location.

Related Technologies

Virtual directory

In computing, the term **virtual directory** has a couple of meanings. It may simply designate (for example in IIS) a folder which appears in a path but which is not actually a subfolder of the preceding folder in the path. However, this article will discuss the term in the context of directory services and identity management.

A virtual directory or **virtual directory server** in this context is a software layer that delivers a single access point for identity management applications and service platforms. A virtual directory operates as a high-performance, lightweight abstraction layer that resides between client applications and disparate types of identity-data repositories, such as proprietary and standard directories, databases, web services, and applications.

A virtual directory receives queries and directs them to the appropriate data sources by abstracting and virtualizing data. The virtual directory integrates identity data from multiple heterogeneous data stores and presents it as though it were coming from one source. This ability to reach into disparate repositories makes virtual directory technology ideal for consolidating data stored in a distributed environment.

As of 2011[1], virtual directory servers most commonly use the LDAP protocol, but more sophisticated virtual directories can also support SQL as well as DSML and SPML.

Industry experts have heralded the importance of the virtual directory in modernizing the identity infrastructure. According to Dave Kearns of Network World,[2] “Virtualization is hot and a virtual directory is the building block, or foundation, you should be looking at for your next identity management project." In addition, Gartner analyst, Bob Blakley[3] said that virtual directories are playing an increasingly vital role. In his report, “The Emerging Architecture of Identity Management,” Blakley wrote: “In the first phase, production of identities will be separated from consumption of identities through the introduction of a virtual directory interface.”

Capabilities of Virtual Directories

Virtual directories can have some or all of the following capabilities:

- Aggregate identity data across sources to create a single point of access.
- Create high-availability for authoritative data stores.
- Act as identity firewall by preventing denial-of-service attacks on the primary data stores through an additional virtual layer.
- Support a common searchable namespace for centralized authentication.
- Present a unified virtual view of user information stored across multiple systems.
- Delegate authentication to backend sources through source-specific security means.
- Virtualize data sources to support migration from legacy data stores without modifying the applications that rely on them.
- Enrich identities with attributes pulled from multiple data stores, based on a link between user entries.

Some advanced identity virtualization platforms can also:

- Enable application-specific, customized views of identity data without violating internal or external regulations governing identity data.Reveal contextual relationships between objects through hierarchical directory structures.
- Develop advanced correlation across diverse sources using correlation rules.
- Build a global user identity by correlating unique user accounts across various data stores, and enrich identities with attributes pulled from multiple data stores, based on a link between user entries.

- Enable constant data refresh for real-time updates through a persistent cache.

Advantages of virtual directories

Virtual Directories:

- Enable faster deployment because users do not need to add and sync additional application-specific data sources
- Leverage existing identity infrastructure and security investments to deploy new services
- Deliver high availability of data sources
- Provide application-specific views of identity data which can help avoid the need to develop a master enterprise schema
- Allow a single view of identity data without violating internal or external regulations governing identity data
- Act as identity firewalls by preventing denial-of-service attacks on the primary data-stores and providing further security on access to sensitive data
- Can reflect changes made to authoritative sources in real-time
- Present a unified virtual view of user information from multiple systems so that it appears to reside in a single system
- Can secure all backend storage locations with a single security policy

Disadvantages

An original disadvantage is public perception of "push & pull technologies" which is the general classification of "virtual directories" depending on the nature of their deployment. Virtual directories were initially designed and later deployed with "Push technologies" in mind, which also contravened with "Privacy laws" in the USA. This is no longer the case. There are, however, other disadvantages in the current technologies.

- The classical virtual directory based on proxy cannot modify underlying data structures or create new views based on the relationships of data from across multiple systems. So if an application requires a different structure, such as a flattened list of identities, or a deeper hierarchy for delegated administration, a virtual directory is limited.
- Many virtual directories cannot correlate same-users across multiple diverse sources in the case of duplicate users
- Virtual directories without advanced caching technologies cannot scale to heterogeneous, high-volume environments.

Sample terminology

- Unify metadata: Extract schemas from the local data source, map them to a common format, and link the same identities from different data silos based on a unique identifier.
- Namespace joining: Create a single large directory by bringing multiple directories together at the namespace level. For instance, if one directory has the namespace "ou=internal,dc=domain,dc=com" and a second directory has the namespace "ou=external,dc=domain,dc=com," then creating a virtual directory with both namespaces is an example of namespace joining.
- Identity joining: Enrich identities with attributes pulled from multiple data stores, based on a link between user entries. For instance if the user joeuser exists in a directory as "cn=joeuser,ou=users" and in a database with a username of "joeuser" then the "joeuser" identity can be constructed from both the directory and the database.
- Data remapping: The translation of data inside of the virtual directory. For instance, mapping “uid” to “samaccountname,” so a client application that only supports a standard LDAP-compliant data source is able to search an Active Directory namespace, as well.
- Query routing: Route requests based on certain criteria, such as “write operations going to a master, while read operations are forwarded to replicas.”

- Identity routing: Virtual directories may support the routing of requests based on certain criteria (such as write operations going to a master while read operations being forwarded to replicas).
- Authoritative source: A "virtualized" data repository, such as a directory or database, that the virtual directory can trust for user data.
- Server groups: Group one or more servers containing the same data and functionality. A typical implementation is the multi-master, multi-replica environment in which replicas process "read" requests and are in one server group, while masters process "write" requests and are in another, so that servers are grouped by their response to external stimuli, even though all share the same data.

Sample Virtual Directory Use Cases

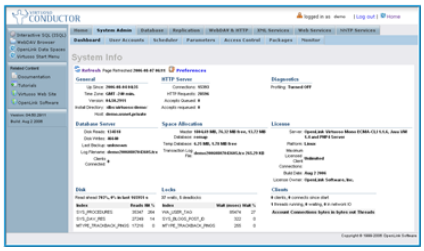
- Integrating multiple directory namespaces to create a central enterprise directory.
- Supporting infrastructure integrations after mergers and acquisitions.
- Centralizing identity storage across the infrastructure, making identity information available to applications through various protocols (including LDAP, JDBC, and web services).
- Creating a single access point for web access management (WAM) tools.
- Enabling web single sign-on (SSO) across varied sources or domains.
- Supporting role-based, fine-grained authorization policies
- Enabling authentication across different security domains using each domain's specific credential checking method.
- Improving secure access to information both inside and outside of the firewall.

References

[1] http://en.wikipedia.org/w/index.php?title=Virtual_directory&action=edit
[2] http://www.networkworld.com/newsletters/dir/2006/0807id1.html
[3] The Emerging Architecture of Identity Management, Bob Blakley, April 16, 2010.

Virtuoso Universal Server

Virtuoso Universal Server



Virtuoso Conductor (Database Administration User Interface)

Developer(s)	OpenLink Software
Stable release	7.1.0 / February 2014
Operating system	Cross-platform
Type	Triplestore, RDBMS, Application server, Web server
License	GPLv2 and proprietary
Website	virtuoso.openlinksw.com [1]

Virtuoso Universal Server is a middleware and database engine hybrid that combines the functionality of a traditional RDBMS, ORDBMS, virtual database, RDF, XML, free-text, web application server and file server functionality in a single system. Rather than have dedicated servers for each of the aforementioned functionality realms, Virtuoso is a "universal server"; it enables a single multithreaded server process that implements multiple protocols. The open source edition of Virtuoso Universal Server is also known as **OpenLink Virtuoso**. The software has been developed by OpenLink Software with Kingsley Uyi Idehen and Orri Erling as the chief software architects.

Database structure

Core database engine

Virtuoso provides an extended object-relational model, which combines the flexibility of relational access with inheritance, run time data typing, late binding, and identity based access. Virtuoso Universal Server database includes physical file and in memory storage and operating system processes that interact with the storage. There is one main process, which has listeners on a specified port for HTTP, SOAP, and other protocols.

Architecture

Virtuoso is designed to take advantage of operating system threading support and multiple CPUs. It consists of a single process with an adjustable pool of threads shared between clients. Multiple threads may work on a single index tree with minimal interference with each other. One cache of database pages is shared among all threads and old dirty pages are written back to disk as a background process.

The database has at all times a clean checkpoint state and a delta of committed or uncommitted changes to this checkpointed state. This makes it possible to do a clean backup of the checkpoint state while transactions proceed on the commit state.

A transaction log file records all transactions since the last checkpoint. Transaction log files may be preserved and archived for an indefinite time, providing a full, recoverable history of the database.

A single set of files is used for storing all tables. A separate set of files is used for all temporary data. The maximum size of a file set is 32 terabytes, for 4G × 8K pages.

Locking

Virtuoso provides dynamic locking, starting with row level locks and escalating to page level locks when a cursor holds a large percentage of a page's rows or when it has a history of locking entire pages. Lock escalation only happens when no other transactions hold locks on the same page, hence it never deadlocks. Virtuoso SQL provides means for exclusive read and for setting transaction isolation.

Transactions

All four levels of isolation are supported: Dirty read, read committed, repeatable read and serializable. The level of isolation may be specified operation by operation within a single transaction. Virtuoso can also act as a resource manager and/or transaction coordinator under Microsoft's Distributed Transaction Coordinator (MS DTC) or the XA standard.

Data integrity

Virtuoso ORDBMS database supports entity integrity and referential integrity. Virtuoso ensures that relationships between records in related tables are valid by enforcing referential integrity. Integrity constraints include:

- NOT NULL - Within the definition of a table, Virtuoso allows data to contain a NULL value. This NULL value is not really a value at all and is considered an absence of value. The constraint of NOT NULL forces a value to be given to a column.
- Unique Key - Uniqueness for a column or set of columns means that the values in that column or set of columns must be different from all other columns or set of columns in that table. A unique key may contain NULL values since they are by definition a unique non-valued value.
- Primary Key - Primary key are much like unique keys except that they are designed to uniquely identify a row in a table. They can consist of a single column or multiple columns. The primary key cannot contain a NULL value.
- CHECK Constraint - Virtuoso provides on a column an integrity constraint that requires certain conditions to be met before the data is inserted or modified. If the checks are not satisfied then the transaction cannot be completed.

Data dictionary

Virtuoso stores all its information about all user objects in the database in the system catalog tables designated by db.dba*.

Components and files

Components

Virtuoso is made up of client and server components. These are the components typically used to communicate with a local or remote Virtuoso server which include:

- Virtuoso Drivers for ODBC, JDBC, ADO.NET and OLE DB
- Conductor, a Web Based Database Administration User Interface
- ISQL (Interactive SQL) and ISQO Utilities
- Documentation and Tutorials
- Samples

All database installation come with two databases, a default database and a demo database.

History

The Virtuoso project was born in 1998 from a merger of the **OpenLink** data access middleware and **Kubl** RDBMS.

Kubl RDBMS

The Kubl ORDBMS was one of a list of relational database systems with roots in Finland. This list also includes MySQL, InnoDB, and Solid RDBMS/Solid Technologies.

As is the case with most technology products, key personnel behind OpenLink Virtuoso, InnoDB, and Solid share periods of professional overlap that provide noteworthy insight into the history of database technology development in Finland. Heikki Tuuri (creator of InnoDB), Ora Lassila (W3C and Nokia Research, a technology lead and visionary in the areas RDF and Semantic Web in general alongside Tim Berners-Lee), and Orri Erling (Virtuoso Program Manager at OpenLink Software) all worked together in a startup company called Entity Systems in Finland - where they were developing Common Lisp and Prolog development environments for the early generation of PC's circa. 1986–88.

Later, Orri Erling worked with VIA International, the developer of VIA/DRE in designing a LISP based object-oriented data access layer atop the company's DBMS product. The core development team of VIA, following the company's demise in 1992, went on to found Solid Technologies under the direction of Artturi Tarjanne.

Heikki Tuuri worked at Solid for a while before starting his own database development project which became InnoDB (acquired by Oracle in 2005).

Orri Erling started his own DBMS development work in 1994, which was to become Kubl. Development of Kubl was initially financed by Infosto Group, publisher of Finland's largest free ads paper, as part of their in-house software development project for their on-line services. The on-line version of *Keltainen Pörssi* was at one time said to be Finland's most popular web site with 500,000 registered users. The Kubl database was prominently displayed in a *"Powered by Kubl"* logo on the search results.

A free trial version of Kubl was made available for download on November 7, 1996.

Kubl was marketed as a high performance lightweight database for embedded use; the development aim was to achieve top scores in Transactions Per Second tests. Pricing of the product was especially favorable to Linux users with a Linux license priced at $199.

Kubl became the cornerstone of OpenLink Virtuoso, after the technology paths of Kingsley Uyi Idehen and Orri Erling crossed in 1998, leading to the acquisition of Kubl by OpenLink Software.

Functionality realms

Virtuoso provides functionality that covers a broad range of traditionally distinct functionality realms as part of a single product offering. The realms include:

- Object-relational database engine for (SQL, XML, RDF and plain text)
- Web services computing platform
- Web application server
- Web content management system (WCMS)
- NNTP-based Discussion Management
- Replication of Homogeneous and Heterogeneous Data
- Mail Storage Sink and (POP3) Service Proxy
- DataPortability

Protocols implemented

In addition to the functionality realms above, the product implements of a broad range of industry standard Web & Internet protocols that includes: HTTP, WebDAV, CalDAV, CardDAV, SOAP, UDDI, WSDL, WS-Policy, WS-Security, WS-ReliableMessaging, WS-Routing, WS-Referral, WS-Attachment, WS-BPEL, SyncML, GData, SPARQL, SPARUL, NNTP

API support

For the database application developer and systems integrator, Virtuoso implements a variety of industry standard data access APIs (client and server) that includes: ODBC, JDBC, OLE DB, ADO.NET, ADO.NET Entity Framework, XMLA

Content syndication and interchange format support

For the Web application developer, content syndicate(s), content publishers, and content consumers, Virtuoso implements support for standards such as: Atom, RSS 2.0, RSS 1.0, OPML, XBEL, FOAF, SIOC

Query language support

SQL, SPARQL (with numerous extensions), XQuery (implementation of Core functions library is seriously incomplete), XPath (1.0 only), XSLT (1.0 only)

Schema definition language support

SQL's Data Definition Language, XML Schema

Usage scenarios

Virtuoso is a solution for the following system integration challenges:

- Enterprise Information Integration (EII)
- Programming Language Independent Web application deployment
- Monolithic application decomposition that leverages the principles of service-oriented architecture
- Web service based enterprise application integration via a significant amount of WS-* protocols support
- Business process management via BPEL
- Semantic Web Data Spaces Generation
- Deployment Platform for injecting RDF-based Linked Data into the Semantic Data Web

Related technology areas

Data management

- Relational database management system
- List of relational database management systems
- Comparison of object-relational database management systems
- Comparison of relational database management systems

Enterprise application, information, and data integration

- Web 2.0
- Enterprise service bus
- Service-oriented architecture

- Enterprise application integration
- Data integration
- Web service
- Semantic Web
- Business Integration Severs Comparison Matrix

Related products and tools

In addition to Virtuoso, OpenLink Software produces several related tools and applications.

- OpenLink Data Spaces a Virtuoso based platform for cost-effective creation and management of Semantic Web / Linked Data Web presence. It provides a data junction box for integrating data across third party Social network service, Blog, File sharing, Shared & Social bookmarking, Wiki, E-mail, Photo Sharing, RSS 2.0, Atom, and RSS 1.1 Content Aggregation services. In addition, to its third party integration functionality, it also includes its own rich collection of Linked Data compliant distributed collaborative applications, across each of the aforementioned Web application realms.
- Universal Data Access Drivers - High-performance data access drivers for ODBC, JDBC, ADO.NET, and OLE DB that provide transparent access to enterprise databases across multiple platforms and databases.

Platforms

Virtuoso is supported on a number of 32- & 64-bit platforms including cross-platform Windows, UNIX (HP, AIX, Sun, DEC, BSD, SCO), Linux (Red Hat, SUSE), and Mac OS X.

Licensing

In April 2006, an open source version of Virtuoso was made available under the GNU General Public License v2. The software is now available in Commercial and Open Source license variants.

References

[1] http://virtuoso.openlinksw.com/

External links

- Main Virtuoso Web Site (http://virtuoso.openlinksw.com/)
- OpenLink Virtuoso (Open-Source Edition) (http://sourceforge.net/projects/virtuoso/) at SourceForge
- OpenLink Virtuoso (Open-Source Edition) (https://github.com/openlink/virtuoso-opensource) at GitHub

Workflow engine

A **workflow engine** is a software application that defines a process, the rules governing process decisions, and routes information.[1] It is a key component in workflow technology and typically makes use of a database server.

A workflow manages and monitors the state of activities, such as the processing and approval of a loan application form, and determines which new activity to transition to according to defined processes (workflows).[2] The actions may be anything from saving an application form in a document management system to sending a reminder e-mail to users or escalating overdue items to management. A workflow engine facilitates the flow of information, tasks, and events.[3] Workflow engines may also be referred to as a Workflow Orchestration Engines.[4]

Workflow engines mainly have three functions:

- Verification of the current status: Check whether the command is valid in executing a task.
- Determine the authority of users: Check if the current user is permitted to execute the task.
- Executing condition script: After passing the previous two steps, workflow engine begins to evaluate condition script in which two processes are carried out, if the condition is true, workflow engine execute the task, and if execution successfully complete, it returns the success, if not, it reports the error to trigger and roll back the change.[5]

A workflow engine is a core technique for task allocation software application, such as BPM in which the workflow engine allocates task to different executors with communicating data among participants. A workflow engine can execute any arbitrary sequence of steps, for example, a healthcare data analysis.

References

[1] http://glossary.businessprocessincubator.com/index.php/6987/workflow-engine
[2] http://docs.oracle.com/cd/B13789_01/workflow.101/b10286/wfapi.htm
[3] http://ceiton.com/CMS/EN/workflow/introduction.html#Benefits
[4] http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/index.jsp?topic=%2Fcom.ibm.sco.doc_2.2%2Fenablement%2Fworkfloworchestration.html
[5] The Workflow Engine Model. The Workflow Engine Model (http://msdn.microsoft.com/en-us/library/aa188337(office.10).aspx) Accessed 1 Dec. 2010.

Metadata Models

Metadata

Metadata is "data about data". The term is ambiguous, as it is used for two fundamentally different concepts (types). **Structural metadata** is about the design and specification of data structures and is more properly called "data about the containers of data"; **descriptive metadata**, on the other hand, is about individual instances of application data, the data content.

Metadata are traditionally found in the card catalogs of libraries. As information has become increasingly digital, metadata are also used to describe digital data using metadata standards specific to a particular discipline. By describing the contents and context of data files, the quality of the original data/files is greatly increased. For example, a webpage may include metadata specifying what language it is written in, what tools were used to create it, and where to go for more on the subject, allowing browsers to automatically improve the experience of users.

Definition

Metadata (metacontent) are defined as the data providing information about one or more aspects of the data, such as:

- Means of creation of the data
- Purpose of the data
- Time and date of creation
- Creator or author of the data
- Location on a computer network where the data were created
- Standards used

For example, a digital image may include metadata that describe how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document's metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document.

Metadata are data. As such, metadata can be stored and managed in a database, often called a metadata registry or metadata repository.[1] However, without context and a point of reference, it might be impossible to identify metadata just by looking at them. For example: by itself, a database containing several numbers, all 13 digits long could be the results of calculations or a list of numbers to plug into an equation - without any other context, the numbers themselves can be perceived as the data. But if given the context that this database is a log of a book collection, those 13-digit numbers may now be identified as ISBNs - information that refers to the book, but is not itself the information within the book.

The term "metadata" was coined in 1968 by Philip Bagley, in his book "Extension of programming language concepts" where it is clear that he uses the term in the ISO 11179 "traditional" sense, which is "structural metadata" i.e. "data about the containers of data"; rather than the alternate sense "content about individual instances of data content" or metacontent, the type of data usually found in library catalogues.[2] Since then the fields of information management, information science, information technology, librarianship and GIS have widely adopted the term. In these fields the word *metadata* is defined as "data about data". While this is the generally accepted definition, various disciplines have adopted their own more specific explanation and uses of the term.

Libraries

Metadata have been used in various forms as a means of cataloging archived information. The Dewey Decimal System employed by libraries for the classification of library materials is an early example of metadata usage. Library catalogues used 3x5 inch cards to display a book's title, author, subject matter, and a brief plot synopsis along with an abbreviated alpha-numeric identification system which indicated the physical location of the book within the library's shelves. Such data help classify, aggregate, identify, and locate a particular book. Another form of older metadata collection is the use by US Census Bureau of what is known as the "Long Form." The Long Form asks questions that are used to create demographic data to find patterns of distribution.

Photographs

Metadata may be written into a digital photo file that will identify who owns it, copyright and contact information, what camera created the file, along with exposure information and descriptive information such as keywords about the photo, making the file searchable on the computer and/or the Internet. Some metadata are written by the camera and some is input by the photographer and/or software after downloading to a computer. However, not all digital cameras enable you to edit metadata; this functionality has been available on most Nikon DSLRs since the Nikon D3 and on most new Canon cameras since the Canon EOS 7D.

Photographic Metadata Standards are governed by organizations that develop the following standards. They include, but are not limited to:

- IPTC Information Interchange Model IIM (International Press Telecommunications Council),
- IPTC Core Schema for XMP
- XMP – Extensible Metadata Platform (an ISO standard)
- Exif – Exchangeable image file format, Maintained by CIPA (Camera & Imaging Products Association) and published by JEITA (Japan Electronics and Information Technology Industries Association)
- Dublin Core (Dublin Core Metadata Initiative – DCMI)
- PLUS (Picture Licensing Universal System).

Video

Metadata are particularly useful in video, where information about its contents (such as transcripts of conversations and text descriptions of its scenes) are not directly understandable by a computer, but where efficient search is desirable.

Web pages

Web pages often include metadata in the form of meta tags. Description and keywords meta tags are commonly used to describe the Web page's content. Most search engines use these data when adding pages to their search index.

Creation of metadata

Metadata can be created either by automated information processing or by manual work. Elementary metadata captured by computers can include information about when an object was created, who created it, when it was last updated, file size and file extension.

For the purposes of this article, an "object" refers to any of the following:

- A physical item such as a book, CD, DVD, map, chair, table, flower pot, etc.
- An electronic file such as a digital image, digital photo, document, program file, database table, etc.

Metadata types

The metadata application is manyfold covering a large variety of fields of application there are nothing but specialised and well accepted models to specify types of metadata. Bretheron & Singley (1994) distinguish between two distinct classes: structural/control metadata and guide metadata. **Structural metadata** are used to describe the structure of database objects such as tables, columns, keys and indexes. **Guide metadata** are used to help humans find specific items and are usually expressed as a set of keywords in a natural language. According to Ralph Kimball metadata can be divided into 2 similar categories: technical metadata and business metadata. **Technical metadata** correspond to internal metadata, *business metadata* - to external metadata. Kimball adds a third category named **process metadata**. On the other hand, NISO distinguishes among three types of metadata: descriptive, structural and administrative. **Descriptive metadata** are the information used to search and locate an object such as title, author, subjects, keywords, publisher; **structural metadata** give a description of how the components of the object are organised; and **administrative metadata** refer to the technical information including file type. Two sub-types of administrative metadata are rights management metadata and preservation metadata.

Metadata structures

Metadata (metacontent), or more correctly, the vocabularies used to assemble metadata (metacontent) statements, are typically structured according to a standardized concept using a well-defined metadata scheme, including: metadata standards and metadata models. Tools such as controlled vocabularies, taxonomies, thesauri, data dictionaries and metadata registries can be used to apply further standardization to the metadata. Structural metadata commonality is also of paramount importance in data model development and in database design.

Metadata syntax

Metadata (metacontent) syntax refers to the rules created to structure the fields or elements of metadata (metacontent). A single metadata scheme may be expressed in a number of different markup or programming languages, each of which requires a different syntax. For example, Dublin Core may be expressed in plain text, HTML, XML and RDF.

A common example of (guide) metacontent is the bibliographic classification, the subject, the Dewey Decimal class number. There is always an implied statement in any "classification" of some object. To classify an object as, for example, Dewey class number 514 (Topology) (i.e. books having the number 514 on their spine) the implied statement is: "<book><subject heading><514>. This is a subject-predicate-object triple, or more importantly, a class-attribute-value triple. The first two elements of the triple (class, attribute) are pieces of some structural metadata having a defined semantic. The third element is a value, preferably from some controlled vocabulary, some reference (master) data. The combination of the metadata and master data elements results in a statement which is a metacontent statement i.e. "metacontent = metadata + master data". All these elements can be thought of as "vocabulary". Both metadata and master data are vocabularies which can be assembled into metacontent statements. There are many sources of these vocabularies, both meta and master data: UML, EDIFACT, XSD, Dewey/UDC/LoC, SKOS, ISO-25964, Pantone, Linnaean Binomial Nomenclature etc. Using controlled vocabularies for the components of metacontent statements, whether for indexing or finding, is endorsed by ISO-25964 [3]: "If both the indexer and the searcher are guided to choose the same term for the same concept, then relevant documents will be retrieved." This is particularly relevant when considering the behemoth of the internet, Google. It simply indexes pages then matches text strings using its complex algorithm, there is no intelligence or "inferencing" occurring. Just the illusion thereof.

Hierarchical, linear and planar schemata

Metadata schema can be hierarchical in nature where relationships exist between metadata elements and elements are nested so that parent-child relationships exist between the elements. An example of a hierarchical metadata schema is the IEEE LOM schema where metadata elements may belong to a parent metadata element. Metadata schema can also be one-dimensional, or linear, where each element is completely discrete from other elements and classified according to one dimension only. An example of a linear metadata schema is Dublin Core schema which is one dimensional. Metadata schema are often two dimensional, or planar, where each element is completely discrete from other elements but classified according to two orthogonal dimensions.

Metadata hypermapping

In all cases where the metadata schemata exceed the planar depiction, some type of hypermapping is required to enable display and view of metadata according to chosen aspect and to serve special views. Hypermapping frequently applies to layering of geographical and geological information overlays.

Granularity

The degree to which the data or metadata are structured is referred to as their granularity. Metadata with a high granularity allow for deeper structured information and enable greater levels of technical manipulation however, a lower level of granularity means that metadata can be created for considerably lower costs but will not provide as detailed information. The major impact of granularity is not only on creation and capture, but moreover on maintenance. As soon as the metadata structures get outdated, the access to the referred data will get outdated. Hence granularity shall take into account the effort to create as well as the effort to maintain.

Metadata standards

International standards apply to metadata. Much work is being accomplished in the national and international standards communities, especially ANSI (American National Standards Institute) and ISO (International Organization for Standardization) to reach consensus on standardizing metadata and registries.

The core standard is ISO/IEC 11179-1:2004 and subsequent standards (see ISO/IEC 11179). All yet published registrations according to this standard cover just the definition of metadata and do not serve the structuring of metadata storage or retrieval neither any administrative standardisation. It is important to note that this standard refers to metadata as the data about containers of the data and not to metadata (metacontent) as the data about the data contents. It should also be noted that this standard describes itself originally as a "data element" registry, describing disembodied data elements, and explicitly disavows the capability of containing complex structures. Thus the original term "data element" is more applicable than the later applied buzzword "metadata".

The Dublin Core metadata terms are a set of vocabulary terms which can be used to describe resources for the purposes of discovery. The original set of 15 classic metadata terms, known as the Dublin Core Metadata Element Set are endorsed in the following standards documents:

- IETF RFC 5013
- ISO Standard 15836-2009
- NISO Standard Z39.85.

Although not a standard, Microformat (also mentioned in the section metadata on the internet below) is a web-based approach to semantic markup which seeks to re-use existing HTML/XHTML tags to convey metadata. Microformat follows XHTML and HTML standards but is not a standard in itself. One advocate of microformats, Tantek Çelik, characterized a problem with alternative approaches:

> Here's a new language we want you to learn, and now you need to output these additional files on your server. It's a hassle. (Microformats) lower the barrier to entry.

Metadata usage

Data virtualization

Data virtualization has emerged as the new software technology to complete the virtualization stack in the enterprise. Metadata are used in data virtualization servers which are enterprise infrastructure components, alongside database and application servers. Metadata in these servers are saved as persistent repository and describe business objects in various enterprise systems and applications. Structural metadata commonality is also important to support data virtualization.

Statistics and census services

Standardization work has had a large impact on efforts to build metadata systems in the statistical community[*citation needed*]. Several metadata standardsWikipedia:Avoid weasel words are described, and their importance to statistical agencies is discussed. Applications of the standardsWikipedia:Avoid weasel words at the Census Bureau, Environmental Protection Agency, Bureau of Labor Statistics, Statistics Canada, and many others are described[*citation needed*]. Emphasis is on the impact a metadata registry can have in a statistical agency.

Library and information science

Libraries employ metadata in library catalogues, most commonly as part of an Integrated Library Management System. Metadata are obtained by cataloguing resources such as books, periodicals, DVDs, web pages or digital images. These data are stored in the integrated library management system, ILMS, using the MARC metadata standard. The purpose is to direct patrons to the physical or electronic location of items or areas they seek as well as to provide a description of the item/s in question.

More recent and specialized instances of library metadata include the establishment of digital libraries including e-print repositories and digital image libraries. While often based on library principles, the focus on non-librarian use, especially in providing metadata, means they do not follow traditional or common cataloging approaches. Given the custom nature of included materials, metadata fields are often specially created e.g. taxonomic classification fields, location fields, keywords or copyright statement. Standard file information such as file size and format are usually automatically included.

Standardization for library operation has been a key topic in international standardization (ISO) for decades. Standards for metadata in digital libraries include Dublin Core, METS, MODS, DDI, ISO standard Digital Object Identifier (DOI), ISO standard Uniform Resource Name (URN), PREMIS schema, Ecological Metadata Language, and OAI-PMH. Leading libraries in the world give hints on their metadata standards strategies.

Metadata and the law

United States

Problems involving metadata in litigation in the United States are becoming widespread.Wikipedia:Manual of Style/Dates and numbers#Chronological items Courts have looked at various questions involving metadata, including the discoverability of metadata by parties. Although the Federal Rules of Civil Procedure have only specified rules about electronic documents, subsequent case law has elaborated on the requirement of parties to reveal metadata. In October 2009, the Arizona Supreme Court has ruled that metadata records are public record.

Document metadata have proven particularly important in legal environments in which litigation has requested metadata, which can include sensitive information detrimental to a party in court.

Using metadata removal tools to "clean" documents can mitigate the risks of unwittingly sending sensitive data. This process partially (see Data remanence) protects law firms from potentially damaging leaking of sensitive data through electronic discovery.

Metadata in healthcare

Australian researches in medicine started a lot of metadata definition for applications in health care. That approach offers the first recognized attempt to adhere to international standards in medical sciences instead of defining a proprietary standard under the WHO umbrella first.

The medical community yet did not approve the need to follow metadata standards despite respective research.[3]

Metadata and data warehousing

Data warehouse (DW) is a repository of an organization's electronically stored data. Data warehouses are designed to manage and store the data whereas the business intelligence (BI) focuses on the usage of the data to facilitate reporting and analysis.[4]

The purpose of a data warehouse is to house standardized, structured, consistent, integrated, correct, cleansed and timely data, extracted from various operational systems in an organization. The extracted data are integrated in the data warehouse environment in order to provide an enterprise wide perspective, one version of the truth. Data are structured in a way to specifically address the reporting and analytic requirements. The design of structural metadata commonality using a data modeling method such as entity relationship model diagramming is very important in any data warehouse development effort.

An essential component of a data warehouse/business intelligence system is the metadata and tools to manage and retrieve the metadata. Ralph Kimball describes metadata as the DNA of the data warehouse as metadata defines the elements of the data warehouse and how they work together.

Kimball et al. refers to three main categories of metadata: Technical metadata, business metadata and process metadata. Technical metadata are primarily definitional, while business metadata and process metadata are primarily descriptive. Keep in mind that the categories sometimes overlap.

- **Technical metadata** define the objects and processes in a DW/BI system, as seen from a technical point of view. The technical metadata include the system metadata which define the data structures such as: tables, fields, data types, indexes and partitions in the relational engine, and databases, dimensions, measures, and data mining models. Technical metadata define the data model and the way it is displayed for the users, with the reports, schedules, distribution lists and user security rights.
- **Business metadata** are a content from the data warehouse described in more user-friendly terms. The business metadata tell you what data you have, where they come from, what they mean and what their relationship is to other data in the data warehouse. Business metadata may also serve as a documentation for the DW/BI system. Users who browse the data warehouse are primarily viewing the business metadata.
- **Process metadata** are used to describe the results of various operations in the data warehouse. Within the ETL process, all key data from tasks are logged on execution. This includes start time, end time, CPU seconds used, disk reads, disk writes and rows processed. When troubleshooting the ETL or query process, this sort of data becomes valuable. Process metadata are the fact measurement when building and using a DW/BI system. Some organizations make a living out of collecting and selling this sort of data to companies - in that case the process metadata becomes the business metadata for the fact and dimension tables. Collecting process metadata is in the interest of business people who can use the data to identify the users of their products, which products they are using and what level of service they are receiving.

Metadata on the Internet

The HTML format used to define web pages allows for the inclusion of a variety of types of metadata, from basic descriptive text, dates and keywords to further advanced metadata schemes such as the Dublin Core, e-GMS, and AGLS[5] standards. Pages can also be geotagged with coordinates. Metadata may be included in the page's header or in a separate file. Microformats allow metadata to be added to on-page data in a way that users do not see, but computers can readily access.

Interestingly, many search engines are cautious about using metadata in their ranking algorithms due to exploitation of metadata and the practice of search engine optimization, SEO, to improve rankings. See Meta element article for further discussion. Studies show that search engines respond to web pages with metadata implementations.[6]

Metadata in the broadcast industry

In broadcast industry, metadata are linked to audio and video Broadcast media to:

- *identify* the media: clip or playlist names, duration, timecode, etc.
- *describe* the content: notes regarding the quality of video content, rating, description (for example, during a sport event, keywords like *goal*, *red card* will be associated to some clips)
- *classify* media: metadata allow to sort the media or to easily and quickly find a video content (a TV news could urgently need some archive content for a subject). For example, the BBC have a large subject classification system, Lonclass, a customized version of the more general-purpose Universal Decimal Classification.

These metadata can be linked to the video media thanks to the video servers. All latest broadcasted sport events like FIFA World Cup or Olympic Games use these metadata to distribute their video content to TV stations through keywords. It's often the host broadcaster who is in charge of organizing metadata through its *International Broadcast Centre* and its video servers. Those metadata are recorded with the images and are entered by metadata operators (*loggers*) who associate in live metadata available in *metadata grids* through software (such as Multicam(LSM) or IPDirector used during FIFA World Cup or Olympic Games).

Geospatial metadata

Metadata that describe geographic objects (such as datasets, maps, features, or simply documents with a geospatial component) have a history dating back to at least 1994 (refer MIT Library page on FGDC Metadata [8]). This class of metadata is described more fully on the Geospatial metadata page.

Ecological and environmental metadata

Ecological and environmental metadata are intended to document the who, what, when, where, why, and how of data collection for a particular study. Metadata should be generated in a format commonly used by the most relevant science community, such as Darwin Core, Ecological Metadata Language, or Dublin Core. Metadata editing tools exist to facilitate metadata generation (e.g. Metavist, Mercury: Metadata Search System, Morpho). Metadata should describe provenance of the data (where they originated, as well as any transformations the data underwent) and how to give credit for (cite) the data products.

Digital music

CDs such as recordings of music will carry a layer of metadata about the recordings such as dates, artist, genre, copyright owner, etc. The metadata, not normally displayed by CD players, can be accessed and displayed by specialized music playback and/or editing applications.

The metadata for compressed and uncompressed digital music is often encoded in the ID3 tag. Common editors such as TagLib support MP3, Ogg Vorbis, FLAC, MPC, Speex, WavPack TrueAudio, WAV, AIFF, MP4 and ASF file formats.

Cloud applications

With the availability of Cloud applications, which include those to add metadata to content, metadata is increasingly available over the Internet.

Metadata administration and management

Metadata storage

Metadata can be stored either *internally*, in the same file as the data, or *externally*, in a separate file. Metadata that are embedded with content is called *embedded metadata*. A data repository typically stores the metadata *detached* from the data. Both ways have advantages and disadvantages:

- Internal storage allows transferring metadata together with the data they describe; thus, metadata are always at hand and can be manipulated easily. This method creates high redundancy and does not allow holding metadata together.
- External storage allows bundling metadata, for example in a database, for more efficient searching. There is no redundancy and metadata can be transferred simultaneously when using streaming. However, as most formats use URIs for that purpose, the method of how the metadata are linked to their data should be treated with care. What if a resource does not have a URI (resources on a local hard disk or web pages that are created on-the-fly using a content management system)? What if the metadata can only be evaluated if there is a connection to the Web, especially when using RDF? How to realize that a resource is replaced by another with the same name but different content?

Moreover, there is the question of data format: storing metadata in a human-readable format such as XML can be useful because users can understand and edit it without specialized tools. On the other hand, these formats are not optimized for storage capacity; it may be useful to store metadata in a binary, non-human-readable format instead to speed up transfer and save memory.

Metadata management

Metadata management is the end-to-end process and governance framework for creating, controlling, enhancing, attributing, defining and managing a metadata schema, model or other structured aggregation, either independently or within a repository and the associated supporting processes (often to enable the management of content). The world Wide Web Consortium (W3C) has identified Governance as a key challenge in the advancement of third generation Web Technologies (Web 3.0, Semantic Web), and a number of research prototypes, such as S3DB, explore the use of semantic modeling to identify practical solutions.

Database management

Each relational database system has its own mechanisms for storing metadata. Examples of relational-database metadata include:

- Tables of all tables in a database, their names, sizes and number of rows in each table.
- Tables of columns in each database, what tables they are used in, and the type of data stored in each column.

In database terminology, this set of metadata is referred to as the catalog. The SQL standard specifies a uniform means to access the catalog, called the information schema, but not all databases implement it, even if they implement other aspects of the SQL standard. For an example of database-specific metadata access methods, see Oracle metadata. Programmatic access to metadata is possible using APIs such as JDBC, or SchemaCrawler.

- Agris: International Information System for the Agricultural Sciences and Technology
- Classification scheme
- Crosswalk (metadata)
- DataONE
- Data Dictionary (aka metadata repository)
- Dublin Core
- Folksonomy
- GEOMS – Generic Earth Observation Metadata Standard
- IPDirector
- ISO/IEC 11179
- Knowledge tag
- Mercury: Metadata Search System
- Meta element
- Metadata Access Point Interface
- Metadata discovery
- Metadata facility for Java
- Metadata from Wikiversity
- Metadata publishing
- Metadata registry
- Metamathematics
- METAFOR Common Metadata for Climate Modelling Digital Repositories
- Microcontent
- Microformat
- Multicam(LSM)
- Ontology (computer science)
- Official statistics
- Paratext
- Preservation Metadata
- SDMX
- Semantic Web
- SGML
- The Metadata Company
- Universal Data Element Framework
- Vocabulary OneSource
- XSD

References

[1] Hüner, K.; Otto, B.; Österle, H.: Collaborative management of business metadata, in: International Journal of Information Management, 2011

[2] "The notion of "metadata" introduced by Bagley".

[3] M. Löbe, M. Knuth, R. Mücke TIM: A Semantic Web Application for the Specification of Metadata Items in Clinical Research (http://ceur-ws.org/Vol-559/Paper1.pdf), CEUR-WS.org, urn:nbn:de:0074-559-9

[4] Inmon, W.H. Tech Topic: What is a Data Warehouse? Prism Solutions. Volume 1. 1995.

[5] National Archives of Australia, AGLS Metadata Standard, accessed 7 January 2010, (http://www.naa.gov.au/records-management/create-capture-describe/describe/AGLS/index.aspx)

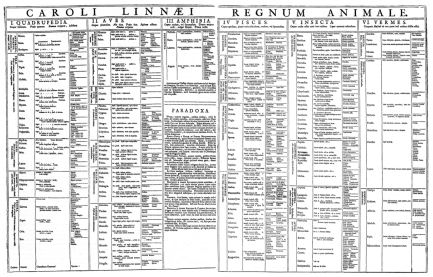
[6] The impact of webpage content characteristics on webpage visibility in search engine results http://web.simmons.edu/~braun/467/part_1.pdf

External links

- Mercury: Metadata Management, Data Discovery and Access (http://mercury.ornl.gov/ornldaac), managed by Oak Ridge National Laboratory Distributed Active Archive Center
- Guardian US interactive team. " A Guardian guide to your Metadata (http://www.guardian.co.uk/technology/interactive/2013/jun/12/what-is-metadata-nsa-surveillance#meta=0000000)." *The Guardian*. Wednesday 12 June 2013.
- Metacrap: Putting the torch to seven straw-men of the meta-utopia (http://www.well.com/~doctorow/metacrap.htm) – Cory Doctorow's opinion on the limitations of metadata on the Internet, 2001
- Retrieving Meta Data from Documents and Pictures Online (http://www.anonwatch.com/?p=9) - AnonWatch
- Understanding Metadata (http://www.niso.org/publications/press/UnderstandingMetadata.pdf) - NISO, 2004
- DataONE (http://www.dataone.org) Investigator Toolkit
- Journal of Library Metadata (http://www.informaworld.com/openurl?genre=journal&issn=1938-6389), Routledge, Taylor & Francis Group, ISSN 1937-5034
- International Journal of Metadata, Semantics and Ontologies (IJMSO) (http://www.inderscience.com/ijmso), Inderscience Publishers, ISSN 1744-263X
- AFC2IC Vocabulary OneSource Tool (https://gcic.af.mil/onesource)
- On metadata and metacontent (http://www.metalounge.org/_literature_52579/Stephen_Machin_â_ON_METADATA_AND_METACONTENT) archiv.org (http://web.archive.org/web/*/http://www.metalounge.org/_literature_52579/Stephen_Machin_â_ON_METADATA_AND_METACONTENT)
- Managing Metadata (http://library.caltech.edu/laura/) blog

Meta-data management

Meta-data management (also known as metadata management, without the hyphen) involves managing data about *other data*, whereby this "other data" is generally referred to as *content* data. The term is used most often in relation to Digital media, but older forms of metadata are catalogs, dictionaries, and taxonomies. For example, the Dewey Decimal Classification is a metadata management system for books developed in 1876 for libraries.



Linnaean taxonomy, a metadata system used historically for grouping animals in zoos, first published in 1735

Metadata schema

Metadata management can be defined as the end-to-end process and governance framework for creating, controlling, enhancing, attributing, defining and managing a metadata schema, model or other structured aggregation system, either independently or within a repository and the associated supporting processes (often to enable the management of content). For web-based systems, URLs, images, video etc. may be referenced from a triples table of object, attribute and value.

Scope

With specific knowledge domains, the boundaries of the metadata for each must be managed, since a general ontology is not useful to experts in one field whose language is knowledge-domain specific.

Metadata manager

If one is in the process of making a knowledge management solution, creating a metadata schema and developing a system in which metadata is managed are very important. In such a project, a dedicated metadata manager may be appointed in order to maintain adherence to metadata and information management standards. [*citation needed*] This is a person who will be responsible for the metadata strategy, and possibly, the implementation. A metadata manager does not need to know about and be involved with everything concerning the solution, but it does help to have an understanding of as much of the process as possible to make sure a relevant schema is developed.

Card catalog and digital media access point

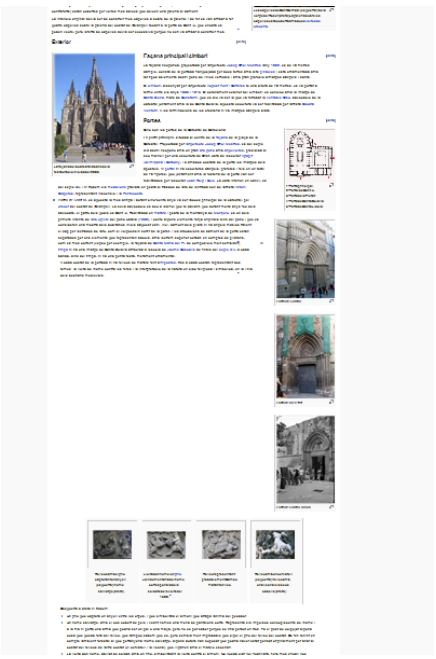
Metadata management over time

Managing the metadata in a knowledge management solution is an important step in a metadata strategy. It is part of the strategy to make sure that the metadata are complete, current and correct at any given time. Managing a metadata project is also about making sure that users of the system are aware of the possibilities allowed by a well-designed metadata system and how to maximize the benefits of metadata. Regular monitoring the metadata to ensure that the schema remains relevant is advised.

Wikipedia metadata

Wikipedia is a project that actively manages metadata for its articles and files. For example, volunteer editors carefully curate new biographical articles based on the notability (*claim to fame*), name, birth, and/or death dates.[1] Similarly, volunteer editors carefully curate new architectural articles based on name, municipality, or geo coordinates.[2] When new articles with a valid alternate spelling are added to Wikipedia that match up to existing articles based on metadata, these are then manually checked and if needed, tagged for merging.[3] When new articles are added that are considered out of scope or otherwise unfit for Wikipedia, these are nominated for deletion.[4] To help keep track of metadata on Wikipedia, the new Wikimedia project Wikidata was established in 2012. Click on the pictures to view more metadata about these images:

This picture of the Barcelona Cathedral was uploaded to the English Wikipedia in 2003 to illustrate its Wikipedia article, and was transferred to Wikimedia Commons in 2007 so it could be used in other language versions of Wikipedia.



This screenprint of the Catalan Wikipedia page on the cathedral features several photos including this one. The screenprint was uploaded to Wikimedia Commons in 2007 soon after the photo was available there, but that article on the Catalan Wikipedia has since been expanded.

References

[1] See the internal Wikipedia project on the English Wikipedia called Wikipedia:WikiProject Biography
[2] See Wikipedia:WikiProject Architecture
[3] See Wikipedia:WikiProject Merge
[4] See Wikipedia:Articles for deletion

Metadatabase

Metadatabase is a database model for (1) **metadata** management, (2) global query of independent databases, and (3) distributed data processing.[1][2][3][4][5] The word *metadatabase* is an addition to the dictionary. Originally, metadata was only a common term referring simply to "data about data", such as tags, keywords, and markup headers. However, in this technology, the concept of metadata is extended to also include such data and knowledge representation as information models (e.g., relations, entities-relationships, and objects), application logic (e.g., production rules), and analytic models (e.g., simulation, optimization, and mathematical algorithms). In the case of analytic models, it is also referred to as a **Modelbase**.[6]

These classes of metadata are integrated with some modeling ontology[7] to give rise to a stable set of meta-relations (tables of metadata). Individual models are interpreted as metadata and entered into these tables. As such, models are inserted, retrieved, updated, and deleted in the same manner as ordinary data do in an ordinary (relational) database. Users will also formulate global queries and requests for processing of local databases through the Metadatabase, using the globally integrated metadata. The Metadatabase structure can be implemented in any open technology for relational databases.

Significance

The Metadatabase technology is developed at Rensselaer Polytechnic Institute at Troy, New York, by a group of faculty and students (see the references at the end of the article), starting in late 1980s. Its main contribution includes the extension of the concept of metadata and metadata management, and the original approach of designing a database for metadata applications. These conceptual results continue to motivate new research and new applications. At the level of particular design, its openness and scalability is tied to that of the particular ontology proposed: It requires reverse-representation of the application models in order to save them into the meta-relations. In theory, the ontology is neutral, and it has been proven in some industrial applications.[8] However, it needs more development to establish it for the field as an open technology. The requirement of reverse-representation is common to any global information integration technology. A way to facilitate it is in the Metadatabase approach is to distribute a core portion of it at each local site, to allow for peer-to-peer translation on the fly.

References

[1] Hsu, C., Bouziane, M., Rattner, L. and Yee, L. "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach", IEEE Transactions on Software Engineering, Vol. SE-17, No. 6, June 1991, pp. 604-624.

[2] Babin, G. and Hsu, C. "Decomposition of Knowledge for Concurrent Processing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 5, 1996, pp 758-772.

[3] Cheung, W. and Hsu, C. "The Model-Assisted Global Query System for Multiple Databases in Distributed Enterprises," *ACM Transactions on Information Systems*, Vol. 14, No.4, Oct 1996, Pages 421-470.

[4] Boonjing, V. and Hsu, C., "A New Feasible Natural Language Database Query Method," *International Journal on Artificial Intelligence Tools*, Vol. 20, No. 10, 2006, pp. 1-8.

[5] Levermore, D., Babin, G., and Hsu, C. "A New Design for Open and Scalable Collaboration of Independent Databases in Digitally Connected Enterprises," Journal of the Association for Information Systems, 2009.

[6] Hsu, C., Service Science: Design for Service Scaling and Transformation, World Scientific and Imperial College Press, 2009. ISBN 978-981-283-676-2, ISBN 981-283-676-4.

[7] Hsu, C. Tao, Y.-C., Bouziane, M and Babin, G. "Paradigm Translations in Integrating Manufacturing Information Using a Meta-model: The TSER Approach," Information Systems Engineering, France, Vol.1, No. 3, pp.325-352, 1993.

[8] Cho, J. and Hsu, C. "A Tool for Minimizing Update Errors for Workflow Applications: the CARD Model," Computers and Industrial Engineering, Vol. 49, 2005, pp 199-220.

External links

- http://viu.eng.rpi.edu/

Metadirectory

A **metadirectory** system provides for the flow of data between one or more directory services and databases, in order to maintain synchronization of that data, and is an important part of identity management systems. The data being synchronized typically are collections of entries that contain user profiles and possibly authentication or policy information. Most metadirectory deployments synchronize data into at least one LDAP-based directory server, to ensure that LDAP-based applications such as single sign-on and portal servers have access to recent data, even if the data is mastered in a non-LDAP data source.

Metadirectory products support filtering and transformation of data in transit.

Most identity management suites from commercial vendors include a metadirectory product, or a user provisioning product.

Open source software

Ldap Synchronization Connector [1] synchronize data from any data source including databases, LDAP directories or files by reading, transforming and comparing these data between the source and the target referentials. These connectors can then be used to continuously synchronize a data source to a directory, for a one shot import or just to compare differences by outputting CSV or LDIF format reports.

References

[1] http://lsc-project.org/wiki/about/start

Metadata controller

Metadata controller (or MDC) is a storage area network (SAN) technology for managing file locking, space allocation and data access authorization. This is needed when several clients are given block level access to the same disk volume.

The abstract for the patent describing this technology can be read here [1]

References

[1] http://www.freepatentsonline.com/7448077.html

Data element

In metadata, the term **data element** is an atomic unit of data that has precise meaning or precise semantics. A data element has:

1. An identification such as a data element name
2. A clear data element definition
3. One or more representation terms
4. Optional enumerated values Code (metadata)
5. A list of synonyms to data elements in other metadata registries Synonym ring

Data elements usage can be discovered by inspection of software applications or application data files through a process of manual or automated Application Discovery and Understanding. Once data elements are discovered they can be registered in a metadata registry.

In telecommunication, the term **data element** has the following components:

1. A named unit of data that, in some contexts, is considered indivisible and in other contexts may consist of data items.
2. A named identifier of each of the entities and their attributes that are represented in a database.
3. A basic unit of information built on standard structures having a unique meaning and distinct units or values.
4. In electronic record-keeping, a combination of characters or bytes referring to one separate item of information, such as name, address, or age.

In the areas of databases and data systems more generally a data element is a concept forming part of a data model. As an element of data representation, a collection of data elements forms a data structure.[1]

In Practice

In practice, data elements (fields, columns, attributes, etc.) are sometimes "over loaded", meaning a given data element will have multiple potential meanings. While a known bad practice, over loading is nevertheless a very real factor or barrier to understanding what a system is doing.

References

[1] Beynon-Davies P. (2004). Database Systems 3rd Edition. Palgrave, Basingstoke, UK

External links

- Association for Enterprise Integration (http://www.afei.org/)
- Federal XML Developer's Guide (http://xml.gov/documents/in_progress/developersguide.pdf)
- ISO/IEC 11179 Standards (http://isotc.iso.org/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm) (see ISO/IEC 11179-3:2003 clause 3.3.36)

Metadata publishing

Metadata publishing is the process of making metadata data elements available to external users, both people and machines using a formal review process and a commitment to change control processes.

Metadata publishing is the foundation upon which advanced distributed computing functions are being built. But like building foundations, care must be taken in metadata publishing systems to ensure the structural integrity of the systems built on top of them.

Definition of metadata publishing

Published metadata has the following characteristics:

1. Metadata structures available to the general public on a public web site or by a download
2. There is a documented review and approval process for adding or updating data elements to the system
3. New releases are made available without disturbing prior versions
4. A publishing organization that makes a commitment to change control process

Benefits of metadata publishing

When classifying benefits of metadata publishing two groups are usually considered. External parties are usually consumers of information that are not part of the publishing organization. Internal parties are usually the various business units or departments within an organization.

Benefits to external parties

1. Allows external systems (both people and agents) to have a clear understanding of the semantics of data elements in a system
2. Allows third parties to build semantic maps between data models and import and export data between systems
3. Promotes service oriented architectures and allow horizontal sharing of information between traditional information silos
4. Allows systems to participate in accurately indexed and federated search processes

Benefits to internal parties

1. allows parties from diverse business units to agree on shared data definitions and separate department or function specific definitions
2. makes Extract, transform, load (ETL) operations more precise for data warehousing
3. allows user interface designers to access a common pool of screen and report header labels
4. promotion of model-driven architecture

Objections to metadata publishing

- Organizations that publish their metadata could make it easier for unauthorized people to find sensitive data if they breach an organization's firewall
- Vendors that publish their metadata risk customers creating tools that could allow their customers to export their data from computer systems therefor making it easier to migrate off of a vendor's system

Core process in metadata publishing

The following are some of the core processes in metadata publishing

1. Gathering of metadata requirements
2. Selection of metadata registry and metadata publishing tools
3. Training of metadata concepts to project participants
4. Stakeholder group formation
5. Metadata harvesting
6. Glossary consolidation
7. Initial upper ontology construction (abstract data elements)
8. Draft data element loading
9. Data element review process
10. Publishing approved metadata elements in a variety of output formats (see below)
11. Creation and maintenance of versions and depreciation of unused or redundant data elements

File format metadata publishing

Organizations that create applications that store data in file systems can also publish metadata definitions. One common way to perform this is to store application data in a compressed XML file format. The XML files can be uncompressed and validated against an external XML Schema. An example of this is done by the Open Source FreeMind tool.

Metadata publishing formats

1. HTML - used for browsing a web site and indexing by text-based search engines
2. Web Ontology Language (OWL) - used by metadata search engines such as Swoogle
3. XML Metadata Interchange (XMI) - OMG standard for exchanging metadata
4. Common Warehouse Metamodel (CMW) - OMG standard for data warehouse metadata
5. Topic maps - an ISO standard for the representation and interchange of knowledge, with an emphasis on the findability of information.
6. KM3 or Kernel Meta Meta Model as used in the Metamodel Zoos. The AtlanticZoo [1] is an open source library of more than 100 metamodels under EPL License. KM3 [2] is a simple Domain Specific Language for specifying metamodels. A number of transformations are available to translate from KM3 to other notations like XMI.

External links

- MetaQuery examples [3] provided by Ambient Webs LLC
- SWED portal [4] provided by WordPressHelp
- Microsoft Metadata Publishing Example [5]

References

[1] http://www.eclipse.org/gmt/am3/zoos/
[2] http://www.eclipse.org/gmt/atl/doc/KernelMetaMetaModel
[3] http://ambientwebs.com/
[4] http://wordpresshelp.org/
[5] http://winfx.msdn.microsoft.com/library/default.asp?url=/library/en-us/indigo_samples/html/78c13633-d026-4814-910e-1c801cffdac7.asp

Metadata registry

A **metadata registry** is a central location in an organization where metadata definitions are stored and maintained in a controlled method.

Use of Metadata Registries

Metadata registries are used whenever data must be used consistently within an organization or group of organizations. Examples of these situations include:

- Organizations that transmit data using structures such as XML, Web Services or EDI
- Organizations that need consistent definitions of data across time, between databases, between organizations or between processes, for example when an organization builds a data warehouse
- Organizations that are attempting to break down "silos" of information captured within applications or proprietary file formats

Central to the charter of any metadata management programme is the process of creating trusting relationships with stakeholders and that definitions and structures have been reviewed and approved by appropriate parties.

Common characteristics of a metadata registry

A metadata registry typically has the following characteristics:

- Protected environment where only authorized individuals may make changes
- Stores data elements that include both semantics and representations
- Semantic areas of a metadata registry contain the meaning of a data element with precise definitions
- Representational areas of a metadata registry define how the data is represented in a specific format, such as in a database or a structured file format (e.g., XML)

Clear separation of semantics and system-specific constraints

Because metadata registries are used to store both semantics (the meaning of a data element) and systems-specific constraints (for example the maximum length of a string) it is important to identify what systems impose these constraints and to document them. For example the maximum length of a string should not change the meaning of a data element.

The International Organization for Standardization (ISO) has published standards for a metadata registry called ISO/IEC 11179 and also ISO15000-3 and ISO15000-4 ebXML registry and repository (regrep) [1] EbXML RegRep

ISO standards

There are two ISO standards which are commonly referred to as metadata registry standards: ISO 11179 and ISO 15000-3. There are some who believe that ISO 11179 and ISO 15000-3 are interchangeable or at least in some way similar. e.g.

"Of interest is that the ISO 11179 model was one of the inputs to the ebXML RIM (registry information model) and so has much functional equivalence to the "registry" region of the ISO 11179 conceptual model." [2]

This is however incorrect. Although the specification ebRIM v2.0 (5 december 2001) says at the beginning in its Design Objectives: "Leverage as much as possible the work done in the OASIS [OAS] and the ISO 11179 [ISO] Registry models" [3] by the time of ebRIM v3.0 (2 May 2005) all reference to ISO/IOEC 11179 is reduced to a mention under informative references on page 76 of 78. [4] It was recognised by some team members that the ebXML RIM data model had no place to store "fine grained artifacts" [5] ie. the data elements which are at the heart of ISO/IEC 11179, but not until 2009 can an explicit and definitive statement from the team be found. [6]

ISO/IEC 11179

ISO/IEC 11179 says that it is concerned with "traditional" metadata: "We limit the scope of the term as it is used here in ISO/IEC 11179 to descriptions of data - the more traditional use of the term." Originally the standard named itself a "data element" registry. It describes data elements: "data elements are the fundamental units of data" and "data elements themselves contain various kinds of data that include characters, images, sound, etc." It also describes a registry with an analogy: "This is analogous to the registries maintained by governments to keep track of motor vehicles. A description of each motor vehicle is entered in the registry, but not the vehicle itself."

ebXML

The ebXML RIM says about its Repository and Registry that it is

- "... capable of storing any type of electronic content such as XML documents, text documents, images, sound and video … RepositorytItems (sic) are stored in a content repository".

It also says that it is

- "... capable of storing standardized metadata that MAY be used to further describe RepositoryItems" which metadata "… are stored in the registry".

It also describes itself with "...this familiar metaphor. An ebXML Registry is like your local library. The repository is like the bookshelves in the library. The repository items in the repository are like book (sic) on the bookshelves." It goes on to say "The registry is like the card catalog … A RegistryObject is like a card in the card catalog."

What should be immediately apparent is that something which holds catalogue cards is not "like" a catalogue, it IS a catalogue.

Unfortunately for a number of organisations that have implemented ebXML RIM to satisfy a requirement for an ISO/IEC 11179 registry, ebXML RIM

- is neither a registry

- nor does it store metadata.

It is

- a "content repository"
- and a "metacontent catalogue".

Metadata registry roles

A metadata registry is frequently set up and administered by an organization's data architect or data modeling team.

Data elements are frequently assigned to data stewards or data stewardship teams that are responsible for the maintenance of individual data elements.

Metadata element workflow

Metadata registries frequently have a formal data element submission, approval and publishing approval process. Each data element should be accepted by a data stewardship team and reviewed before data elements are published. After publication change control processes should be used.

Metadata navigation, search and publishing

Metadata registries are frequently large and complex structures and require navigation, visualization and searching tools. Use of hierarchical viewing tools are frequently an essential part of a metadata registry system. Metadata publishing consists of making data element definitions and structures available to both people and other systems.

Examples of public metadata registries

- Agency for Healthcare Research and Quality- United States Health Information Knowledgebase (USHIK) [7]
- Apelon Medical Registry [8]
- Australian Institute of Health and Welfare [9]
- Dublin Core Metadata Registry [10]
- Knowledge Network for Biocomplexity [11]
- Cancer Data Standards Repository [12]
- Global Justice XML Data Model (GJXDM) [13]
- Minnesota Department of Education Metadata Registry (K-12 Data)[14]
- National Information Exchange Model [15]
- NIST ebXML Registry for HL7 / HIMSS / IHE [16]
- Open Metadata Registry (formerly the National Science Digital Library (NSDL) Metadata Registry) [17]
- US Department of Defense Metadata Registry (requires sponsored registration) [18]
- US Environmental Protection Agency - Environmental Data Registry [19]

See also Geographic information directories: Geospatial metadata#Metadata directories

Metadata registry vendors / solutions

In alphabetical order:

- a.k.a. software by Synercon [20]
- ASG Rochade [21]
- Data Advantage Group MetaCenter [22]
- Data Consulting Group [23]
- IBM Business Glossary, Metadata Server & Metadata Workbench [24]
- InfoLibrarian Metadata Integration Framework [25]
- Informatica Metadata Manager and Business Glossary [26]
- Jumper 2.0 open-source Enterprise 2.0 metadata registry
- Masai Technologies M:GRID [27]
- Octagon Research Solutions ViewPoint MDR [28]
- Oracle MDS [29]
- SAS Metadata Repository [30]
- Software AG's webMethods OneData Metadata Registry - formerly Data Foundations [31]
- The Society of Motion Picture and Television Engineers Metadata Dictionary; Registry of Metadata Element Descriptions [32]
- freebXML Registry, A royalty-free open source project implementing OASIS ebXML RegRep standard [33]
- Wellfeet Software's WellGEO RegREP product provides an integrated Registry and Repository specialized for Geographical and Semantic Information management [34]

References

Open Forums on Metadata Registries, in reverse chronological order:

- 11th International Forum on Metadata Registries (2008 - Sydney, Australia) [35]
- 10th International Forum on Metadata Registries (2007 - NYC, USA) [36]
- 9th International Forum on Metadata Registries (2006 - Kobe, Japan) [37]
- 8th International Forum on Metadata Registries (2005 - Berlin, Germany) [38]
- 7th International Forum on Metadata Registries (2004 - Xian, China) [39]
- On ISO 11179 versus ebXML (in "On metadata and metacontent") [40]

References

[1] http://xml.coverpages.org/ISO-ebXML.html
[2] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.6331&rep=rep1&type=pdf
[3] http://xml.coverpages.org/OASIS-ebXML-RIM-v20.pdf
[4] http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf
[5] http://www.stylusstudio.com/xmldev/200503/post70270.html
[6] http://sourceforge.net/mailarchive/forum.php?thread_name=4ACBC943.8090609%40wellfleetsoftware.com&forum_name=ebxmlrr-tech
[7] http://ushik.ahrq.gov/
[8] http://sage.wherever.org/deployment/deployment.html
[9] http://meteor.aihw.gov.au/content/index.phtml/itemId/181162
[10] http://dublincore.org/dcregistry/
[11] http://knb.ecoinformatics.org
[12] http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/cadsr
[13] http://www.it.ojp.gov/jxdm/
[14] http://education.state.mn.us/mde-dd
[15] http://www.niem.gov
[16] http://hcxw2k1.nist.gov:9080/
[17] http://metadataregistry.org

[18] http://metadata.dod.mil
[19] http://www.epa.gov/edr
[20] http://www.a-k-a.com.au
[21] http://www.asg.com/Products/View/ASG-Rochade.aspx?lang=en-US
[22] http://www.dag.com/
[23] http://www.dcgroupinc.com/
[24] http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r0/index.jsp?topic=/com.ibm.swg.im.iis.productization.iisinfsv.overview.doc/topics/cisomsmetaserver.html
[25] http://www.infolibcorp.com
[26] http://www.informatica.com/uk/metadata-management/
[27] http://www.masaitechnologies.com
[28] http://www.octagonresearch.com/metadata-registry.html
[29] http://www.oracle.com/technetwork/developer-tools/jdev/metadataservices-fmw-11gr1-130345.pdf
[30] http://www.sas.com/technologies/bi/appdev/base/metadatasrv_factsheet.pdf
[31] http://www.softwareag.com/corporate/solutions/soa/mdm/business_needs/metadata.asp
[32] http://store.smpte.org/product-p/rp%200210.10-2007.htm
[33] http://ebxmlrr.sourceforge.net
[34] http://www.wellfleetsoftware.com/products
[35] http://metadataopenforum.org/index.php?id=36,187,0,0,1,0
[36] http://metadataopenforum.org/index.php?id=22,0,0,1,0,0
[37] http://www.tiu.ac.jp/org/openforum2006/
[38] http://www.berlinopenforum.de/
[39] http://www.cnis.gov.cn/openforum/english/meeting2/index.htm
[40] http://www.metalounge.org/_literature_52579/Stephen_Machin_%E2%80%93_ON_METADATA_AND_METACONTENT

Metadata facility for Java

The **Metadata Facility for Java** is a specification for Java that defines an API for annotating fields, methods, and classes as having particular attributes that indicate they should be processed in specific ways by development tools, deployment tools, or run-time libraries.

The specification was developed under the Java Community Process as JSR 175, and was released as a part of J2SE 5.0 (Tiger).

External links

- JSR 175 [1] *A Metadata Facility for the Java Programming Language*
- JSR 250 [2] *Common Annotations* (defines common Java SE and Java EE annotations)
- JSR 270 [3] *Pluggable Annotation Processing API* (defines a pluggable interface for developing build-time annotation processors)

References

[1] http://www.jcp.org/en/jsr/detail?id=175
[2] http://www.jcp.org/en/jsr/detail?id=250
[3] http://www.jcp.org/en/jsr/detail?id=269

Object Management Group

Object Management Group	
Abbreviation	OMG
Motto	We Set the Standard
Formation	1989
Headquarters	109 Highland Ave Needham, Massachusetts
Website	www.omg.org [1]

Object Management Group (**OMG**) is an international, open membership, not-for-profit computer industry standards consortium. OMG Task Forces develop enterprise integration standards for a wide range of technologies and an even wider range of industries. OMG's modeling standards enable powerful visual design, execution and maintenance of software and other processes. Originally aimed at standardizing distributed object-oriented systems, the company now focuses on modeling (programs, systems and business processes) and model-based standards.

Overview

OMG provides only specifications, and does not provide implementations. But before a specification can be accepted as a standard by OMG, the members of the winning submitter team must guarantee that they will bring a conforming product to market within a year. This is an attempt to prevent unimplemented (and unimplementable) standards.

Other private companies or open source groups are encouraged to produce conforming products and OMG is attempting to develop mechanisms to enforce true interoperability.

OMG hosts technical meetings for its members and interested nonmembers. The Technical meetings provide a neutral forum to discuss, develop and adopt standards that enable software interoperability for a wide range of industries including: Finance, Manufacturing, CORBA, Security and more. In December 2013, the TC meeting will be in Santa Clara, California; in March 2014, in Reston, Virginia; in June 2014, in Boston, MA; in September 2014 in Austin, Texas, in September 2014, in Long Beach, California; and in March 2015, they return to Reston, VA. [2]

History

Founded in 1989 by eleven companies (including Hewlett-Packard, IBM, Sun Microsystems, Apple Computer, American Airlines and Data General), OMG's initial focus was to create a heterogeneous distributed object standard. The founding executive team included Christopher Stone, Richard Soley, Bill Hoffman [3] and John Slitz. As of November, 2012, the leadership includes Chairman and CEO Richard Soley, President and COO Bill Hoffman [3] and Vice President and Technical Director Andrew Watson.

Since 2000, the OMG's International Headquarters have been located in Needham, Massachusetts. In November, 2012, the headquarters was moved from 140 Kendrick St to 109 Highland Ave.

OMG Headquarters

The goal was a common portable and interoperable object model with methods and data that work using all types of development environments on all types of platforms.

In 1997, the Unified Modeling Language (UML) was added to the list of OMG adopted technologies. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering.

In June 2005, the Business Process Management Initiative (BPMI.org) and OMG announced the merger of their respective Business Process Management (BPM) activities to form the Business Modeling and Integration Domain Task Force (BMI DTF).

In 2006 the BPMN language specification was adopted as a standard by OMG.

In 2007 the Business Motivation Model (BMM) was adopted as a standard by the OMG. The BMM is a metamodel that provides a vocabulary for corporate governance and strategic planning and is particularly relevant to businesses undertaking governance, regulatory compliance, business transformation and strategic planning activities.

In 2009 OMG, together with the Software Engineering Institute at Carnegie Mellon, launched the Consortium of IT Software Quality (CISQ). CISQ brings together industry executives from Global 2000 IT organizations, system integrators, outsourcers, and package vendors to jointly address the challenge of standardizing the measurement of IT software quality and to promote a market-based ecosystem to support its deployment.

In 2011 OMG formed the Cloud Standards Customer Council [4]. Founding sponsors included CA, IBM, Kaavo, Rackspace and Software AG. The CSCC is an OMG end user advocacy group dedicated to accelerating cloud's successful adoption, and drilling down into the standards, security and interoperability issues surrounding the transition to the cloud. The Council is not a standards organization, but will complement existing cloud standards efforts and establish a core set of client-driven requirements to ensure cloud users will have the same freedom of choice, flexibility, and openness they have with traditional IT environments. CSCC is open to all end-user organizations.

In September, 2011, the OMG Board of Directors unanimously voted to adopt the Vector Signal and Image Processing Library (VSIPL) as the latest OMG specification. Work for adopting the specification was led by Mentor Graphics' Embedded Software Division, RunTime Computing Solutions, The Mitre Corporation as well as the High Performance Embedded Computing Software Initiative (HPEC-SI). VSIPL is an application programming interface (API) defined by an open standard developed by embedded signal and image processing hardware and software vendors, academia, application developers, and government labs. VSIPL and VSIPL++ contain hundreds of functions used for common signal processing kernel and other computations. These functions include basic arithmetic, trigonometric, transcendental, signal processing, linear algebra, and image processing. The VSIPL family of libraries has been implemented by multiple vendors for a range of processor architectures, including x86, PowerPC, Cell, and NVIDIA GPUs. VSIPL and VSIPL++ are designed to achieve high performance, increase programmer productivity and maintain portability across a range of processor architectures. Additionally, VSIPL++ was designed from the start to include support for parallelism.

Late 2012 early 2013, The Object Management Group Board of Directors has adopted the Automated Function Point (AFP) specification.[5] The push for adoption was led by the Consortium for IT Software Quality (CISQ). AFP provides a standard for automating the popular Function Point measure according to the counting guidelines of the International Function Point User Group (IFPUG).

Hot Topics

On the company's website, there is a Hot Topics [6] page featuring a few of the different technology areas that are currently trending. As of December 4, 2013, the hot topics include: Data Distribution Service, Financial Industry Business Ontology (FIBO), Security Fabric, and Software-defined networking(SDN).

OMG products

Common Object Request Broker Architecture

At its founding, OMG set out to create the initial Common Object Request Broker Architecture (CORBA) standard which appeared in 1991. CORBA is a standard that enables software components written in multiple computer languages and running on multiple computers to work together (i.e., it supports multiple platforms). OMG has also developed a core set of standards adapting CORBA for embedded and real-time systems. Implementations of real time CORBA are widely used in control systems in ships and aircraft.

Data Distribution Service

Data Distribution Service for real-time systems (DDS) is a specification of a publish/subscribe middleware for distributed systems created in response to the need to augment CORBA with a data-centric publish-subscribe specification.[7]

Model Driven Architecture

OMG evolved towards modeling standards by creating the standard for Unified Modeling Language (UML) followed by related standards for

- Meta-Object Facility (MOF),
- XML Metadata Interchange (XMI)
- MOF Query/Views/Transformation (QVT).
- Model to text transformation language (MOFM2T).

These together provide the foundation for Model Driven Architecture (MDA), and related set of standards, building upon the success of UML and MOF.

Systems Modeling Language (SysML), a modeling language based on UML for use in Systems Engineering, has been standardized in collaboration with INCOSE.

Significant progress has also been made in bringing the world of UML modeling and the Semantic Web together through the adoption of the *Ontology Definition Metamodel* which relates UML models in a standard way with RDF and Web Ontology Language (OWL) models.

Semantics of Business Vocabulary and Business Rules (SBVR) is a landmark for the OMG, the first OMG specification to incorporate the formal use of natural language in modeling and the first to provide explicitly a model of formal logic. Based on a fusion of linguistics, logic, and computer science, and two years in preparation, SBVR provides a way to capture specifications in natural language and represent them in formal logic so they can be machine-processed. SBVR is an integral part of MDA.

Architecture Driven Modernization

Architecture Driven Modernization (ADM) is the reverse of MDA. ADMTF is an OMG group similar to ADTF with high potential.

Knowledge Discovery Metamodel (KDM), a common intermediate representation for existing software systems and their operating environments. Knowledge Discovery Metamodel is designed as the OMG's foundation for software modernization and software assurance. Knowledge Discovery Metamodel uses Meta-Object Facility to define an XMI interchange format between tools that work with existing software and an abstract interface for the next-generation assurance and modernization tools.

The Software Process Engineering Metamodel (SPEM) is an OMG-standard for Meta-Process Modeling.

Abstract Syntax Tree Metamodel (ASTM), a modeling language for fine grained reverse engineering.

Semantics of Business Vocabulary and Business Rules (SBVR) and KDM are designed as two parts of a unique OMG Technology Stack for software analytics related to existing software systems. KDM defines an ontology related to software artifacts and thus provides an initial formalization of the information related to a software system. SBVR is further used to formalize complex compliance rules related to the software.

Domain models

- Business models : OMG manages a number of standards for business modeling, including BPMN, the Business Motivation Model (BMM) and the Semantics of Business Vocabulary and Business Rules (SBVR) specification.
- Verticals : Considerable progress has also been made in developing vertical model-based standards in the healthcare, finance, telecommunications, manufacturing, software-defined radio, space/ground systems communications and some dozen other technology areas.

Software assurance and regulatory compliance

New activities have been initiated to address important concerns of *Regulatory Compliance* and *Software Assurance*, building upon the base standards of MDA.

Certification

OMG offers a number of professional certifications:

- OCEB - OMG Certified Expert in Business Process Management (BPM)[8]
- OCUP - OMG Certified UML Professional[9]
- OCSMP - OMG Certified Systems Modeling Professional[10]
- OCRES - OMG Certified Real-time and Embedded Systems Specialist[11]

In 2013, OMG announced that it has updated its OCEB program to reflect improvements made to the BPMN specification in Version 2.0. As of December, 2013, only the updated OCEB 2 Fundamental certification level has been made available with Business Intermediate and Technical Intermediate updates expected to be released in the next few months.[12]

Notes

[1] http://www.omg.org/
[2] OMG Upcoming Events (http://www.omg.org/news/schedule/upcoming.htm/)
[3] http://www.omg.org/contacts/hoffman.htm
[4] http://www.cloud-council.org/
[5] http://www.omg.org/news/releases/pr2013/01-17-13.htm>
[6] http://www.omg.org/hot-topics/index.htm
[7] As of December 2005 the latest standard is DDS 1.2 (http://www.omg.org/technology/documents/formal/data_distribution.htm), with version 1.3 currently available to OMG members.
[8] OCEB - OMG Certified Expert in Business Process Management (BPM) (http://www.omg.org/oceb/)
[9] OCUP - OMG Certified UML Professional (http://www.omg.org/uml-certification/)
[10] OCSMP - OMG Certified Systems Modeling Professional (http://www.omg.org/ocsmp/)
[11] OCRES - OMG Certified Real-time and Embedded Systems Specialist (http://www.omg.org/ocres/)
[12] BPM Certification Program Updated By OMG (http://www.omg.org/news/releases/pr2013/11-19-13.htm/)

References

- This article is based on material taken from Object Management Group (http://foldoc.org/index.cgi?Object+Management+Group) at the Free On-line Dictionary of Computing prior to 1 November 2008 and incorporated under the "relicensing" terms of the GFDL, version 1.3 or later.

External links

- Object Management Group website (http://www.omg.org/)
- Model Driven Architecture website (http://www.omg.org/mda/)
- Architecture-Driven Modernization website (http://adm.omg.org)
- Software Assurance website (http://swa.omg.org)
- UML Certification Program (http://www.omg.org/uml-certification/)
- CISQ website (http://www.it-cisq.org/)

Semantics of Business Vocabulary and Business Rules

The **Semantics of Business Vocabulary and Business Rules** (SBVR) is an adopted standard of the Object Management Group (OMG) intended to be the basis for formal and detailed natural language declarative description of a complex entity, such as a business. SBVR is intended to formalize complex compliance rules, such as operational rules for an enterprise, security policy, standard compliance, or regulatory compliance rules. Such formal vocabularies and rules can be interpreted and used by computer systems. SBVR is an integral part of the OMG's Model Driven Architecture (MDA).

Overview

The SBVR defines the vocabulary and rules for documenting the semantics of business vocabularies, business facts, and business rules; as well as an XMI schema for the interchange of business vocabularies and business rules among organizations and between software tools.

SBVR allows the production of business vocabularies and rules; vocabulary plus rules constitute a shared domain model with the same expressive power of standard ontological languages. SBVR allows multilingual development, since it is based on separation between symbols and their meaning. SBVR enables making business rules accessible to software tools, including tools that support the business experts in creating, finding, validating, and managing business rules, and tools that support the information technology experts in converting business rules into implementation rules for automated systems.

SBVR uses OMG's Meta-Object Facility (MOF) to provide interchange capabilities MOF/XMI mapping rules, enable generating MOF-compliant models and define an XML schema. SBVR proposes Structured English as one of possibly many notations that can map to the SBVR Metamodel.

SBVR and Knowledge Discovery Metamodel (KDM) are designed as two parts of a unique OMG Technology Stack for software analytics related to existing software systems. KDM defines an ontology related to software artifacts and thus provides an initial formalization of the information related to a software system. SBVR can be further used to formalize complex compliance rules related to the software.

Background

Business rules represent the primary means by which an organization can direct its business, defining the operative way to reach its objectives and perform its actions.

A rule-based approach to managing business and the information used by that business is a way of *identifying and articulating the rules which define the structure and control the operation of an enterprise* it represents a new way to think about enterprise and its rules, in order to enable a complete business representation made by and for business people. Business rules can play an important role in defining business semantics: they can influence or guide behaviours and support policies, responding to environmental situations and events. *Semantics of Business Vocabulary and Business Rules* (SBVR) is the OMG implementation of the business rules approach.

History

In June 2003 OMG issued the Business Semantics of Business Rule (BSBR) Request For Proposal, in order to create a standard *to allow business people to define the policies and rules by which they run their business in their own language, in terms of the things they deal with in the business, and to capture those rules in a way that is clear, unambiguous and readily translatable into other representations.* The SBVR proposal was developed by the Business Rules Team, a consortium organized in August 2003 to respond to the BSBR RFP.[1][2]

In September 2005, The Business Modeling and Integration Task Force and the Architecture Board of the Object Management Group approved the proposal *Semantics of Business Vocabulary and Business Rules (SBVR)* to become a final adopted specification in response to the RFP. Later SBVR proposal was ratified by the Domain Technical Committee (DTC), approved of the OMG Board of Directors, and SBVR finalization task force was launched to convert the proposal into ISO/OMG standard format and perform final editing prior to release as an OMG formal specification.

In January 2008, the finalization phase was completed and the Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0 formal specification was released and is publicly available[3] at the Catalog of OMG Business Strategy, Business Rules and Business Process Management Specifications [4] web page.

Conceptual Formalization

SBVR is a landmark for the OMG, the first OMG specification to incorporate the formal use of natural language in modeling and the first to provide explicitly a model of formal logic. Based on a fusion of linguistics, logic, and computer science, and two years in preparation, SBVR provides a way to capture specifications in natural language and represent them in formal logic so they can be machine-processed.

Methodologies used in software development are typically applied only when a problem is already formulated and well described. The actual difficulty lies in the previous step, that is describing problems and expected functionalities. Stakeholders involved in software development can express their ideas using a language very close to them, but they usually are not able to formalize these concepts in a clear and unambiguous way. This implies a large effort in order to interpret and understand real meanings and concepts hidden among stakeholders' words. Special constraints on syntax or predefined linguistic structures can be used in order to overcome this problem, enabling natural language to well represent and formally define problems and requirements.

The main purpose of natural language modelling is hence to make natural language suitable for conceptual modelling. The focus is on semantic aspects and shared meanings, while syntax is thought in a perspective based on formal logic mapping.

Conceptualization and representation play fundamental roles in thinking, communicating, and modeling. For each concept there is a triad of 1) the concept in our minds, 2) the real-world things conceptualized by the concept, and 3) a representation of the concept that we can use to think and communicate about the concept and its corresponding real-world things. (Note that real-world things include both concrete things and representations of those concrete things as records and processes in operational information systems.)

A conceptual model is a formal structure representing a possible world, comprising a conceptual schema and a set of facts that instantiate the conceptual schema. The conceptual schema is a combination of concepts and facts of what is possible, necessary, permissible, and obligatory in each possible world. The set of facts instantiates the conceptual schema by assertion to describe one possible world. A rule is a fact that asserts either a logical necessity or an obligation. Obligations are not necessarily satisfied by the facts; necessities are always satisfied.

SBVR contains a vocabulary for conceptual modeling and captures expressions based on this vocabulary as formal logic structures. The SBVR vocabulary allows one to formally specify representations of concepts, definitions, instances, and rules of any knowledge domain in natural language, including tabular forms. These features make SBVR well suited for describing business domains and requirements for business processes and information systems

to implement business models.

Fact-orientation

People communicate facts, that is the fact is the unit of communication. The fact-oriented approach enables multidimensional categorization.

- The fact-oriented approach supports time changeability.
- The fact-oriented approach provides semantic stability.
- The fact-oriented approach enables extensibility and reuse.
- The fact-oriented approach involves breaking down compound fact types into elementary (atomic) ones.

Conceptual formalization describes a business domain, and is composed of 1) a conceptual schema (fact structure) and 2) a population of ground facts. A business domain (universe of discourse) comprises those aspects of the business that are of interest.

The schema declares:

- the relevant fact types (kinds of ground fact, e.g. *Employee works for Department*)
- the relevant business rules (typically constraints or derivation rules).

A fact is a proposition taken to be true by the business. Population facts are restricted to elementary and existential facts.

Constraints can be static or dynamic:

- A static constraint imposes a restriction on what fact populations are possible or permitted, for each fact population taken individually e.g. *Each Employee was born on at most one Date.*
- A dynamic constraint imposes a restriction on transitions between fact populations

e.g. *a person's marital status may change from single to married, but not from divorced to single*

Derivation of facts.

- Derivation means either, how a fact type may be derived from one or more other fact types e.g.
  - *Person1 is an uncle of Person2 if Person1 is a brother of some Person3 who is a parent of Person2*
- Or, how a noun concept (object type) may be defined in terms of other object types and fact types e.g.
  - *Each FemaleAustralian is a Person who was born in Country 'Australia' and has Gender 'Female'*

Rule-based approach

Rules play a very important role in defining business semantics: they can influence or guide behaviours and support policies, responding to environmental situations and events. This means that rules represent the primary means by which an organization can direct its business, defining the operative way to reach its objectives and perform its actions.

The rule-based approach aims to address two different kinds of users:

- it addresses business communities, in order to provide them with a structured approach, based on a clear set of concepts and used to access and manage business rules;
- it addresses IT professionals, in order to provide them with a deep understanding about business rules and to help them in models creation. The rules-based approach also helps bridge the rift that can occur between the data managers and the software designers.

The essence of the rule-based conceptual formalizations is that *rules build on facts, and facts build on concepts as expressed by* terms

This mantra is memorable, but a simplification since in SBVR: Meaning is separate from expression; Fact Types (Verb Concepts) are built on Noun Concepts; Noun Concepts are represented by Terms; and Fact Types are

represented by Fact Symbols (verb phrases).

Rule statements are expressed using either alethic modality or deontic modality and require elements of modal logic as formalization.

SBVR **Structural Business Rules** use two alethic modal operators:

it is necessary that …

it is possible that …

SBVR **Operative Business Rules** use two deontic modal operators:

it is obligatory that …

it is permitted that …

Structural business rules (static constraints) are treated as alethic necessities by default, where each state of the fact model corresponds to a possible world. Pragmatically, the rule is understood to apply to all future states of the fact model, until the rule is revoked or changed. For the model theory, the necessity operator is omitted from the formula. Instead, the rule is merely tagged as a necessity. For compliance with Common Logic, such formulae can be treated as irregular expressions, with the necessity modal operator treated as an uninterpreted symbol.

If the rule includes exactly one deontic operator, e.g. O (obligation), and this is at the front, then the rule may be formalized as Op, where p is a first-order formula that is tagged as obligatory. In SBVR, this tag is assigned the informal semantics: it ought to be the case that p (for all future states of the fact model, until the constraint is revoked or changed). From a model-theoretic perspective, a model is an interpretation where each non-deontic formula evaluates to true, and the model is classified as: a permitted model if the p in each deontic formula (of the form Op) evaluates to true, otherwise the model is a forbidden model (though still a model). This approach removes any need to assign a truth value to expressions of the form Op.

SBVR is formal logic with a natural language interface

SBVR is for modeling in natural language. Based on linguistics and formal logic, SBVR provides a way to represent statements in controlled natural languages as logic structures called semantic formulations. SBVR is intended for expressing business vocabulary and business rules, and for specifying business requirements for information systems in natural language. SBVR models are declarative, not imperative or procedural. SBVR has the greatest expressivity of any OMG modeling language. The logics supported by SBVR are typed first order predicate logic with equality, restricted higher order logic (Henkin semantics), restricted deontic and alethic modal logic, set theory with bag comprehension, and mathematics. SBVR also includes projections, to support definitions and answers to queries, and questions, for formulating queries. Interpretation of SBVR semantic formulations is based on model theory. SBVR has a MOF model, so models can be structurally linked at the level of individual facts with other MDA models based on MOF.

SBVR is aligned with Common Logic – published by ISO as ISO/IEC 24707:2007.

SBVR captures business facts and business rules that may be expressed either informally or formally. Business rule expressions are formal only if they are expressed purely in terms of: fact types in the pre-declared schema for the business domain, certain logical/ mathematical operators, quantifiers etc. Formal rules are transformed into a logical formulation that is used for exchange with other rules-based software tools. Informal rules may be exchanged as un-interpreted comments. An approach to automatically generate SBVR business rules from natural language specification is presented in.

SBVR and other OMG standards

SBVR specification defines a metamodel and allows to instance it, in order to create different vocabularies and to define the related business rules; it is also possible to complete these models with data suitable to describe a specific organization. the SBVR approach provides means (i.e. mapping rules) to translate natural language artifacts into MOF-compliant artifacts; this allows to exploit all the advantages related to MOF (repository facilities, interchangeability, tools, …).

Several MDA-related OMG works in progress are expected to incorporate SBVR, including:

- Business Process Definition Metamodel (BPDM)
- Organization Structure Metamodel (OSM)
- Business Motivation Model (BMM)
- UML Profile for Production Rule Representation (PRR)
- UML Profile for the Department of Defense Architecture Framework/Ministry of Defense(Canada) Architecture Framework (DoDAF/MODAF).
- Knowledge Discovery Metamodel (KDM)
- Wider interest in SBVR– Semantic Web, OASIS

The Ontology Definition Metamodel (ODM) has been made compatible with SBVR, primarily by aligning the logic grounding of the ISO Common Logic specification (CL) referenced by ODM with the SBVR Logical Formulation of Semantics vocabulary. CL itself was modified specifically so it potentially can include the modal sentence requirements of SBVR. ODM provides a bridge to link SBVR to the Web Ontology Language for Services (OWL-S), Resource Description Framework Schema (RDFS), Unified Modeling Language (UML), Topic Map (TM), Entity Relationship Modeling (ER), Description Logic (DL), and CL.

Other programs outside the OMG are adopting SBVR. The Digital Business Ecosystem (DBE), an integrated project of the European Commission Framework Programme 6, has adopted SBVR as the basis for its Business Modeling Language.[*citation needed*] The World Wide Web Consortium (W3C) is assessing SBVR for use in the Semantic Web, through the bridge provided by ODM.[*citation needed*] SBVR will extend the capability of MDA in all these areas.

References

[1] Co-submitters of SBVR were:

- Adaptive, Business Rule Solutions LLC, Business Semantics Ltd, Hendryx & Associates, MEGA, Neumont University, Unisys Corporation

[2] Supporters of SBVR are:

- Automated Reasoning Corporation, Business Rules Group, Fujitsu Ltd, Hewlett-Packard Company, InConcept, LibRT, KnowGravity Inc, Model Systems, Ness Technologies, Perpetual Data Systems, Sandia National Laboratories, The Rule Markup Initiative, X-Change Technologies Group

[3] Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0 (formal) (http://www.omg.org/spec/SBVR/1.0/)

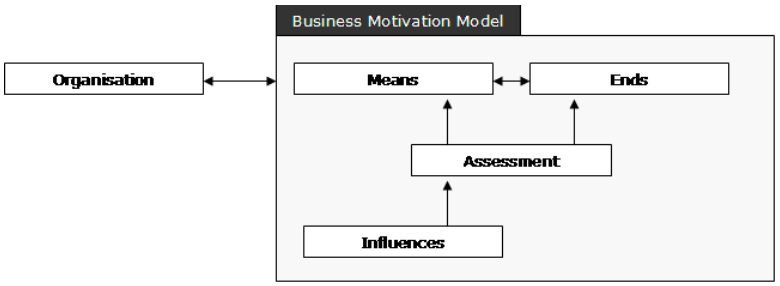
[4] http://omg.org/technology/documents/bms_spec_catalog.htm#SBVR

External links

- Business Rules Group (http://www.businessrulesgroup.org/sbvr.shtml)

Business Motivation Model

The **Business Motivation Model** (BMM) in enterprise architecture provides a scheme and structure for developing, communicating, and managing business plans in an organized manner.[1] Specifically, the Business Motivation Model does all of the following:



Business Motivation Model

- identifies factors that motivate the establishing of business plans;
- identifies and defines the elements of business plans; and
- indicates how all these factors and elements inter-relate.

BMM Elements

"BMM captures business requirements across different dimensions to rigorously capture and justify why the business wants to do something, what it is aiming to achieve, how it plans to get there, and how it assesses the result."[2]

The main elements of BMM are:

- Ends: *What* (as oppose to *how*) the business wants to accomplish
- Means: *How* the business intends to accomplish its *ends*
- Directives: The *rules* and *policies* that *constrain* or *govern* the available means
- Influencers: Can cause changes that affect the organization in its employment of its Means or achievement of its Ends. Influencers are *neutral* by definition.
- Assessment: A *judgment* of an Influencer that affects the organization's ability to achieve its Ends or use its Means.

BMM History

Initially developed by the Business Rules Group (BRG) [3]

"Since 1997, the BRG has focused its energies on understanding business Rules from a business perspective. This in turn required a full, business-oriented understanding of how the elements of business plans should be organized. The BRG found that although many professionals have used planning methodologies over the years, no standard existed in that area, and many of the basic concepts were hazy and ad hoc."[4]

"In September 2005, the Object Management Group (OMG) voted to accept the Business Motivation Model as the subject of a Request for Comment (RFC). This meant that the OMG was willing to consider the Business Motivation Model as a specification to be adopted by the OMG, subject to comment from any interested parties. Adoption as an OMG specification carries the intention that the Business Motivation Model would, in time, be submitted to the International Organization for Standardization (ISO) as a standard."[5]

In August 2008 version 1.0 was released by OMG.

In May 2010 version 1.1 of BMM specification [6] was released and as of July 2011 it is the latest stable release.

Referenced Standards

Semantics of Business Vocabulary and Business Rules (SBVR)

Other related frameworks are POLDAT and the Zachman Framework.

Notes

[1] OMG BMM Specification v1.1, page 1 (http://www.omg.org/spec/BMM/1.1/PDF/)
[2] http://www.ibm.com/developerworks/rational/library/08/0401_amsden/
[3] http://www.businessrulesgroup.org/
[4] The Business Motivation Model *Business Governance in a Volatile World*, Release 1.3, September 2007, page ix (http://www.businessrulesgroup.org/second_paper/BRG-BMM.pdf)
[5] http://www.businessrulesgroup.org/bmm.shtml
[6] http://www.omg.org/spec/BMM/1.1/

External references

- The Business Motivation Model (http://www.businessrulesgroup.org/bmm.shtml) Business Governance in a Volatile World, Release 1.3, Business Rules Group (2007)
- From the Business Motivation Model to Service Oriented Architecture (http://www.jot.fm/issues/issue_2008_11/column6/index.html), by Birol Berkem, *Journal Of Object Technology* vol.7, no.8– (2008)
- The Business Motivation Model *Business Governance in a Volatile World*, Release 1.3, September 2007 (http://www.businessrulesgroup.org/second_paper/BRG-BMM.pdf)

Business Process Definition Metamodel

The **Business Process Definition Metamodel** (**BPDM**) is a standard definition of concepts used to express business process models (a metamodel), adopted by the OMG (Object Management Group). Metamodels define concepts, relationships, and semantics for exchange of user models between different modeling tools. The exchange format is defined by XSD (XML Schema) and XMI (XML for Metadata Interchange), a specification for transformation of OMG metamodels to XML. Pursuant to the OMG's policies, the metamodel is the result of an open process involving submissions by member organizations, following a Request for Proposal [1] (RFP) issued in 2003. BPDM was adopted in initial form in July 2007, and finalized in July 2008.

BPDM provides abstract concepts as the basis for consistent interpretation of specialized concepts used by business process modelers. For example, the ordering of many of the graphical elements in a BPMN (Business Process Modeling Notation) diagram is depicted by arrows between those elements, but the specific elements can have a variety of characteristics. For example, all BPMN events have some common characteristics, and a variety of specific events are designated by the type of circle and the icon in the circle. The abstract BPDM concepts ensure implementers of different modeling tools will associate the same characteristics and semantics with the modeling elements to ensure models are interpreted the same way when moved to a different tool. Users of the modeling tools do not need to be concerned with the abstractions-they only see the specialized elements.

BPDM extends business process modeling beyond the elements defined by BPMN and BPEL to include interactions between otherwise-independent business processes executing in different business units or enterprises (choreography). A choreography can be specified independently of its participants, and used as a requirement for the specification of the orchestration implemented by a participant. BPDM provides for the binding of orchestration to choreography to ensure compatibility. Many current business process models focus on specification of executable business processes that execute within an enterprise (orchestration).

The BPDM specification addresses the objectives of the OMG RFP [1] on which it is based:

- BPDM "will define a set of abstract business process definition elements for specification of executable business processes that execute within an enterprise, and may collaborate between otherwise-independent business processes executing in different business units or enterprises."
- common metamodel to unify the diverse business process definition notations that exist in the industry containing semantics compatible with leading business process modeling notations.
- A metamodel that complements existing UML metamodels so that business processes specifications can be part of complete system specifications to assure consistency and completenes.
- The ability to integrate process models for workflow management processes, automated business processes, and collaborations between business units.
- Support for the specification of web services choreography, describing the collaboration between participating entities and the ability to reconcile the choreography with supporting internal business processes.
- The ability to exchange business process specifications between modeling tools, and between tools and execution environments using XMI.

The RFP seeks to "improve communication between modelers, including between business and software modelers, provide flexible selection of tools and execution environments, and promote the development of more specialized tools for the analysis and design of processes."

For exchange of business process models, BPDM is an alternative to the existing process interchange format XPDL (XML Process Definition Language) from the WfMC (Workflow Management Coalition). The two specifications are similar in that they can be used by process design tools to exchange business process definitions. They are different in that BPDM provides a specification of semantics integrated in a metamodel, and it includes additional modeling capabilities such as choreography, discussed above. In addition, XPDL has many implementations, though only some support for XPDL 2.x, needed for interchanging BPMN. BPDM implementations are in preparation, including support for BPMN, and translation to XPDL.

External links

- BPDM Tutorial [2]
- Design Rationale [3] (see Section 4, also Sections 7.6 and 7.9).
- Other introductory presentations [4]
- Web pages showing metamodels [5] in UML notation
- Specification documents, in two parts:
  - Common Infrastructure [6] (see Section 4.4.1.1 for an overview of metamodeling).
  - Process Definition [7].

References

[1] http://www.omg.org/cgi-bin/doc?bei/03-01-06
[2] http://doc.omg.org/omg/08-06-32
[3] http://doc.omg.org/bmi/08-09-07
[4] http://www.conradbock.org/#BPDM
[5] ftp://ftp.omg.org/pub/docs/dtc/08-05-11/pages/188c21b53f42002f.htm
[6] http://doc.omg.org/dtc/08-05-07
[7] http://doc.omg.org/dtc/08-05-10

Knowledge Discovery Metamodel

Knowledge Discovery Metamodel (**KDM**) is publicly available specification from the Object Management Group (OMG). KDM is a common intermediate representation for existing software systems and their operating environments, that defines common metadata required for deep semantic integration of Application Lifecycle Management tools. KDM was designed as the OMG's foundation for software modernization, IT portfolio management and software assurance. KDM uses OMG's Meta-Object Facility to define an XMI interchange format between tools that work with existing software as well as an abstract interface (API) for the next-generation assurance and modernization tools. KDM standardizes existing approaches to knowledge discovery in software engineering artifacts, also known as software mining.

History of KDM

- In November 2003, the OMG's Architecture-Driven Modernization Task Force recommended, and the Platform Technical Committee issued, the Knowledge Discovery Metamodel (KDM) RFP. The objective of this RFP was to *provide a common repository structure to represent information about existing software assets and their operating environment.* The goal of KDM was defined as *exchanging information related to transformation of existing software assets*. The RFP stated that KDM shall provide *the ability to document existing systems, discover reusable components in existing software, support transformations to other languages and to MDA, or enable other potential transformations. The Knowledge Discovery Metamodel will also enable information about existing software artifacts to be exchanged among different tools. This will enable vendors that specialize on certain languages, platforms or types of transformations to deliver customer solutions in conjunction with other vendors.*
- The original KDM RFP is available to OMG members for download (document lt/03-11-04 [1]).
- Throughout 2004 and 2005 12 companies collaborated to prepare a joint response to the KDM RFP. More than 30 organizations from 5 countries have contributed to the development and review of the KDM specification.
- In May 2006, the Team's submission -- the Knowledge Discovery Metamodel (KDM) -- moved into the finalization stage of the OMG's standards adoption process. The OMG adopted Specification for KDM became publicly available (OMG document ptc/06-06-07).
- In March 2007 the KDM Finalization Task Force finished the finalization stage of the OMG's standards adoption process. The formal KDM specification KDM 1.0 is available from OMG (http://www.omg.org/spec/KDM/1.0/KDM 1.0]).
- KDM Analytics maintains Open portal for KDM news, reference and education materials and tools [2]

Overview of KDM

The goal of KDM is to ensure interoperability between tools for maintenance, evolution, assessment and modernization. KDM is defined as a metamodel that can be also viewed as an ontology for describing the key aspects of knowledge related to the various facets of enterprise software. KDM support means investment into the KDM ecosystem - a growing open-standard based cohesive community of tool vendors, service providers, and commercial components.

KDM represents entire enterprise software systems, not just code. KDM is a wide-spectrum entity-relationship representation for describing existing software. KDM represents structural and behavior elements of existing software systems. The key concept of KDM is a *container*: an entity that owns other entities. This allows KDM to represent existing systems at various degrees of granularity.

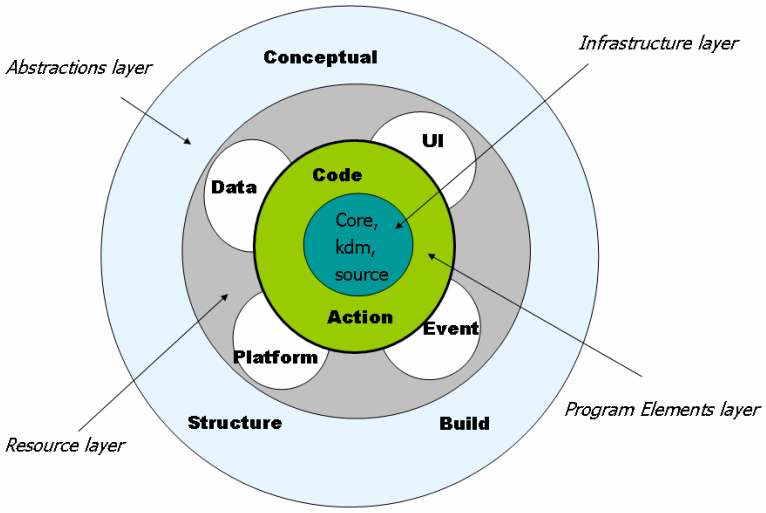
KDM defines precise semantic foundation for representing behavior, the so-called *micro-KDM*. It provides a high-fidelity intermediate representation which can be used, for example, for performing static analysis of existing software systems. micro-KDM is similar in purpose to a Virtual machine for KDM, although KDM is not an executable model, or a constraint model, but a representation of existing artifacts for analysis purposes.

KDM facilitates incremental analysis of existing software systems, where the initial KDM representation is analyzed and more pieces of knowledge are extracted and made explicit as KDM to KDM transformation performed entirely within the KDM technology space. The steps of the knowledge extraction process can be performed by tools, and may involve the analyst.

KDM is the uniform language- and platform- independent representation. Its extensibility mechanism allows addition of domain-, application- and implementation-specific knowledge.

Architecture of KDM

KDM packages are arranged into the following four layers:



Knowledge Discovery Metamodel consists of 12 packages arranged into 4 layers.

Infrastructure Layer

The KDM Infrastructure Layer consists of the **Core**, **kdm**, and **Source** packages which provide a small common core for all other packages, the inventory model of the artifacts of the existing system and full traceability between the meta-model elements as links back to the source code of the artifacts, as well as the uniform extensibility mechanism. The Core package determines several of patterns that are reused by other KDM packages. Although KDM is a meta-model that uses Meta-Object Facility, there is an alignment between the KDM Core and Resource Description Framework (RDF).

Program Elements Layer

The Program Elements Layer consists of the Code and Action packages.

- The **Code package** represents programming elements as determined by programming languages, for example data types, procedures, classes, methods, variables, etc. This package is similar in purpose to the Common Application Meta-model (CAM) from another OMG specification, called Enterprise Application Integration (EAI). KDM Code package provides greater level of detail and is seamlessly integrated with the architecturally significant views of the software system. Representation of datatypes in KDM is aligned with ISO standard ISO/IEC 11404 (see also General Purpose Datatypes).
- The **Action package** captures the low level behavior elements of applications, including detailed control- and data flow between statements. Code and Action package in combination provide a high-fidelity intermediate representation of each component of the enterprise software system

Resource Layer

The Resource Layer represents the operational environment of the existing software system. It is related to the area of Enterprise Application Integration (EAI).

- **Platform package** represents the operating environment of the software, related to the operating system, middleware, etc. including the control flows between components as they are determined by the runtime platform
- **UI package** represents the knowledge related to the user interfaces of the existing software system
- **Event package** represents the knowledge related to events and state-transition behavior of the existing software system
- **Data package** represents the artifacts related to persistent data, such as indexed files, relational databases, and other kinds of data storage. These assests are key to enterprise software as they represent the enterprise metadata. The KDM Data package is aligned with another OMG specification, called Common Warehouse Metamodel (CWM)

Abstractions Layer

The Abstraction Layer represents domain and application abstractions.

- **Conceptual package** represent business domain knowledge and business rules, insofar as this information can be mined from existing applications. These packages are aligned with another OMG specification, called Semantics of Business Vocabulary and Business Rules (SBVR)
- **Structure package** describes the meta-model elements for representing the logical organization of the software system into subsystems, layers and components
- **Build** package represents the engineering view of the software system

External links

- KDM 1.0 specification [3]
- Object Management Group (OMG) [4]
- Open KDM portal and tools from KDM Analytics [2]
- OMG Architecture-Driven Modernization Task Force [5]
- DSTC initial submission [6]
- SBVR link [7]
- Software Hypermodel Blueprint Portal for Open Source Software - TSRI's instantiations of ASTM+KDM+SMM [8]
- Open Source Components from MoDisco Eclipse project [9]

References

[1] http://www.omg.org/cgi-bin/doc?lt/03-11-04
[2] http://www.kdmanalytics.com/kdm
[3] http://www.omg.org/technology/documents/modernization_spec_catalog.htm
[4] http://www.omg.org
[5] http://adm.omg.org
[6] http://www.itee.uq.edu.au/~sse/seminars/2004/05jul04.html
[7] http://www.businessrulesgroup.org/sbvr.shtml
[8] http://www.softwarerevolution.com/portals
[9] http://wiki.eclipse.org/MoDisco/KDM

Resources, events, agents (accounting model)

Resources, events, agents (**REA**) is a model of how an accounting system can be re-engineered for the computer age. REA was originally proposed in 1982 by William E. McCarthy as a generalized accounting model, and contained the concepts of resources, events and agents (McCarthy 1982).

REA is a popular model in teaching accounting information systems (AIS). But it is rare in business practice—companies cannot easily dismantle their legacy systems to meet REA's radical demands. Workday, Inc., IBM Scalable Architecture for Financial Reporting, REATechnology, and ISO 15944-4 are exceptions.

The REA model gets rid of many accounting objects that are not necessary in the computer age. Most visible of these are debits and credits—double-entry bookkeeping disappears in an REA system. Many general ledger accounts also disappear, at least as persistent objects; e.g., accounts receivable or accounts payable. The computer can generate these accounts in real time using source document records.

REA treats the accounting system as a virtual representation of the actual business. In other words, it creates computer objects that directly represent real-world-business objects. In computer science terms, REA is an ontology. The real objects included in the REA model are:

- goods, services or money, i.e., resources
- business transactions or agreements that affect resources, i.e., events
- people or other human agencies (other companies, etc.), i.e., agents

These objects contrast with conventional accounting terms such as asset or liability, which are less directly tied to real-world objects. For example, a conventional accounting asset such as goodwill is not an REA resource.

There is a separate REA model for each business process in the company. A business process roughly corresponds to a functional department, or a function in Michael Porter's value chain. Examples of business processes would be sales, purchases, conversion or manufacturing, human resources, and financing.

At the heart of each REA model there is usually a pair of events, linked by an exchange relationship, typically referred to as the "duality" relation. One of these events usually represents a resource being given away or lost, while the other represents a resource being received or gained. For example, in the sales process, one event would be "sales"—where goods are given up—and the other would be "cash receipt", where cash is received. These two events are linked; a cash receipt occurs in exchange for a sale, and vice versa. The duality relationship can be more complex, e.g., in the manufacturing process, it would often involve more than two events (see Dunn et al. [2004] for examples).

REA systems have usually been modeled as relational databases with entity-relationship diagrams, though this is not compulsory. Workday uses an in-memory database. This is similar to Palantir Technologies' Object Model.

> "The breakthrough that Workday achieves is to move away from a fixed database structure. The SQL database in a traditional business management application defines the meaning of data into the table structure, and that is its achilles' heel. Workday's database tables reflect the needs of the object-oriented application architecture — there are just three tables, for 'instances', attributes' and 'references' — and the data and definitions stored in the table are instantiated only when the application runs. The definitions are therefore as easy to change as the data."

The philosophy of REA draws on the idea of reusable Design Patterns, though REA patterns are used to describe databases rather than object oriented programs, and are quite different from the 23 canonical patterns in the original designs pattern book by Gamma et al. Research in REA emphasizes patterns (e.g., Hruby et al. 2006). Here is an example of the basic REA pattern:

The pattern is extended to encompass commitments (promises to engage in transactions, e.g., a sales order), policies, and other constructs. Dunn et al. (2004) provide a good overview at an undergraduate level (for accounting majors),

while Hruby et al. (2006) is an advanced reference for computer scientists.

REA is a continuing influence on the electronic commerce standard ebXML, with W. McCarthy actively involved in the standards committee. The competing XBRL GL standard however is at odds with the REA concept, as it closely mimics double-entry book-keeping.

Further reading

- Hruby, P., Kiehn, J., Scheller, C. V. (2006). *Model-Driven Design Using Business Patterns*. Springer. ISBN 3-540-30154-2
- Dunn, C., Cherrington, J. O., Hollander, A. S. (2004) *Enterprise Information Systems: A Pattern-Based Approach*. McGraw-Hill/Irwin. ISBN 0-07-240429-9
- Hollander, A. S., Denna, E., Cherrington, J. O. (1999) *Accounting, Information Technology, and Business Solutions*. McGraw-Hill/Irwin. ISBN 0-256-21789-0
- Geerts, L. G., McCarthy, E. W. (2002, Vol.3) *An Ontological Analysis of the Primitives of the Extended-REA Enterprise Information Architecture*. The International Journal of Accounting Information Systems, pp. 1–16
- Geerts, L. G., McCarthy, E. W. (2000) *The Ontological Foundation of REA Enterprise Information Systems*. Working paper, Michigan State University
- McCarthy, E. W. (July 1982) *The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment*. The Accounting Review, pp. 554–78

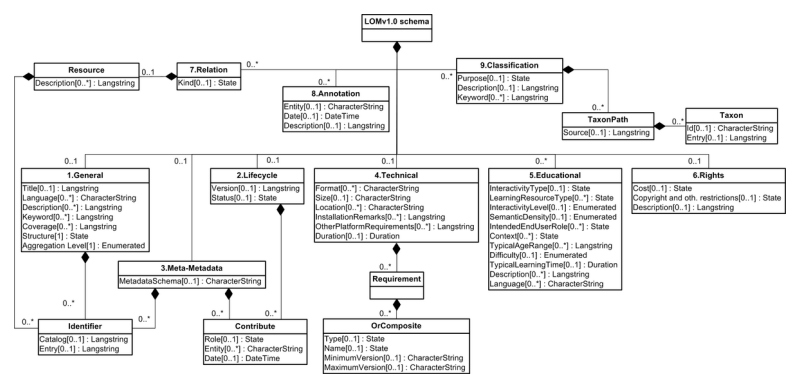
Footnotes

External links

- REA Enterprise Source Ontology (http://www.msu.edu/user/mccarth4/rea-ontology/).
- REA Technology (http://reatechnology.com/).

Learning object metadata

Learning Object Metadata is a data model, usually encoded in XML, used to describe a learning object and similar digital resources used to support learning. The purpose of learning object metadata is to support the reusability of learning objects, to aid discoverability, and to facilitate their interoperability, usually in the context of online learning management systems (LMS).



A schematic representation of the hierarchy of elements in the LOM data model

The IEEE 1484.12.1 – 2002 Standard for Learning Object Metadata is an internationally-recognised open standard (published by the Institute of Electrical and Electronics Engineers Standards Association, New York) for the description of "learning objects". Relevant attributes of learning objects to be described include: type of object; author; owner; terms of distribution; format; and pedagogical attributes, such as teaching or interaction style.

IEEE 1484.12.1 – 2002 Standard for Learning Object Metadata

In brief

The IEEE working group that developed the standard defined learning objects, *for the purposes of the standard,* as being "any entity, digital or non-digital, that may be used for learning, education or training." This definition has struck many commentators as being rather broad in its scope, but the definition was intended to provide a broad class of objects to which LOM metadata might usefully be associated rather than to give an instructional or pedagogic definition of a learning object. *IEEE 1484.12.1* is the first part of a multipart standard, and describes the LOM data model. The LOM data model specifies which aspects of a learning object should be described and what vocabularies may be used for these descriptions; it also defines how this data model can be amended by additions or constraints. Other parts of the standard are being drafted to define bindings of the LOM data model, i.e. define how LOM records should be represented in XML and RDF (*IEEE 1484.12.3* and *IEEE 1484.12.4* respectively). This article focuses on the LOM data model rather than issues relating to XML or other bindings.

IMS Global Learning Consortium is an international consortium that contributed to the drafting of the IEEE Learning Object Metadata (together with the ARIADNE Foundation [1]) and endorsed early drafts of the data model as part of the IMS Learning Resource Meta-data specification (IMS LRM, versions 1.0 – 1.2.2). Feedback and suggestions from the implementers of IMS LRM fed into the further development of the LOM, resulting in some drift between version 1.2 of the IMS LRM specification and what was finally published at the LOM standard. Version 1.3 of the IMS LRM specification realigns the IMS LRM data model with the IEEE LOM data model and specifies that the IEEE XML binding should be used. Thus, we can now use the term 'LOM' in referring to both the IEEE standard and version 1.3 of the IMS specification. The IMS LRM specification also provides an extensive *Best Practice and Implementation Guide*, and an *XSL transform* that can be used to migrate metadata instances from the older versions of the IMS LRM XML binding to the IEEE LOM XML binding.

Technical details

How the data model works

The LOM comprises a **hierarchy of elements**. At the first level, there are nine categories, each of which contains sub-elements; these sub-elements may be simple elements that hold data, or may themselves be aggregate elements, which contain further sub-elements. The semantics of an element are determined by its context: they are affected by the parent or container element in the hierarchy and by other elements in the same container. For example, the various *Description* elements (1.4, 5.10, 6.3, 7.2.2, 8.3 and 9.3) each derive their context from their parent element. In addition, description element 9.3 also takes its context from the value of element 9.1 *Purpose* in the same instance of *Classification*.

The data model specifies that some elements may be repeated either individually or as a group; for example, although the elements 9.3 (*Description*) and 9.1 (*Purpose*) can only occur once within each instance of the *Classification* container element, the *Classification* element may be repeated - thus allowing many descriptions for different purposes.

The data model also specifies the **value space** and **datatype** for each of the simple data elements. The value space defines the restrictions, if any, on the data that can be entered for that element. For many elements, the value space allows any string of Unicode character to be entered, whereas other elements entries must be drawn from a declared list (i.e. a controlled vocabulary) or must be in a specified format (e.g. date and language codes). Some element datatypes simply allow a string of characters to be entered, and others comprise two parts, as described below:

- **LangString** items contain Language and String parts, allowing the same information to be recorded in multiple languages
- **Vocabulary** items are constrained in such a way that their entries have to be chosen from a controlled list of terms - composed of Source-Value pairs - with the Source containing the name of the list of terms being used and the Value containing the chosen term
- **DateTime** and **Duration** items contain one part that allows the date or duration to be given in a machine readable format, and a second that allows a description of the date or duration (for example “mid summer, 1968”).

When implementing the LOM as a data or service provider, it is not necessary to support all the elements in the data model, nor need the LOM data model limit the information which may be provided. The creation of an application profile allows a community of users to specify which elements and vocabularies they will use. Elements from the LOM may be dropped and elements from other metadata schemas may be brought in; likewise, the vocabularies in the LOM may be supplemented with values appropriate to that community.

Requirements

The key requirements for exploiting the LOM as a data or service provider are to:

- Understand user/community needs and to express these as an application profile
- Have a strategy for creating high quality metadata
- Store this metadata in a form which can be exported as LOM records
- Agree a binding for LOM instances when they are exchanged
- Be able to exchange records with other systems either as single instances or *en masse*.

Related specifications

There are many metadata specifications; of particular interest is the Dublin Core Metadata Element Set (commonly known as Simple Dublin Core, standardised as *ANSI/NISO Z39.85 – 2001*), which provides a simpler, more loosely-defined set of elements with some overlap with the LOM, and which is useful for sharing metadata across a wide range of disparate services. The Dublin Core Metadata Initiative is also working on a set of terms which allow the Dublin Core Element Set to be used with greater semantic precision (Qualified Dublin Core). The Dublin Education Working Group aims to provide refinements of Dublin Core for the specific needs of the education community. Details of Dublin Core can be found at the Dublin Core website [2].

Many other education-related specifications allow for LO metadata to be embedded within XML instances, such as: describing the resources in an IMS Content Package or Resource List; describing the vocabularies and terms in an IMS VDEX (Vocabulary Definition and Exchange) file; and describing the question items in an IMS QTI (Question and Test Interoperability) file. Details of these can be found at the IMS Global website [3].

The IMS Vocabulary Definition and Exchange (VDEX) specification has a double relation with the LOM, since not only can the LOM provide metadata on the vocabularies in a VDEX instance, but VDEX can be used to describe the controlled vocabularies which are the value space for many LOM elements.

LOM records can be transported between systems using a variety of protocols, perhaps the most widely used being OAI-PMH.

Application profiles

UK LOM Core

For UK Further and Higher Education, the most relevant family of application profiles are those based around the *UK LOM Core*.[4] The UK LOM Core is currently a draft schema researched by a community of practitioners to identify common UK practice in learning object content, by comparing 12 metadata schemas. UK LOM is currently legacy work, it is not in active development.

CanCore

CanCore provides detailed guidance for the interpretation and implementation of each data element in the LOM standard. These guidelines (2004) constitute a 250-page document, and have been developed over three years under the leadership of Norm Friesen, and through consultation with experts across Canada and throughout the world. These guidelines are also available at no charge from the CanCore Website [5].

ANZ-LOM

ANZ-LOM [6] is a metadata profile developed for the education sector in Australia and New Zealand. The profile provides interpretations of metadata structures and illustrates how to apply controlled vocabularies, especially using the "classification" element. It is supported by detailed examples of learning resource metadata, including regional vocabularies. The ANZ-LOM profile was first published by The Le@rning Federation (TLF) in January, 2008.

Vetadata

The Australian Vocational Training and Education (VET) sector uses an application profile of the IEEE LOM called Vetadata [7]. The profile contains five mandatory elements, and makes use of a number of vocabularies specific to the Australian VET sector. This application profile was first published in 2005. The Vetadata and ANZ-LOM profiles are closely aligned.

NORLOM

NORLOM [8] is the Norwegian LOM profile. The profile is managed by NSSL [9] (The Norwegian Secretariat for Standardization of Learning Technologies)

ISRACore

ISRACORE [10] is the Israeli LOM profile. The Israel Internet Association (ISOC-IL [11]) and Inter University Computational Center (IUCC [12]) have teamed up to manage and establish an e-learning objects database.

SWE-LOM

SWE-LOM [13] is the Swedish LOM profile that is managed by IML [14] at Umeå University as a part of the work with the national standardization group TK450 at Swedish Standards Institute.

TWLOM

TWLOM [15] is the Taiwanese LOM profile that is managed by Industrial Development and Promotion of Archives and e-Learning Project [16]

LOM-FR

LOM-FR [17] is a metadata profile developed for the education sector in France. This application profile was first published in 2006.

NL LOM

NL LOM [18] is the Dutch metadata profile for educational resources in the Netherlands. This application profile was the result of merging the Dutch higher education LOM profile with the one used in primary and secondary Dutch education. The final version was released in 2011.

LOM-CH

LOM-CH [19] is a metadata profile developed for the education sector in Switzerland. It is current available in French and German. This application profile was published in 2013.

Others

Other application profiles are those developed by the Celebrate project[20] and the metadata profile that is part of the SCORM reference model.[21]

References

[1] http://www.ariadne-eu.org/
[2] http://dublincore.org/
[3] http://imsglobal.org/
[4] http://zope.cetis.ac.uk/profiles/uklomcore/
[5] http://cancore.tru.ca/en/
[6] http://www.ndlrn.edu.au/standards_for_digital_resources/metadata/metadata_profile.html
[7] http://e-standards.flexiblelearning.net.au/vetadata/index.htm
[8] http://www.itu.no/nssl/NORLOM
[9] http://www.nssl.no
[10] http://www.iucc.ac.il/lo/repository_en.htm
[11] http://www.isoc.org.il/
[12] http://www.iucc.ac.il/
[13] http://www.swe-lom.se
[14] http://www.iml.umu.se
[15] http://standard.teldap.tw/node/3
[16] http://idp.ncu.edu.tw/english/

[17] http://www.lom-fr.fr/
[18] http://wiki.surf.nl/display/nllom/Home
[19] http://ctie.educa.ch/fr/lom-ch-version-10
[20] European Schoolnet, CELEBRATE Application Profile (http://web.archive.org/web/20071225053548/http://www.eun.org/ww/en/pub/celebrate_help/application_profile.htm) (2003).
[21] ADL, SCORM (http://www.adlnet.gov/capabilities/scorm#tab-learn).

External links

E-mail Lists

- The CETIS Metadata and Digital Repository special interest group (http://metadata.cetis.ac.uk/) has two email lists:
- The SIG list (http://www.jiscmail.ac.uk/lists/CETIS-METADATA.html) will keep you up to date on all SIG activities and spec development related to metadata and digital repositories
- The LOM-cataloguing list (http://www.jiscmail.ac.uk/lists/LOM-CATALOGUING.html) is for queries relating to the creation and management of LOM descriptions.

Metadata Editors

- Reload (http://www.reload.ac.uk/) includes an open-source editor for IMS LRM.
- Curriculum Online Tagging Tool (http://www.curriculumonline.gov.uk/SupplierCentre/taggingtool.htm) is used for creating metadata for online resource made available to schools in England.
- LomPad (https://sourceforge.net/projects/lompad/) Lompad is a metatagging tool for learning object. It is bilingual(Fr,En) supports the IEEE-lom spec and SCORM, CANCORE and Normetic profiles.

Repositories/Catalogues

- MERLOT (http://www.merlot.org)(Multimedia Education Resource for Learning and Online Teaching) is a free and open peer-reviewed collection of learning and online teaching materials and faculty-developed services contributed and used by an international education community. It is an international consortium of over 20 institutions (and systems) of higher education, industry partners, professional organizations and individuals devoted to identifying, peer reviewing, organizing and making available existing WWW resources in a range of academic disciplines for use by higher education faculty and students.
- OER Commons (http://oercommons.org) is a freely accessible online library for teachers and others to search, discover, share, and evaluate Open Educational Resources (OER) and other freely available instructional materials. OER Commons is a project created by ISKME (http://www.iskme.org), an independent non-profit organization based in Half Moon Bay, California.
- OpenScout (http://learn.openscout.net) is a hub for open educational resources for business and management education and training. It has international materials from around the globe and provides tools for adapting resources to the context.
- Learning Resource Exchange (http://lreforschools.eun.org) from European Schoolnet offers more than 200,000 multilingual Open Educational Resources for the K-12 sector from more than 50 content providers.
- Desire2Learn Learning Repository (http://www.desire2learn.com/learningrepository/) is a repository system with rich support for IEEE LOM, Dublin Core, CanCore, and other customisable and templatable profiles. It also supports harvesting of metadata from other repository/catalogue systems. It is used by hundreds of educational and commercial organisations.
- Intrallect intraLibrary (http://www.intrallect.com/products) is a commercial repository system, with rich support for the IEEE LOM, including customisable application profiles
- The Xtensis Open Architecture is a foundation for building customised learning object repository and learning content management systems.

- HarvestRoad Hive (http://www2.harvestroad.com.au/cgi-bin/hr/loadframes_tmp.cgi?lcms) is a commercial federated digital repository system (based out of Australia), i.e. it is designed share content across multiple repositories.
- The Jorum project (http://www.jorum.ac.uk) uses Intrallect intraLibrary (http://www.intrallect.com/products) for its ongoing service.
- The National Learning Network, using Xtensis, is a UK repository of over 2000 learning objects, commissioned by Becta and the LSC.
- Kursnavet (http://kursnavet.cfl.se) (Swedish), is a repository containing around 11 000 (2007) unique learning objects, free of use non-commersial. From the Swedish Agency for Flexible Learning, CFL (http://www.cfl.se).
- LearnAlberta.ca is a Canadian learning object repository maintained by Alberta Education, the Province of Alberta's education Ministry.
- The Learning Object Repository Network (http://lorn.flexiblelearning.net.au) (LORN) is Australia's national learning object repository network for the vocational education and training (VET) sector. It contains nearly 3000 learning objects.
- Agrega is Spain's national learning object repository network. It contains nearly 100.000 learning objects.
- Spindeln (http://itforpedagoger.skolverket.se/hitta_material/) (the Spider), is a Swedish learning object repository brokerage service that connects a number of Swedish LORs. Spindeln is provided by the Nat. Agency for Education.
- North Carolina Learning Object Repository - The k-20 learning object repository used by the State of North Carolina and managed by the North Carolina Community College System.
- EQUELLA (www.equella.com (http://www.equella.com)) is a digital repository implemented by the editorial giant Pearson that integrates with various learning management systems, supports IMS packages and all metadata schemas including IEEE LOM.

Resources on the Internet

- (http://cancore.athabascau.ca/en/) is a thorough element-by-element guide to implementing the IEEE LOM.
- IMS Global Learning Consortium Learning resource meta-data specification (http://www.imsglobal.org/metadata/).
- XML Binding Specification (http://ltsc.ieee.org/wg12/files/IEEE_1484_12_03_d8_submitted.pdf).
- Final draft for LOM standard (15 July 2002 / PDF) (http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf).
- LOM Java API (http://sourceforge.net/projects/lom-j/).
- A mapping between the IEEE LOM and IMS Learning Resource Metadata (http://www.intrallect.com/support/metadata/ims2lom_metadata_mapping.htm)
- Metadata? Thesauri? Taxonomies? Topic Maps! Making sense of it all (http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html)

Learning object

A **learning object** is "a collection of content items, practice items, and assessment items that are combined based on a single learning objective". The term is credited to Wayne Hodgins when he created a working group in 1994 bearing the name though the concept was first described by Gerard in 1967. Learning objects go by many names, including content objects, chunks, educational objects, information objects, intelligent objects, knowledge bits, knowledge objects, learning components, media objects, reusable curriculum components, nuggets, reusable information objects, reusable learning objects, testable reusable units of cognition, training components, and units of learning.

Learning objects offer a new conceptualization of the learning process: rather than the traditional "several hour chunk", they provide smaller, self-contained, re-usable units of learning.

They will typically have a number of different components, which range from descriptive data to information about rights and educational level. At their core, however, will be instructional content, practice, and assessment. A key issue is the use of metadata.

Learning object design raises issues of portability, and of the object's relation to a broader learning management system.

Definitions

The Institute of Electrical and Electronics Engineers (IEEE) defines a learning object as "any entity, digital or non-digital, that may be used for learning, education or training".

Chiappe defined Learning Objects as: "A digital self-contained and reusable entity, with a clear educational purpose, with at least three internal and editable components: content, learning activities and elements of context. The learning objects must have an external structure of information to facilitate their identification, storage and retrieval: the metadata."

The following definitions focus on the relation between learning object and digital media. RLO-CETL, a British inter-university Learning Objects Center, defines "reusable learning objects" as "web-based interactive chunks of e-learning designed to explain a stand-alone learning objective". Daniel Rehak and Robin Mason define it as "a digitized entity which can be used, reused or referenced during technology supported learning".

Adapting a definition from the Wisconsin Online Resource Center, Robert J. Beck suggests that learning objects have the following key characteristics:

- Learning objects are a new way of thinking about learning content. Traditionally, content comes in a several hour chunk. Learning objects are much smaller units of learning, typically ranging from 2 minutes to 15 minutes.
- Are self-contained – each learning object can be taken independently
- Are reusable – a single learning object may be used in multiple contexts for multiple purposes
- Can be aggregated – learning objects can be grouped into larger collections of content, including traditional course structures
- Are tagged with metadata – every learning object has descriptive information allowing it to be easily found by a search

Components

The following is a list of some of the types of information that may be included in a learning object and its metadata:

- General Course Descriptive Data, including: course identifiers, language of content (English, Spanish, etc.), subject area (Maths, Reading, etc.), descriptive text, descriptive keywords
- Life Cycle, including: version, status
- Instructional Content, including: text, web pages, images, sound, video
- Glossary of Terms, including: terms, definition, acronyms
- Quizzes and Assessments, including: questions, answers
- Rights, including: cost, copyrights, restrictions on Use
- Relationships to Other Courses, including prerequisite courses
- Educational Level, including: grade level, age range, typical learning time, and difficulty. [IEEE 1484.12.1:2002]
- Typology as defined by Churchill (2007): presentation, practice, simulation, conceptual models, information, and contextual representation [1]

Metadata

One of the key issues in using learning objects is their identification by search engines or content management systems.[*citation needed*] This is usually facilitated by assigning descriptive learning object metadata. Just as a book in a library has a record in the card catalog, learning objects must also be tagged with metadata. The most important pieces of metadata typically associated with a learning object include:

1. **objective:** The educational objective the learning object is instructing
2. **prerequisites:** The list of skills (typically represented as objectives) which the learner must know before viewing the learning object
3. **topic:** Typically represented in a taxonomy, the topic the learning object is instructing
4. **interactivity:** The Interaction Model of the learning object.
5. **technology requirements:** The required system requirements to view the learning object.

Mutability

A mutated learning object is, according to Michael Shaw, a learning object that has been "re-purposed and/or re-engineered, changed or simply re-used in some way different from its original intended design". Shaw also introduces the term "contextual learning object", to describe a learning object that has been "designed to have specific meaning and purpose to an intended learner".

Portability

Before any institution invests a great deal of time and energy into building high-quality e-learning content (which can cost over $10,000 per classroom hour),[2] it needs to consider how this content can be easily loaded into a Learning Management System. It is possible for example, to package learning objects with SCORM specification and load it in Moodle Learning Management System or Desire2Learn Learning Environment.

If all of the properties of a course can be precisely defined in a common format, the content can be serialized into a standard format such as XML and loaded into other systems. When it is considered that some e-learning courses need to include video, mathematical equations using MathML, chemistry equations using CML and other complex structures, the issues become very complex, especially if the systems needs to understand and validate each structure and then place it correctly in a database.[*citation needed*]

Criticism

In 2001, David Wiley criticized learning object theory in his paper, The Reusability Paradox [3] which is summarized by D'Arcy Norman [4] as, *If a learning object is useful in a particular context, by definition it is not reusable in a different context. If a learning object is reusable in many contexts, it isn't particularly useful in any.* In Three Objections to Learning Objects and E-learning Standards [5], Norm Friesen, Canada Research Chair in E-Learning Practices at Thompson Rivers University, points out that the word *neutrality* in itself implies *a state or position that is antithetical ... to pedagogy and teaching.*

References

[1] Churchill, D. (2007). Towards a useful classification of learning objects. *Educational Technology Research & Development, 55(5)*, 479-497.
[2] Rumble, Greville. 2001. The Cost and Costing of Networked Learning. Journal of Asynchronous Learning Networks, Volume 5, Issue 2.
[3] http://web.archive.org/web/20041019162710/http:/rclt.usu.edu/whitepapers/paradox.html
[4] http://www.darcynorman.net/2003/08/21/addressing-the-reusability-paradox/
[5] http://www.mendeley.com/research/three-objections-to-learning-objects/

Further reading

- Beck, Robert J. (2009), *"What Are Learning Objects?", Learning Objects, Center for International Education, University of Wisconsin-Milwaukee,* (http://www4.uwm.edu/cie/learning_objects.cfm?gid=56), retrieved 2009-10-23.
- Learning Technology Standards Committee (2002), *Draft Standard for Learning Object Metadata. IEEE Standard 1484.12.1* (http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf) (PDF), New York: Institute of Electrical and Electronics Engineers, retrieved 2008-04-29.
- Rehak, Daniel R.; Mason, Robin (2003), "Engaging with the Learning Object Economy", in Littlejohn, Allison, *Reusing Online Resources: A Sustainable Approach to E-Learning*, London: Kogan Page, pp. 22–30, ISBN 978-0-7494-3949-1.
- Shaw, Michael (October 2003), "(Contextual and Mutated) Learning Objects in the Context of Design, Learning and (Re)Use" (http://www.shawmultimedia.com/edtech_oct_03.html), *Teaching and Learning with Technology*, retrieved 2008-04-29
- Chiappe, Andres.; Segovia, Yasbley; Rincon, Yadira (2007), "Toward an instructional design model based on learning objects" (http://www.springerlink.com/content/u84w63873vq77h2h/?p=41be7fbeef9648ee9b554f1835112005&pi=6), in Boston, Springer, *Educational Technology Research and Development*, Boston: Springer, pp. 671–681, ISBN Special:BookSources/1042-1629(Print)1556-6501(Online)Spanish Draft available in "Blog de Andrés Chiappe - Objetos de Aprendizaje"(http://andreschiappe.blogspot.com/2007/09/que-es-un-objeto-de-aprendizaje-what-is.html)|1042-1629(Print)1556-65 Draft available in "Blog de Andrés Chiappe - Objetos de Aprendizaje"(http://andreschiappe.blogspot.com/2007/09/que-es-un-objeto-de-aprendizaje-what-is.html) Check `|isbn=` value (help), retrieved 2008-08-21.
- Northrup, Pamela (2007), *Learning Objects for Instruction: Design and Evaluation* (Book), USA: Information Science Publishing.
- Hunt, John P.; Bernard, Robert (2005), *"An XML-based information architecture for learning content", IBM developerWorks,* (http://www.ibm.com/developerworks/xml/library/x-dita9a), retrieved 2005-08-05.
- Churchill, D. (2007). Towards a useful classification of learning objects. *Educational Technology Research & Development, 55(5)*, 479-497.

Innayah: Creating An Audio Script with Learning Object, unpublished, 2013.

External links

- The Learning Objects (http://www4.uwm.edu/cie/learning_objects.cfm?gid=55) at Milwaukee's Center for International Education.

OGML

Ontology Grounded Metalanguage (OGML) is a metalanguage like MOF. The goal of OGML is to tackle the difficulties of MOF:[1] linear modeling architecture, ambiguous constructs and incomprehensible/unclear architecture. OGML provides a nested modeling architecture with three fixed layers (models, languages and metalanguage), therefore it is clear how the different models conform to each other and can be handled. Constructs in OGML are chosen from the science of Ontology, making the distinction between properties / objects and classes / objects very clear. This commitment makes explicit certain oddities of the definition of, for example, relations. Furthermore, OGML provides an explicit notion of instantiation:[2] model elements encode their types and languages define the semantics of instantiation. This extra information is needed in the relative modeling architecture to distinguish between structural and conceptual views on models, for example: we may want to view a UML model as an instance of the Object language and an instance of the Class model (Clabject). By providing this dual view on metamodel layer and on the language layer, OGML provides a very precise modeling architecture and an expressive way to deal with models.

References

[1] Atkinson, C.; Kuhne, T. (2003). Model-driven development: a metamodeling foundation"
[2] Laarman, A. (2009). An Ontology-Based Metalanguage with Explicit Instantiation"

External links

- Official website (http://fmt.cs.utwente.nl/~laarman/ogml)

Article Sources and Contributors

Data management *Source*: http://en.wikipedia.org/w/index.php?oldid=599124498 *Contributors*: A5b, Abaqus, Aksi great, Altes, AnmaFinotera, Antandrus, Anthony, Ash Crow, Auniqueuserid, Bobo192, Bovineone, Carly805, Cascadeuse, ChemGardener, CliffC, Craig Stuntz, Cwbash, Dawnseeker2000, Duke Ganote, Edcolins, Elockid, Happysailor, Hegde Pooja, IvanLanin, J.delanoy, Jackollie, Jaeger5432, JamesBWatson, Jfsarayvi, Jschmidt163, KeyStroke, Kku, Know I AM, Kuru, L mahonsmith, Labscientist, Lilac Soul, Mandarax, Markcowan, Martarius, Matthew Stannard, Mdd, Michael Hardy, MrOllie, NameIsRon, Nicolas1981, Nighthawk2050, Nurg, Nww mag, Oli Filth, Opagecrtr, Qwfp, RHaworth, Rhobite, RoyBoy, Sandymok, Seb az86556, Siroxo, SoftwareSalesRep, Stormie, Syaskin, Tchan012, Tetracube, Tide rolls, Tontine991, Troels Arvin, Vcmohanonline, Wilhkar, Wimbojambo, Winton.Smith, Xpclient, Ykhwong, Ze miguel, Zhenqinli, ZimZalaBim, 140 anonymous edits

Database *Source*: http://en.wikipedia.org/w/index.php?oldid=599045026 *Contributors*: *drew, 05winsjp, 069952497a, 10285658sdsaa, 10metreh, 110808028 amol, 16@r, 206.31.111.xxx, 25or6to4, 28421u2232nfenfcenc, 28nebraska, 2D, 4twenty42o, 65.10.163.xxx, APH, Aaron Brenneman, Abhikumar1995, AddWittyNameHere, Addihockey10, Aditya gopal3, Admfirepanther, Adrian J. Hunter, Aepanico, Afluegel, Afree10, Ahodgkinson, Ahoerstemeier, Ahy1, Aitias, Aj.robin, Akamad, Al Wiseman, Alain Amiouni, Alansohn, Alasdair, Ale jrb, Allan McInnes, Allecher, Alpha Quadrant (alt), Alphax, Alzpp, Amaraiel, Amaury, Amd628, Anders Torlind, Andonic, Andre Engels, Andrewferrier, AndriuZ, Angela, Anikingos, AnjaliSinha, AnmaFinotera, Ann Stouter, AnonUser, Anonymous Dissident, Antandrus, Antrax, Apparition11, Arbitrarily0, Arcann, Arctic Kangaroo, Argon233, Arjun01, Armen1304, ArnoLagrange, Arthena, Arthur Rubin, Arved, ArwinJ, AsceticRose, Asyndeton, AtheWeatherman, Atkinsdc, Autumn Wind, AutumnSnow, Avenged Eightfold, AwamerT, Ayecee, AzaToth, Baa, Babbling.Brook, Barneca, Bbatsell, Bbb23, Bblank, Bcartolo, Bcontins, Beeblebrox, Beetstra, Beland, Ben Ben, Ben-Zin, Benni39, BentlijDB, Bentogoa, Bernd in Japan, Beta M, Betterusername, Bharath357, Bjcubsfan, Bkhouser, Blackguard SF, Blanchardb, BlindEagle, Blood Red Sandman, BluCreator, Bluemask, Bluerocket, BobStepno, Bobblewik, Bogdangiusca, Bogey97, Boing! said Zebedee, Boli1107, Bongwarrior, Bowlderizer, Branzman, Brick Thrower, BrokenSphere, BryanG, Btilm, Bubba hotep, Bunnyhop11, Burner0718, Buzzimu, Bwhynot14, C12H22O11, CIreland, COMPFUNK2, CableCat, Calabe1992, Call me Bubba, Callanecc, Calliopejen1, Caltas, Calutuigor, Cambalachero, Cambapp, Cammo33, Camw, Can't sleep, clown will eat me, CanadianLinuxUser, CanisRufus, Canterbury Tail, Cantras, Capricorn42, Captain-n00dle, Captain-tucker, Carbonite, CardinalDan, Carliitaeliza, Caster23, CasualVisitor, Cavanagh, Cenarium, CesarB, Cevalsi, Ceyjan, Chaojoker, Cheolsoo, Chester Markel, Childzy, Chirpy, Chocolateboy, ChorizoLasagna, Chrax, Chris 73, Chris G, ChrisGualtieri, Chrislk02, Chrism, Christophe.billiottet, Chriswiki, Chtuw, Chuckhoffmann, Chuunen Baka, Clarince63, Clark89, Click23, Closedmouth, Colindolly, Colonies Chris, Cometstyles, Commander Keane, Compfreak7, Comps, Constructive, Conversion script, Corvoe, Courcelles, Cpereyra, Cpl Syx, Cpuwhiz11, Craftyminion, Craig Stuntz, Crashdoom, Credema, Crucis, Cryptic, Culverin, Cyan, Cybercobra, Cyberjoac, CynicalMe, D. Recorder, DARTH SIDIOUS 2, DEddy, DFS454, DJ Clayworth, DVD R W, DVdm, DamnRandall, Dan100, Dancayta, Dancter, Danhash, Daniel.Cardenas, DanielCD, Danieljamesscott, Danim, Dart88, Darth Mike, Darth Panda, Darthvader023, Davewild, David Fuchs, David0811, Davidcopperfield123, Dbates1999, Dbfirs, DePiep, Dead3y3, DeadEyeArrow, DeadlyAssassin, Deathlasersonline, Decrease789, DeirdreGerhardt, Denisarona, DerBorg, DerHexer, Deville, Dgw, Diamondland, DigitalEnthusiast, Discospinster, Djordjes, Djsasso, Dkastner, Dkwebsub, Doc glasgow, Doddsy1993, Dogposter, Donama, Doniago, Donner60, DougBarry, Dougofborg, Doulos Christos, DragonLord, Dreadstar, Dreamyshade, Drivenapart, Drumroll99, Dsimic, Duyanfang, Dwolt, Dysepsion, E23, Eagleal, Earlypsychosis, EarthPerson, EastTN, Echartre, Eddiecarter1, Eddiejcarter, Edgar181, Edgarde, Edivorce, Edward, Eeekster, Ejrrjs, ElKevbo, ElijahLloyd97, Elwikipedista, Epbr123, Epicgenius, Era7bd, Eric Bekins, Eric Burnett, Eric3K, Ericlaw02, Erikrj, Escape Orbit, Etxrge, EugeneZelenko, EvergreenFir, Everyking, Evildeathmath, Excirial, Exor674, Explicit, Eyesnore, Ezeu, FFGeyer, Fabartus, Fang Aili, FatalError, Favonian, Feedmecereal, FetchcommsAWB, Feydey, Fieldday-sunday, Filx, Finlay McWalter, Flewis, Flubeca, Fluffernutter, Flyer22, FlyingToaster, Fooker69, Fortdj33, Foxfax555, Fraggle81, Frankman, Franky21, Franl, Fratrep, Freebiekr, Frsparrow, Frze, Fubar Obfusco, Furrykef, Fuzzie, Fæ, G12kid, GDonato, GHe, GLaDOS, Gadfium, Gail, Gajurahman, Galzigler, Garyzx, Gburd, Giftlite, Gilliam, Ginsengbomb, Ginsuloft, Girl2k, Gishac, Glacialfox, Glane23, GlenPeterson, GnuDoyng, Gogo Dodo, GoingBatty, Gonfus, Gozzy345, Graeme Bartlett, GraemeL, Graham87, GrayFullbuster, GregWPhoto, Gregfitzy, GregorB, Grim23, Grsmca, Grstain, Gsallis, Gscshoyru, Gwizard, Gzkn, Haakon, Hadal, HaeB, Hamtechperson, Hankhuck, HappyInGeneral, Harej, Hasek is the best, HeliXx, Helixblue, Helloher, HexaChord, Heymid, Heysim0n, Hgilbert, Hotstaff, Hshoemark, Hugsandy, Huntthetroll, Hurricane111, HybridBoy, Hydrogen Iodide, IComputerSaysNo, IElonex!, Iced Kola, Igoldste, IkamusumeFan, Imfargo, Imnotminkus, Imran, Informatwr, Insineratehymn, Inspector 34, Intgr, IrfanSha, IronGargoyle, Ironman5247, Isfisk, Itafran2010, ItsZippy, Ixfd64, J.delanoy, JCLately, JForget, JJdaboss, JLaTondre, JMRyan, Ja 62, JaGa, Jab843, Jabby11, Jack Greenmaven, Jackacon, Jackfork, JamesBWatson, JamesMoose, Jan1nad, Jandalhandler, Jarble, Jasimab, Jasper Deng, Jauerback, Javert, Jaxl, Jay, Jb-adder, Jclemens, Jdlambert, JeffTan, JeffreyYasskin, Jennavecia, JephapE, Jerome Charles Potts, Jesant13, JewishMonser69, Jk2q3jrklse, Jmanigold, Jni, JoanneB, Joel7687, Joffeloff, John Vandenberg, John of Reading, Johnuniq, Jojalozzo, Jonathan Webley, JonathanFreed, Jondel, Jonearles, Jonwynne, Joshnpowell, Joshwa1234567890, Journalist, Jschnur, Jstaniek, JunWan, Jvhertum, Jwoodger, Jwy, KILLERKEA23, Kanonkas, Karlhahn, Karmafist, Katalaveno, Keenan Pepper, Keilana, Kekekecakes, Kellyk99, Kenny sh, Kevins, KeyStroke, Khazar2, Khoikhoi, Kiand, Kimberly ayoma, Kimera Kat, King of Hearts, Kingius, Kingpin13, KingsleyIdehen, Kivar2, Kkailas, Knbanker, Koavf, Kocio, Komal.Ar, KoshVorlon, KotetsuKat, Kozmando, Kraftlos, Krashlandon, Kslays, Kukini, Kunaldeo, Kungfuadam, Kuru, Kushal one, Kvasilev, Kwiki, KyraVixen, Kzzl, L Kensington, LC, Lamp90, LaosLos, Larsinio, Latka, Lbausalop, Leaderofearth, Leandrod, LeaveSleaves, LeeHam2007, Leonnicholls07, LessHeard vanU, Levin, Levin Carsten, Lexo, Lflores92201, Lfstevens, Lguzenda, Lights, LindaEllen, Lingliu07, Lingwitt, Linkspamremover, LittleOldMe, LittleWink, Llyntegid, Lod, Logan, Lotje, Lovefamosos, Lovelac7, Lowellian, Lradrama, Lsschwar, LuK3, Lucyin, Lugia2453, Luizfsc, Luna Santin, M.badnjki, M4gnum0n, MBisanz, MECiAf., MER-C, MJunkCat, Machdohvah, Madhava 1947, Madth3, Magioladitis, MainFrame, Majorly, Makecat, Malvikiran, Mandarax, Manikandan 2030, Mannafredo, Marasmusine, Mark Arsten, Mark Renier, MarkSutton, MartinSpamer, Materialscientist, Mathewforyou, Mato, Matthewrbowker, Matticus78, Mattisse, Maty18, Maury Markowitz, Max Naylor, Maxferrario, Maxime.Debosschere, Maxmarengo, Mayur, Mazca, Mboverload, McGeddon, Mdd, Meaghan, Mean as custard, Mediran, Megatronium, MelbourneStar, Melody Lavender, Melucky2getu, Mentifisto, Menublogger, Mercy11, Methnor, Mhkay, Michael Hardy, Michael Slone, Microchip08, Mike Dillon, Mike Rosoft, Mike Schwartz, Mike99999, MikeSy, Mikeblas, Mikey180791, MilerWhite, Millermk, Milo99, Mindmatrix, Minimac, Minna Sora no Shita, Mkeranat, Moazzam chand, Mojo Hand, Moreschi, Morwen, MrNoblet, Mrozlog, Mrt3366, Mspraveen, Mugaliens, Mukherjeeassociates, Mulad, Mumonkan, Mushroom, Mwaci11, Mxn, N1RK4UDSK714, N25696, NAHID, NSR, Nafclark, Namlemez, Nanshu, NathanBeach, NawlinWiki, NetManage, Netizen, NewEnglandYankee, Ngpd, Nick, Nicoosuna, Niteowlneils, Nk, Noah Salzman, Noctibus, Noldoaran, Nomonomnom, Northamerica1000, Northernhenge, Nsaa, Nurg, Ocaasi, Oda Mari, Odavy, Oddbodz, Ohka-, Oho1, Oli Filth, Olinga, OllieFury, OnePt618, Optakeover, OrgasGirl, Oroso, OverlordQ, P2Peter, PJM, PaePae, Pak21, PappaAvMin, Parzi, PatrikR, Paul August, Paul Drye, Paul E Ester, Paul Foxworthy, Paulinho28, Pcb21, Pdcook, Peashy, Pee Tern, PeeAeMKay, Pengo, PeregrineAY, Peruvianllama, Pete1248, Peter Flass, Peter Karlsen, Peter.C, Pgk, Phantomsteve, Pharaoh of the Wizards, Phearlez, PhilKnight, Philip Trueman, Philippe, Phinicle, Phoenix-wiki, Piano non troppo, Pillefj, Pingveno, Pinkadelica, Pjoef, Plrk, Pnm, Poeloq, Pol098, Poor Yorick, Poterxu, Praba tuty, Prabash.A, Prari, Prashanthns, Pratyya Ghosh, PrePress, Preet91119, Proofreader77, Prunesqualer, Psaajid, Psb777, Puchiko, Pvjohnson, Pyfan, Quadell, Qwertykris, Qwyrxian, Qxz, R'n'B, R3miixasim, RIH-V, RadioFan, RadioKirk, Rafaelschp, Railgun, Rajat Kant Singh, Rakeki, Raspalchima, Ravinjit, Ray Lightyear, RayGates, RayMetz100, RazorXX8, Rdsmith4, Reaper Eternal, Reatlas, Refactored, Regancy42, Reidh21234, RenamedUser01302013, Rettetast, RexNL, Rhobite, Rich Farmbrough, Ricky81682, Ringbang, Rishu arora11, Riverraisin, Rj Haseeb, Rjwilmsi, Robert Merkel, Robert Skyhawk, Robocoder, Robth, Rocketrod1960, Rockonomics, Rohitj.iitk, Romanm, Rotanagol, Rothwellisretarded, Roux, Rursus, Ruud Koot, Ryager, Ryanslater, Ryanslater2, Ryulong, S.K., SAE1962, SDSWIKI, SFK2, SJP, SWAdair, Sae1962, Saiken79, Salvio giuliano, Sam Barsoom, Sam Korn, SamJohnston, Samir, Samster0708, Sander123, Sandman, Sango123, Sarchand, SarekOfVulcan, Satellizer, SatuSuro, Saturdayswiki, Savh, ScMeGr, Sceptre, Seanust 1, Seaphoto, SebastianHelm, Serketan, Several Pending, Sewebster, Shadowjams, Shadowseas, Sheeana, Shipmaster, Shirulashem, Siebren, Silly rabbit, Simeon, Simetrical, SimonMorgan, Sintaku, Sir Nicholas de Mimsy-Porpington, Sissi's bd, Siteobserver, Sjakkalle, Sjc, Skamecrazy123, Skybrian, Slakr, Sleske, SnoFox, Somchai1029, Sonett72, Sonia, Soosed, Sophus Bie, Soumark, SpK, Spartaz, Spazure, Spdegabrielle, SpikeTorontoRCP, SpuriousQ, Squids and Chips, Srdju001, Srikeit, Ssd, StaticVision, Stdazi, Stephen Gilbert, Stephenb, Stevertigo, Stifle, Stirling Newberry, Storm Rider, Strike Eagle, Strongsauce, Stuhacking, Sucker666, Sudarevic, Suffusion of Yellow, Sunrocket89, SuperHamster, Supertouch, Supreme Deliciousness, Supten, SwisterTwister, SymlynX, Sythy2, Tabletop, Tablizer, TakuyaMurata, TalkyLemon, Tasc, Tazmaniacs, Technopat, Tgeairn, Th1rt3en, Thatperson, The Anome, The Thing That Should Not Be, The Wiki Octopus, The wub, TheGrimReaper NS, TheNewPhobia, Thedjatclubrock, Thehulkmonster, Theimmaculatechemist, Theodolite, Theory of deadman, Thingg, Think outside the box, Thinktdub, Thomasryno, ThumbFinger, Thumperward, Tictacsir, Tide rolls, Tim Q. Wells, TimBentley, Timbits82, Title302, TittoAssini, Tobias Bergemann, Tolly4bolly, Tomatronster, Tonydent, Toquinha, Tpbradbury, Treekids, TrentonLipscomb, Trevor MacInnis, Triwbe, Troels Arvin, Trusilver, Tualha, Tudorol, Tuhl, Turlo Lomon, Turnstep, Twebby, Twelvethirteen, TwistOfCain, TwoTwoHello, Twsx, TyA, Tyler, UberScienceNerd, Ubiq, Ubiquity, Ugebgroup8, Ulric1313, Ultraexactzz, Uncle Dick, Unyoyega, VNeumann, Vary, Velella, Verajohne, Versus22, Veryprettyfish, Vespristiano, Victor falk, Vikreykja, Vincent Liu, Vipinhari, Vishnava, Visor, Vivacewwxu, VoxLuna, Vrenator, W mccall, W163, WOSlinker, Waggers, Waveguy, Wavelength, Weetoddid, Welsh, Werdna, Widefox, Widr, Wifione, Wik, Wiki alf, Wiki tiki tr, Wikidrone, Wikipelli, WikiuserNI, Willking1979, Wimt, Windsok, Winterst, Wipe, Wmahan, Woohookitty, Woseph, Writeread82, Wulfila, Wwmbes, Wya 7890, Xfact, Xhelllox, Xin0427, Yintan, Yossman007, ZenerV, Zhaofeng Li, Zhenqinli, Zipircik, Zippanova, ZooPro, Zro, Zundark, Zzuuzz, Σ, М И Ф, كاشف عقيل, 雷大伟, 3213 anonymous edits

Database system *Source*: http://en.wikipedia.org/w/index.php?oldid=544587256 *Contributors*: *drew, 05winsjp, 069952497a, 10285658sdsaa, 10metreh, 110808028 amol, 16@r, 206.31.111.xxx, 25or6to4, 28421u2232nfenfcenc, 28nebraska, 2D, 4twenty42o, 65.10.163.xxx, APH, Aaron Brenneman, Abhikumar1995, AddWittyNameHere, Addihockey10, Aditya gopal3, Admfirepanther, Adrian J. Hunter, Aepanico, Afluegel, Afree10, Ahodgkinson, Ahoerstemeier, Ahy1, Aitias, Aj.robin, Akamad, Al Wiseman, Alain Amiouni, Alansohn, Alasdair, Ale jrb, Allan McInnes, Allecher, Alpha Quadrant (alt), Alphax, Alzpp, Amaraiel, Amaury, Amd628, Anders Torlind, Andonic, Andre Engels, Andrewferrier, AndriuZ, Angela, Anikingos, AnjaliSinha, AnmaFinotera, Ann Stouter, AnonUser, Anonymous Dissident, Antandrus, Antrax, Apparition11, Arbitrarily0, Arcann, Arctic Kangaroo, Argon233, Arjun01, Armen1304, ArnoLagrange, Arthena, Arthur Rubin, Arved, ArwinJ, AsceticRose, Asyndeton, AtheWeatherman, Atkinsdc, Autumn Wind, AutumnSnow, Avenged Eightfold, AwamerT, Ayecee, AzaToth, Baa, Babbling.Brook, Barneca, Bbatsell, Bbb23, Bblank, Bcartolo, Bcontins, Beeblebrox, Beetstra, Beland, Ben Ben, Ben-Zin, Benni39, BentlijDB, Bentogoa, Bernd in Japan, Beta M, Betterusername, Bharath357, Bjcubsfan, Bkhouser, Blackguard SF, Blanchardb, BlindEagle, Blood Red Sandman, BluCreator, Bluemask, Bluerocket, BobStepno, Bobblewik, Bogdangiusca, Bogey97, Boing! said Zebedee, Boli1107, Bongwarrior, Bowlderizer, Branzman, Brick Thrower, BrokenSphere, BryanG, Btilm, Bubba hotep, Bunnyhop11, Burner0718, Buzzimu, Bwhynot14, C12H22O11, CIreland, COMPFUNK2, CableCat, Calabe1992, Call me Bubba, Callanecc, Calliopejen1, Caltas, Calutuigor, Cambalachero, Cambapp, Cammo33, Camw, Can't sleep, clown will eat me, CanadianLinuxUser, CanisRufus, Canterbury Tail, Cantras, Capricorn42, Captain-n00dle, Captain-tucker, Carbonite, CardinalDan, Carliitaeliza, Caster23, CasualVisitor, Cavanagh, Cenarium, CesarB, Cevalsi, Ceyjan, Chaojoker, Cheolsoo, Chester Markel, Childzy, Chirpy, Chocolateboy, ChorizoLasagna, Chrax, Chris 73, Chris G, ChrisGualtieri, Chrislk02, Chrism, Christophe.billiottet, Chriswiki, Chtuw, Chuckhoffmann, Chuunen Baka, Clarince63, Clark89, Click23, Closedmouth, Colindolly, Colonies Chris, Cometstyles, Commander Keane, Compfreak7,

Database management system *Source*: http://en.wikipedia.org/w/index.php?oldid=584529352 *Contributors*: *drew, 05winsjp, 069952497a, 10285658sdsaa, 10metreh, 110808028 amol, 16@r, 206.31.111.xxx, 25or6to4, 28421u2232nfenfcenc, 28nebraska, 2D, 4twenty42o, 65.10.163.xxx, APH, Aaron Brenneman, Abhikumar1995, AddWittyNameHere, Addihockey10, Aditya gopal3, Admfirepanther, Adrian J. Hunter, Aepanico, Afluegel, Afree10, Ahodgkinson, Ahoerstemeier, Ahy1, Aitias, Aj.robin, Akamad, Al Wiseman, Alain Amiouni, Alansohn, Alasdair, Ale jrb, Allan McInnes, Allecher, Alpha Quadrant (alt), Alphax, Alzpp, Amaraiel, Amaury, Amd628, Anders Torlind, Andonic, Andre Engels, Andrewferrier, AndriuZ, Angela, Anikingos, AnjaliSinha, AnmaFinotera, Ann Stouter, AnonUser, Anonymous Dissident, Antandrus, Antrax, Apparition11, Arbitrarily0, Arcann, Arctic Kangaroo, Argon233, Arjun01, Armen1304, ArnoLagrange, Arthena, Arthur Rubin, Arved, ArwinJ, AsceticRose, Asyndeton, AtheWeatherman, Atkinsdc, Autumn Wind, AutumnSnow, Avenged Eightfold, AwamerT, Ayecee, AzaToth, Baa, Babbling.Brook, Barneca, Bbatsell, Bbb23, Bblank, Bcartolo, Bcontins, Beeblebrox, Beetstra, Beland, Ben Ben, Ben-Zin, Benni39, BentlijDB, Bentogoa, Bernd in Japan, Beta M, Betterusername, Bharath357, Bjcubsfan, Bkhouser, Blackguard SF, Blanchardb, BlindEagle, Blood Red Sandman, BluCreator, Bluemask, Bluerocket, BobStepno, Bobblewik, Bogdangiusca, Bogey97, Boing! said Zebedee, Boli1107, Bongwarrior, Bowlderizer, Branzman, Brick Thrower, BrokenSphere, BryanG, Btilm, Bubba hotep, Bunnyhop11, Burner0718, Buzzimu, Bwhynot14, C12H22O11, CIreland, COMPFUNK2, CableCat, Calabe1992, Call me Bubba, Callanecc, Calliopejen1, Caltas, Calutuigor, Cambalachero, Cambapp, Cammo33, Camw, Can't sleep, clown will eat me, CanadianLinuxUser, CanisRufus, Canterbury Tail, Cantras, Capricorn42, Captain-n00dle, Captain-tucker, Carbonite, CardinalDan, Carliitaeliza, Caster23, CasualVisitor, Cavanagh, Cenarium, CesarB, Cevalsi, Ceyjan, Chaojoker, Cheolsoo, Chester Markel, Childzy, Chirpy, Chocolateboy, ChorizoLasagna, Chrax, Chris 73, Chris G, ChrisGualtieri, Chrislk02, Chrism, Christophe.billiottet, Chriswiki, Chtuw, Chuckhoffmann, Chuunen Baka, Clarince63, Clark89, Click23, Closedmouth, Colindolly, Colonies Chris, Cometstyles, Commander Keane, Compfreak7, Comps, Constructive, Conversion script, Corvoe, Courcelles, Cpereyra, Cpl Syx, Cpuwhiz11, Craftyminion, Craig Stuntz, Crashdoom, Credema, Crucis, Cryptic, Culverin, Cyan, Cybercobra, Cyberjoac, CynicalMe, D. Recorder, DARTH SIDIOUS 2, DEddy, DFS454, DJ Clayworth, DVD R W, DVdm, DamnRandall, Dan100, Dancayta, Dancter, Danhash, Daniel.Cardenas, DanielCD, Danieljamesscott, Danim, Dart88, Darth Mike, Darth Panda, Darthvader023, Davewild, David Fuchs, David0811, Davidcopperfield123, Dbates1999, Dbfirs, DePiep, Dead3y3, DeadEyeArrow, DeadlyAssassin, Deathlasersonline, Decrease789, DeirdreGerhardt, Denisarona, DerBorg, DerHexer, Deville, Dgw, Diamondland, DigitalEnthusiast, Discospinster, Djordjes, Djsasso, Dkastner, Dkwebsub, Doc glasgow, Doddsy1993, Dogposter, Donama, Doniago, Donner60, DougBarry, Dougofborg, Doulos Christos, DragonLord, Dreadstar, Dreamyshade, Drivenapart, Drumrol99, Dsimic, Duyanfang, Dwolt, Dysepsion, E23, Eagleal, Earlypsychosis, EarthPerson, EastTN, Echartre, Eddiecarter1, Eddiejcarter, Edgar181, Edgarde, Edivorce, Edward, Eeekster, Ejrrjs, ElKevbo, ElijahLloyd97, Elwikipedista, Epbr123, Epicgenius, Era7bd, Eric Bekins, Eric Burnett, Eric3K, Ericlaw02, Erikrj, Escape Orbit, Etxrge, EugeneZelenko, EvergreenFir, Everyking, Evildeathmath, Excirial, Exor674, Explicit, Eyesnore, Ezeu, FFGeyer, Fabartus, Fang Aili, FatalError, Favonian, Feedmecereal, FetchcommsAWB, Feydey, Fieldday-sunday, Filx, Finlay McWalter, Flewis, Flubeca, Fluffernutter, Flyer22, FlyingToaster, Fooker69, Fortdj33, Foxfax555, Fraggle81, Frankman, Franky21, Franl, Fratrep, Freebiekr, Frsparrow, Frze, Fubar Obfusco, Furrykef, Fuzzie, Fæ, G12kid, GDonato, GHe, GLaDOS, Gadfium, Gail, Gajurahman, Galzigler, Garyzx, Gburd, Giftlite, Gilliam, Ginsengbomb, Ginsuloft, Girl2k, Gishac, Glacialfox, Glane23, GlenPeterson, GnuDoyng, Gogo Dodo, GoingBatty, Gonfus, Gozzy345, Graeme Bartlett, GraemeL, Graham87, GrayFullbuster, GregWPhoto, Gregfitzy, GregorB, Grim23, Grsmca, Grstain, Gsallis, Gscshoyru, Gwizard, Gzkn, Haakon, Hadal, HaeB, Hamtechperson, Hankhuck, HappyInGeneral, Harej, Hasek is the best, HeliXx, Helixblue, Helloher, HexaChord, Heymid, Heysim0n, Hgilbert, Hotstaff, Hshoemark, Hugsandy, Huntthetroll, Hurricane111, HybridBoy, Hydrogen Iodide, IComputerSaysNo, IElonex!, Iced Kola, Igoldste, IkamusumeFan, Imfargo, Imnotminkus, Imran, Informatwr, Insineratehymn, Inspector 34, Intgr, IrfanSha, IronGargoyle, Ironman5247, Isfisk, Itafran2010, ItsZippy, Ixfd64, J.delanoy, JCLately, JForget, JJdaboss, JLaTondre, JMRyan, Ja 62, JaGa, Jab843, Jabby11, Jack Greenmaven, Jackacon, Jackfork, JamesBWatson, JamesMoose, Jan1nad, Jandalhandler, Jarble, Jasimab, Jasper Deng, Jauerback, Javert, Jaxl, Jay, Jb-adder, Jclemens, Jdlambert, JeffTan, JeffreyYasskin, Jennavecia, JephapE, Jerome Charles Potts, Jesant13, JewishMonser69, Jk2q3jrklse, Jmanigold, Jni, JoanneB, Joel7687, Joffeloff, John Vandenberg, John of Reading, Johnuniq, Jojalozzo, Jonathan Webley, JonathanFreed, Jondel, Jonearles, Jonwynne, Joshnpowell, Joshwa1234567890, Journalist, Jschnur, Jstaniek, JunWan, Jvhertum, Jwoodger, Jwy, KILLERKEA23, Kanonkas, Karlhahn, Karmafist, Katalaveno, Keenan Pepper, Keilana, Kekekecakes, Kellyk99, Kenny sh, Kevins, KeyStroke, Khazar2, Khoikhoi, Kiand, Kimberly ayoma, Kimera Kat, King of Hearts, Kingius, Kingpin13, KingsleyIdehen, Kivar2, Kkailas, Knbanker, Koavf, Kocio, Komal.Ar, KoshVorlon, KotetsuKat, Kozmando, Kraftlos, Krashlandon, Kslays, Kukini, Kunaldeo, Kungfuadam, Kuru, Kushal one, Kvasilev, Kwiki, KyraVixen, Kzzl, L Kensington, LC, Lamp90, LaosLos, Larsinio, Latka, Lbausalop, Leaderofearth, Leandrod, LeaveSleaves, LeeHam2007, Leonnicholls07, LessHeard vanU, Levin, Levin Carsten, Lexo, Lflores92201, Lfstevens, Lguzenda, Lights, LindaEllen, Lingliu07, Lingwitt, Linkspamremover, LittleOldMe, LittleWink, Llyntegid, Lod, Logan, Lotje, Lovefamosos, Lovelac7, Lowellian, Lradrama, Lsschwar, LuK3, Lucyin, Lugia2453, Luizfsc, Luna Santin, M.badnjki, M4gnum0n, MBisanz, MECiAf., MER-C, MJunkCat, Machdohvah, Madhava 1947, Madth3, Magioladitis, MainFrame, Majorly, Makecat, Malvikiran, Mandarax, Manikandan 2030, Mannafredo, Marasmusine, Mark Arsten, Mark Renier, MarkSutton, MartinSpamer, Materialscientist, Mathewforyou, Mato, Matthewrbowker, Matticus78, Mattisse, Maty18, Maury Markowitz,

Types of DBMS *Source*: http://en.wikipedia.org/w/index.php?oldid=598218773 *Contributors*: Askari786, Autoerrant, Chris the speller, ChrisGualtieri, Cymru.lass, Fraggle81, Happysailor, Hgilbert, Joshua Issac, Katharineamy, Malcolma, Materialscientist, MusikAnimal, SchreiberBike, Shyammohankanojia, Theimmaculatechemist, Tolly4bolly, Topbanana, Widr, 84 anonymous edits

Data store *Source*: http://en.wikipedia.org/w/index.php?oldid=595148567 *Contributors*: Beland, Beta m, Frap, Jarble, Juzeris, Rilak, Sae1962, Textractor, 7 anonymous edits

Technical Information Project *Source*: http://en.wikipedia.org/w/index.php?oldid=575017625 *Contributors*: Cander0000, Danim, Doctus, Malcolma, Rich Farmbrough, SimonP, Stumps

Data modeling *Source*: http://en.wikipedia.org/w/index.php?oldid=582234686 *Contributors*: A. Parrot, AGK, ASHPvanRenssen, Ahoerstemeier, AndriesVanRenssen, Andy Dingley, Antonielly, Arpabr, Asanka Rajapaksha, AutumnSnow, Axeltroike, Bo Jacoby, Bongwarrior, Brick Thrower, Certus01, Charles T. Betz, Chris the speller, Cristan, DJPohly, Dan Polansky, Danim, Dave Hay, DavidCHay, DeadEyeArrow, Denoir, Dicklyon, Diego Grez, Dr. JJenkins, Duncharris, EagleFan, EddyVanderlinden, Egrabczewski, Elsendero, Elwikipedista, Epbr123, Evert r, Findling67, FocalPoint, Friendlydata, George100, Georgesd, Gmelli, Gregbard, Guy Harris, Harburg, Hardywang, Helwr, Hi Yo Epona!, Jeff3000, Jjalexand, John of Reading, Jr johnstone, Katonal, KeyStroke, Kku, Kubanczyk, Kuru, LarRan, Leandrod, Libcub, Luna Santin, MER-C, ManoMarks, Mark Renier, Martinig, Mbasuchit1, Mdd, Metaframe, MikeLang, Mkluwe, Muhammod13, Mutant, Nav102, Neenadavies, Oxymoron83, Pandit 2008, PascaleKelly, Paul Bassin, Peppage, PeterFV, Petrb, Phc3719, Qwyrxian, RFMack, Razorbliss, Reedy, Rhododendrites, RichMorin, Richard Bruce Bradford, Rjwilmsi, Rodasmith, Rpajares, Sameen.r, Sava chankov, Seoegb123, Shamatt, Siroxo, Some standardized rigour, Stevenj, Strike Eagle, Sweetmoose6, Tdrewry, Techtonik, The ken evans, Thinktdub, Turnstep, TylerRick, Vegaswikian, Veinor, Vjson, Vrfour, Walter Görlitz, Westyfield, Winterst, Wisesabre, Wymwe5, XP1, ZimZalaBim, Zondor, Zvika, ^demon, 194 anonymous edits

Data model *Source*: http://en.wikipedia.org/w/index.php?oldid=594133776 *Contributors*: 16@r, AGK, AS, Akinyemi, Aleksd, Algebraist, Anca, Angrytoast, Arthena, Bambam229, Bgwhite, Bill Malloy, Binary TSO, Bryan Derksen, CanadianLinuxUser, Chris G, ChrisGualtieri, Closedmouth, DARaynor, Danim, David.Horat, DerBorg, Derek Ross, Disooqi, Dmccreary, Dr. JJenkins, Dreftymac, ESRIRedlands, EagleFan, EddyVanderlinden, Edward, Egrabczewski, Encycloshave, Equilibrioception, Erianna, Findling67, FlyHigh, FocalPoint, Frap, George100, Gorbag42, Heirpixel, Hmains, Humane Earth, Informatwr, Intellectual Superior, J04n, Jack Greenmaven, Jaideraf, Jan Hidders, Jeff.Hull, Joel7687, John, John of Reading, Kandreyev, KeyStroke, Kku, Kotecky, Kylu, Lauratech4ever, Leandrod, Libcub, LilHelpa, Loretta Mahon Smith, M4gnum0n, MacFreek, Mann jess, Marc Venot, Mark Arsten, Mark Renier, Mdd, MeekMark, Mhkay, Michael Hardy, Nasnema, Neenadavies, NetRolller 3D, Niceguyedc, OnceAlpha, Ott, Paul Bassin, Paul W, Phelun, Phoebe, Phsource, Pmm infomgmt, Prenju, Quantumobserver, R'n'B, RFMack, Ravinjit, Razorbliss, Recurring dreams, Reedy, Rettetast, Rhododendrites, Rjwilmsi, Rknasc, Ronz, Rpresser, Saigyo, Sarang, Schandi, Sfan00 IMG, Skittleys, Smhdog, Tabletop, Taka, Tentinator, Theserialcomma, Tjarrett, Udo Altmann, Wbm1058, Widr, Wiki8127, Woohookitty, Wymwe5, Yamamoto Ichiro, Yintan, Zhangmoon618, ZimZalaBim, Zondor, 183 anonymous edits

Database model *Source*: http://en.wikipedia.org/w/index.php?oldid=598035327 *Contributors*: AGK, ARUNKUMAR P.R, AgadaUrbanit, Airplaneman, Alansohn, Amy whattt, AutumnSnow, BOOSA SANDEEP, Beland, Bill Slawski, CharlesBarouch, Cybercobra, Danim, Decrease789, Dkwebsub, Dwils098, ENeville, Edward, FatalError, Isacdaavid, J.delanoy, Jabbba, Jbolden1517, JoyMundy, Jwoodger, LaosLos, Magomaitin, MainFrame, Mark Renier, Materialscientist, Mdd, Mihai-gr, Mindmatrix, Minimac, Minna Sora no Shita, Mr. Vernon, Nn123645, Philip Trueman, Porterjoh, Razorbliss, Richramos, Roenbaeck, Sun Creator, Tim1357, Vegaswikian, Wikipelli, Woohookitty, 86 anonymous edits

Database design *Source*: http://en.wikipedia.org/w/index.php?oldid=596778531 *Contributors*: A. B., Agrumer, Allecher, Allen3, Alpha Quadrant, Andrewman327, AnmaFinotera, Anna Lincoln, ArchCarrier, Arthur Rubin, Bhanks, Binksternet, Bobo192, Bruce s r, Capricorn42, Codeczero, Compfreak7, Cschutijser, Cyrius, DVdm, Danim, David Delony, DavidCHay, Dethomas, DhAlcojor, Disnarda, Dr. Zombieman, Dr1819, Dreikin, Ejiskolo, ElKevbo, Elwikipedista, Ewebxml, Falcon8765, Focus22, Fraggle81, Friendlydata, Giteshtrivedi, Grsmca, Handcover, Hrundi Bakshi, Jason Quinn, Jon Awbrey, Jwoodger, KeyStroke, Kku, Ksnow, LAX, Lguzenda, LilHelpa, Lingliu07, M4gnum0n, Magioladitis, Maimai009, Mark Arsten, Mark Renier, Matt Beard, Mattbrundage, Mdd, Mentifisto, Mikimik, Miquonranger03, Mixer, MrOllie, Myasuda, Neerajadsul, Ohnoitsjamie, Oxguy3, Petrb, Puchiko, RMFan1, Ravenmichael, Ravinjit, Rbellika, Rednblu, RichardVeryard, Sanskritkanji, Seaphoto, Shamanx, SkyWalker, Smokris, Soap, Sobloku, Tarotcards, The Letter J, The wub, Tkgd2007, Troels Arvin, Turgan, TyA, Ubiquity, Vigyani, White Trillium, Widr, Xcentaur, Xhienne, Zzuuzz, 175 anonymous edits

Conceptual schema *Source*: http://en.wikipedia.org/w/index.php?oldid=599166250 *Contributors*: Baz.77.243.99.32, Bobby122, Charles Matthews, Crysb, Danim, DavidCHay, Destynova, EagleFan, Gregbard, ILovePlankton, J'raxis, Johannes Simon, Jojalozzo, Jsmethers, Karthikshanth, KeyStroke, Khalid hassani, Kku, Libcub, Mark Renier, MaryEFreeman, Mdd, Modify, Nichtich, Ps07swt, Qu3a, Rbrewer42, RedWolf, Rich Farmbrough, Sminthopsis84, Stevertigo, Tarotcards, Tsarris, Walter Görlitz, Woohookitty, Zanaq, 43 anonymous edits

Data structure diagram *Source*: http://en.wikipedia.org/w/index.php?oldid=585756035 *Contributors*: Alansohn, Atlant, Begewe, BoP, Chasrmartin, Danim, Dl2000, EagleFan, Fvw, GAllegre, Gamejumperx, GregorB, Guy0307, Heirpixel, Hu12, IzzyIzumi133, Jerome Charles Potts, Joseph Dwayne, KSchutte, Mdd, Michael Hardy, Mjchonoles, Niceguyedc, NickelShoe, Redbull gives you wheels, RichardVeryard, Torchiest, Vishwa mohan 49, Wael Ellithy, Welsh, 10 anonymous edits

Hierarchical database model *Source*: http://en.wikipedia.org/w/index.php?oldid=599197054 *Contributors*: 10metreh, A5b, AJackl, Al E., Alansohn, Aleksd, Alison, Andy Dingley, Arthuralbano, BD2412, Behringerdj, Beland, BenFrantzDale, Benwing, Brick Thrower, Bryan Derksen, Buickid, Bulwersator, CanisRufus, Capricorn42, Charles Matthews, Chase me ladies, I'm the Cavalry, Chrisahn, Clam0p, Colin Kimbrell, DJNW, Dandv, Danim, Dareservire, Dnnk, Dpm64, DragonHawk, Dreftymac, EagleOne, Elwikipedista, Enric Naval, Epbr123, Finlay McWalter, François Robere, Fred Bradstadt, Fredrik, Fyyer, Gdm, Gurch, Gökhan, Harikishore, Heezy, HenryHayes, Horta, Hoymkot, IOLJeff, Iancarter, Jalesh, John Vandenberg, Jpbowen, Karnesky, Kate, Larsinio, Leonard Vertighel, MER-C, MaGa, Magioladitis, Marcus Luccas, Mark Renier, Mdd, Mindmatrix, Moe Epsilon, Nfriedly, Odie5533, PatrickR, Pearle, Petergreer, Philcha, RJFJR, Random832, Razorbliss, Reedy, Reverie98, RickBeton, Ricktroll, Rony fhebrian, RussBlau, S.K., SDSWIKI, Senator Palpatine, Skittleys, Smjg, Sobreira, Spiritia, SqlPac, Stevietheman, Sumooqp, SymlynX, TeaDrinker, Tengai, Tentinator, TheArguer, Thomasmschaefer, Titoxd, Veryprettyfish, Vikreykja, Volland, Vrenator, WAJWAJ, Wainstead, Widr, Wiki alf, Ziounclesi, 176 anonymous edits

Network model *Source*: http://en.wikipedia.org/w/index.php?oldid=593564188 *Contributors*: AVB, Aditya, Adrianwn, Anonymous Dissident, Barkeep, Beland, Bgwhite, Blackguard SF, BonsaiViking, Brick Thrower, Bucketsofg, Can't sleep, clown will eat me, Capricorn42, Cybercobra, Danim, Darkov, David Eppstein, Dbates1999, DeadEyeArrow, Dnnk, Dpm64, Elwikipedista, Enric Naval, Expensivehat, GermanX, Greentryst, Gurch, Iwbrowse, Jamelan, Jpbowen, KGasso, Kate, Kbrose, KingsleyIdehen, Krawi, Kuru, Lifebaka, MER-C, Magomaitin, Mange01, Mark Renier, Master of Puppets, Mdd, Meca999, Media lib, Mhkay, Mikeo, Mindmatrix, Mrgreenwald, Nibuod, Nick, Ocrow, Pavel Vozenilek, Peter Flass, Petergreer, Phatfish, Phe, Philip Trueman, Pvjohnson, RMFan1, Razorbliss, Rholton, Rony fhebrian, S.K., SJP, Sania.17, ShaunES, Shoujun, Shulini, Spiritia, Synchronism, The Thing That Should Not Be, Tomrud, Toyota prius 2, Vanished user 9i39j3, Vikreykja, Yuriz, ZimZalaBim, 119 anonymous edits

Navigational database *Source*: http://en.wikipedia.org/w/index.php?oldid=543783567 *Contributors*: Beland, Danim, Derek R Bullamore, Dysprosia, Edward Z. Yang, Elwikipedista, Ewaddell, Exelban, FatalError, Gnomz007, Golick, Gzornenplatz, Itai, Jatkins, Jeepday, Kinema, Lotje, Mdomenjoud, Mhkay, NormanGray, Pauk, Pol098, Reedy, SatuSuro, Sava chankov, Stan Shebs, Tablizer, 39 anonymous edits

Entity-relationship model *Source*: http://en.wikipedia.org/w/index.php?oldid=493924960 *Contributors*: 1-555-confide, 16@r, AJackl, Ahy1, Alksentrs, Alleborgo, Amaury, Amitchaudhary, Anarchitect, Andrejj, Andrewpmk, Ann Stouter, Anshuk, Apparition11, Ardonik, Arronax50, Arunloboforever, Asavari24, Aupward, Axeltroike, Ayush Samantroy, Bencoder, Benthompson, Bhanks, BigSmoke, Bignose, Bilby, Bobo192, Bolo1729, Borgx, BozMo, Brick Thrower, Bsodmike, CDV, Caltas, CapitalR, Causa sui, Cedric.claidiere, Centrx, Charlesbc, ChrisEich, Claygate, Cohesion, Colin Angus Mackay, Corb3126, Craigwb, Ctxppc, DStoykov, Danhuby, Danim, Datapolitical, DavRosen, Dburbank, DeadEyeArrow, Deragon, Diligentdogs, Diptanshu.D, Discospinster, Diza, Doctorambient, Dodo bird, Domesticenginerd, Doscmaker, Doug Bell, Dougher, Dream of Nyx, Duke Ganote, Dumelow, ERfan111, EWikist, EagleFan, Edward, Egrabczewski, Eirikr, Ekillaby, Elsendero, Elwikipedista, Erics, FERITEJ, Fan-1967, Fieldday-sunday, Finell, Firsfron, Flyer22, Forage, Ftiercel, Fuper, Gebe, GeorgeBarnick, Godrickwok, Gonwin, GraemeL, GreyCat, Gronky, Gsmgm, HTS3000, Hannu lehikoinen, Hapos, HerrSchnapps, HexaChord, HisSpaceResearch, Hu, Hu12, Hugsandy, Huntsclan, ITBlair, Ian channing, Informatwr, Ixfd64, J.delanoy, JD554, JLaTondre, Ja 62, Jan Hidders, Jandalhandler, Jason Quinn, Jay, Jay Gatsby, Jeff3000, JerkerNyberg, Jerome Charles Potts, Jim1138, Johncartmell, Johncurrier, Jojalozzo, Jolta, Jonathan Webley, Jordanthelewis, JorisvS, Jwilkinson, Kalotus, Karlcuya, Karnesky, Kate, Keilana, Kemorgan, KennethJ, KeyStroke, Khalid hassani, Khazar2, Kirklm, Klausness, Kliford, Koavf, Kocio, Kop, Kushalbiswas777, Lambmj, Lancew, Legaia, Leggattst, Libcub, Ligulem, Liotier, LockeShocke, Lprichar, MBisanz, Macadk, Makecat, Malcolmxl5, Mann jess, Manning Bartlett, Marekich, Mark Arsten, Mark Renier, Master of Puppets, Materialscientist, Matthew 1130, Mcowpland, Mdd, Meegs, MelbourneStar, Mflatischler, Mhkay, Michael Hardy, Michael miceli, Miriup, Misteror, Mjb, Mogism, Mojo Hand, Motyka, MrOllie, Msnicki, NBthee, NSLE, Nareshyalagala, Nickcarr, Nishadha, Nn123645, Northernhenge, Northumbrian, Nt777, Octahedron80, Ohnoitsjamie, OsamaBinLogin, Ottomachin, Patrick Deelen, Patriotic dissent, Paul Bassin, Paulgeorgebassin, Paullewis4372, Pde, Perrydegner, Perter Nokia, Petr Dlouhý, Plhale667, Pm master, Pmmulligan, Pooryorick, Pratyya Ghosh, Puffin, Raffen, Ralph Corderoy, Rambaldi47, Raven4x4x, Ravinjit, Rejeantremblay, René, Riccardo.fabris, RichardVeryard, Roadmr, RobertG, RobertRoggenbuck, Robsta, RockMFR, Ronjouch, Ronz, Rosattin, Rp, Rsrikanth05, Rvstephenson, Sae1962, SarekOfVulcan, Saturday, Saurabha5, Saustin1, Sbachmn, Scott, ScottHJohnson, Sean Whitton, Slovakjoe, SoftwareDeveloper, Softwiki, Sstrauch, Stanislav Nowak, Star767, Stephenb, SteveChervitzTrutane, SteveLichtman, Stuque, Stwalkerster, Sulai, Supten, Svirf, THEN WHO WAS PHONE?, Talion86, Tdrewry, Texture, The Illusive Man, The undertow, TheMattrix, Thejoshwolfe, Threazy, Tide rolls, Tijfo098, Tolly4bolly, Toyota prius 2, Tumble, Turnstep, TwoTwoHello, TypoBoy, Udit90, Udo Altmann, UnitedStatesian, Vespristiano, Vishnava, Vrenator, Wdhoke, Whadda, Wikiwawawa, Willking1979, Wireless friend, Wizgha, Wsandiego, Wwmbes, Xilaworp, Yahya Abdal-Aziz, Zanaq, Zhenyu, Zondor, ZweiOhren, ^demon, 616 anonymous edits

Has-a *Source*: http://en.wikipedia.org/w/index.php?oldid=568231498 *Contributors*: Arthena, CesarB, Chris Howard, CodeCaster, Dark Horse, Doddodoo10110100, Honnza, J. Finkelstein, Jmt4b04d4v, JonHarder, MrOllie, Niroshan S, Ottomachin, PFHLai, PJonDevelopment, Philip Trueman, Piccadilly, Remuel, Rich Farmbrough, Rjgodoy, Sadangel, Srinivas, Vicki Rosenzweig, Vmnit, WereSpielChequers, Wmbolle, 18 anonymous edits

Many-to-many (data model) *Source*: http://en.wikipedia.org/w/index.php?oldid=594029415 *Contributors*: Abdull, Adys, Andrewman327, BD2412, ChrisGualtieri, Davedev15, Destynova, Earcanal, George100, Kussembayev, Leifbk, Materialscientist, Mohd.nayaz, Mysdaao, Pesto, Pluke, Thecheesykid, Tim.spears, Van der Hoorn, Wally77, 12 anonymous edits

Enhanced Entity-Relationship Model *Source*: http://en.wikipedia.org/w/index.php?oldid=497853600 *Contributors*: Boccobrock, Derek Ross, HerrSchnapps, JorisvS, Libcub, M4gnum0n, Materialscientist, Mdd, Ptbarnumma1, Tide rolls, Tijfo098, Woohookitty, 25 anonymous edits

Weak entity *Source*: http://en.wikipedia.org/w/index.php?oldid=542526920 *Contributors*: Babbling.Brook, Bender235, Crypticstargate, Danim, Destynova, Dodo bird, Gareth Owen, J'raxis, JForget, Jsmethers, Jwoodger, Mixer, Ottomachin, Qu1j0t3, Rholton, Ritchy, Ron Ritzman, Rustushki, Salam32, Sanfranman59, Stdazi, Tarquin, TheParanoidOne, Woohookitty, Zondor, 40 anonymous edits

Associative Entities *Source*: http://en.wikipedia.org/w/index.php?oldid=514991230 *Contributors*: Abdull, Ahsile, Ayla, Brian Crawford, Danim, Giraffedata, JorisvS, RHaworth, Shareb, Tfga, 10 anonymous edits

Structured-Entity-Relationship-Model *Source*: http://en.wikipedia.org/w/index.php?oldid=544562337 *Contributors*: Jokestress, KateH, Steffen Thomas

Barker's Notation *Source*: http://en.wikipedia.org/w/index.php?oldid=573432330 *Contributors*: Egrabczewski, Gillyweed, GregorB, Jason Quinn, Johncartmell, Magioladitis, Mhkay, Nposs, Pkubiak, RichardVeryard, Tassedethe, 4 anonymous edits

Peter Chen *Source*: http://en.wikipedia.org/w/index.php?oldid=594493629 *Contributors*: Adamdhal, Afcool83, Ahoerstemeier, Ancheta Wis, Bananappl, Bilderbear, Bloodshedder, BorgHunter, Cbdorsett, Cigano, Cimberwolf, DJ Clayworth, Eaefremov, EagleOne, Ejrrjs, Erianna, Esincl1, EvenT, Gilliam, GregorB, GreyCat, J04n, JLaTondre, Jamelan, Janm67, Jerrch, Jpalazzo, KYPark, KeyStroke, Klemen Kocjancic, LilHelpa, LogicalDash, MarkSweep, Mdd, Megartasm93, Mhkay, Michael miceli, Multivariable, Neilc, Nineball, Nlu, OS2Warp, Omnipaedista, Ph.eyes, Plainriverbank, Pnahuy, RDBrown, RexNL, SparsityProblem, Stephenb, Tassedethe, Yuriz, 58 anonymous edits

Unified Modeling Language *Source*: http://en.wikipedia.org/w/index.php?oldid=597053232 *Contributors*: 0, 212.33.53.xxx, 217.162.105.xxx, A. Parrot, AGK, Abelson, Abune, Acather96, Acockrum, AdrianLozano, AdultSwim, Af1n, Afxid, Agentv, Aholub, Alansohn, Alexey.kudinkin, Alexf, Alksentrs, Allan McInnes, Allens, Aludstartups, Ancheta Wis, Anders Nilsson, Andre Engels, Andres Agudelo, Andrewman327, Angela, Anish7, Antandrus, Antonielly, Aquatics, Arjun024, Armand3496, Arnabbh, Astatine211, BarretB, BarryNorton, Bebates, Bechemel, Beetstra, Beland, Ben modeler, BenAveling, BenFrantzDale, Beorhast, Bergsten, Bernard.szlachta, Bevo, Bhanks, Birge, Black Falcon, BoP, Bob22232, Bobdc, Booyabazooka, Borgx, Boson, Brick Thrower, BrownHornet, Bryan Derksen, Cactus.man, Caesura, Camw, Can't sleep, clown will eat me, CardinalDan, CarrieVS, Cathness, Ccsingle07, Cdiggins, Chairboy, ChrisG, ChristianPetro, Clarepawling, Classicalecon, Closedmouth, Cloud ponderer, Cmdrjameson, Cncmaster, Comatose51, Cometstyles, Conti, Corb3126, Corrie.engelbrecht, Craigy144, Crazynas, CrniBombarder!!!, Crysb, Cyberhitesh, Danakil, Danim, Daveryan, David KELLER, David Martland, DavidSaff, Dayewalker, Dbarashev, De-Wiki-observer, DePiep, DeQuire, Deeray82, Demian12358, Demiurg, DennisDaniels, Deon Steyn, DerHexer, Derek Ross, Dethomas, Dhbernstein, Dirk Riehle, Dirkbike, Discospinster, Diyoev, Dmccreary, Dobbin2000, Doradus, Doral, Duncharris, Dvdrtrgn, EPHERE, Eastlaw, Edrawing, Educres, Edward, ElTyrant, Emakarov, Emerks, Emmjoyit, EngineerScotty, Enviroboy, Erkan Yilmaz, EuroCarGT, Everyking, Ewebxml, Faizan, Falcon8765, Famarsha, Fatespeaks, Fdp, Femto, Fieldday-sunday, Figureskatingfan, Firien, FishSpeaker, Fl, Folajimi, Fotomut, Fraggle81, FranckBarbier, Frentos, GLari, Gardar Rurak, GeneralCheese, George100, GeorgeBills, Gerald Mylog, Glenford, Glrx, Gorpik, Gparent, Graphity, Gregbard, Hadal, HamburgerRadio, Hannes Hirzel, Happysailor, Harriv, Harshadsamant, Hedleya, Herbee, Hervegirod, Hhielscher, Hhtracy, Hohohoneysingh, Hongooi, Hpetya, Hu12, Hydrogen Iodide, IMSoP, ITocapa, Intelliw, Iridescent, Isarra (HG), Ivpen, Ixfd64, J Di, JDP90, JLaTondre, JWenting, Ja 62, Jakohn, Jamestochter, Jancewic, Jay, Jcarroll, Jdmarshall, Jeffq, Jems421, JensMDD, JeremySmyth, Jerome Charles Potts, Jim1138, Jjalexand, Jmcrispin, Joconnor, JohnnyGeetar, Jojalozzo, Jonik, Jonkerz, JorgeGG, Jorgjimen, Josef Sábl cz, Joyous!, Jpbowen, Jpiw, Jriddell, Jrutley, Jschnur, Julesd, Julthep, JustAnotherJoe, Kadamwiki, Kakurady, Karategeek6, Kataniza, Kenneth Vergil, Khalid hassani, Khishig.s, Kierano, Kilva, Kimrew, Kingpin13, Kire87, Kishorekumar 62, Kmccoy, Koavf, KrakatoaKatie, Krishvanth, Kronn8, Ksumedh, Kuratowski's Ghost, KyleSummers, Kyng, Lafeber, Lashiec, Learn4life, Lecafe1, LeeHunter, Leggattst, Leibniz, Leszek Jańczuk, Levin, LiDaobing, Ligulem, Lloydsmart, Logonos, Loqi, Luckyherb, Lwoodyiii, M4gnum0n, MDE, MME, MVerkerk, Malleus Fatuorum, Manu31415, Mardus, Maria C Mosak, Mark Foskey, Mark Renier, Mark.murphy, Marsve, MartinSpamer, Martinig, Martpol, Materialscientist, MaxHund, Maximumbrownidge, Mayumashu, Mdd, MegoMars, Mgillett, MicahDCochran, Michael Hardy, MichaelBillington, Mike Rosoft, Mikhail Ryazanov, Mintleaf, Mirosamek, Misteror, Mjchonoles, Mo0, Monymirza, Morgankevinj huggle, Mortense, MrOllie, Mrgamedesign, Mrockman, Msa, Mxn, Myguest, Myselfram, Mysidia, Naayagan, Nagarjunag, Narcsisuss, Nasa-verve, Nashrul Hakiem, Nealmcb, Niceguyedc, Nick1nildram, Nikai, Nioski, Nixeagle, Noah Salzman, O watkins, OVERM1ND, Obradovic Goran, Octahedron80, Ohnoitsjamie, Oicumayberight, Oleg Alexandrov, Oli Filth, Opptic, Ottomachin, Ozten, Pengo, Peterkimrowe, Petrb, Pgerts, Philip Trueman, PhilippWeissenbacher, Philippe, Piano non troppo, Pierre5018, Piksou, Pindakaas, Pmerson, Poccil, Pointillist, Pojo, Polluks, Poor Yorick, Pp3223, Ppelleti, PrestonH, Profvalente, Progdev, Project2501a, QVT, RJFJR, Rajesh.jadhav.18, RayGates, Rcdeboer, Reaper Eternal, Red-eyed demon, Remi0o, Rich Farmbrough, Richard-VS-545, Riverdusty, Rob Burbidge, Robertbowerman, Rodasmith, Rodii, Ronark, Ronz, Roux, Royboycrashfan, Rspeer, Rsrikanth05, Ryadav, Ryan shell, S ried, S.K., SAE1962, Sae1962, Sahuagin, Salix alba, Salty-horse, Salvar, Samliew, Sannse, Sarav62, SarekOfVulcan, Sbwoodside, Sciurinæ, Scott, ScottWAmbler, Seanhan, Seattlenow, SebastianBreier, Sega381, Senak, Serkanbirkan, Shadowjams, Shanes, Siyu LI, Sketch-The-Fox, Skr15081997, Slashme, Sleepyhead81, Smsarmad, Snoyes, Softzen, Spacelib, Spasemunki, Sprachpfleger, SpyMagician, Steven Kelly, Stfg, Stiang, StuffOfInterest, Suffusion of Yellow, Sven Manguard, TastyPoutine, Taw, Tawaregs08.it, Tayal01, Techman224, Technopilgrim, TeleOffice, Tellyaddict, TerraFrost, Tevirselrahc, That Guy, From That Show!, TheAMmollusc, Thejavaguy, Thowa, Thumperward, ThurnerRupert, Timo Honkasalo, Timothyclethbridge, Tobias Hoevekamp, Toddfast, Tom Jenkins, TommyG, TowerDragon, Treisijs, Triwbe, Twasserman, Um9penmc, VOGELLA, Vanished user g454XxNpUVWvxzlr, Vardhanw, Vespristiano, Victor Snezhko, Vim-Hogar, Vocaro, Vrsane, Wael Ellithy, Walter Görlitz, WalterGR, Wht p0wer, Wikipelli, Willking1979, Windknigth, Wiretse, Wmahan, Woohookitty, Xactium, Xihr, Xmen253, Xtifr, Yakushima, Yamamoto Ichiro, Yamla, Yellowdesk, Zanemoody, Zappernapper, Zergwyn, Zoicon5, Zolkash, Zuckon, Šedý, 甲虫, 1155 anonymous edits

The Third Manifesto *Source*: http://en.wikipedia.org/w/index.php?oldid=593172260 *Contributors*: AutumnSnow, Cathy Linton, Cybercobra, Darren Duncan, DaveVoorhis, Derek Ross, Elwikipedista, EngineerScotty, Ericjs, Falcor84, FatalError, Filemon, Hairy Dude, IMSoP, Jacobko, Jan Hidders, Jsnx, Leandrod, Loadmaster, MrJones, Nkocharh, PeterCanthropus, Rjwilmsi, Rp, Tobias Bergemann, Zzrbiker, 17 anonymous edits

Three schema approach *Source*: http://en.wikipedia.org/w/index.php?oldid=580021825 *Contributors*: Arthur Schnabel, Boson, Bumatic, Ewebxml, J04n, Jerome Charles Potts, Kwiki, LilHelpa, Mark Renier, Mdd, MrOllie, Nitro2k01, Razorbliss, Schmloof, White gecko, 9 anonymous edits

White pages schema *Source*: http://en.wikipedia.org/w/index.php?oldid=558845075 *Contributors*: Joy, MacGyverMagic, MarkWahl, PKT, UkPaolo, 2 anonymous edits

Anchor Modeling *Source*: http://en.wikipedia.org/w/index.php?oldid=570899764 *Contributors*: Crysb, Ctxppc, MrOllie, Roenbaeck, Wadeperson, 5 anonymous edits

Bachman diagram *Source*: http://en.wikipedia.org/w/index.php?oldid=509646166 *Contributors*: Alansohn, Atlant, Begewe, BoP, Chasrmartin, Danim, Dl2000, EagleFan, Fvw, GAllegre, Gamejumperx, GregorB, Guy0307, Heirpixel, Hu12, IzzyIzumi133, Jerome Charles Potts, Joseph Dwayne, KSchutte, Mdd, Michael Hardy, Mjchonoles, Niceguyedc, NickelShoe, Redbull gives you wheels, RichardVeryard, Torchiest, Vishwa mohan 49, Wael Ellithy, Welsh, 10 anonymous edits

Bitemporal Modeling *Source*: http://en.wikipedia.org/w/index.php?oldid=538316254 *Contributors*: Bill Slawski, Iohannes Animosus, Khazar, Nafsadh, Roenbaeck, 1 anonymous edits

Bitemporal data *Source*: http://en.wikipedia.org/w/index.php?oldid=588060406 *Contributors*: Charles Matthews, Greenrd, Joelm, Lotje, Momo san, Nibblus, Quarl, Sadads, Steinsky, Twothongs, Yan Kuligin, 2 anonymous edits

IDEF1X *Source*: http://en.wikipedia.org/w/index.php?oldid=572608640 *Contributors*: ChrisGualtieri, DavidHarkness, Erkan Yilmaz, Favonian, Gordon.harding, Guanajuato, Highlandwolf, J04n, Mark Renier, Mcclarke, Mdd, NETLSTEP, Nick Stavros, Osnetwork, R'n'B, Rillian, Schmloof, 雁太郎, 12 anonymous edits

Universal Data Element Framework *Source*: http://en.wikipedia.org/w/index.php?oldid=597937924 *Contributors*: Akihabara, ChrisHodgesUK, Curt Garcia, DEddy, Discospinster, Dmccreary, Evmako, Gaius Cornelius, Lesser Cartographies, Michael Hardy, RayGates, Ringbang, Ruud Koot, Schuldt, Will Beback Auto, 31 anonymous edits

Terminology model *Source*: http://en.wikipedia.org/w/index.php?oldid=593139378 *Contributors*: Andrewaskew, Beland, CompRhetoric, Magioladitis, Odaba, Phantomsteve, ReinhardK, TheJJJunk, Typo Sheriff, 1 anonymous edits

Georelational data model *Source*: http://en.wikipedia.org/w/index.php?oldid=300966112 *Contributors*: Canis Lupus, ESRIRedlands, Malcolma

Semantic data model *Source*: http://en.wikipedia.org/w/index.php?oldid=531530512 *Contributors*: 4th-otaku, Adi4094, AdjustShift, AndriesVanRenssen, Danim, Mark Renier, Mdd, MilerWhite, Par0328, QuakeSim, Rstower, 5 anonymous edits

Relational Model/Tasmania *Source*: http://en.wikipedia.org/w/index.php?oldid=599290616 *Contributors*: Czj, Danim, Egrabczewski, Haaninjo, Leandrod, RBarryYoung, Rjwilmsi, The Anome, Tijfo098, Walter Görlitz, Ynhockey, 68 anonymous edits

Relational model *Source*: http://en.wikipedia.org/w/index.php?oldid=594409600 *Contributors*: 130.94.122.xxx, 62.114.199.xxx, A930913, Adamcscott, Altenmann, AndrewWTaylor, AndrewWarden, AndyKali, AnonMoos, Arthur Rubin, Ashrust, Asukite, Audiodude, AutumnSnow, Aytharn, BD2412, BMF81, Babbling.Brook, Bblfish, Beland, Bento00, Bobo192, BonsaiViking, Brick Thrower, Brion VIBBER, Budloveall, Bulwersator, CBM, Cadr, Cathy Linton, Cconnett, ChaosControl, Chessphoon, Chrisahn, Chrissi, Conti, Conversion script, Craig Stuntz, Crashoffer12345, Crosbiesmith, DARTH SIDIOUS 2, Damian Yerrick, Danim, DaveVoorhis, David Eppstein, Derek Ross, Dreadstar, Drunken Pirate, EagleFan, Ehn, Elwikipedista, Emx, Enric Naval, Erik Garrison, Evildeathmath, Furrykef, Fyrael, Gadfium, Gary, Gary D, Giftlite, Gilliam, Grassnbread, Greenrd, Gregbard, GregorB, Gurch, Hans Adler, Helvetius, Hyacinth, Ideogram, Iluvitar, Immunize, Irishguy, Ixfd64, J04n, JCLately, Jadedcrypto, Jalesh, Jan Hidders, Jarble, Jbolden1517, Jeff3000, JesseW, Jklowden, Jmabel, Joelm, Jon Awbrey, Jpbowen, Kassie, Kendrick Hang, Ketiltrout, Khalid hassani, Kimchi.sg, Kjkolb, Klausness, Korrawit, Lahiru k, Larsinio, Leandrod, Leifbk, Lethe, Lfstevens, Lopifalko, MER-C, Madjestic, Magioladitis, Maokart444, MarXidad, Marc Venot, Mark Renier, Materialscientist, Mathprodigy20, Matt Deres, Mblumber, Mckaysalisbury, Mdd, Metaeducation, Mets501, Mhkay, Michael Hardy, MilerWhite, Mindmatrix, Moogwrench, Muntfish, NSash, Nad, Nascar1996, Neilc, Niteowlneils, NonDucor, Nsd, Ocaasi, Ocrow, Ozten, Pablothegreat85, Paul Foxworthy, Pitix, Pmsyyz, Pol098, PsychoAlienDog, Quazak Zouski, R'n'B, Razorbliss, Rbrwr, Reedy, Reyk, RonaldKunenborg, Ronhjones, Rp, Rursus, Ruud Koot, S.K., Sae1962, Sdorrance, Seraphim, SeventyThree, Sietse Snel, Simetrical, SimonP, Sonett72, Spartan-James, Spellcast, SpuriousQ, SqlPac, SteinbDJ, Stevertigo, THEN WHO WAS PHONE?, Tatrgel, Teknic, The-G-Unit-Boss, Tjic, Tobias Bergemann, Tohobbes, Tony1, Toreau, Troels Arvin, Tualha, Turnstep, Vikreykja, Welsh, Wgsimon, Windharp, Winhunter, Wjhonson, Woohookitty, Zklink, 303 anonymous edits

Relational database *Source*: http://en.wikipedia.org/w/index.php?oldid=595288951 *Contributors*: *Kat*, 01001, 127, 217.162.105.xxx, 64.192.12.xxx, Abdull, Abolen, Adamcscott, Adamrush, Admrboltz, Agateller, Ahoerstemeier, Alain Amiouni, Alansohn, Andre Engels, Angela, Anuja297, Appzter, Astheg, AutumnSnow, Avoided, Banes, Beland, Benderjaii, Beno1000, Bitnine, Bobo2000, Boothy443, Booyabazooka, Bpalitaa, Brennamack, Brick Thrower, Bsdlogical, CALR, Calmer Waters, Calvernaz, Chris.Giles, Chrislk02, Conversion script, Cpiral, Craig Stuntz, Crosbiesmith, Cww, DARTH SIDIOUS 2, DVdm, Dandv, Danim, Dannydaman9, Darth Mike, Dave6, David Delony, Dfeuer, DinosaursLoveExistence, Dionyziz, Dirk P Broer, DoorsAjar, DouglasCalvert, Download, Drgs100, Dschwart11, Dumbledad, EagleFan, Eik Corell, El C, ElKevbo, Emperorbma, Fabrictramp(public), FatalError, FayssalF, Ferkelparade, Fidimayor, Fieldday-sunday, Filiocht, Findling67, FlyingDoctor, Francs2000, Fratrep, Fred Bradstadt, Freediving-beava, Frigotoni, Fuddle, Gaur1982, Gerbrant, Giftlite, Glane23, GoingBatty, Graham87, HJ Mitchell, Hapsiainen, Harold f, Herostratus, Hmrox, Hp-NJITWILL, I do not exist, ILikeBeer, IRP, Ideogram, Iohannes Animosus, J.delanoy, JCLately, JLaTondre, JaGa, Jacobrothstein, Jan Hidders, Jarble, JeremySmyth, Jerry-VA, Jitendraapi, Jncraton, John Vandenberg, Johnuniq, Jon Awbrey, Juro2351, Jwoodger, Jóna Þórunn, K faiad, KingsleyIdehen, Klausness, KnowledgeOfSelf, Kostmo, Kraron, Kristensson, Krogstadt, Kuru, Lalapicklem, Lamp90, Larsinio, Leandrod, Lfstevens, Linlasj, Logthis, Looxix, Luna Santin, LyricalCat, MC MasterChef, MER-C, Mac, MainlyDigGrammar, Mandarax, Manop, Mark Renier, Mark T, Mav, Mblumber, Mckaysalisbury, Merlion444, Metroking, Michael Hardy, Michael Hodgson, Mikeblas, MilerWhite, Mindmatrix, Msikma, NHRHS2010, Nannahara, Nanshu, Neo-Jay, Nisantha04, Niteowlneils, Nocohen, Ns.code, Odie5533, Odysseus1479, Olinga, Oursinees324, OwenBlacker, Oxymoron83, Pablo323, Pdcook, Pearle, Philcha, Pietdesomere, Pinkadelica, Psb777, Psychcf, Quitchy, Rasmus Faber, RayGates, Rchertzy, Rfl, Riggz0rer, Romanm, Rrburke, Rsduhamel, SandyGeorgia, Scouser0phil, Sequologist, Sfe1, Sgiovannini, Shinju, Sir Nicholas de Mimsy-Porpington, Sir Vicious, Sjschultz, Slightlyusefulcat, Smjg, Solipsist, Sonett72, Specialbrad, Spiritia, Spudtater, SqlPac, Stare at the sun, SteinbDJ, Steve Casburn, Stryn, Super48paul, Supten, TJRC, Tcncv, Tcnuk, Ted Longstaffe, Teles, TheDJ, Thingg, Tijfo098, Tobias Bergemann, Todd Vredevoogd, Tom harrison, Triddle, Triwbe, Troels Arvin, Ttguy, Turnstep, UKAmerican, Utcursch, Vespristiano, Wavelength, Wesley, Wfrys, Wolfraem, Wolfsbane2k, Xiong, Xphile2868, Zahid Abdassabur, Zipircik, 597 anonymous edits

Relational database management system *Source*: http://en.wikipedia.org/w/index.php?oldid=597631897 *Contributors*: 16@r, AMD, Abb615, Acrider, Afabbro, Aldie, Ale And Quail, Alonemayank, Altenmann, Anastrophe, Anvish, Anwar saadat, Apokrif, Athaenara, AutumnSnow, BL, BMF81, Ballin Insane10, Beland, Bgibbs2, Bob hoskins, BodyTag, Bonadea, Borgx, Brassrat70s, Bressan, Brick Thrower, Brilliantwiki, Cactus26, Chaitrabhat7, Chris Roy, Cnb, Craig Stuntz, Crosbiesmith, Cryptic, Cwitty, DB 103245, Darx9url, Davedx, Daverocks, Dewritech, DigitalEnthusiast, Dockurt2k, Elwikipedista, EoganOD, Faizan, FatalError, Fatehyab ahmed, Flata, Frze, George Rodney Maruri Game, Gilliam, Grunt, Gurch, HEAdrian, Heimstern, I dream of horses, II MusLiM HyBRiD II, Igoldste, Igor Yalovecky, Ikhzter, J.delanoy, J36miles, JCLately, JFM, JHMM13, JamesBWatson, Jameshfisher, Jan Hidders, Jdthood, Jim1138, Jnlin, Joao.matos, Josemanimala, Joseph Dwayne, Joseph chennai, Jtgerman, Kaihsu, Karada, Kate, Kernel.package, KeyStroke, Kingston Dominik, Klausness, Kotika98, Kuru, Larsinio, Leandrod, Lfstevens, LinguistAtLarge, Lowellian, Lulu of the Lotus-Eaters, Mangoe, Mark Renier, Maximaximax, Mckaysalisbury, MelbourneStar, MikeSchinkel, Mikeblas, Mindmatrix, Minghong, Mintleaf, Mr4top, Mxn, Neilc, Nicks100, NuclearWarfare, Nylex, Oberiko, Obradovic Goran, Ohnoitsjamie, Ohyoko, Optakeover, Palica, Paulcolmer, Payal2820, PhiLiP, Pi, Pichpich, Pratyya Ghosh, Quest for Truth, RedHillian, Reddi, Reedy, Rfl, Rhobite, Robert Brockway, Rror, Sasquatch, Setppo, Shabbirbhimani, Shepazu, Smyth, Sparkiegeek, SqlPac, Stevegiacomelli, Szajd, Tablizer, TallMagic, TenPoundHammer, Tentinator, Tolly4bolly, Tomcat66 g500, Troels Arvin, Turnstep, UnDeRTaKeR, Uriber, Useight, VTPG, Vanished user qkqknjitkcse45u3, Vegaswikian, Vincent Liu, WIKIWIZWORKER, Wykypydya, Xcasejet, Xphile2868, Лев Дубовой, 396 anonymous edits

Life cycle of a relational database *Source*: http://en.wikipedia.org/w/index.php?oldid=580913175 *Contributors*: Altenmann, Arthur Schnabel, Boson, Chris the speller, Crysb, Cwmhiraeth, DStoykov, EagleFan, Ewebxml, GoingBatty, Jalal saeed, Jon Awbrey, Lfstevens, LilHelpa, Magioladitis, Materialscientist, Mato, Mikesc86, MrOllie, Pearle, RB90, Reconsider the static, Ronnam, Rrabins, TommyG, V4us, Woodshed, Ysw1987, 9 anonymous edits

Logical data model *Source*: http://en.wikipedia.org/w/index.php?oldid=597093633 *Contributors*: Apugazh, Beland, Brick Thrower, Christianvinter, DavidCHay, Fl, Francois Cartier, Gaslight1900, Haakondahl, Hooperbloob, Jalwikip, Jonnich, Josve05a, Kalotus, KeyStroke, Kitdaddio, Kku, Lucky 6.9, Magioladitis, Mark Renier, MaryEFreeman, Matthew 1130, MauriceJFox3, Mcclarke, Mdd, Minna Sora no Shita, N8d1, Omnipaedista, Pukekobird, Razorbliss, RitigalaJayasena, Rodasmith, SchreiberBike, Signalhead, Sumanda, Walter Görlitz, Xyzzyplugh, Zidanie5, 44 anonymous edits

Logical schema *Source*: http://en.wikipedia.org/w/index.php?oldid=542129253 *Contributors*: AHowlett, Alksentrs, Bearcat, Beland, CanadianCaesar, Ciphers, Cyrius, Danim, Dawynn, F-UCKMODS, Harda, JakobVoss, KeyStroke, Mark Renier, Mdd, Paul A, Tinucherian, Varlaam, Филатов Алексей, 17 anonymous edits

Relation (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=598177523 *Contributors*: AndrewWarden, Asfreeas, Asocall, AutumnSnow, Crosbiesmith, Ed Poor, Fratrep, Georgeryp, GreatWhiteNortherner, Icairns, Lfstevens, MaD70, Mark Renier, MusiKk, NickCT, Nigwil, Rob Bednark, Subversive.sound, Talhahabib00, Tijfo098, Universalss, Vieque, 14 anonymous edits

Table (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=593321831 *Contributors*: 12george1, 16@r, Abdull, Ajraddatz, Alai, AlanS1951, Arcann, AutumnSnow, Blanchardb, Bobgrey89, Bobo192, Bongwarrior, Bruxism, C.Fred, Cbrunschen, Coleeey, Correogsk, Cyfal, DARTH SIDIOUS 2, Danim, Djtgamer101, Dreftymac, Dzlinker, Epbr123, Excirial, FattyMcjimmy, Feder raz, Funnyfarmofdoom, Gurch, IMSoP, IanCarter, J36miles, Jamelan, Jerome Charles Potts, Kdib987, Krishna Vinesh, Larsinio, LeonardoGregianin, Lfstevens, Mandy121, Mark Renier, Materialscientist, Mblumber, Mikeblas, Mikeo, Mindmatrix, Morad86, N0nr3s, Nibs208, Nikuwap, Ofus, Pyfan, Quentar, S.K., Sae1962, Scs, Senator2029, Sietse Snel, SimonP, Sippsin, Sonett72, SqlPac, Stolze, TheParanoidOne, TommyG, Turnstep, Txomin, Versageek, Wellskallison, Widefox, Yug1rt, Zhenqinli, 102 anonymous edits

Tuple *Source*: http://en.wikipedia.org/w/index.php?oldid=596625995 *Contributors*: 1ForTheMoney, 213.253.39.xxx, 216.28.46.xxx, 3ICE, Alzarian16, Andres, Anonymous Dissident, Anonymous101, Art Maddox, Arthena, AtomicDragon, AugPi, Avoided, Azimuth1, BD2412, Beefyt, Br77rino, Bsod2, CBM, COGDEN, CRGreathouse, CecilTyme, Cek, Charles Matthews, ChrisGualtieri, Circeus, Compvis, Conversion script, Cpiral, Crestu, Crystallina, Cybercobra, Daelin, Denisarona, Dominus, DopefishJustin, Double sharp, Doug Bell, Dr Greg, Dratman, Dreadstar, Dreftymac, Duncan.france, ENeville, EdC, Edupedro, Egriffin, EpicDream86, Exor674, Flavio Guitian, Fredrik, Gadget850, Gazpacho, Georgia guy, Giftlite, Grundle2600, Hairy

Dude, Hans Adler, Hirvinen, Hyacinth, Iamthedeus, Icktoofay, Ikh, InverseHypercube, Isnow, Ivan Štambuk, Jan Hidders, Jarble, Jeff G., Jefferson2011, Jim.belk, Jim.henderson, Jitse Niesen, John Vandenberg, JohnyDog, Jon Awbrey, Jorge Stolfi, Joshua Issac, Jowa fan, Jrtayloriv, Jtle515, KSmrq, Kbjr14, KeyStroke, Kidlittle, Kietotheworld, King of Hearts, Kku, Knakts, Kowarschick, Lambiam, Lmgottschalk, LokiClock, Lumbercutter, MaNeMeBasat, Madmarigold, Magnus.de, Maksim-e, MarSch, MathMartin, Matthiaspaul, Meters, Mfc, Mfwitten, Michael Hardy, Mike Fikes, Mikevpol, Minesweeper, Miserlou, Moala, Mormegil, Ms2ger, Nastaris, Negi(afk), Ninrouter, NoirNoir, Olaf, Oleg Alexandrov, Patrick, Paulmiko, Pcap, Plastikspork, Pmanderson, PrestonH, Quondum, Racerx11, RandomDSdevel, Revth, Rnddim, Robofish, Runtime, Ruud Koot, S. Neuman, SDC, Sae1962, Safalra, Salamehar, Salix alba, Sderose, Simeon H, Simon12, Sinan Taifour, SoHome, Spoon!, Spooony, Strait, Supmansesis, Suxs62, Tarquin, Tcncv, Tentacles, Thumperward, TigerShark, Tobias Bergemann, Trylks, Velho, Vlad Patryshev, WatchHawk, Wavelength, Wiki alf, Wikiknol, Wikiolap, Woohookitty, Wshun, Zephalis, 204 anonymous edits

Row (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=593691488 *Contributors*: 2help, Allen3, Asfreeas, CommonsDelinker, D4g0thur, Danim, David H Braun (1964), Flip, GLaDOS, Gail, GermanX, Glacialfox, GregorySmith, Jamespurs, Jerroleth, Jmabel, KKramer, KeyStroke, Liujiang, Mark Renier, Mark T, Mxg75, Mzuther, O.Koslowski, Oyauguru, Pnm, Pol098, Retodon8, Rjd0060, Ronhjones, Shaka one, Sietse Snel, SootySwift, Troels Arvin, Yamamoto Ichiro, 32 anonymous edits

Attribute domain *Source*: http://en.wikipedia.org/w/index.php?oldid=525501974 *Contributors*: Cedars, Crashoffer12345, David Eppstein, EsonLinji, Exeunt, Goldenrowley, JaGa, Libcub, Michael Hardy, Oddchr, Yewrajesh, 8 anonymous edits

Candidate key *Source*: http://en.wikipedia.org/w/index.php?oldid=595456608 *Contributors*: Acroterion, AlexPlank, Amikake3, AndrewWarden, Arravikumar, Axiomsofchoice, Brick Thrower, Bryant1410, Captmjc, Charles Matthews, Crosbiesmith, CyborgTosser, DMacks, DVdm, Dharmabum420, Docu, Ejrrjs, Epicgenius, Eric22, FuthaMukker, Hans Adler, HenningThielemann, J.delanoy, Jan Hidders, Jleedev, Jorge Stolfi, Josephbui, JoshDuffMan, Kalyson, KeyStroke, LanguageMan, Mark Renier, Massic80, Materialscientist, MiloszD, Mindmatrix, Mwtoews, Nabav, Neilc, Obradovic Goran, Patriotic dissent, Possession, ProcerusDecor, Prodizy, Rbrewer42, Rholton, Richaraj, Ronald S. Davis, SqlPac, Sreyan, Stolkin, Torzsmokus, Weedwhacker128, Yahya Abdal-Aziz, ZeroOne, Δ, 石庭豐, 98 anonymous edits

Unique key *Source*: http://en.wikipedia.org/w/index.php?oldid=591714839 *Contributors*: Aberdeen01, Ahoerstemeier, Akyadav324, Alessandro57, Alexjbest, AmandeepJ, Ambuj.Saxena, Andre.psantos, Baa, Blue Square Thing, Boson, Causa sui, ChrisGualtieri, Ctimko, DarkFalls, DeadEyeArrow, Dgc03052, Dougher, Drphilharmonic, Ewebxml, Fabrivera99, Faizan, Feraudyh, Fnielsen, Frap, Gurch, Hike395, J.delanoy, JHunterJ, Jberkus, Jbodilytm, Jdeperi, Jesdisciple, Joe.dolivo, Jorge Stolfi, Jyothisdavid4u, KeithB, L'Aquatique, L337 kybldmstr, LittleOldMe, Loren.wilton, Mahemoff, Materialscientist, Mindmatrix, Minimac, Mwtoews, Nabav, Natalie Erin, Northamerica1000, O.Koslowski, Obradovic Goran, Pevernagie, Praveentech, ProcerusDecor, Raja200682, Ratarsed, RobIII, Spartaz, Special Cases, Spinality, Stolze, Subversive.sound, Themusicgod1, Thumperward, Tijfo098, TommyG, Troels Arvin, Unixxx, Velavan, Vjosullivan, Wensceslao, Whitejay251, Wiki.Tango.Foxtrot, Winterst, X201, Yintan, Ykliu, Zzuuzz, 119 anonymous edits

Natural key *Source*: http://en.wikipedia.org/w/index.php?oldid=598366020 *Contributors*: =JACK=, AndyRaffle, AnthonyMastrean, Arcturus, Brossow, Bryant1410, Darksun, Giladbr, Jacksheriff, Jpyamamoto09, KeyStroke, Kjkolb, Kokjeroen, Lkesteloot, Mark T, Pne, Rising, Tfitzg, Timhowardriley, Troels Arvin, Veryverysorry, 14 anonymous edits

Key field *Source*: http://en.wikipedia.org/w/index.php?oldid=597666750 *Contributors*: Aberdeen01, Ahoerstemeier, Akyadav324, Alessandro57, Alexjbest, AmandeepJ, Ambuj.Saxena, Andre.psantos, Baa, Blue Square Thing, Boson, Causa sui, ChrisGualtieri, Ctimko, DarkFalls, DeadEyeArrow, Dgc03052, Dougher, Drphilharmonic, Ewebxml, Fabrivera99, Faizan, Feraudyh, Fnielsen, Frap, Gurch, Hike395, J.delanoy, JHunterJ, Jberkus, Jbodilytm, Jdeperi, Jesdisciple, Joe.dolivo, Jorge Stolfi, Jyothisdavid4u, KeithB, L'Aquatique, L337 kybldmstr, LittleOldMe, Loren.wilton, Mahemoff, Materialscientist, Mindmatrix, Minimac, Mwtoews, Nabav, Natalie Erin, Northamerica1000, O.Koslowski, Obradovic Goran, Pevernagie, Praveentech, ProcerusDecor, Raja200682, Ratarsed, RobIII, Spartaz, Special Cases, Spinality, Stolze, Subversive.sound, Themusicgod1, Thumperward, Tijfo098, TommyG, Troels Arvin, Unixxx, Velavan, Vjosullivan, Wensceslao, Whitejay251, Wiki.Tango.Foxtrot, Winterst, X201, Yintan, Ykliu, Zzuuzz, 119 anonymous edits

Compound key *Source*: http://en.wikipedia.org/w/index.php?oldid=595348234 *Contributors*: Alksub, Bblfish, ElenionAncalima, Ewebxml, Furrykef, Govorun, Imroy, KeyStroke, Lealvf, Leszek Jańczuk, Mallag, MrOllie, Norm, Oli Filth, Patriotic dissent, ProcerusDecor, Rror, Whitejay251, Wiki.Tango.Foxtrot, Zzuuzz, 38 anonymous edits

Foreign key *Source*: http://en.wikipedia.org/w/index.php?oldid=599084143 *Contributors*: 16@r, Abdull, Amix.pal, Anbu121, AndrewWarden, Arichnad, Arthur Schnabel, Arunsinghil, Aurochs, AutumnSnow, Beesforan, Biochemza, Brick Thrower, Can't sleep, clown will eat me, Cander0000, Causa sui, Clarificationgiven, Cory Donnelly, Cpiral, Cww, DHN, Darth Panda, Derek Balsam, DireWolf, Dobi, Dougher, EWikist, Eldavan, Electricnet, Entropy, FatalError, Feder raz, Fluffernutter, Flyer22, Frap, Govorun, GregorB, Gsm1011, IanHarvey, JHunterJ, Jadriman, Jesselong, Jim1138, Jk2q3jrklse, Jlenthe, Joebeone, John of Reading, KeyStroke, Kf4bdy, Kubntk, Larsinio, Marcusfriedman, Mark Renier, MexicanMan24, Mike Rosoft, Mikeblas, MikeyTheK, Mindmatrix, Minimac, Mmtrebuchet, Mogism, Mormegil, MrOllie, Mrt3366, Natalie Erin, Ngriffeth, O.Koslowski, Obradovic Goran, PPOST, Pbwest, Peak, Polypus74, RIL-sv, Reedy, Rjwilmsi, Rror, Rsrikanth05, SDS, Salvatore Ingala, Sboosali, Seanohagan, Selfworm, Semaperepelitsa, Semperf, Shreyasjoshis, Species8473, Staecker, Stolze, Svenmathijssen, Tarquin, Tgeairn, The Thing That Should Not Be, TheExtruder, Threeacres, Timhowardriley, TobiasPersson, Troels Arvin, Unordained, Waskyo, WikHead, Zipz0p, Δ, 石庭豐, 242 anonymous edits

Persistent Object Identifier *Source*: http://en.wikipedia.org/w/index.php?oldid=402510379 *Contributors*: Lear's Fool, Vjosullivan

Cardinality (data modeling) *Source*: http://en.wikipedia.org/w/index.php?oldid=583255988 *Contributors*: 10285658sdsaa, Alan Liefting, Andreas Kaufmann, AndrewMWebster, Atltom331, Beaddy1238, Bragcat, Brunson943, Cst17, Destynova, Gengiskanhg, George Henry Archibald, GregorB, Hemmingsen, JackStoneS, Jay, Jczubeck, Linforest, Mandarax, Mark Renier, Materialscientist, Nagarjunag, Oleg Alexandrov, Onjacktallcuca, RainbowCrane, SAE1962, Stephen Fenner, Sylvain Leroux, Valentinejoesmith, Vksgeneric, 29 anonymous edits

Recordset *Source*: http://en.wikipedia.org/w/index.php?oldid=541704854 *Contributors*: Amicron, Byapparov, Ed Poor, Fufthmin, Jaksmata, Jamelan, MER-C, Nakon, Piotrus, Soumyasch, WadeSimMiser, Warren, Wavelength, Woodshed, Xpclient, 17 anonymous edits

Superkey *Source*: http://en.wikipedia.org/w/index.php?oldid=587816569 *Contributors*: AndrewWarden, Anog, AutumnSnow, Boson, CeleronNutcase, CharlotteWebb, ChrisGualtieri, ColinFine, Crosbiesmith, Dawynn, Fatherlinux, Fimp, Igor Yalovecky, IronGargoyle, James Crippen, Jan Hidders, Jorge Stolfi, Jusdafax, Jwulff, Katieh5584, KeyStroke, Kranix, LOL, Larsinio, M. Frederick, Magioladitis, Mark Renier, Metron4, Michaelcomella, Mikeblas, Millermk, Mindmatrix, Nabav, Pimlottc, ProcerusDecor, Reedy, Rhoerbe, SpuriousQ, Sss41, Stbrob, The Thing That Should Not Be, TheParanoidOne, Tobias Bergemann, Torzsmokus, Twarther, Voidxor, Welsh, Wikitanvir, Yay unto the Chicken, Zzuuzz, Ox, Δ, 石庭豐, 80 anonymous edits

Integrity constraints *Source*: http://en.wikipedia.org/w/index.php?oldid=595171674 *Contributors*: A412, Abdull, Ae86, Agilius, Bulwersator, Byte, Donner60, Edison, Edward, FreeFlow99, Jim1138, Jpvinall, Materialscientist, Nbarth, O paladin o, Pgr94, RenniePet, Ritesh104, Rubicon, Salvar, SqlPac, Sujitrumane, Tide rolls, Tyros1972, Victor falk, 79 anonymous edits

Check Constraint *Source*: http://en.wikipedia.org/w/index.php?oldid=527094467 *Contributors*: Abdull, Aeroplanepics0112, Andreas Kaufmann, Donner60, Edward, EdwardH, LegitimateAndEvenCompelling, MER-C, Mark Renier, Mogism, Normash, Quaker5567, RHaworth, Soluch, SqlPac, Stolze, Tizio, UberCryxic, Unordained, 7 anonymous edits

Propagation constraint *Source*: http://en.wikipedia.org/w/index.php?oldid=526067105 *Contributors*: Elwikipedista, Isopropyl, Jaavaaguru, Jeff3000, Jessemerriman, Libcub, Michael Hardy, PKT, Snodnipper, William Avery, 7 anonymous edits

Transition constraint *Source*: http://en.wikipedia.org/w/index.php?oldid=596528574 *Contributors*: Bastawhiz, CALR, Edward, GregorB, John254, LilHelpa, SqlPac, 1 anonymous edits

Wide and narrow data *Source*: http://en.wikipedia.org/w/index.php?oldid=575367615 *Contributors*: -5-, EoGuy, Melcombe, PaulHurleyuk, RadioFan, Sadads, Zven, 1 anonymous edits

Universal relation assumption *Source*: http://en.wikipedia.org/w/index.php?oldid=554422333 *Contributors*: Rpyle731, Tijfo098, VanishedUserABC, 1 anonymous edits

Reference table *Source*: http://en.wikipedia.org/w/index.php?oldid=554404536 *Contributors*: Andrew Gray, Caerwine, DJ Clayworth, Duke Ganote, Edward, EmanWilm, Goldenrowley, Hrainian, Kappa, Libcub, MONGO, Paul Carpenter, Rjwilmsi, RxS, Sadads, Schmiteye, Ylem, 6 anonymous edits

Junction table *Source*: http://en.wikipedia.org/w/index.php?oldid=528486926 *Contributors*: Alzarian16, Ayla, Bobo192, Cruftcraft, Gabrielsroka, Haakondahl, Kuru, Leifbk, Lord Snoeckx, Macha, Mr.98, Quercus solaris, Rajashar, Rawlogic, Rharner, Srikant.sharma, TheSteve2, 21 anonymous edits

Nested set model *Source*: http://en.wikipedia.org/w/index.php?oldid=596272031 *Contributors*: 0x24a537r9, AManWithNoPlan, Ariel., DaRentzDue, Damianrt, Dougher, GregorB, Grondilu, Ivolucien, Malcolma, Mark Renier, Meinhard Benn, Paul Ebermann, Polytx, RickBeton, Rp, Sherahm, Tassedethe, Tonyrogerson, Zygmuntjr, 23 anonymous edits

Information schema *Source*: http://en.wikipedia.org/w/index.php?oldid=565234112 *Contributors*: AvicAWB, Bricelam, Greenrd, Jason Quinn, Jmorgan, Johnrussell13, Pablitobof, Pnm, Rudament, Rwwww, Tokek, Zeke pbuh, 12 anonymous edits

Codd's 12 rules *Source*: http://en.wikipedia.org/w/index.php?oldid=596505601 *Contributors*: AndrewWTaylor, AutumnSnow, BMF81, Bluemoose, Bobrayner, Brick Thrower, Bulwersator, BuzCo, Catgut, Charles Matthews, Cherkash, Cntras, Danim, Delldot, Dwiedeback, Elf, Excirial, Fiftytwo thirty, GVF, GregorB, Gwernol, Hadal, HaeB, Ijliao, Jamesontai, Jianhui67, Jim no.6, Julesd, JustinWW05, KeyStroke, Khalid, Khalid hassani, Kreca, Lampica, Leandrod, LuK3, Mate2code, Mathmo, Mckaysalisbury, Mhkay, Morwen, MrDolomite, Nasnema, Niceguyedc, Oneiros, Paul Magnussen, Petrb, Pgk, Pingveno, Recnilgiarc, Rene Mas, RobJ1981, Seaphoto, Simplyharsh, SmackoVector, Snigbrook, SqlPac, StasMalyga, Stirling Newberry, Suffusion of Yellow, SwisterTwister, Szombara88, Thompson.matthew, Troels Arvin, Turnstep, Uncle G, Vaceituno, Vanished user 1234567890, Versus22, Widefox, Wikipelli, Woohookitty, Wymanro,

Zvar, 160 anonymous edits

Edgar F. Codd *Source*: http://en.wikipedia.org/w/index.php?oldid=593166429 *Contributors*: Accurizer, Albanaco, Alma Pater, AlphaAqua, Ancheta Wis, Andrei Stroe, Apc005, Asiananimal, Asymmetric, AutumnSnow, BMF81, Bazj, Bbsrock, Bobblewik, Bovineone, Btomp, Cdrdata, Cole is ugly, Crosbiesmith, Cyro, D6, DJPhazer, DanielBeaver, David Eppstein, DavidRoe, Dbeardsl, DePiep, Derek Ross, Dmeddy, Dmsar, Drjefflehewphd, Duds 2k, Duncan.Hull, Duncharris, Dysprosia, E23, EagleOne, Ed Poor, Eggy1609, Egrabczewski, Elwikipedista, EnOreg, Farrellm, Ferkel, Finlay McWalter, Francs2000, Fred Bradstadt, Frosted14, Gdm, Gentgeen, Giftlite, Girl2k, Gjd001, Harvester, HeartofaDog, Infrogmation, Isidore, Ivan Štambuk, Ixfd64, JHMM13, Jan Hidders, Janm67, Jay, Jedward, John Vandenberg, Jon Awbrey, Jost Riedel, Jpbowen, K.lee, KF, Kbdank71, Kelson, KeyStroke, Kpjas, Lampica, Leandrod, Lugnuts, MER-C, Man0ai, Mark Renier, Mav, Mckaysalisbury, Mdd, Mhkay, Michael Hardy, Mindmatrix, MrDolomite, Nchanin, Necrothesp, Neilc, Nima Baghaei, Ningauble, Nocat50, Nonick, Notheruser, Omnipaedista, Patrick-br, Paul Magnussen, Pavel Vozenilek, Pedant17, Peter James, Phil Boswell, Pointillist, Pqowieuryt, Prakash Nadkarni, Qu1j0t3, QubitOtaku, RDBrown, Reedy, Resurgent insurgent, Rich Farmbrough, RitKill, Rjpwilliams, Rjwilmsi, Rl, Ronfagin, Ropcat, Rossenglish, Rschroev, Rsocol, Sannse, Schoen, Seefrank, Shinx, Skäpperöd, Soarhead77, Some jerk on the Internet, St33lbird, Stefanomione, Stirling Newberry, Thumb10.40, TimothyChenAllen, Timrollpickering, Tkynerd, Toddst1, Vdgr, Vernanimalcula, Vfp15, VivaEmilyDavies, Vonfraginoff, Wahwah, WeißNix, Wernher, Widefox, Wikiolap, Zzuuzz, پووپا, 211 anonymous edits

Relational algebra *Source*: http://en.wikipedia.org/w/index.php?oldid=595242188 *Contributors*: Agquarx, Alain Amiouni, Alan Liefting, Alansohn, AlecTaylor, AndrewWarden, Anuj royal, Arthur Rubin, Arunloboforever, Austinflorida, AutumnSnow, Banazir, BiT, Blahedo, Blaisorblade, Brick Thrower, Bug, Bulwersator, CALR, CRGreathouse, Cdrdata, Charvest, Chewings72, Chocolateboy, Chris the speller, Clawed, Cmdrjameson, Combatentropy, Cometstyles, CountMacula, Cryout, Cybercobra, Cycchina, DaveVoorhis, Davidfstr, Davnor, Derbeth, Dessources, Dhanuthilaka, DoriSmith, Download, Drowne, Drunken Pirate, EagleFan, Ed g2s, Edcolins, Egmontaz, Egriffin, Elektron, Elwikipedista, Esalder, Ezrakilty, Fabian Pijcke, Falcor84, Flyhighplato, Fresheneesz, FuFoFuEd, Gazpacho, Geira, Giftlite, Gregbard, GregorB, Hadal, Hans Adler, Hasanv, Hughitt1, Hussaibi, Hypergraph, IceCreamAntisocial, Infestor, IvanLanin, JackPotte, Jamesx12345, Jan Hidders, Jan1nad, Jarble, Javert16, JingguoYao, Jleedev, Joebolte, JohnyDog, Jon Awbrey, Joseph Dwayne, Jsnx, Juansempere, Justin W Smith, Kanenas, Keegan, KelvSYC, Khalid hassani, Kinaro, Kjetil r, Kku, Klausness, Klpn81, KnightRider, LOL, Lambiam, Larsinio, Leaflord, Lemycanh, Lfstevens, LtWorf, Magic5ball, Maksim-e, Mandries, Mani1, Mark Renier, Matthiaspaul, Mckaysalisbury, Mcthree, Mdd, Mets501, Michael Hardy, Michealt, Mikeblas, Mindmatrix, Msnicki, Myheimu, Nbarth, NewEnglandYankee, Ntmatter, O.Koslowski, Ocranom, Oleg Alexandrov, PanagosTheOther, Peruvianllama, Peter.vanroose, Pgan002, Phamthelong, Polluxian, Popol1991, Qwertyus, R'n'B, Rathgemz, Reedy, Rgrimson, Rishig327, Rjwilmsi, Rleyton, Rsrikanth05, Ruakh, Rursus, Salix alba, Sam Staton, Samppi111, Sbrenesms, Scf1984, Schmid, Sdorrance, ShadowPhox, Shreyasjoshis, Sir Nicholas de Mimsy-Porpington, Slgcat, SnowFire, Sspecter, Stephan202, Tablizer, Tgeairn, The undertow, Tijfo098, Tommy2010, Tompsci, Troels Arvin, Twimoki, Vaucouleur, Vegpuff, Viperlight89, Wavelength, Way2veers, Wayne Slam, We64, Wifki, Wikipelli, Wrp103, Xcpenguin, Yoosofan, Zink Dawg, Ziusudra, と ある 白い猫, 348 anonymous edits

Projection (relational algebra) *Source*: http://en.wikipedia.org/w/index.php?oldid=582370918 *Contributors*: AK456, AndrewWarden, Clarkcj12, Dfass, Drowne, EagleFan, Hyju, Itsmejudith, Jon Awbrey, Joseph Dwayne, KelvSYC, Light current, Linas, Liveste, Nbarth, Niceguyedc, Salix alba, SchreiberBike, Seaphoto, Shreyasjoshis, Srleffler, Subversive.sound, Tablizer, Tijfo098, 10 anonymous edits

Rename (relational algebra) *Source*: http://en.wikipedia.org/w/index.php?oldid=589442268 *Contributors*: AndrewWarden, Blahedo, EagleFan, Niceguyedc, R'n'B, SchreiberBike, 1 anonymous edits

Selection (relational algebra) *Source*: http://en.wikipedia.org/w/index.php?oldid=546529115 *Contributors*: AndrewWarden, Blahedo, EagleFan, Gregbard, Hyju, Joseph Dwayne, KelvSYC, MattGiuca, TheBigGuy, 3 anonymous edits

Generalized selection *Source*: http://en.wikipedia.org/w/index.php?oldid=504532278 *Contributors*: Dessources, EagleFan, Gregbard, Joseph Dwayne, KelvSYC, Malo, Niceguyedc, Rettetast, SchreiberBike, 1 anonymous edits

Range query *Source*: http://en.wikipedia.org/w/index.php?oldid=555197663 *Contributors*: Canonperson, Erechtheus, LilHelpa, Nikhilkrgvr, Qwertyus, 2 anonymous edits

Monotonic query *Source*: http://en.wikipedia.org/w/index.php?oldid=592582044 *Contributors*: Bearcat, Jesse V., Mmmjacob1275, PJmdJIm, Rp, Tom Morris, 6 anonymous edits

Recursive join *Source*: http://en.wikipedia.org/w/index.php?oldid=552183561 *Contributors*: AtholM, Danim, Dthomsen8, Falcor84, Malcolma, Rich Farmbrough, Sderose, Unhingedloon

Relvar *Source*: http://en.wikipedia.org/w/index.php?oldid=599192851 *Contributors*: Andreas Kaufmann, AndrewHowse, AndrewWarden, Anypodetos, Apyule, Brick Thrower, Bryant1410, DannyAsher, Elwikipedista, Leandrod, LyricalCat, M. Frederick, Mark Renier, Tijfo098, Tinucherian, Tobias Bergemann, Wikipelli, 14 anonymous edits

Relational calculus *Source*: http://en.wikipedia.org/w/index.php?oldid=541136897 *Contributors*: AutumnSnow, Cdrdata, Elwikipedista, Gregbard, Guppyfinsoup, Jan Hidders, Jim1138, Joieko, Jpbowen, Kku, Leandrod, Lfstevens, Mark Renier, Michael Hardy, Mikeblas, Mindmatrix, Omnipaedista, Opabinia regalis, Pewwer42, Remuel, Robert L Pendleton, Rsrikanth05, SqlPac, TheTito, 26 anonymous edits

Tuple relational calculus *Source*: http://en.wikipedia.org/w/index.php?oldid=599033615 *Contributors*: 19more sphinx elbows, Andre Engels, Banazir, Bunnyhop11, Charles Matthews, ChicXulub, Coffeehood, ColdFeet, DH85868993, Danim, Dirk Beyer, Draicone, Drunken Pirate, Elwikipedista, Gregbard, J.delanoy, Jan Hidders, Jpbowen, Jrdioko, KelvSYC, Khalid hassani, Kosebamse, Lajm, Leafnode, Magister Mathematicae, Mark Renier, Martarius, Maury Markowitz, Mets501, Michael Hardy, Omnipaedista, Ps ttf, RJFJR, Reedy, Shanel, Soobrickay, TPK, Tennin, Topbanana, 28 anonymous edits

Query language *Source*: http://en.wikipedia.org/w/index.php?oldid=585643683 *Contributors*: ASHPvanRenssen, Adarw, Ahoerstemeier, Ahunt, Albert688, Alex Ashdealer, Amin Hashem, AmirMehri, AndrewWarden, BCable, Bacchus123, Beaton1131, BenAveling, Bkonrad, Chtirrell, CxQL, DEng, Danakil, Danim, Davidfstr, Deepugn, Devourer09, Diamondland, ERfan111, Edward, Ehajiyev, Elwikipedista, Frieda, Groovenstein, Grutness, HanielBarbosa, Honys, Ihenriksen, Inverse.chi, IvanLanin, Jay42, Joerg Kurt Wegner, John Vandenberg, John of Reading, Jonathan.mark.lingard, KeyStroke, Kwiki, Larsinio, Logiphile, Manifestation, MarXidad, Mark Arsten, Mark Renier, Markhobley, Mgreenbe, Mhkay, MichaelSpeer, Mild Bill Hiccup, Msnicki, NGC 2736, Nikola Smolenski, Ojigiri, OsamaK, Peter Gulutzan, Retireduser1111, Rfl, SarekOfVulcan, Shekhardtu, Slipstream, Soumyasch, Srandrews, Steven Walling, Svick, Tassedethe, Techno.modus, Throbblefoot, TommyG, Toussaint, Trevor MacInnis, Troels Arvin, Usien6, Valafar, Vanished user qkqknjitkcse45u3, Vmenkov, Wikiolap, Xodlop, ZygmuntKrynicki, 48 anonymous edits

Data Definition Language *Source*: http://en.wikipedia.org/w/index.php?oldid=573407226 *Contributors*: 16@r, Abdull, Academic Challenger, Aitias, Alansohn, Aleenf1, Andrewman327, Ankit Maity, Anna Lincoln, Arctic Kangaroo, Aymatth2, BL, Babbage, Bentogoa, BioStu, Boshomi, Bposert, C628, Casperthegost, Cecilkorik, Chupon, CodeNaked, Cometstyles, Cybersprocket, Danim, Decoy, Dgw, Eags, Elektrik Shoos, Eliazar, Elonka, Emvee, Entropy, Epicgenius, Fresheneesz, Ftiercel, Gcdinsmore, Goplat, GregorB, Heimstern, Hu12, Isotope23, JLaTondre, JakobVoss, Jason Quinn, JoeB, Johnathan29, Juansempere, Katieh5584, Kbrose, KeyStroke, Lectonar, Luna Santin, Mark Renier, Martin451, Materialscientist, Mayur, Mhkay, Mindmatrix, Modster, MrRadioGuy, Nate Silva, NerdyScienceDude, NickGarvey, RHaworth, Riki, Rjwilmsi, Roberticus, Rstinejr, Santoshijd, Shriram, Sjgilbertz, Skew-t, SkyWalker, Snay2, SqlPac, Squids and Chips, Svick, Tobias382, Tomsimtim, Trevorbjork, Tumble, Unyoyega, Vy0123, WOSlinker, WadeSimMiser, WeißNix, Whywhenwhohow, WikHead, Winterst, Wwwwolf, Zanimum, 182 anonymous edits

Varchar *Source*: http://en.wikipedia.org/w/index.php?oldid=575097900 *Contributors*: AlisonW, Anphanax, Brenthale, Elwikipedista, HMSSolent, Jeff Song, Lewissall1, MasKa, Mikeblas, Mr. Vernon, NicM, Octahedron80, Shamesspwns, SueHay, Thumperward, 34 anonymous edits

Data Manipulation Language *Source*: http://en.wikipedia.org/w/index.php?oldid=573407301 *Contributors*: Andy Dingley, Aneeshpu, Booyabazooka, CanisRufus, Chrisahn, Chupon, CodeNaked, Danim, Danlev, Discospinster, Frazzydee, Gilliam, Isotope23, Jasper Deng, Jay, Jcfried, Kbrose, KeyStroke, KnightRider, Markhobley, Minghong, Mr. Wheely Guy, Nevcao, OMouse, Plavozont, RHaworth, Rcsprinter123, Shanes, Sikon, SqlPac, Super48paul, Tavahom, Tobias Bergemann, Tony Fox, Vacation9, Viveksharma474, Vmenkov, 52 anonymous edits

Create, read, update and delete *Source*: http://en.wikipedia.org/w/index.php?oldid=595592212 *Contributors*: 5994995, Absinf, Alex2222, Alvin-cs, Ant, Antonielly, Apokrif, Atki4564, BD2412, Banzaimonkey, Beatupbutterfly, Bovski, CanadianLinuxUser, Chris Purcell, DaGizza, Demonkoryu, Dhartung, Donperk, Dreftymac, Eraldito, FeRD NYC, Fgrinder, Fluffernutter, Fred Bradstadt, Gary, Garylhewitt, Gjs238, Gobbleswoggler, Gogo Dodo, Guoxiao281, Herbee, Iancarter, Intgr, James Harvard, Jarble, Jim1138, Jleedev, Jmkim dot com, Kazvorpal, Kbrose, KeithTyler, Kenyon, Khalid hassani, KnightRider, Korg, Kozuch, L Kensington, Leonus, Lord Zoner, LucQ, Luís Felipe Braga, Lyverbe, Mark McColl, Mark Renier, MarkusStolze, Martnym, Max Terry, Meltingwax, Metal.lunchbox, Mike Blackney, Mikeblas, Mindmatrix, Mr e guest79, Mzajac, Napoleonsacrebleu, NawlinWiki, Nepenthes, OguzOzkeroglu, Pinethicket, Pmcm, RJFJR, Railwayfan2005, Rayngwf, Remuel, Richnice, RickScott, Robert K S, RobertG, Robertbowerman, Sannse, Sawall, Stephan Leeds, Syhon, Techi2ee, Thorwald, Thüringer, Troels Arvin, Unixxx, Vectro, Wesley, Winterst, Zanerock, Zegoma beach, 194 anonymous edits

SQL *Source*: http://en.wikipedia.org/w/index.php?oldid=598309401 *Contributors*: 28421u2232nfenfcenc, 28bytes, 62.253.64.xxx, 64.168.29.xxx, A.R., Aaron Brenneman, Aarthib123, Abuyakl, Acroterion, ActiveSelective, Adashiel, Admrboltz, Adrian.walker, AgentCDE, Ahecht, Ahoerstemeier, Aidan W, Ajcumming, Aka042, Alai, Alalia 17, AlanUS, Alansohn, Alec.korba, Alexius08, Alexrexpvt, AlistairMcMillan, AllanManangan, Allens, Allstarecho, Alon, Alpha 4615, Alterego, Alvin-cs, Amaury, AnandKumria, Anclation, Andr3w, Andre Engels, AndrewWTaylor, Andrewpmk, Andrisi, Andy Dingley, Angela, Anoko moonlight, Anonymous Dissident, Antiqueight, Aou, Apienczy, Arcann, Aresgunther, Arvi, Aseld, Ashley Y, Asix, Asqueella, Avillia, Avé, Az1568, B15nes7, B6nb3k, BD2412, BL, Badseed, Baiji, Banannamal, Barefootguru, Basil.bourque, Beetstra, Behringerdj, Beland, Ben D., Ben-Zin, BenFrantzDale, Beno1000, Bernd vdB, Bevo, Bill Huffman, Bkwillwm, BlindEagle, Blonkm, Bobdc, Bobstay, Bodmerb, Bomazi, Bonadea, Bongwarrior, Booles, Boshomi, Bovineone, Branko, Brassrat70s,

Bricaniwi, Brick Thrower, Bryan Derksen, Bstan72, Bumm13, Bunnyhop11, Burgher, Butterscotch, CALR, CF84, CRGreathouse, Caltas, Caltrop, CambridgeBayWeather, Camw, Can't sleep, clown will eat me, Canterbury Tail, Capnchicken, Carbo1200, Cedar101, Channard, Charles Matthews, Chealer, Cherry Cotton, Chowbok, ChrisGualtieri, Christian75, Chtirrell, Cic, Closedmouth, Coffeehood, Combatentropy, Cometstyles, Compfreak7, Comps, Conversion script, Coolboy1234, Coremayo, Countersubject, Cpiral, Cpl Syx, Craig Stuntz, Crosbiesmith, Crowne, Crysb, Cuchullain, Cuckoosnest, Cwolfsheep, Cww, CyberShadow, Cybercobra, D6, DH85868993, Damian Yerrick, Dan Atkinson, DanBishop, DanP, Danakil, Dancter, Danim, Davep.org, David Nemati, DavidWikiuser, Davidfreesefan23, Davidhorman, Dbtfz, Dcoetzee, DePiep, Decon1985, Decoy, Deepaksogani, DefaultLocale, Deflective, Delikedi, Denelson83, Dennis forbes, DennisWithem, Detjo, Dhies88, Dianelos, DigitalEnthusiast, Dittaeva, Dlohcierekim's sock, Dlyons493, Dmsar, Doctorx0079, Docu, Donner60, DopefishJustin, Download, Dpr, DrJolo, DragonHawk, Dreadstar, Dreftymac, Drgrussell, Dsan, Duncharris, Dysprosia, Długosz, EarthPerson, Eboyjr, Ed Poor, Edward, Edwinludo, Egrabczewski, Ehajiyev, Ehheh, ElectricRay, Elfguy, Elkman, Eloquence, Elwikipedista, Emperorbma, Entoaggie09, Enviroboy, Epbr123, Epicgenius, Escape Orbit, Evildeathmath, Ezeu, Fabius.83, Faizan, Falcor84, Fastily, FatalError, Favonian, Feder raz, FelisLeo, Ferdna, Finalius, Fleurydj, Flyer22, FrancoGG, Francs2000, Frando, Franl, Frap, Frecklefoot, Fred Bradstadt, Frosted14, Ftiercel, Furrykef, Fæ, GHe, Gabrielsroka, Gadfium, Galoubet, Gawat123, Gefstar1, Gennaro Prota, Ghettoblaster, Gil Gamesh, Gilliam, Gioto, GnuDoyng, Goethean, Goki, Gopher292, Goplat, Gosub, GraemeL, Grafen, Graham87, GrandPoohBah, Grauenwolf, Greco-German, Greg Lindahl, GregCovey, Gregbard, GregorB, Grigrim, Ground Zero, Gsapient, Guy Macon, Gzuckier, Hairy Dude, Hakkinen, Hankwang, Hardeeps, Hari, Harlock jds, Harperska, Harry, Haza-w, Herbythyme, Heron, Hippo75, Hirzel, Hongooi, Hu12, Huzarus, Hydrargyrum, Hymek, I am One of Many, Iancarter, Ida Shaw, Ilaydakinalan, Incnis Mrsi, Infojunkie23, Inonit, Intgr, Ioannes Pragensis, Iolar, ItsMeowAnywhere, Ivan Pozdeev, J.delanoy, JCLately, JLaTondre, Jacek79, JackPotte, Jackacon, Jacobnibu, JamesNK, Jamesday, Jan Hidders, Jasper Deng, Jay, Jay-Sebastos, Jaysweet, Jc3s5h, Jdlambert, Jeenware, Jeff G., Jeffq, Jeffrey Smith, Jerome Charles Potts, Jezmck, Jimbo Wales, Jimhark, Jimmytharpe, JnRouvignac, Joaquin008, Joelm, Joeyjojo Junior, John Vandenberg, JohnCD, JohnOwens, Johnmc, Joieko, Jon Awbrey, Jonas AGX, Jonearles, Jonik, Jorge Stolfi, Jpk, Jpvinall, Jschnur, Jsmethers, Jsnx, Julesd, Jusdafax, Just James, Justinkaz, Jwissick, Jzcool, Jzel6201, KD5TVI, Kaimiddleton, Karl-Henner, Kelti, Kesava84, KeyStroke, KingsleyIdehen, Kitch, Kizor, Kku, Klausness, Koavf, Kop, Kovianyo, Kpelt, Kreca, Kruusamägi, Kubra, Kuru, Kvdveer, Kwamikagami, KymFarnik, La comadreja, Lambiam, Lankiveil, Larsinio, Lavarock86, Leandrod, Leave Mr Mitchell alone!, Legacypath, Lfstevens, Liao, Libcub, Littlebluenick, Loadmaster, Logiphile, Lost.goblin, Lotje, Lousyd, Love1710, Loveenatayal, Lowellian, Ltruett, Lucian Sunday, Lukaseder, LuoShengli, M7, MacTed, Macaldo, Mackseem, Magioladitis, Magnus Manske, Mahamoty, Mahanga, Mandarax, Mark Arsten, Mark Renier, Mark T, Mark10011, Markhobley, MartinRinehart, Master2841, Materialscientist, MathiasRav, Mathmo, Matt.smart, MatthewIreland, MaxEnt, Maximaximax, Mblumber, Mdchachi, Mdd, Meh222, MeltBanana, Mendel, Mentifisto, Merphant, Mersperto, Metroking, Mgt88drcr, Mhkay, Michael Hardy, MichaelSpeer, Michele.alessandrini, Mikeblas, Mild Bill Hiccup, Mindmatrix, Minghong, MinorContributor, Miskin, Mitch Ames, MoRsE, Modster, Mogism, Mojodaddy, Moontube, MrOllie, Mt.toth, Mxn, Mzajac, NYKevin, Nasa-verve, Naveen0665, NawlinWiki, Nbarth, Neilc, Netfriend, Neum, NiceGuyEduardo, NickGarvey, Nikola Smolenski, Njbooher, Njol, Nohat, Norm mit, Nricardo, Nsd, Nurg, O.Koslowski, Octahedron80, Od Mishehu, Oli Filth, Olivier Debre, Oljtn, Ondrejk, Opticyclic, Orzetto, Osarius, Osmodiar, Paul Foxworthy, Paul Mackay, Paul.mcjones, Paulmlieberman, Peruvianllama, Peter James, Peter.vanroose, Peterl, Philip Trueman, Philippe Perrault, Piano non troppo, PierreAbbat, Piet Delport, Platypus222, Plcsys, Plrk, Plugwash, Pmcjones, Poepkop, Pogonomyrmex, Poloolop, Possum, Pouya, Power piglet, Powerlord, Pparazorback, Prashanthns, Puffin, Pukkie, Qu1j0t3, Quantum00, Quebec99, Quintote, R'n'B, RBarryYoung, RW Marloe, Racklever, RadioKirk, RainbowOfLight, Randomned, Ravi555, Razorx, Rdeleonp, Rdmil, Rdsmith4, RedWolf, Reedy, Reikon, RevRagnarok, Rfc1394, Rfl, Rhobite, Ribaw1, Riffic, Rjwilmsi, Rklawton, Roadrunner, RobertG, RockMFR, Ronstone, Royote, Rp, Ruakh, Rursus, Ruud Koot, Rwwww, S figueiredo, S.Örvarr.S, SF007, Safay, Sailee5, Salix alba, SallyBoseman, Salvio giuliano, Sam Korn, Sander Säde, Saramar10, SarekOfVulcan, Saric, Savh, Sbose7890, Sbvb, Scanos, Schaef, Schuetzm, SciberDoc, Scs, Seaphoto, Sergioledesma, Severa, Sgkay, Shizhao, Shuipzv3, Siafu, Simetrical, Sinisterstuf, SiobhanHansa, Sippsin, Sjakkalle, Sjc, Skreyola, SkyWalker, Slavering.dog, Smartweb, Smeyerinnewton, Smithhogg, Smu95rp, Somme111, SouthLake, Spectrum266, Spezied, Sql pol, SqlPac, Sstrader, Starbeard, Starius, Stephenb, Steve Alvesteffer, Stevietheman, Stifle, Stratadrake, Subflux, SuezanneC Baskerville, SummerWithMorons, Suranjan91, Svick, Swanyboy2, Syzy, Ta bu shi da yu, Tablizer, Tagtool, Tasc, Tcncv, Tcutcher, Tgeller, The Anome, The Letter J, The Thing That Should Not Be, TheExtruder, Thekaleb, Theone256, Thetorpedodog, Thirtyfootscrew, Thom2729, ThomasStrohmann, Three-quarter-ten, Thumperward, Thunderbritches, TigerShark, Tiha, Tijfo098, TimTay, Timwi, Tizio, Tjmoel, Tobias Bergemann, Todd Vierling, Tomtheeditor, Toreau, Traal, Traxer, Trevorloflin, Trlovejoy, Troels Arvin, TunaSushi, Turnstep, Twsx, Ucanlookitup, Ulf Abrahamsson, Uncle G, Unforgettableid, Unknown W. Brackets, Vanished user qkqknjitkcse45u3, Velella, Ville Oikarinen, Vincenzo.romano, Viridity, Visor, Vitund, Vivek.pandey, Vivio Testarossa, VoluntarySlave, Vonfraginoff, Vrenator, WOSlinker, Wadamja, Wakebrdkid, Washburnmav, Wei.cs, Weialawaga, Wernher, Wesley, West.andrew.g, Weylinp, Widr, WikHead, Wikiklrsc, William Avery, Winchelsea, Wingnut909, Wiretse, Wisq, Wmaheriv, Wragge, Wulfila, X-Fi6, XP1, XSTRIKEx6864, Xdenizen, Xibe, Xjhx001, Yakovsh, Yeneesha, Yeng-Wang-Yeh, Ysangkok, Yzchang, Zhenqinli, Zigger, Zundark, Ævar Arnfjörð Bjarmason, 1669 anonymous edits

SQL-92 *Source*: http://en.wikipedia.org/w/index.php?oldid=585833595 *Contributors*: Bendzh, Bovineone, BradBeattie, CRGreathouse, Carl.bunderson, ChrisGualtieri, DBBell, F, Ghettoblaster, JVz, JakobVoss, Mark Renier, Mdmcginn, Mortense, SqlPac, The Fortunate Unhappy, TheParanoidOne, Tijfo098, Unforgettableid, Wenli, 18 anonymous edits

SQL:1999 *Source*: http://en.wikipedia.org/w/index.php?oldid=593166502 *Contributors*: Abatishchev, Abdull, AllanManangan, BD2412, Baest, CRGreathouse, ColdShine, Dragonfi, Dsimic, DynV, FrankHamersley, LilHelpa, Miracle Pen, NeonMerlin, Sémaphore, The Fortunate Unhappy, Tijfo098, Петър Петров, 6 anonymous edits

SQL:2003 *Source*: http://en.wikipedia.org/w/index.php?oldid=593085273 *Contributors*: Alastairgbrown, Aurelienge, Avé, Bovineone, CRGreathouse, CYCC, ChrisGualtieri, Duster.Cleaner, Friendlydata, Galaxy07, JVz, Jeltz, John Vandenberg, Jowa fan, Mariuz, Mark Renier, Pekaje, SqlPac, Tdedonno, The Fortunate Unhappy, TheParanoidOne, Tijfo098, Touko vk, Wavelength, Ysangkok, 23 anonymous edits

SQL:2008 *Source*: http://en.wikipedia.org/w/index.php?oldid=586865327 *Contributors*: AllanManangan, Allstarecho, AzertyFab, CRGreathouse, ChrisGualtieri, Dfetter, GregorB, Highpush76, Itai, Kku, Klemen Kocjancic, Linan, Pykello, The Fortunate Unhappy, Touko vk, Troels Arvin, 8 anonymous edits

SQL:2011 *Source*: http://en.wikipedia.org/w/index.php?oldid=587859797 *Contributors*: AzertyFab, ChrisGualtieri, Fetterless, GoingBatty, Intgr, John of Reading, Krauss, Lenticel, Peter.vanroose, Tijfo098, Ysangkok, 6 anonymous edits

Data *Source*: http://en.wikipedia.org/w/index.php?oldid=597921689 *Contributors*: 192.75.241.xxx, 208.245.214.xxx, A.M.Lewis, Aadal, Aarktica, Abramjackson, Achaemenes II, Acronymsical, Adam M. Gadomski, Aditya, AdjustShift, Adrian J. Hunter, Ahoerstemeier, Alansohn, Alasdair, Aldie, Aldrichg, Aleenf1, Alexius08, AlistairMcMillan, Amd628, Andrew Levine, AndrewHowse, Angrytoast, Antandrus, Apollo, Arikanari, Aroundthewayboy, Arthena, Arthur Rubin, Astral, Atif.t2, Avoided, Beetstra, Begewe, Beyondthislife, Bhadani, Bhumi thakor, Bikeable, Blake Burba, Blue Hoopy Frood, Bluemask, Bluerasberry, Bobo192, Bomac, Bongwarrior, Boson, Boxplot, Bped1985, Bpilstrom, Bruce1ee, COMPATT, COMPFUNK2, CRGreathouse, Calabe1992, Calvin 1998, Capricorn42, Ccloudies, Cenarium, Christopher Denman, Chuunen Baka, Clenny93, Cnwilliams, Coemgenus, Colincbn, Connectel, Conversion script, Cool moe dee 345, Coracles, Corpx, Corti, Cpiral, Cpl Syx, CustardJack, Cybercobra, DJDunsie, DancingPhilosopher, Daniel Mietchen, Daniel.Cardenas, DanielDeibler, Dar-Ape, Dastoger Bashar, Daven200520, Dawnseeker2000, Dcljr, Decltype, Deejaye6, Deflective, Denisarona, DerHexer, Dfletter, Dhaluza, Dmarquard, Dmccreary, Dmyersturnbull, Donald swager, Donner60, Dontdoit, Dotancohen, Dr. JJenkins, Dreadstar, Dthomsen8, Duncharris, Dungodung, Dusty relic, Duxwing, Dycedarg, EJF, EagleFan, Ec5618, Edgar181, El C, ElKevbo, Elitejoe24, Elonka, Emperorbma, Epbr123, Eric, Espoo, Ewawer, Excirial, Faramir333, Flyer22, Fnlayson, Folder18, Former user, Frozenevolution, Fudgcker33, Funandtrvl, Furrykef, G716, GDibyendu, GKantaris, GMBosman, Gaelen S., Galoubet, Gareth Griffith-Jones, GauravDevD, Gbbinning, Geneb1955, General Wesc, Georgea76, Georgethorne, Giftlite, Glane23, Glenn, Gogo Dodo, Goldendragon2000, Goldwein, Goodnightmush, GrahamN, Gregbard, GregorB, Gsonnenf, Guaka, Gurch, Happysailor, Harej, Hateless, Headbomb, Heimstern, HexaChord, Hierarchivist, HkQwerty, Huntthetroll, Hydrargyrum, IGeMiNix, IanMSpencer, Icseaturtles, Idunno271828, IlyaHaykinson, Incnis Mrsi, Informatwr, Inkwina, Insanity Incarnate, Invincible Ninja, Ioannes Pragensis, J.delanoy, JForget, JHunterJ, JMK, Jack Greenmaven, JamesBWatson, January, Jdlambert, Jeffrey Bosboom, Jensgb, Jeremy1989, Jgallihugh, Jim1138, Jjchingo, Jkl, Joe Kress, Joebeone, Joeblakesley, Joffan, John Reaves, John Schulien, Johnflan, Johnuniq, Jojhutton, Jondel, Jschwa1, Jtneill, KGasso, Kablammo, Kayau, Kb3edk, Kbolino, Kbrose, KellyCoinGuy, Kencf0618, Kenny sh, Kevin Saff, KeyStroke, Kku, Krutikaa, Kubanczyk, Kumudupinto, Kuru, Kurykh, Kushalbiswas777, Kvan, Kwamikagami, Lambiam, Lapsus Linguae, LedgendGamer, Leithp, Lights, Lil.Mizz.Charmz, LilyKitty, Logan, Loupeter, Luccas, MER-C, Machine Elf 1735, Magdalena Szarafin, Mandarax, Marasmusine, Marek69, Mark Arsten, Mark Renier, Materialscientist, MattTM, Mav, Mayumashu, Mayur, Mbubel, Mdd, Me lkjhgfdsa, Meiskam, Mercurydaac, Metricopolus, Michael Hardy, Mike Rosoft, Miss Madeline, Mr Stop 2000-2000-1999 to 2006 versus TCP, Mxn, NHRHS2010, NawlinWiki, Neonback, Netoholic, NewEnglandYankee, Nick, Nigelloring, Niteowlneils, Nuwewsco, O.Koslowski, Oberst, Octahedron80, Onecanadasquarebishopsgate, Oxymoron83, PRF Brown, PTJoshua, Pacifistpanda, Passportguy, Pcj, Pearle, Pengo, PeterSymonds, Petrb, PhilKnight, Philip Trueman, Piano non troppo, Pinethicket, Pontsticill, Pooresd, Prari, Prepare2fire, Qasim malik, Quadell, Quietbritishjim, R3m0t, Radon210, RainbowOfLight, Rajsandhi, RandomP, Ravedave, RayGates, Rdococ, RealityApologist, Red Slash, RedWolf, RenamedUser01302013, RevRagnarok, Rhododendrites, Riana, Rkitko, Robbie andrew, RodC, Romanm, Roozbeh, Rrburke, Sa'y, Saehry, Salvio giuliano, Sanjayanand, Satellizer, Sbfw, Sbwoodside, SchreiberBike, Scott MacLean, Seaphoto, Secretlondon, Serggasp, Sergio.ballestrero, Sesu Prime, Shadowjams, Shipmaster, Sina2, Sjakkalle, Skempe42, Skittleys, Sky Attacker, Sleekgray, Snowded, SoCalSuperEagle, SpaceFlight89, Stepp-Wulf, Stevenson-Perez, Subramanyam.avvaru, Sugsez, Supertouch, Suriel1981, Swtimaginations123, THF, Tahmina.tithi, Tcncv, Teifif, Tgeairn, The Anome, The Rationalist, The wub, Thinktdub, TimBentley, TimVickers, Tin, Tobias Bergemann, Tommy2010, Tompsci, Tracytheta, Trepeche, Triwbe, Trusilver, TubularWorld, TutterMouse, TuukkaH, Tuxedo junction, Twilightloverforev, TwoTwoHello, Venullian, VictorianMutant, Vihelik, Vlmastra, Voxii, WBTtheFROG, Ward20, Wasbeer, Wavelength, Wayne Slam, Webclient101, Welsh, Wereon, WhatisFeelings?, Whooooooknows, Widr, WikidSmaht, Wikipelli, Wile E. Heresiarch, WinterSpw, WojPob, Woohookitty, Wwmbes, XKL, Xhaoz, Xiaopo, Yoshkayoshka, Zero Thrust, Zerokewl, Zipircik, Zzuuzz, ماني, 684 anonymous edits

Metadata *Source*: http://en.wikipedia.org/w/index.php?oldid=598958437 *Contributors*: 0612, 121a0012, 16@r, 210.49.109.xxx, 7, AGK, AGToth, Acdg, Ahoerstemeier, Akhristov, Al Adriance, Alansohn, Albert ip, AlexChurchill, Algocu, AlistairMcMillan, Allen Moore, Allens, Allthingsace, Amr.rs, Andrea Parri, Andrew Gray, Andrew Werdna, AndrewHowse, AndrewRH, Androo, Andy Dingley, AnmaFinotera, Anna Lincoln, Anna Montull, Anniebiogirl, Antandrus, Anthony Borla, Antonielly, Apapadop, Armaced, Arpabr, Artaxiad, Arthena, Avicennasis, BD2412, BackwardsBoy, Balajiveera, Beardo, Beetstra, Belbernard, BenRogers, Benbludget, Beyer, Bhny, Bigpinkthing, Biker JR, BioPupil, Blackphiber, Blitzmut, Boardhead, Btwied, BurntSky, CFilm, CRJO-CRJO, CTZMSC3, Cab88, Canyouhearmenow, Carly805, Carolyn22789, Catgut, Cdc, Ceceliadid, Celticst, Cfwschmidt, Chandan Patel, Charles T. Betz, CharlesC, Chievous, ClaudioFerreira, Cnash11, Coinchon, Cole2, Conversion script, Crimsonmargarine, Crysb, Cureden, DEddy, DGG, DMacks, Dalbon, DanBri, Daniel5127, DarkSaber2k, Daswani.Amit, Dave Hay, David B in Canberra, David Gerard, David Woolley, David.T.Bath, DavidPKendal, Davidryan168, DePiep, Deborah-jl, Dedmonds, Demerara, Denverjeffrey, Devarakondar, Diannaa, Dicklyon, Dmccreary, Do better, Doug Bell, Dramalho, Drrwebber, Dsimic, Dwlegg, ECEstats, Eeheeheeheeheeheehee, El benito, Elf, Elving, Emperorbma, Emre D., EnDumEn, Enfresdezh, Equilibrioception, Eric Blatant, Error, Europrobe, EvenT, Evercat, Everyking, Evil Monkey, Exe, Face.real, Feldermouse, FlamingSilmaril, Flarn2006, Fmccown,

Francs2000, FredMBrown, FreplySpang, Fritzpoll, Fuzheado, GRAHAMUK, GVogeler, Gaius Cornelius, Galoubet, Garion96, Gautam3, Gbevin, Genealbert, GhostGirl, Gioto, Girl2k, GoingBatty, Graham87, Graybeal, Green caterpillar, GregLindahl, GregorB, Guffydrawers, Gurch, Gywst, H10130, Halaster, HarryAlffa, Harryzilber, Heron, Hiplibrarianship, Hmains, Hugonius, Hvn0413, Hvs, IRowlands, Iancarter, Iluvcapra, Inkington, Isiaunia, Itai, J Milburn, J04n, JRR Trollkien, Jacobolus, Jamacfarlane, Jarble, Jasongao99, Jdmbellevue, Jeffrey Mall, Jehochman, Jerome Charles Potts, Jewers, Jgjournalist, Jim1138, Jinzhanguwm, Jleedev, Jmundo, Joejoejoejoejoejoejoejoejoe, John Hubbard, John Vandenberg, JohnRonald, JonHarder, Jonasalmeida, Jordgette, JosebaAbaitua, Jschwa1, Juliancolton, Juro2351, Jusdafax, Just plain Bill, K6ka, Kazrak, Keilana, Kenta, Kevin B12, KeyStroke, Khalid hassani, Khym Chanur, Killiondude, Kim Bruning, Kku, Klower, Kraftlos, Kuru, Kvng, Kylemew, LOL, Lando Calrissian, Lazy Techie, Lcme, LeeHunter, Les733, Lethesl, Lheuer, Ligulem, LinaMishima, Ljagerman, Lotje, Lowellian, Luckyz, Lulu of the Lotus-Eaters, M4gnum0n, MDE, MacGyverMagic, Malyctenar, Mamizou, Manfred-jeu, Mani1, Maork, Marchitelli, Mark Renier, Markhurd, Masgatotkaca, Matthewvelie, Matěj Grabovský, Maurice Carbonaro, Mav, Mboverload, Mdd, MetaWorker, Metajohng, Michael Hardy, MikeLynch, Mild Bill Hiccup, Millmoss, Miss Madeline, Mjb, Modify, Mrbradleyt, Mudkipzss, Mushin, Mwestby828, Mxn, Nadimghaznavi, Natalya, NawlinWiki, Neo-Jay, Nerdonpurpose, Newbyguesses, Nf1234, Nichtich, Nickg, Niduzzi, Night Gyr, Nixdorf, Nopetro, Northgrove, Notinasnaid, Nowa, Nscwikiedt, Nyq, Nyttend, Octahedron80, Ojw, Omassey, Omnipaedista, OnceAlpha, Ondertitel, Ottomachin, Pathoschild, Patrick, PatrickFisher, Paul Asman, Peciv, Peepeefigjam123, PerryTachett, Pete Short, Phantomsteve, Phillipgi, Pigsonthewing, PinkAmpersand, Pinkgirl9595, Pinku.nagpal, Poor Yorick, Poornima vijayan, Prince of Strings, Protonk, Public Menace, Purslane, R'n'B, Rabbit67890, Raffaele Megabyte, Ramu50, Rarcher88, RayGates, RedWolf, ResearchRave, Reswobslc, Rexdeaz, Rhagen7, Richard.decal, Rjwilmsi, Rnathanday, Robth, Ronvelig, Ronz, Rougieux, Rspeer, S. Cruz, S.K., SCEhardt, SEWilco, SFK2, Sallyrenee, Santamoly, Schandi, ScienceGolfFanatic, SebastianHelm, Sgb, Shaddim, Shadowjams, Shanebdavis, Sieuthulin, SilkTork, Sk19842, Skizzik, Sky Attacker, Slacka123, Smalljim, Snaxe920, Soler97, Soliloquial, Somewherepurple, Soocom1, Soumyasch, Spalding, Spdegabrielle, Spropis, SqueakBox, Srobak, StaticGull, Steffclarke, Stephen B Streater, Stevertigo, Stevietheman, Stolkin, Strike Eagle, StuartGilbert, Stuartyeates, Sualeh, Superm401, Syaskin, Synctext, TJRC, TYelliot, TaintedMustard, Tajymoid, Tarquin, The Epopt, TheDude813, Thetawave, Thetimperson, Tikiwont, Timtrent, Tipiac, Tkmi.itu.dk, Tnekevets, Tobias Bergemann, Tokailoverock, Tongbram, Tony1212, Tosahilchopra, Tregonsee, Treybien, Trilliumz, Tverbeek, Typhoon, Udayan.warnekar, Uliwitness, UninvitedCompany, UnitedStatesian, Utcursch, Vegard, Vigilius, Vik-Thor, Vincent jonsson, VitalyTarasov, Volphy, Wafulz, Warren, Whatyousage44, WhisperToMe, Wik, Wikky Horse, Wimmeljan, Wingwalker13, Wireless friend, Woodshed, Writ Keeper, WriterHound, Yerpo, Ylvabarker, Yonkie, ZackMartin, Zundark, Zyb5586, 697 anonymous edits

Table *Source*: http://en.wikipedia.org/w/index.php?oldid=593321831 *Contributors*: 12george1, 16@r, Abdull, Ajraddatz, Alai, AlanS1951, Arcann, AutumnSnow, Blanchardb, Bobgrey89, Bobo192, Bongwarrior, Bruxism, C.Fred, Cbrunschen, Coleeey, Correogsk, Cyfal, DARTH SIDIOUS 2, Danim, Djtgamer101, Dreftymac, Dzlinker, Epbr123, Excirial, FattyMcjimmy, Feder raz, Funnyfarmofdoom, Gurch, IMSoP, IanCarter, J36miles, Jamelan, Jerome Charles Potts, Kdib987, Krishna Vinesh, Larsinio, LeonardoGregianin, Lfstevens, Mandy121, Mark Renier, Materialscientist, Mblumber, Mikeblas, Mikeo, Mindmatrix, Morad86, N0nr3s, Nibs208, Nikuwap, Ofus, Pyfan, Quentar, S.K., Sae1962, Scs, Senator2029, Sietse Snel, SimonP, Sippsin, Sonett72, SqlPac, Stolze, TheParanoidOne, TommyG, Turnstep, Txomin, Versageek, Wellskallison, Widefox, Yug1rt, Zhenqinli, 102 anonymous edits

Column *Source*: http://en.wikipedia.org/w/index.php?oldid=598877966 *Contributors*: AbsoluteFlatness, Arcann, CesarB, CommonsDelinker, Danim, Dreftymac, Frietjes, Fæ, GermanX, Huiren92, Jmabel, KeyStroke, Mark Renier, Mark T, Mzuther, N370, Petrb, RJFJR, Sae1962, Sietse Snel, SqlPac, 石庭豐, 13 anonymous edits

Field *Source*: http://en.wikipedia.org/w/index.php?oldid=584584966 *Contributors*: 16@r, 220 of Borg, Alan012, Alanlevin, Antandrus, Anwar saadat, Arcann, BenTels, Btx40, CWenger, CanisRufus, Ceyockey, Codename Lisa, Cybercobra, Dcoetzee, Dreftymac, Garyzx, Hjal, JeffTan, Luís Felipe Braga, Magioladitis, Mark Arsten, Mark Renier, Michael Hardy, My another account, Niceguyedc, Nuujinn, Patrick, PersOnLine, RadioFan, Sae1962, Sderose, Sjakkalle, Skizzik, Spoon!, Taemyr, TakuyaMurata, Tgeairn, Tizio, Tommy2010, WadeSimMiser, Waltpohl, Widr, Xqsd, 45 anonymous edits

Row *Source*: http://en.wikipedia.org/w/index.php?oldid=593691488 *Contributors*: 2help, Allen3, Asfreeas, CommonsDelinker, D4g0thur, Danim, David H Braun (1964), Flip, GLaDOS, Gail, GermanX, Glacialfox, GregorySmith, Jamespurs, Jerroleth, Jmabel, KKramer, KeyStroke, Liujiang, Mark Renier, Mark T, Mxg75, Mzuther, O.Koslowski, Oyauguru, Pnm, Pol098, Retodon8, Rjd0060, Ronhjones, Shaka one, Sietse Snel, SootySwift, Troels Arvin, Yamamoto Ichiro, 32 anonymous edits

Data type *Source*: http://en.wikipedia.org/w/index.php?oldid=596770015 *Contributors*: AThing, Acrollins, Agrophobe, Alik Kirillovich, Amaury, Andrew J. MacDonald, Andyjsmith, Anghammarad, Antonielly, Arjayay, Armageddon11, Arthena, Arthur Rubin, B4hand, Bender235, Bensin, Benwing, Blanzik, Bookinvestor, Busfault, Canaima, Captain-n00dle, Causa sui, Chealer, Chilvan, Christian75, Cybercobra, DVdm, DanBishop, Danielcreech, David Shay, DeadEyeArrow, Dejawolf, Diannaa, Dreftymac, Duke Ganote, EAderhold, EdC, Eghanvat, Elunah, Elwikipedista, Epicgenius, Esap, Faizan, Falcon8765, Floatjon, Friendlydata, Fyrael, Garyzx, Gaspercat, Gendut, GeorgeBills, Ghettoblaster, Greenstruck, Guoguo12, Halaster, HappyDog, Hardyplants, Hash Dollar, Hmmwhatsthisdo, Hooperbloob, HumphreyW, JCLately, Jb-adder, Jeh, Johnteslade, Kbrose, Khullah, KrakatoaKatie, Krischik, Kwiki, L Kensington, LaMona, Lambiam, Leotohill, Libcub, Logiphile, M4gnum0n, MC10, Maelor, Maian, Manifestation, Mark Renier, Martin Bravenboer, Materialscientist, MattGiuca, Mawode, Mbxp, Mdd, Michal Jurosz, MilerWhite, Millermk, Minimac, Minna Sora no Shita, Mjb, Mr Stephen, MrTsuna20, MusikAnimal, Mxcatania, Myconix, NHSavage, Neelix, Neilc, Neo63, Nibuod, Nightstallion, Omnipaedista, Oroso, Pcap, Peter.C, Peterdjones, Public Menace, R'n'B, RHaworth, Raffaele Megabyte, Rahulghose, Raymondwinn, Reaper Eternal, Renku, Resoru, Rettetast, Robbiemorrison, Ru.va, Ruud Koot, S, S.Örvarr.S, Shell Kinney, Slon02, Slugger, SoWhy, Socrates2008, TakuyaMurata, Technopat, Teji, The Thing That Should Not Be, Thecheesykid, Thincat, Tide rolls, Tobias Bergemann, Tokigun, Triwbe, Trlovejoy, Ttwaring, TutterMouse, TuukkaH, WereSpielChequers, Wickorama, Widr, Wikiji, Wimt, 304 anonymous edits

Select *Source*: http://en.wikipedia.org/w/index.php?oldid=596469701 *Contributors*: 2aprilboy, Aavindraa, Ahoerstemeier, Ajmas, Andr3w, Arcann, Avé, Berny68, Bovineone, Cedar101, Centrx, Chester Markel, CodeNaked, Cww, Da404lewzer, Dancter, Danielluyo, Data64, Dp462090, Dparvin, Ed Poor, EvanCarroll, Faradayplank, Gadfium, GargoyleMT, Graden, GregorB, GreyCat, Gurch, Helix84, Isotope23, JLaTondre, JaGa, Jay, John Vandenberg, Jon CIT694, Jpmague, Jpo, Julien.palard, Kayvee, Ketiltrout, LHOON, Larsinio, LinguistAtLarge, Locke Cole, Lowellian, Maashatra11, Macrakis, Mais oui!, Mark Renier, MarkusWinand, Materialscientist, Mikeblas, Mild Bill Hiccup, Mindmatrix, Mormegil, MrOllie, Musiphil, Mxn, Ngpd, Nic Waller, Niketanp, Octahedron80, OdiProfanum, PaD, Paul Pogonyshev, Piano non troppo, Plrk, Randycjones, Resilvajr77, S.K., Shoone, SpeedyGonsales, Spetzgrad, SqlPac, Sqlinfo, Stolze, Svick, Tedmund, TerriersFan, The Fortunate Unhappy, Tomhubbard, Troels Arvin, Twsx, Unforgiven24, Unknown W. Brackets, Victor falk, WOSlinker, Wgillett, WikHead, Wmenton, Wwphx, Xjhx001, Yug1rt, 140 anonymous edits

Result set *Source*: http://en.wikipedia.org/w/index.php?oldid=546486497 *Contributors*: Abdull, Charles Matthews, Dawynn, Junkyardprince, Mikeblas, Stolze, The Anome, Welsh, 2 anonymous edits

Synonym (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=597201989 *Contributors*: Abdull, AutumnSnow, ChrisGualtieri, Cptrwizd1990, Eeekster, EuroCarGT, Frze, Jerome Charles Potts, Jjoeknow, Kal 9371, Nlapierre, Santoshijd, Waldir, 11 anonymous edits

Alias (SQL) *Source*: http://en.wikipedia.org/w/index.php?oldid=578203131 *Contributors*: Aavindraa, Abdull, AllyD, ChrisGualtieri, Jeepday, Racklever, Shukla.rohit09, 4 anonymous edits

Insert *Source*: http://en.wikipedia.org/w/index.php?oldid=591563644 *Contributors*: Adanbrown, Ahoerstemeier, Alvin-cs, Amarsir, Antandrus, Ben2k9, Blcm, Bshow, Cedar101, Chester Markel, ChrisGualtieri, Dancter, DaveChild, DeadEyeArrow, Dparvin, Drake Wilson, Dze27, ESkog, Ed Poor, Edward, Flyer22, Frap, Ftiercel, Fzzzy, GSwarthout, Gadfium, GargoyleMT, Ginsengbomb, GregorB, GreyCat, Griffinofwales, Iamavandalizer, Isotope23, JLaTondre, Jason Quinn, Jercos, Julien.palard, Kingmotley, Larsinio, LinguistAtLarge, Mark Renier, Melonkelon, Mikeblas, Mindmatrix, Mnb20, Mounaam, MrOllie, Neilc, Nogoodshitname, PauloCalipari, Pnieloud, Pope on a Rope, Reswobslc, Riki, Rjwilmsi, Slackwise, SqlPac, Stolze, Svick, TeunSpaans, The Fortunate Unhappy, Troels Arvin, Tsijun, Unknown W. Brackets, Utkwes, Yug1rt, 116 anonymous edits

Update *Source*: http://en.wikipedia.org/w/index.php?oldid=590943276 *Contributors*: Abdull, Amux, Back ache, Cedar101, CodeNaked, Dbolton, Eugene-elgato, Gadfium, GargoyleMT, GregorB, GreyCat, Hmzabeeh62, Ipeattie, Isotope23, Kakoui, Larsinio, LinguistAtLarge, Mark Renier, Mikeblas, Mild Bill Hiccup, Mindmatrix, Nic Waller, PauloCalipari, Puffin, Reedy, S.K., Sean.hoyland, ShaunOfTheLive, SqlPac, Stolze, Sumail, The Fortunate Unhappy, Timc, Troels Arvin, Uncle Milty, Xaje, Yug1rt, Zain Ebrahim111, 69 anonymous edits

Merge *Source*: http://en.wikipedia.org/w/index.php?oldid=592473649 *Contributors*: Abdull, Az1568, Bovineone, Bunnyhop11, Burtonator, Bwperrin, Cedar101, Chuunen Baka, Cowtowncoder, Edam, Ftiercel, Greenrd, GreyCat, Hmrox, Krauss, Magioladitis, Mark Renier, Mecanismo, Mikeblas, MrOllie, Patriotic dissent, Reedy, Rgauf, Samdorr, Scrooll, SqlPac, Sqlinfo, Stolze, The Fortunate Unhappy, Tijfo098, Unordained, Xaje, Xenodevil, Ysangkok, 39 anonymous edits

Delete *Source*: http://en.wikipedia.org/w/index.php?oldid=540405972 *Contributors*: Aaronmatthews, Abdull, Anakin101, Bongwarrior, Bruce1ee, Candid Dauth, Chick Bowen, Closedmouth, Dancter, Fnielsen, Frap, GargoyleMT, GregorB, GreyCat, Intgr, Isotope23, JNMofMV, Jason Quinn, Jewetnitg, Jwoodger, Kaimiddleton, Kratos 84, LHOON, Larsinio, Lectonar, LinguistAtLarge, MBisanz, MJGR, Mary*wu, Mikeblas, Mindmatrix, Nabilla00, Nic Waller, OdiProfanum, Slon02, Some jerk on the Internet, SqlPac, The Fortunate Unhappy, Troels Arvin, Turnstep, Unixxx, Wavelength, Yug1rt, 69 anonymous edits

Join *Source*: http://en.wikipedia.org/w/index.php?oldid=598513898 *Contributors*: 28421u2232nfenfcenc, 28bytes, Abdull, Adamrmoss, Addshore, Ajgorhoe, Ajraddatz, Alansohn, Alessandro57, Alexey Izbyshev, Alexius08, Algorithm, AlistairMcMillan, Altaïr, Amniarix, Andr3w, AndrewN, AndrewWTaylor, Andy Dingley, Angryxpeh, Aranel, Arcann, Azazyel, Back ache, Banazir, BenFrantzDale, Benatkin, Bevo, Bilby, BitPoet, Bluezy, Bobnewstadt, Bogdanmionescu, BrandonCsSanders, Bunnyhop11, Cabe6403, Caltas, Can't sleep, clown will eat me, Canterbury Tail, CardinalDan, Cbuckley, Cedar101, Cherkash, Chris55, Chrismacgr, Church of emacs, Chzz, Cintari, Cmrudolph, Conrad.Irwin, Constantine Kon, CousinJohn, Crypticstargate, Cynthia Blue, DARTH SIDIOUS 2, DBBell, DBigXray, DRAGON BOOSTER, Damian Yerrick, Dan Forward, David Eppstein, David.m.needham, Davidjhp, DeRien, Decrease789, Defenestrate, Dhdawe, Dobi, Dparvin, DruidZ, Dtwong, EJSawyer, ESkog, Ed Poor, Ed g2s, Edward, Elektrik Shoos, Elwikipedista, Erwin, Esofomuso, Faithlessthewonderboy, Flyingcheese, FourtySix&Two, Fox2k11, Frap, Fred Bradstadt, Fremsoft, Friedo, Fundamentisto, Futurix, Gadfium, Gail, GeorgeVarghese12, Ghiraddje, Gilliam, Gintsp, Glaisher, Glane23, Gmacar, Goethean, Gogo Dodo, Golbez, GrayFullbuster, GregorB, Gwandoya, Haymaker, Hdt83, Hebrews412, Herbythyme, HorsePunchKid, Hpetya, Hsdav, Hu12, Hudson2013, Insanephantom, Io Katai, Iridescence, Iridiumcao, Isotope23, Itmozart, JCLately, Jatin.ramanathan, Jeepday, Jlhollin, Johannes Simon, Jpatokal, Julesd, Juliano, JustinRosenstein, Jwalantsoneji, Jwoodger, Kaelar, Kayau,

Kazvorpal, Khazar2, Kingmotley, Kite07712, Klausness, Krassonkel, Lambiam, Landon1980, Larsinio, LateToTheGame, Laug, Leeannedy, Lemming, LeoHeska, Likejune, Loren.wilton, Lousyd, Lucio, Lyonzy90, MER-C, Mandarax, MarchHare, Mark Renier, Markhurd, Martinvie, Mate2code, Materialscientist, MaxMahem, Mbarbier, Mbloore, Mckaysalisbury, MelbourneStar, Mentifisto, Mfyuce, MiguelM, Mike.lifeguard, Mike929t, Mikeblas, Mild Bill Hiccup, Mindmatrix, Mitar, Mmichaelc, Mojo Hand, Moogwrench, MrOllie, Murphydactyl, Musiphil, Mwtoews, NFD9001, Nbarth, Neilc, NewEnglandYankee, Newtman, Nichtich, Nirion, Nithinhere, Noitidart, Nricardo, Oddbodz, OdiProfanum, Oliverlyc, Omicronpersei8, OracleGuy, Orange Suede Sofa, OrangeDog, Oshah, OverlordQ, Palosirkka, Pasteurizer, Paulwehr, Pedant17, Perijove, Pgan002, Philip Trueman, PhilipMW, Pinaldave, Plustgarten, Pne, Pnm, Possum, Prakash Nadkarni, PseudoSudo, Pseudomonas, Quebec99, Qviri, Rahi1234, Ramanna.Sathyanarayana, Raztus, Rburhum, Redhanker, Reedy, Rjohnson84, Rocketrod1960, Roga Danar, RokerHRO, RoyGoldsmith, Russellsim, Sam Allison, Santhoshseeks, Saravananganesh, SarekOfVulcan, Scolebourne, Seaphoto, Shadowjams, Sharon.delarosa, Shinmawa, Silvaran, SimonP, Sippsin, Smjg, Smtchahal, Soluch, SpK, Spitfire8520, Spropis, SpuriousQ, SqlPac, Sreecanthr, Ssavelan, St33lbird, Stevecudmore, Steven Zhang, Stolze, Storkk, Subversive.sound, SynergyBlades, Ta bu shi da yu, TangentCube, Taral, Tbsdy lives, Tfischer, The Fortunate Unhappy, The Thing That Should Not Be, Thekaleb, Thekingfofa, Thumperward, Tijfo098, Timflutre, Titusjan, Tjphall, Troels Arvin, Tuntable, Tweisbach, Unknown W. Brackets, Unschool, Urhixidur, Vikashrajan, Viriditas, Vsotnikov, Wasell, Wavelength, Widefox, WinContro, Woohookitty, WouterBolsterlee, X96lee15, Xanderiel, Yangshuai, Ysangkok, Zanemoody, 1188 anonymous edits

Set operations *Source*: http://en.wikipedia.org/w/index.php?oldid=591124367 *Contributors*: AlanBarrett, Cmweiss, CodeNaked, Dcoetzee, Epbr123, Ftiercel, Gintsp, Googl, GregorB, Isotope23, John Cline, Kaimiddleton, Larsinio, LinguistAtLarge, Mark Renier, Mikeblas, Mindmatrix, MrOllie, Noformation, QNeX, Racklever, Reedy, S.K., SqlPac, Stolze, The Fortunate Unhappy, 47 anonymous edits

Commit *Source*: http://en.wikipedia.org/w/index.php?oldid=540490661 *Contributors*: Ajgorhoe, Amalas, Arashb31, Bhny, Donovan Heinrich, G7yunghi, Gonchibolso12, Gpvos, Gtrmp, JCLately, Jthiesen, Lmalecki, Mark Renier, Michael Slone, Mikeblas, Nabla, Radagast83, RedWolf, RockMFR, Schapel, SqlPac, Stevertigo, Yuvysingh, Zucchinidreams, 25 anonymous edits

Rollback *Source*: http://en.wikipedia.org/w/index.php?oldid=596932342 *Contributors*: Alexius08, Apokrif, Brick Thrower, Craig Stuntz, Emperorbma, Feinoha, Gadfium, Ibrahim Husain Meraj, J.s.banger, JCLately, JonHarder, Khalid hassani, M4gnum0n, Mark Renier, Mattisse, Mediran, Mikeblas, Mindmatrix, Oxymoron83, Pegship, PierceG, Piotrus, Pne, Rfl, SimonP, Turnstep, Unknown W. Brackets, Vedant, Widr, Wikiklrsc, 28 anonymous edits

Truncate *Source*: http://en.wikipedia.org/w/index.php?oldid=569291165 *Contributors*: Abdull, BeSeeingYou, Bgwhite, BlackSmith.Fi, Bunnyhop11, Chrisleonard, Er sachinagarwal, Ftiercel, GargoyleMT, GreyCat, Igno2, Isotope23, Jay Johnston, Jidanni, Larsinio, LinguistAtLarge, Mark Renier, Mikeblas, Mysdaao, Nathanator, NewWikiMan, Nicopedia, NotAnonymous0, PaD, QNeX, SqlPac, Stangel, The Fortunate Unhappy, Troels Arvin, Zawersh, 26 anonymous edits

View (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=563578355 *Contributors*: Abdull, Acjelen, Alai, Andrew.george.hammond, Anthony Appleyard, Blowdart, Boson, BullRunner2009, Cedar101, Christian75, ClementSeveillac, Dcoetzee, Dfrg.msc, Edwardzhu, Elcasc, Ewebxml, Excirial, FernandoAires, Fragment, Galador, JDHeinzmann, JForget, JLaTondre, JeepdaySock, Jerome Charles Potts, Joaquin008, Jobbin, Jtgerman, Jwoodger, Kibbled bits, Kku, Kuru, Larsinio, Lythic, Mark Renier, Materialscientist, Mathmo, Matinict, Mikeblas, Mindmatrix, MrOllie, Muchium, Nothings, Quentar, Quuxplusone, RaBa, Raeky, Rfl, Ricardo rivaldo, Rjwilmsi, Roux, Rsrikanth05, Ruud Koot, S.K., Sappy, Scrool, Sippsin, Smalljim, Stolze, TheDamian, TommyG, Toyota prius 2, Troels Arvin, UncleDouggie, WmLGann, Woohookitty, Wwphx, Z.E.R.O., Zerodeux, Zhenqinli, 148 anonymous edits

Materialized view *Source*: http://en.wikipedia.org/w/index.php?oldid=592406928 *Contributors*: Alai, Alan Canon, Andreas Kaufmann, Arunyadav007, AutumnSnow, Bakert, Brick Thrower, Cedar101, Centrx, Choess, Cowb0y, Cpollach, Czarkoff, Damian Yerrick, Dcoetzee, DragonLord, Frap, Hopkinsju, Jandalhandler, Jerome Charles Potts, JnRouvignac, Kapitikaka, Levin, Mark Renier, Materialscientist, Mikeblas, Millermk, Mr. Credible, Neilc, Remy B, Santoshijd, Sippsin, Svilenv, Timneu22, William Avery, Woodshed, Yjacolin, 47 anonymous edits

Hierarchical query *Source*: http://en.wikipedia.org/w/index.php?oldid=521802527 *Contributors*: Aavindraa, Abolen, Cedar101, Ctxppc, Dfgriggs, Ego White Tray, GregorB, Hertzsprung, Jianhui67, John Vandenberg, Kweetal, Loudenvier, Márcio Mazza, QNeX, Rimio, Sippsin, Smjg, Sobreira, Tijfo098, Trappist the monk, Troels Arvin, 16 anonymous edits

Hint (SQL) *Source*: http://en.wikipedia.org/w/index.php?oldid=569276145 *Contributors*: JLaTondre, Jandalhandler, Johnrussell13, Midom, Miyagawa, Nikola Smolenski, Ryoga Godai, Yms, 3 anonymous edits

SQL/CLI *Source*: http://en.wikipedia.org/w/index.php?oldid=544833061 *Contributors*: BD2412, SqlPac, Touko vk, Ysangkok, 2 anonymous edits

SQL/JRT *Source*: http://en.wikipedia.org/w/index.php?oldid=579411932 *Contributors*: BD2412, Download, Mark Renier, SqlPac, Tijfo098, Touko vk, Ysangkok, 1 anonymous edits

SQL/MED *Source*: http://en.wikipedia.org/w/index.php?oldid=544833077 *Contributors*: BD2412, Fundou, Intgr, Piotrek0, SqlPac, Touko vk, Ysangkok, 6 anonymous edits

SQL/OLB *Source*: http://en.wikipedia.org/w/index.php?oldid=586092570 *Contributors*: Jackie, Kelti, Lukaseder, SqlPac, Tijfo098, Touko vk, Ysangkok, 2 anonymous edits

SQL/PSM *Source*: http://en.wikipedia.org/w/index.php?oldid=584226902 *Contributors*: Abdull, Greenrd, GregorB, Jdlambert, Jesse Viviano, Josve05a, RossPatterson, SqlPac, Tijfo098, Ysangkok, 10 anonymous edits

SQL/Schemata *Source*: http://en.wikipedia.org/w/index.php?oldid=544833116 *Contributors*: Abdull, Edward, Friendlydata, JakobVoss, SqlPac, Touko vk, Troels Arvin, Ysangkok, 1 anonymous edits

StreamSQL *Source*: http://en.wikipedia.org/w/index.php?oldid=532085618 *Contributors*: Davnor, DeanMarkTaylor, Denisarona, Dlyons493, FF2010, Hu12, Kimbly, Kotepho, Lmatt, Neilc, Osmouk, R'n'B, RayGates, Sbmarketing, Tibbetts2c, 9 anonymous edits

DBase *Source*: http://en.wikipedia.org/w/index.php?oldid=595728618 *Contributors*: ABLsaurusRex, Ahoerstemeier, Alacevic, Alan.hawk, AlanM1, Albanaco, Aldie, Alexf, Andrzej P. Wozniak, Arcann, Arny, Astatine211, Awg1010, BD2412, Barefootguru, Bemoeial, Bigown, Blaxthos, BobStepno, Bovineone, Bubba73, Bulwersator, Cander0000, CatharticMoment, Cdc, Cedar101, CesarB, Chowbok, Chris Roy, Chris the speller, Chriscf, Christian List, Clint9kc, Cyrius, David Delony, Dbasellc, Dde0apb, EJSawyer, Echion2, Edward, Ego White Tray, Emallove, Erik Bachmann, Ernyd, Frecklefoot, Gcm, Geeoharee, GerardKeating, GermanX, Giraffedata, GroveGuy, Gschizas, HDCase, Hibase, IMSoP, Japo, Jesster79, Jimgawn, John of Reading, JohnLHawkins, Jojalozzo, Josephf, Jost Riedel, Kaihsu, Kawasemi, Kelly Martin, Kevin.r.marshall, Khazar2, Kmund, KnightRider, Kozuch, Kristjan.Jonasson, Kuru, Kwiki, Leviathan, Lexie 40, Liftarn, MER-C, Magioladitis, Majestic27, Manassehkatz, Mark Renier, Maroux, Martarius, Matthewdunsdon, Maury Markowitz, Maxsiseditor, MeekMark, Mhedblom, Mika1h, Mk*, Mkaydatabi, Mrozlog, Octolad, OlEnglish, Pamar, Paul Stansifer, Peter Boughton, Peterl, PhilHibbs, Pol098, Qwertyus, RScheiber, Ramalho, Rboatright, Recital, RedWolf, Redd Foxx 1991, Reedy, Rich Farmbrough, Root 42, Sandwiches99, SarekOfVulcan, Scatteredpixels, Sdteffen, Sergios2, Shlomital, Sreejithk2000, Stan Brin, Sull900, Surv1v4l1st, Tablizer, Tannin, Taviso, Theopolisme, Timc, TubularWorld, Unimath, UrsaFoot, Vanished user kjij32ro9j4tkse, Verdatum, Vk2cz, Welsh, Wernher, Will Nitschke, Xcasejet, Yworo, Zenohockey, Zer0431, Zoicon5, Δ, 에멜무지로, 221 anonymous edits

Clip (compiler) *Source*: http://en.wikipedia.org/w/index.php?oldid=561487502 *Contributors*: Andrzej P. Wozniak, Bigown, BillWSmithJr, CesarB, Gioto, Gschizas, JLaTondre, Magioladitis, Mnemoc, Museo8bits, Pegship, Ruud Koot, Samw, 6 anonymous edits

Clipper (programming language) *Source*: http://en.wikipedia.org/w/index.php?oldid=598322901 *Contributors*: Aa5779, Aivosto, Alkresin, Andrzej P. Wozniak, Anthonycarrabino, Balek, Bigown, Billinghurst, Bogdangiusca, Bovineone, Cander0000, Chester br, Choster, Danakil, DataWraith, Davep.org, David Jordan, Davidmaxwaterman, Dralison, Dysprosia, EdgeOfEpsilon, Edward, Ekjon Lok, EmilioSilva, Epolk, Everyking, GerardKeating, Grafxsoft, Graham87, Guidod, Hmains, I601, Iaen, John Vandenberg, John of Reading, Johnmperry, Jpfagerback, Jvhertum, LiDaobing, Luckas Blade, Maxsiseditor, Mazarin07, Mbelgrano, McGeddon, MetaManFromTomorrow, Mnemoc, Notbyworks, RTC, Recital, Reisio, Ricvelozo, Ruud Koot, SarekOfVulcan, Satori, Scott McNay, Stevietheman, Tannin, Thumperward, Ubiquity, Unother, Wernher, 93 anonymous edits

Flagship compiler *Source*: http://en.wikipedia.org/w/index.php?oldid=580425851 *Contributors*: Balek, Bigown, Cander0000, Cedar101, CommonsDelinker, EagleFan, Frap, Gadfium, GerardKeating, Rjwilmsi, Surv1v4l1st, Tobias Bergemann, Woohookitty, 12 anonymous edits

FoxPro 2 *Source*: http://en.wikipedia.org/w/index.php?oldid=467937278 *Contributors*: Al Lemos, Alexf, Andrewman327, ArglebargleIV, BillWSmithJr, CRGreathouse, Cander0000, ChrisGualtieri, Davemc50, Evanhaas1111, Fragglet, Georgeryp, Gioto, GoForMoe, GoingBatty, GregorB, Guffydrawers, Gurch, Hu12, Igor2004, Jamestaylor, Josi.ow, Logan, MaGioZal, Magioladitis, Mazin07, Mmerlinn, Museo8bits, Neustradamus, Psychonaut, Recital, RingtailedFox, SarekOfVulcan, Shoowak, Superp, Thumperward, Wavelength, WideArc, Wwardw, 53 anonymous edits

Harbour (software) *Source*: http://en.wikipedia.org/w/index.php?oldid=588501033 *Contributors*: Alex degarate, Alkresin, Andrzej P. Wozniak, Arny, Bigown, Billinghurst, Casablanca2000in, Cedar101, CommonsDelinker, Cybercobra, Daniel.Cardenas, Danim, Davep.org, Dl2000, EoGuy, Everyking, Falcor84, Fsgiudice, Gadfium, Gioto, Glacialfox, Glenn, Gongshow, I like Burke's Peerage, Jesse V., John of Reading, Mahmoud Fayed, Maury Markowitz, Mbelgrano, McGeddon, Memories of lost time, Mmreich, Msfclipper, Muhandes, Museo8bits, Nick Number, Ohconfucius, Pegship, R'n'B, Reiknir, Rjwilmsi, Ronchristie, Ruud Koot, Schmloof, SchreiberBike, Sourov0000, Spidermario, Surv1v4l1st, The Banner Turbo, Thumperward, Wavelength, Yworo, 112 anonymous edits

Visual FoxPro *Source*: http://en.wikipedia.org/w/index.php?oldid=598531551 *Contributors*: Abhkum, Aivosto, Albanaco, Ale jrb, Alexf, Andre Engels, Andrzej P. Wozniak, AnimalFriend, ArmadilloFromHell, Aspnetajax, Audriusa, Auntof6, Avoided, AxelBoldt, Balrog, Bigown, BillWSmithJr, Blanchardb, Brat32, CBM, CRGreathouse, Cander0000, Captain Trips, Cdw1952, Charvex, Codename Lisa, CraigBailey, Danakil, Dkalweit, EEMIV, EdBever, Edleafe, Ekjon Lok, Elwhitney, Ericsalim, Eurleif, Faisal.akeel, Fazilati, Funke, Galaad2, GeorgeMoney, Gioto, Gombang, Greenrd, Grstain, Gudeldar, Haggisbingo, Halleyscomet, HedgeHog, Hilmarz, Icseaturtles, Igor2004, Imroy, Introvert, JCLately, Jasabella, Jazz2013, Jeltz, JesseHogan, Joshuamai, Jutiphan, Jvhertum, JzG, Kawasemi, Kentcurtis, Kesla, Kuru, Kypr8, Larryq34, Lexie 40, Lovelac7, Luna Santin, Magioladitis, Mark Miller, Mathmo, Maury Markowitz, Mellery, Mikaey, Mikeblas, Mikenolte, Mlaffs, Mmerlinn, Modster, Mortense, Mxn, Nazgul02, Norm mit, Notbyworks, Notinthisworld, O.Koslowski, Pol098, Pro bug catcher, Rboatright, Recital, RedWolf, Rexmorgan, Rich Farmbrough, Rrabins, Rsocol, Ruud Koot, Ryan2845, RyanNerd, Safety Cap, Samt3, SarekOfVulcan, Shizhao, Soumyasch, StevenBlack, Stevesawyer, Stovetopcookies, Surv1v4l1st, Ta bu shi da yu, TamarGranor, Thalter, TheCatalyst31, Thingg, Thumperward, Tiffer, ToddClausen, ToolmakerSteve, Topbanana, Ubzy, VX, Vespina, VictorEspina, Waza, WideArc, Widr, Wik, Wikipelli, William Allen Simpson, Wizardman, Zaiken, 313 anonymous edits

Visual Objects *Source*: http://en.wikipedia.org/w/index.php?oldid=590059815 *Contributors*: AndrewHowse, Bigown, Bollerpedia, Curb Safe Charmer, DNewhall, Danakil, Davep.org, Gadfium, Grafxsoft, Michael32710, PJonDevelopment, Philip Trueman, RTC, Sam Hocevar, Siroxo, WhiteOak2006, 14 anonymous edits

XBase *Source*: http://en.wikipedia.org/w/index.php?oldid=598321674 *Contributors*: Andrzej P. Wozniak, Anthonycarrabino, Astatine211, Balek, Bigown, BigrTex, Bryan Derksen, CRGreathouse, Can't sleep, clown will eat me, Cander0000, Chavamf, Christian List, DNewhall, Erik Bachmann, Eumolpo, Everyking, GerardKeating, GoingBatty, Grafxsoft, Hibase, Jamelan, Jayden54, Khazar2, Kuru, Lexie 40, Luís Felipe Braga, Maxsiseditor, MeekMark, Mhedblom, Michael B. Trausch, Minghong, Mnemoc, Nasnema, Otermin, Rboatright, Richardinsydney, Ron Pinkas, Seqos, Sergios2, Sftd, Sketchmoose, Squids and Chips, StevenBlack, Ta bu shi da yu, Yworo, 78 anonymous edits

XBase++ *Source*: http://en.wikipedia.org/w/index.php?oldid=572430431 *Contributors*: Andrzej P. Wozniak, Bigown, CRGreathouse, Cander0000, Der Messer, DutiesAtHand, J04n, Otermin, PlasmaDragon, TheParanoidOne, TheRingess, Tide rolls, °, 22 anonymous edits

XHarbour *Source*: http://en.wikipedia.org/w/index.php?oldid=575340584 *Contributors*: Andrzej P. Wozniak, ArglebargleIV, Barticus88, Bigown, Brandon, CRGreathouse, Cedar101, Crystallina, CyberSkull, Cybercobra, Danim, DavidWBrooks, Dd75sg, Dewritech, DouglasGreen, Dpv, Eadmund, Evil Monkey, Ewlyahoocom, Fsgiudice, G7game, Graham87, Jeff3000, John of Reading, Kbrose, KnightRider, Lionel Elie Mamane, Lsuff, Mikeblas, Ospalh, PhilippWeissenbacher, Ron Pinkas, Sam Hocevar, Sfan00 IMG, Squids and Chips, Stewartadcock, Turnstep, Zahid Abdassabur, Алексей Густов, 85 anonymous edits

QUEL query languages *Source*: http://en.wikipedia.org/w/index.php?oldid=580732132 *Contributors*: Alexkon, AutumnSnow, Cathy Linton, Cedar101, DragonLord, Eaefremov, Ebersphi, Elwikipedista, Intgr, Jay, Jeenware, JmA, Jwoodger, Lfstevens, Lowellian, MaD70, Maury Markowitz, Mcrute, Mikaey, Neilc, NuclearWizard, Plvekamp, Romatt, SqlPac, Troels Arvin, Waggers, Wavelength, William Avery, 22 anonymous edits

Query by Example *Source*: http://en.wikipedia.org/w/index.php?oldid=594018429 *Contributors*: Aecis, Apokrif, AutumnSnow, Bulwersator, CBDunkerson, Calvin199, Carlos23diaz, Cdrdata, Cokoli, Cybercobra, DXBari, Danim, DeadEyeArrow, Derbeth, Diego Moya, Georgewilliamherbert, Habbie, JForget, JnRouvignac, Khazar2, Mark Renier, Mblumber, Norm mit, Pnm, Prakash Nadkarni, Ruud Koot, SabbeRubbish, Soumyasch, Ta bu shi da yu, Tablizer, That Guy, From That Show!, Tijfo098, Tmurray-kchisholm, Torzsmokus, 40 anonymous edits

SQR *Source*: http://en.wikipedia.org/w/index.php?oldid=595037459 *Contributors*: AndrewHowse, Avalon, Avoided, Bovineone, ChrisGualtieri, Dhconsult, Dsmccalmont, Frze, JLaTondre, Kungming2, Last digit of pi, Leandrod, Maimai009, Manish.c, Natesgate, Noisy, Novasource, Rl, Sjoerddejong, Srinivas48k, TheParanoidOne, 41 anonymous edits

.QL *Source*: http://en.wikipedia.org/w/index.php?oldid=530119649 *Contributors*: Alksentrs, Blaisorblade, Cander0000, Cesium 133, Chowbok, Ehajiyev, Henning Makholm, JLaTondre, Kephir, Kwamikagami, Nemti, QuentinUK, Zetawoof, 2 anonymous edits

Yahoo! query language *Source*: http://en.wikipedia.org/w/index.php?oldid=528731497 *Contributors*: AtticusX, Diego Moya, Followmarko, Markandey, OsamaK, Ottawahitech, PeregrinoGris, Ronz, Steven Walling, Tuankiet65, Woohookitty, Zundark, 10 anonymous edits

YQL (programming language) *Source*: http://en.wikipedia.org/w/index.php?oldid=523482415 *Contributors*: AtticusX, Diego Moya, Followmarko, Markandey, OsamaK, Ottawahitech, PeregrinoGris, Ronz, Steven Walling, Tuankiet65, Woohookitty, Zundark, 10 anonymous edits

YANG *Source*: http://en.wikipedia.org/w/index.php?oldid=583349766 *Contributors*: Brandon, Bubbles.way, Cmoberg, Curtbeckmann, Diogenes00, Hiro42, Jac16888, Joelja, Lockley, MLauba, MatjazVrecko, Mogism, Screwgoth, SoWhy, Vishwavasu, 8 anonymous edits

WQL *Source*: http://en.wikipedia.org/w/index.php?oldid=575542715 *Contributors*: Bryan Derksen, Drclue, Ghettoblaster, Mindmatrix, Omnipaedista, Rypcord, Tarekradi, TubularWorld, Warren, 5 anonymous edits

Versa (query language) *Source*: http://en.wikipedia.org/w/index.php?oldid=544112603 *Contributors*: A5b, Chimezieogbuji, Fleminra, Henning Makholm, Ian Spackman, James McStub, Jerryobject, John Vandenberg, Keryst, KingsleyIdehen, MeltBanana, Mjb, Pmc, Runtime, Scott, Zundark, 1 anonymous edits

SPARQL *Source*: http://en.wikipedia.org/w/index.php?oldid=595769941 *Contributors*: A5b, Adrian.walker, Afreet, Akampman, Akuckartz, AndersFeder, Andy Dingley, Beland, Bobdc, Brettz9, Bwhmather, Bzane, Cburnett, Cedar101, Chendy, Chris the speller, Danja, Dmccreary, Doctus, Dront, Ekphraster, EricP, Fingerz, Fleminra, GraphStar, Gthb, Holygoat, Indeyets, Int21h, Jamelan, Jasy jatere, Jerryobject, Jmrose, Jodi.a.schneider, Joejimbo, John Vandenberg, JustinHill1980, JzG, Karima Rafes, Kcoy, Kennyluck, Khazar, Khazar2, KingsleyIdehen, Kjetil, KnightRider, Koavf, Lastorset, LeeFeigenbaum, Ls261, MacTed, Maramsujith, Materialscientist, Mdd, MeltBanana, Mhgrove, Mortenf, Newty25, Nikosbik, Nischayn22, Pluma, ProfessorBaltasar, Proofreader77, Ricmitch, Risi, Rjwilmsi, Robertvan1, Romeokienzler, Sabre ball, SarekOfVulcan, Shepard, Shvahabi, SimenH, Suruena, Syneil, Tchetchenko, Theweirdguy, Thorwald, Thumperward, Timboliu, Tulcod, Ultimatewisdom, Universimmedia, Waveform, Wesley, Wolvever, Woodart, Yaron K., Zviedris, 106 anonymous edits

RDF query language *Source*: http://en.wikipedia.org/w/index.php?oldid=544532322 *Contributors*: Andy Dingley, Fleminra, ForrestCroce, Ian Spackman, Jamelan, Jerryobject, Jleedev, John Vandenberg, John of Reading, Juanjobnt, MovGP0, Rjwilmsi, Rollxx, Toussaint, Universimmedia, 8 anonymous edits

Access query language *Source*: http://en.wikipedia.org/w/index.php?oldid=598578774 *Contributors*: A bit iffy, Adamdaley, Angela, Anonymoues, Avicennasis, ChrisGualtieri, Damian Yerrick, Fang Aili, Jerryobject, Kelapstick, Kephir, Klemen Kocjancic, Logan, Markalex, Merovingian, Phil Boswell, Slipstream, Stuartyeates, Suruena, Template namespace initialisation script, TobyJ, YUL89YYZ, 4 anonymous edits

Nonprocedural language *Source*: http://en.wikipedia.org/w/index.php?oldid=522904192 *Contributors*: Borkfisch, CRGreathouse, David Biddulph, Funandtrvl, Legaia, Philip Trueman, Rich Farmbrough, TuukkaH, Waynesalhany, Xp54321, Your Lord and Master, 12 anonymous edits

Facebook Query Language *Source*: http://en.wikipedia.org/w/index.php?oldid=551741865 *Contributors*: Bearcat, Bostwickenator, Dawynn, Fabrictramp, Farras Octara, Frap, Greerjacob, Hazel-roo, Katharineamy, Lauromolina, PamD, Pjoef, Postcard Cathy, Tabledhote, This, that and the other, 5 anonymous edits

Active database *Source*: http://en.wikipedia.org/w/index.php?oldid=562484888 *Contributors*: Appkausman, Beland, ChrisGualtieri, Danim, Duncan.Hull, ErrantX, Greenrd, Imanhelal, Intgr, Mjatuvt, Mjeu, Rich Farmbrough, SarekOfVulcan, Uffanzi, 7 anonymous edits

Database trigger *Source*: http://en.wikipedia.org/w/index.php?oldid=594815893 *Contributors*: Abdull, Acha11, Adarshramesh, Bevo, BobHindy, Brick Thrower, Bucketsofg, Cadillac, Can't sleep, clown will eat me, Cedar101, ClamDip, ClanCC, CodeNaked, DanBishop, DanielcWiki, Deineka, Denisarona, Derbeth, Dffgd, Dirkbb, Fizalhaji, Fæ, Grondemar, Gurch, HMSSolent, Hazard-SJ, Heron, Hu12, Jerome Charles Potts, John of Reading, Jyujin, Knakts, L337 kybldmstr, Larsinio, Lugia2453, M2Ys4U, Magioladitis, Mark Renier, Matinict, Mecanismo, Mike Rosoft, Mikeblas, Mindmatrix, Minna Sora no Shita, Mlpearc, Mortense, MrOllie, Mschlindwein, NathanBeach, Nickleus, Niteowlneils, Noah Salzman, Noelweichbrodt, PaD, Pimlottc, Pinethicket, RJFJR, Ramkrish, Reedy, Rimonon, Ross Fraser, Rsrikanth05, S.K., Sae1962, Sampsonvideos, Sippsin, SluggoOne, Stolze, SuperHamster, Superhilac, Svick, Tlaresch, Troels Arvin, Unordained, Windofkeltia, Yourbane, Yrithinnd, علی ویکی, 297 anonymous edits

Stored procedure *Source*: http://en.wikipedia.org/w/index.php?oldid=597923607 *Contributors*: 0goodiegoodie0, 3Jane, Aaadamaa, Abdull, Aleenf1, Amaury, Andreas Kaufmann, Andy.ruddock, Aravind V R, AvicAWB, Avé, Berny68, Bevo, BobHindy, Bobo192, Bovineone, Brenan99, Brianray, Calane83, Cedear, ChristopherGautier, ClementSeveillac, Coachbudka, Cww, DVdm, Dcoetzee, Derbeth, Dogsgomoo, Dougher, Drake Redcrest, Dreamofthedolphin, Duster.Cleaner, EdgeOfEpsilon, Elockid, EvanCarroll, EvanSeeds, Farazbs20, Favonian, Flashspot, Frap, Frecklefoot, Fred Bradstadt, Friendlydata, Gambhava, Gilliam, GoldenTorc, Graham87, GregorB, Harryboyles, Homestarmy, Honeplus, Hu12, IO Device, Ichimonji10, Izogi, JCLately, Jay, Jeffreyarcand, Jeltz, Jim1138, Jogloran, KeyStroke, Kmsimon, Kuru, Kvdveer, Kyledmorgan, Larsinio, Lewissall1, Lights, Luckypayal, M4gnum0n, MER-C, Mariolina, Markblue, Marr75, Martincamino, Materialscientist, Matticus78, Mayur, Merbabu, Michael@nosivad.com, Mikeblas, Mikesheffler, MilerWhite, Mindmatrix, Mitchandsherri, Modster, Moe Epsilon, MrJones, MrOllie, Mschlindwein, NYCDA, Neilc, Nickdc, Nsaa, Ohiostandard, Pedro, Petersap, Pinethicket, Pravs, Primalmoon, Pseudonym, Rajeearul, Red Thrush, Regani, RevRagnarok, Rich Farmbrough, Riki, Rror, Rythie, S.Örvarr.S, Sava chankov, Scadavidson, Scgtrp, Sdorrance, SimonP, Sippsin, Sqlinfo, Stevietheman, Stolze, SymlynX, Taeshadow, Thane, Thumperward, Tijfo098, Tobias Bergemann, Troels Arvin, Unyoyega, Velella, Winston Chuen-Shih Yang, Xenium, Xzilla, Zhenqinli, Σ, 458 anonymous edits

PL/SQL *Source*: http://en.wikipedia.org/w/index.php?oldid=598963037 *Contributors*: Aarem, Adreamsoul, Aegicen, Akadruid, Alan Millar, Alansohn, Alex.g, Allens, Andy Dingley, Anna Lincoln, Aravind730, Arthena, Artichoker, Bgaskin, Bovineone, Brianray, Bwefler, Can't sleep, clown will eat me, CanisRufus, Carbidfischer, CasperBraske, Centrx, Chappy84, Chowbok, Chrisrimmer, Chrissawer, Chutzpan, Cm10497, Codename Lisa, Cometstyles, Comp123, Conversion script, Craigy144, Cwjolley, Cybercobra, Damian Yerrick, Davipo, Derbeth, DevastatorIIC, Drovetto, E0steven, EagleOne, Edans.sandes, Eduardofeld, Engmark, Eustress, Fdesing, Ffangs, Fraggle81, Frap, Fsilva27, Fubar Obfusco, Graham87, GrandPoohBah, Greensburger, GregorB, Helix84, Hoo man, Hroptatyr, Hu12, Hydrogen Iodide, Icey, Japo, Jay, Jdforrester, Jim1138, Jlam, Johayek, John Vandenberg, Jonathan321, Josemanimala, Kalapratik, Kekedada, KeyStroke, Kjartang, Kmorozov, Krischik, Kuru, Kusunose, Leandrod, Lerdsuwa, Lexikorn, Linlasj, Loren.wilton, Lotje, Lugnad, Luiscosta, Madhukar83, Mahahahaneapneap, Maheshvenna, Mark Renier, MarkMankins, Materialscientist, Mav12321, Mdchachi, Michael Devore, Michig, Mikeblas, Miker@sundialservices.com, Mild Bill Hiccup, Mintleaf, Mogism, Monkeycheetah, MrOllie, Naudefj, Nickdc, Nicolas1981, Nuno Tavares, Nurmuhammad, Patclaffey, PaulBoxley, Pedant17, Peterl, PhilKnight, Pjetter, Prunesqualer, Quadell, Quale, Ramkrish, Rashid alen, RedWolf, Rednblu, Rjwilmsi, Rmosler2100, Ruud Koot, S.K., SarekOfVulcan, Scr0llwheel, Seaphoto, Seb az86556, Securiger, Sef96121, Shadowjams, Shell Kinney, Shimeru, ShlomoS, Shoaler, Sippsin, Siskus, Sjc, Skarkkai, SlowJog, Soosed, Stevietheman, Stryn, TMSTKSBK, Tertrih, The cat ncl uk, ThisIsDennis, TimHall, Tolly4bolly, Turnstep, VMS Mosaic, VictorAnyakin, Villarinho, Vipinhari, Wadsworth, Wernher, Whitehatnetizen, Winterst, Yonatan, Ziguang, 510 anonymous edits

SQL PL *Source*: http://en.wikipedia.org/w/index.php?oldid=580033358 *Contributors*: Berny68, ChrisGualtieri, Discospinster, Jdlambert, Malcolma, MilerWhite, Rwwww, Squids and Chips, Tijfo098, Yutsi, 4 anonymous edits

SQL programming tool *Source*: http://en.wikipedia.org/w/index.php?oldid=593173698 *Contributors*: Avoiceinthesea, Beetstra, Bradwery, ChrisGualtieri, Davie4125, Dawnseeker2000, Devart, Eth000, Hazarbx, Hu12, Istoyanov, JLaTondre, Jsorr, Lfstevens, Mark Arsten, MaxSherbinin, Michael Slone, PLVB, Pagess63, Qpniteflyqp, RHaworth, Rjwilmsi, RobertJLove, Sasha.sheinberg, Sole Soul, Strannikk, Themfromspace, Yurad, 19 anonymous edits

User-defined function *Source*: http://en.wikipedia.org/w/index.php?oldid=595591894 *Contributors*: AKGhetto, ANNAfoxlover, Abdull, Amcguinn, Andreas Kaufmann, Andrzej P. Wozniak, Billinghurst, Cedar101, Chowbok, Chriscf, Crystallina, DPRoberts534, Dede2008, Frap, Giraffedata, Gpvos, Haeinous, Jansan, Kdehl, Lankiveil, Materialscientist, Mercy, Mikeblas, MilerWhite, Mitsukai, Mortense, Mpin, MrX, NaBUru38, Petersap, Piranna, RevRagnarok, ReyBrujo, SchreiberBike, Seqsea, SmilesALot, Stolze, Tobias Bergemann, Xypron, ZacBowling, 35 anonymous edits

Cursor (databases) *Source*: http://en.wikipedia.org/w/index.php?oldid=597274583 *Contributors*: Abdull, Abi79, Agujero Negro, Aitias, Alik Kirillovich, BlastOButter42, Catgut, Cedar101, Christian75, Cwolfsheep, Danielx, DarkFalls, Darth Panda, DigitalEnthusiast, Dmccreary, Doug Bell, Ejdzej, Epbr123, Feder raz, Ffu, Fieldday-sunday, Greenrd, Habitmelon, Haleyga, Hutcher, Ilyanep, Ivantalk, Jamestochter, Janigabor, Jerome Charles Potts, Julesd, Justinc, Kamots, Kubieziel, Larsinio, Mark Renier, Mikeblas, Mindmatrix, Miten.morakhia, MrOllie, Mwtoews, Nagae, NawlinWiki, Nbarth, NeonMerlin, NewEnglandYankee, OsamaK, PhilHibbs, Primalmoon, RHaworth, RandyFischer, Reedy, Richi, RomanSpa, S.K., S.Örvarr.S, Sachzn, SarekOfVulcan, SimonP, Sippsin, SkyWalker, Stolze, Tbannist, TechTony, Tobias Bergemann, Underpants, Wallie, Winston Chuen-Shih Yang, Wjasongilmore, Wknight94, Xiphoris, 96 anonymous edits

SQL Problems Requiring Cursors *Source*: http://en.wikipedia.org/w/index.php?oldid=573367245 *Contributors*: Cedar101, ErrantX, Fabrictramp, Jncraton, Lfstevens, M.O.X, Malcolma, Mortense, SimonP, Somewherepurple, 7 anonymous edits

WxSQLite3 *Source*: http://en.wikipedia.org/w/index.php?oldid=546283969 *Contributors*: Ardric47, Cn.clover, Danim, Howdybob, Rich Farmbrough, Saga City, Shkutkov Michael, Tinucherian, Utelle, Xezbeth, 3 anonymous edits

Application programming interface *Source*: http://en.wikipedia.org/w/index.php?oldid=598743454 *Contributors*: 213.121.101.xxx, 24.108.233.xxx, 24.93.53.xxx, 4483APK, 64.105.112.xxx, 90, AaronL, Aarsalankhalid, AbdulKhaaliq2, Adah, Addshore, Ae-a, Aeons, Aeternus, Ahunt, Ahzahraee, Airplaneman, Alan d, Alex43223, Altaf.attari86, Altaïr, Altenmann, Amanuse, Ancheta Wis, Andkore, Andre Engels, Andremi, Andres, Andrey86, Andy16666, Anteru, Apoltix, Arindra r, Arjayay, Arthur Davies Sikopo, Aruton, Ashamerie, Asydwaters, Atheken, Atreys, Aude, Auntof6, Avk15gt, Awg1010, Bamyers99, Bdesham, Bearcat, Betterusername, Bevo, Bhat sudha, Bikingviking, Blackcats, Blerglebleng, Bobo192, Boing! said Zebedee, Bookiewookie, Borgx, Boyprose, Brianski, Bryan Derksen, BryanG, Bryanmonroe, C5st4wr6ch, CYD, Calton, Calvin 1998, CanisRufus, Capricorn42, Cedar101, Chadsmith729, Chameleon, Chealer, Chicago god, Chiefcoolbreeze, ClaudiaHetman, CloudNine, Colonoh, Consult.kirthi, Conversion script, Coolbloke94, Cornelia Gamst, Courcelles, Cybercobra, Cynical, DShantz, Damian Yerrick, Daniduc, Danja, Daperata, Darklight, Davejohnsan, Davemck, David Gerard, Davron, Dawidl, Deadbeef, DeeKay64, Deepugn, Dennislees, Denny, Derek farn, Detnos, Dewritech, Didym, Diego Moya, Diomidis Spinellis, Dipskinny, Discospinster, Dmarquard, Download, Dqpeck, Dr Marcus Hill, Dr.mmbuddekar, Dreadstar, Drewmeyers, Dschach, Dsimic, Dsnelling, Dylan620, ENeville, EVula, EagleMongoose, Ebessman, Econterms, Ed Poor, Edcolins, Edinwiki, Edwardkerlin, Efa, Egmontaz, Ehn, Elf, Ellmist, Eloquence, Enochlau, Enric Naval, Epbr123, Epicgenius, Eric Agbozo, Espoo, Excirial, Faizan, Farrwill, Ffffloyd, Fieldday-sunday, Fitch, Fraggle81, Frap, Freakimus, Frecklefoot, FrummerThanThou, Funvill, Fæ, GRAHAMUK, Gak, GeoffPurchase, Giftlite, Graham87, Greensburger, Griznant, Gryllida, HUB, Hadal, Harryboyles, Hashar, HenkeB, Heraclius, Hfastedge, Hires an editor, Hmains, Humu, Husond, IShadowed, InShaneee, Infinitycomeo, Itai, Ivan007, Ixfd64, Izno, J3st, JHunterJ, JLaTondre, JWSchmidt, Jakuzem, JamesMLane, Jamesx12345, Jarble, Jbolden1517, Jengod, Jenova20, Jerryobject, Jesant13, Jesse V., Jianhui67, Jidanni, Jitse Niesen, Jleedev, Jmclaury, JoeB34, John of Reading, John.n-irl, JohnBlackburne, Johnuniq, Jtowler, JulesH, Julien, Julienj, K12u, Kbolino, Kernel Saunters, Kichik, Kickin' Da Speaker, Kim Bruning, King of Hearts (old account 2), KnowledgeOfSelf, Kocio, Kompowiec2, Kurdo777, Landon1980, Lavers, Lee Daniel Crocker, Legoktm, Lekshmann, Leoholbel, Liempt, Limbo socrates, Liorma, LizardJr8, Lockeownzj00, Looc4s, Lord Chamberlain, the Renowned, Loren.wilton, Loshsu, Lotje, Lsmll, Lupin, M5, MZMcBride, Mac, Mange01, Manop, Martyn Lovell, Marudubshinki, Materialscientist, Mattknox, Mav, Mbeychok, Meldraft, Mflore4d, Michael Hardy, Michael Hodgson, Michaelas10, Miguel, Mihai cartoaje, Miketwardos, Minghong, Minirogue, Miohtama, Mogism, MoreThanMike, Morg, Mountain, Mpbaumg, Mquigley8, MrOllie, Mrh30, Mshivaram.ie22, Mwarf, My name is not dave, Myc2001, Nanshu, Neelix, NewEnglandYankee, Nopetro, Notinasnaid, Nsda, Obradovic Goran, Ocean Shores, Ochado, Ohnoitsjamie, Ohspite, Old Guard, OlegMarchuk, Omniscientest, Oshwah, Oulrij, Oxymoron83, Pascal.Tesson, PaymentVision, Peak, Pearll's sun, Pengo, Pepe.agell, Percede, Pertolepe, Pgimeno, Philip Trueman, Phuzion, Piano non troppo, Plamka, Poweroid, Prari, PublicAmpersand, Pwforaker, Queenmomcat, Quinet, Quux, Qwertyus, Qx2020, Qxz, R'n'B, RJHall, Rackspacecloud, Raise exception, Randomalious, RedWolf, Redlentil, Reemrevnivek, Renatko, Rich Farmbrough, RichMorin, RichSaunders, Riluve, Rktur, Robert K S, Robneild, Rocastelo, Ronocdh, RoyBoy, Rponamgi, Rs rams, Rudyray, Rwwww, SERIEZ, SHCarter, SMC, SQGibbon, ST47, Sakhal, Salvatore Ingala, Sam Korn, Scohil, Scott Ritchie, Seaphoto, Sega381, Seth Nimbosa, Sfmontyo, Shaun, Shlomif, SimonTrew, SirSandGoblin, Sj, Skeejay, Skysmith, Slady, Slurrymaster, Smalljim, Snippy the heavily-templated snail, SocialRadiusOly, Sodium, Softwareqa, Solusinaeternum, Soumyasch, Spencer, Spikey, SqueakBox, Sriharsh1234, Status, Stephenb, Stephenchou0722, SteveBaker, Steven J. Anderson, Steven-arts, Sun Creator, Suruena, Syvanen, Szajd, T-borg, TCN7JM, Ta bu shi da yu, Tantrumizer, TastyPoutine, Tedickey, Teryx, Teutonic Tamer, The Anome, The Thing That Should Not Be, TheSoundAndTheFury, TheresaWilson, Thiotimoline, Thisisborin9, Thryduulf, Tide rolls, Tijuana Brass, Tony1, Torchiest, Tpbradbury, TrOlLLin1212, Transpar3nt, TrentonLipscomb, Troymccluresf, Tseay11, Ulugen, Uriyan, Utype, Uzume, Varworld, Velella, Vikramtheone, Visvadinu, Vkorpor, Voidxor, WTRiker, WaltBusterkeys, Wapcaplet, Wavelength, Whatsnxt, Whitehorse212, Widr, WikHead, Willy-os, Wjejskenewr, Wysprgr2005, Xqt, Yaris678, YordanGeorgiev, Yurik, Zachlipton, Zeno Gantner, Zero Thrust, Zhinz, ZimZalaBim, Zodon, 계정명뭘로하지, 935 anonymous edits

Structured Query Language Interface *Source*: http://en.wikipedia.org/w/index.php?oldid=470830881 *Contributors*: Andreas Kaufmann, Cander0000, Cgapperi, Danim, Santana

Database transaction *Source*: http://en.wikipedia.org/w/index.php?oldid=598390179 *Contributors*: 16@r, Adi92, Ajk, Al3ksk, AmandeepJ, AnmaFinotera, Appypani, Babbage, Bgwhite, Billinghurst, Binksternet, Burschik, CharlotteWebb, ChrisGualtieri, Clausen, Comps, Craig Stuntz, DCEdwards1966, Damian Yerrick, Daniel0524, Dauerad, Derbeth, DnetSvg, Forderud, Fratrep, Geniac, Georgeryp, Gerd-HH, Gf uip, Ghettoblaster, GregRobson, Haham hanuka, Hbent, Hede2000, Highguard, HumphreyW, Intgr, JCLately, Jarble, Jason Quinn, Jeltz, Karel Anthonissen, KellyCoinGuy, KeyStroke, Khukri, Kirananils, Larsinio, Leeborkman, Lingliu07, Lubos, Luc4, Lysy, M4gnum0n, Mark Renier, Matiash, MegaHasher, Mike Schwartz, Mikeblas, Mindmatrix, Mintleaf, Neilc, Nixdorf, OMouse, Obradovic Goran, Owen, Paul Foxworthy, Pcap, Pepper, Prakash Nadkarni, RedWolf, RichMorin, Rocketrod1960, Roesser, SAE1962, Sandrarossi, SebastianHelm, Sobia akhtar, SqlPac, Stevag, T0m, Thisismyusername96, Timo, Triwger, Troels Arvin, Turnstep, WeißNix, Zerksis, Zhenqinli, 110 anonymous edits

Prepared statement *Source*: http://en.wikipedia.org/w/index.php?oldid=599156692 *Contributors*: Audriusa, Bachrach44, Bgwhite, Cvf-ps, Dcoetzee, Johnrussell13, Jroughgarden, McLibra, Rsrikanth05, Sidjoshy, Two Bananas, Wes.turner, 11 anonymous edits

Open Database Connectivity *Source*: http://en.wikipedia.org/w/index.php?oldid=434753869 *Contributors*: AKGhetto, AdventurousSquirrel, AlistairMcMillan, Allens, Andreas Kaufmann, AndriuZ, Andy Dingley, Arch dude, Auric, AvicAWB, Avé, Beevvy, Bigpru, BobGibson, BonsaiViking, Borgx, Bovineone, BryEllis, Bunnyhop11, Cander0000, CanisRufus, Canterbury Tail, Charlesgold, Chealer, ClaudioSantos, Computafreak, Craig Stuntz, DFulbright, Danim, David Gerard, Derbeth, Discospinster, Dittaeva, DragonHawk, Drewgut, Eglobe55, Electrolite, Epim, Everlasting Winner, Gcm, GreyCat, Gwern, Harry Wood, Inzy, Irish all the way, JLaTondre, Jandalhandler, Jay, Jerome Charles Potts, Jkelly, Jklowden, John of Reading, JonathanMonroe, Jonesey95, Julesd, KeyStroke, KingsleyIdehen, Kuru, Kyouteki, Kzafer, Larsinio, Lkstrand, Lowellian, Lurcher300b, MacTed, Magnus.de, Manop, Mark Renier, Markhurd, Martijn Hoekstra, Materialscientist, Maury Markowitz, Maximaximax, MeltBanana, Michael Hardy, Mikeblas, Mindmatrix, Minesweeper, Minghong, Mintleaf, Misog, Mitsukai, NapoliRoma, Nikos 1993, Nixdorf, NoahSussman, Not Sure, Orlady, Orpheus, Oxda, Pajz, Paul Foxworthy, Pedant17, Pmsyyz, Polluks, Power piglet, PrisonerOfIce, Quuxa, Raffaele Megabyte, Rajkumar9795, Reconsider the static, Reedy, RenniePet, Rjwilmsi, RomanSpa, Rrabins, Seanwong, Sega381, Spellmaster, Sspecter, Struway, Swhalen, The Anome, The wub, Thumperward, Tide rolls, Tin, Todorojo, TommyG, Viridae, Wez, Whpq, Wikipelli, William Avery, Winterst, Woohookitty, Ysangkok, Yug1rt, Хаш-Эрдэнэ, 234 anonymous edits

Java Database Connectivity *Source*: http://en.wikipedia.org/w/index.php?oldid=598090482 *Contributors*: AS, Abdull, Andreas Kaufmann, Andrewman327, Andy Dingley, Arcann, Atozjava, Audriusa, Beetstra, Bkrakesh, Blaine-dev, Brick Thrower, Bunnyhop11, CambridgeBayWeather, Cameltrader, Cander0000, Caomhin, Captmjc, Cherkash, Chowbok, ChrisGualtieri, Chrismacgr, Coldacid, Danim, Darth Molo, Dddelfin, Derbeth, Dinosaurdarrell, DivideByZero14, Doug Bell, Drewgut, Edward, Ehheh, Ehn, Elandon, Elaz85, Epicgenius, Eumolpo, Evgeni Sergeev, Explanator, FOOL, Faisal.akeel, Ferengi, Ferrans, Fikus, Forage, Frecklefoot, Fred Bradstadt, Fuhghettaboutit, Fyyer, GLumba, GraemeL, Graham87, GreyCat, Harryboyles, Iain.dalton, Insomnia64, JDvorak, JLaTondre, Jacks435, Jameboy, Jay, Jems421, JeremySteinJeronimo, Jjaazz, Joffeloff, Josepant, Josephw, Julesd, Katieh5584, Kaydell, KingsleyIdehen, Klausness,

Lenaic, Leszek4444, M4design, M4gnum0n, MER-C, MacTed, Mark Arsten, Mark Renier, Materialscientist, MeltBanana, Mindmatrix, Mintleaf, Moribunt, MrOllie, Nlevitt, Noq, Pako, Pillefj, Pinecar, Pinkyf, Piojo, Pne, Poor Yorick, Praslisa, Ravenmewtwo, RedWolf, Reedy, Rm1507, SJK, Sae1962, Schw3rt, Sfmontyo, ShaunMacPherson, Simran.preet90, Sjc, SmackEater, Sohail.sikora, Sqlinfo, Srecd, Sualeh, Supreme geek overlord, Tcncv, The Anome, The Aviv, Theresa knott, Tom 99, Twsx, Unclejedd, Warren, Widr, Wikipelli, Winterst, Yaniv Kunda, Yaxh, Zundark, Дарко Максимовић, 何少仪, 231 anonymous edits

ODBC *Source*: http://en.wikipedia.org/w/index.php?oldid=597274086 *Contributors*: AKGhetto, AdventurousSquirrel, AlistairMcMillan, Allens, Andreas Kaufmann, AndriuZ, Andy Dingley, Arch dude, Auric, AvicAWB, Avé, Beevvy, Bigpru, BobGibson, BonsaiViking, Borgx, Bovineone, BryEllis, Bunnyhop11, Cander0000, CanisRufus, Canterbury Tail, Charlesgold, Chealer, ClaudioSantos, Computafreak, Craig Stuntz, DFulbright, Danim, David Gerard, Derbeth, Discospinster, Dittaeva, DragonHawk, Drewgut, Eglobe55, Electrolite, Epim, Everlasting Winner, Gcm, GreyCat, Gwern, Harry Wood, Inzy, Irish all the way, JLaTondre, Jandalhandler, Jay, Jerome Charles Potts, Jkelly, Jklowden, John of Reading, JonathanMonroe, Jonesey95, Julesd, KeyStroke, KingsleyIdehen, Kuru, Kyouteki, Kzafer, Larsinio, Lkstrand, Lowellian, Lurcher300b, MacTed, Magnus.de, Manop, Mark Renier, Markhurd, Martijn Hoekstra, Materialscientist, Maury Markowitz, Maximaximax, MeltBanana, Michael Hardy, Mikeblas, Mindmatrix, Minesweeper, Minghong, Mintleaf, Misog, Mitsukai, NapoliRoma, Nikos 1993, Nixdorf, NoahSussman, Not Sure, Orlady, Orpheus, Oxda, Pajz, Paul Foxworthy, Pedant17, Pmsyyz, Polluks, Power piglet, PrisonerOfIce, Quuxa, Raffaele Megabyte, Rajkumar9795, Reconsider the static, Reedy, RenniePet, Rjwilmsi, RomanSpa, Rrabins, Seanwong, Sega381, Spellmaster, Sspecter, Struway, Swhalen, The Anome, The wub, Thumperward, Tide rolls, Tin, Todorojo, TommyG, Viridae, Wez, Whpq, Wikipelli, William Avery, Winterst, Woohookitty, Ysangkok, Yug1rt, Хаш-Эрдэнэ, 234 anonymous edits

OLE DB provider *Source*: http://en.wikipedia.org/w/index.php?oldid=579084599 *Contributors*: Danim, Edward, MacTed

OLE DB *Source*: http://en.wikipedia.org/w/index.php?oldid=565599009 *Contributors*: AKGhetto, Alansohn, Alexf, Armbrust, BWCNY, Berntie, Boshomi, CesarB, ChrisGualtieri, Cst17, DS1953, Danim, Dienstag, Drewgut, Epim, FatalError, Fiach6383, Frigotoni, GreyCat, Haakon, Hu12, Inwind, JLaTondre, Jamelan, Jtem, Jutiphan, KingsleyIdehen, Kmote, Kriani, Luís Felipe Braga, MacTed, Maury Markowitz, MeltBanana, MooNFisH, NSR, Ngpd, O mide@yahoo.com, Phoenicyan, Polluks, Rajkumar9795, Ronk01, Shewitt.au, Soumyasch, SqlPac, Ta bu shi da yu, Thalter, Warren, Wikiolap, Winterst, 84 anonymous edits

UnixODBC *Source*: http://en.wikipedia.org/w/index.php?oldid=594054009 *Contributors*: Amillar, Andreas Kaufmann, Azylber, Bvssatish, Craig Stuntz, Frap, Gronky, Ilikeeatingwaffles, JLaTondre, Jengelh, Lurcher300b, Magioladitis, Michael B. Trausch, Nikhiltri, Peteralexharvey, 小wing, 11 anonymous edits

IODBC *Source*: http://en.wikipedia.org/w/index.php?oldid=561484835 *Contributors*: AKGhetto, Andreas Kaufmann, Bryan Derksen, Craig Stuntz, Danim, Frap, GreyCat, JulianBridle, Kenyon, KingsleyIdehen, Magioladitis, MeltBanana, Mitsukai, Moreati, OwlofDoom, Revoltex, Welsh, 5 anonymous edits

Pool (computer science) *Source*: http://en.wikipedia.org/w/index.php?oldid=579557722 *Contributors*: Buddy23Lee, Dsimic, Frap, Jonkerz, Juansempere, 4 anonymous edits

Connection pool *Source*: http://en.wikipedia.org/w/index.php?oldid=595107943 *Contributors*: Abdull, Acdx, Alai, BMacZero, Beaddy1238, CrashTestWolf, Czarkoff, Davidjxyz, Dfrg.msc, DigitalEnthusiast, Discospinster, Falcon8765, Frap, Ginsuloft, Greenrd, GregRobson, Harro5, Iamaelephant, Inconexo, Jim1138, Karmafist, M4bwav, M4gnum0n, Magioladitis, Materialscientist, Michael93555, Neerajtuteja, NielsenGW, Nivus, Nrworld, Oluies, Pacific202, Reedy, Rich257, Rlest, Slon02, Svick, Tobias Bergemann, Tumb, Wine Guy, Xerixe, Zr40, 76 anonymous edits

Remote Database Access *Source*: http://en.wikipedia.org/w/index.php?oldid=573974030 *Contributors*: Alecperkins, B. Wolterding, BD2412, Cander0000, Charles Matthews, Folajimi, Greenrd, J04n, JonHarder, Kbdank71, Pegship, Pogogunner, Rl, Rror, Star Mississippi, Tassedethe, 13 anonymous edits

SQLJ *Source*: http://en.wikipedia.org/w/index.php?oldid=586092529 *Contributors*: Bunnyhop11, Cedar101, ColinMcNeily, DE103252, Danim, John Vandenberg, Joncnunn, Kelti, LilHelpa, Lunacrescens, Mark Renier, Medovina, MilerWhite, P.Y.Python, Thumperward, Tijfo098, Wikispaceman, Winterst, Ysangkok, 10 anonymous edits

Native Queries *Source*: http://en.wikipedia.org/w/index.php?oldid=431107730 *Contributors*: Foxxygirltamara, Greenrd, Hugo 87, Mikeblas, Rjolly, TheParanoidOne, 3 anonymous edits

Meta-SQL *Source*: http://en.wikipedia.org/w/index.php?oldid=568186268 *Contributors*: Andrewman327, Cithowha, Dlrohrer2003, Gary, GregorB, Gunnala, RJFJR, RockMFR, Serenaacw, Squids and Chips, Trixie05, 6 anonymous edits

ADO.NET *Source*: http://en.wikipedia.org/w/index.php?oldid=592766948 *Contributors*: AKGhetto, AS, Alansohn, Ancheta Wis, Andreas Kaufmann, BWCNY, Beland, Bohumir Zamecnik, Bovineone, Brettaryan, CardinalDan, Carmichael, Chocrates, Chris the speller, Closedmouth, Danim, DarkFalls, Darth Mike, Dcoetzee, Dgies, Dhavalhirdhav, DigitalEnthusiast, Dockurt2k, DragonLord, Dufresne77, E2rd, Edward, Eng.ahmedm, FayssalF, Fraggle81, Frankamand, Fuzzy Logic, GB fan, Ganymead, GermanX, Gonzobrains, GraemeL, GreyCat, Heavenly Crypt, Igloobone, Ilikeverin, IndianGeneralist, Intgr, J. Finkelstein, JLaTondre, Jasonshortphd, Jhoppe, Jutiphan, KingsleyIdehen, Leotohill, Lichtgestalt, MER-C, MacTed, Mdd, Mediran, MeltBanana, Mfhobbs, Mike Schwartz, Mikeblas, MooNFisH, Mortense, MrOllie, Mulad, Niteowlneils, Northamerica1000, Nummer29, Oo7565, Pbb, Quintinwillison, Radiant chains, RainbowOfLight, Rd232, Renaissancee, Rich Farmbrough, Richard R White, Rwwww, S.Örvarr.S, SYSS Mouse, Saxbryn, Sharkoftheday, Sikon, Snow Blizzard, Solarra, Soumyasch, Stainedglasscurtain, Stephenb, Surachit, Svick, Tagus, Tcncv, Tentacle Monster, Terp96, Tiedyeina, Toussaint, TwoMartiniTuesday, Vivio Testarossa, W.D., Warren, WikHead, William Avery, Winterst, Xpclient, Yarin Kaul, 263 anonymous edits

List of relational database management systems *Source*: http://en.wikipedia.org/w/index.php?oldid=594329678 *Contributors*: *drew, AaaghItsMrHell, Adamblang, Adrian J. Hunter, AlistairMcMillan, Alureiter, Amalthea, Anas2048, Arcann, Armadillo ECM, Armen1304, Baojia, Basil.bourque, Beland, BoxSoft, Bp0, Bressan, CasperGoodwood, Cburnett, Cgfdmc, ChandraASGI, CharlieTesta*24, Closedmouth, Countersubject, CovenantD, Craig Stuntz, Crosbiesmith, Cubridorg, DEddy, DRady, Darren Duncan, DatabACE, Davidsheiman, Dbaxter0, DerHexer, Dfetter, Dougbertram, Eeekster, Einarkristjan, Ekraft, Elf, Fraise, Fschupp, Ggeldenhuys, Glange90411, Greenrd, Grstain, Gsingh, HJ Mitchell, HappyCamper, Herostratus, Hertzsprung, Igloobone, Imroy, Jam02, JasonThePirate, Jberkus, Jerome Charles Potts, JethroElfman, Jimgawn, Jimmi Hugh, JohnGray, JohnnyMrNinja, Jonasfagundes, Jost Riedel, Jpetersen74, Jtdunlop, Karnesky, KingsleyIdehen, Kognitio, Kriplozoik, Leandrod, Legolas558, Lfstevens, Lowellian, Lpetrazickis, Luke Lonergan, MainFrame, Mark Renier, Markoprima, MarylandArtLover, Mcaisse, Mekong Bluesman, Mikeblas, Mindmatrix, Minghong, MrBoo, Mrozlog, Mtasic, Neilrieck, Ngpd, Nick Number, Nicolas1981, Niteowlneils, Ocherkashin, OlivierWeb, Paul A, Pelagius333, Pengo, Pgillman, Pjrm, Prolog, Pwinkler4185, Radagast83, Ramanna.Sathyanarayana, Rcorcs, Reedy, Requestion, Rich Farmbrough, Ross Burgess, Rrabins, Ryandaum, Sappy, ScottDavis, Sergsav, Snarius, Sqlinfo, StevenBlack, Stuboy, Sushi500, Syrthiss, Ted nw, Tehnic49, Tentinator, Thomashilbert, Thryduulf, TommyG, Troels Arvin, Tstevelt, Turnstep, Vmatikov, Vtbl, W163, WOSlinker, Waw2010, Whouk, Wikiolap, Williamtimm, Zimbabwer, 186 anonymous edits

Comparison of relational database management systems *Source*: http://en.wikipedia.org/w/index.php?oldid=597231624 *Contributors*: 90, 96cores, A5b, Abu badali, Adono, Aeriform, Agavenwurm, Akagel, Alexandre.Morgaut, Alvaro.Monge, Analoguedragon, Anas2048, AndrewCowie, Angoca, Asqueella, Astral v, Axelstudios, Az29, BBCWatcher, Basil.bourque, Beetstra, Beland, Beta m, Bezik34, Bgwhite, Bigown, Bjkeefe, BlackCatN, Bmfrosty, Bogdangiusca, Brick Thrower, Brierand, Brookie, Brulath, CCFS, CRAIZ3D, Calador109, Carp3, Cazito, Ceaton55, Chachka, Chendy, Cheolsoo, Chris the speller, Chriskl, Cjcollier, Clanie, ClementSeveillac, Clieu, Comp.arch, Connor Behan, Coolboy1234, Corrado, CovenantD, Craig Stuntz, Cubridorg, Curps, DanBishop, DancingMan, Danmcg.au, Dark ixion, Darthsco, DavidMCEddy, Davidsheiman, DeTru711, DeirdreGerhardt, Dfetter, Dionyziz, DocendoDiscimus, Donhalcon, Dougdp, DrThompson, Drilnoth, Duckbill, Dveeden, Dvgeorge, Ean5533, Edward, Eli lilly, Equalizer777, EvanCarroll, Findling67, Fivelittlemonkeys, Fixesfixes, Florian Sening, Fonsie, Frap, Fratrep, Fredrik, Gbgsimulationjon, Gcalis, Gczffl, Geordee, Georgeryp, Gilad.maayan, Gintsp, Gkanel, Glange90411, Glmeece, Gmaxwell, Goeldner, Gogo Dodo, Gongshow, Greenman, GreyCat, Gudeldar, Guerrabraga, HHempelmann, HappyCamper, HarrisonFisk, Hasegeli, Hibethy, Hrgwea, Hu12, Hz.tiang, Igloobone, Improv, Imroy, Indexheavy, Inessa4ever, Intgr, Isaac Sanolnacov, JJay, JLaTondre, Jamesday, Jawsper, Jberkus, Jbicik, Jeltz, Jerome Charles Potts, Jethro555, Jevansen, Jhonjairoroa87, Jim.Callahan,Orlando, Jin.Takahashi, Jmachat, Jmnbatista, Jojalozzo, Jon207, Jot1109, Jpupier, Judyburk, Juha001, Kadishmal, Kaelfischer, Karnesky, Kayvee, Keldar, Kempeth, Kenfar, Kent Heiner, KingsleyIdehen, KiwiBiggles, Kmorozov, Kognitio, Kozmando, Kweetal, Kweetal nl, Kytti khat, LHCgrp, LarsHolmberg, Larsinio, Latios, Leandrod, Leandrpf, Lfstevens, Lguzenda, LiX, Lightblade, Lmxspice, Lowellian, Lowmeus, Lucas Malor, Lzcubs1, M Th vat, Maarten Hermans, MacTed, Maheshgadgil, Manifoldtop, Maple10, Mariuz, Mark Renier, Marklark, MeekMark, Mhagman, Mikeblas, Mikluce, Mindmatrix, Minghong, Misery, Mo ainm, Moocha, Moralis, MrBoo, Mtasic, Mwtoews, Mywikie, Mithrandir, NaibStilgar, NapoliRoma, Naraht, Narendra Sisodiya, Naviworx, Ne0Freedom, Neilc, Ngpd, Nhantdn, Nickdc, Nicolas.fortin, Niqueco, Niteowlneils, Nodulation, Noonand, Noxia, OKIsItJustMe, Odinblade, Ofbarea, Olivier Debre, PCJockey, Patheticcockroach, PentoMcGreno, Peterl, Petri Krohn, Pgan002, PhilHorder, PieterDeBruijn, Plesatejvlk, Plumcreek, Pnv82, Proofreader77, Pvjohnson, Pwsegal, Quebec99, Radagast83, Radio15dude, RandalSchwartz, RaniaSOUSSI, Reedy, Reisio, Rhaas, Rhobite, Rich Farmbrough, RickBeton, Rimio, Rjwilmsi, Robert K S, Rsocol, Rudi.Leibbrandt, Rupert160, Ruud Koot, Ryguasu, S.K., SDSWIKI, SLi, Seashorewiki, Sehbueno, Shenme, Shusseina, Sietse Snel, SixSkys, Slaweks, Slyzius, Snarius, Snarpel, Soeren1611, Sqlboy, Sqlinfo, Stephenw32768, Stuboy, TRauMa, Tabletop, Taichi, TallMagic, Tarjei Knapstad, Taylorsharpemac, TechPurism, Tedmcneal, Terfilo, Textractor, Tfischbeck, Tharakan, The Anome, ThomasMueller, Thunderbird2, Thunderbritches, TimTay, Timwi, Tlaresch, TobiasPersson, TommyG, Tranemonet, Troels Arvin, Turanyuksel, Turnstep, Udittmer, Ukuechle, Unicard-ic, Veliscu Ovidiu, Victorwss, Vincenzo.romano, Vrenator, W3bbo, WOSlinker, Waldir, Waveform, Waw2010, Waynelwarren, WereSpielChequers, Whimsley, Wielewaal, Wikirosi, Wild Pansy, Will henderson, William Avery, Williamtimm, Wiml, Wmahan, Woohookitty, Wtuvell, Xpclient, Xprotocol, Yourbane, Yukuku, Yzchang, Zero0w, Zollhausring, Zsoltika, М И Ф, 943 anonymous edits

Comparison of database tools *Source*: http://en.wikipedia.org/w/index.php?oldid=597409975 *Contributors*: Aflorin27, Ali bayeh, Anas2048, Andrea.ippo, Andy.goryachev, AnmaFinotera, Avoiceinthesea, BGVA3, Baldolaf, Beetstra, Bennyz, Bernd vdB, Billinghurst, Boquia, Bradwery, Bunnyhop11, C5st4wr6ch, Captain Conundrum, ChanOJ, Chiragpinjar, Chris the speller, ChrisGualtieri, Convertin, Dabron, Dandv, Danim, Dbcoo, Dburbank, Delusion23, Devart, Diracleo, Dmcgahey, Donose.mihai, Doscmaker, Dreeves, Drrwebber, Dzrihen, EWayte, Edu tzu, Egandrews, Eliovir, Ellen Vasil, Euagelos, EvergreenFir, Eyal001, FBachofner, Findling67, Fratrep, Frehone, Gambhava, George Schmidt, GoingBatty, Gonesoft, Guoc, Hardywang, Henri 47, Hffmgb899, Highspam, Hu12, IBM Informix Doc, Iamserious, Ixnayus, JLaTondre, JaGa, Jakub Vrána, Jan.prochazka, Jason Quinn, Javaguy223, Jayam02, Jayron32, Jepafi, Jerome Charles Potts, Jevansen, Jokes Free4Me, Jreferee, Keepiru, Klisanor, Kriplozoik, Krystaleen, Kulyttle, Kvng, Leegrissom, Lieutdan13, LukeAlvila, Lyf2002, Magnuskjelland, MariahX, Marina Nastenko,

Mark Renier, MartyAcks, Mattmorr, MaxSherbinin, Mblumber, Mburtscher, Mfz25, Michaelmior, Mindmatrix, Mitchellfx, Mkblackstone, Mnasman, Mortense, MrOllie, Muhandes, Mul14, Mwtoews, Mydboladm, NBthee, NaibStilgar, Nick Number, Nickbiebuyck, Nijel, Nraboy, Oberst, Odinblade, Ole.andre.karlson, Oleg Fedorov, Oski Jr, Pagess63, Palosirkka, Pgsnake, Philipolson, Pigman, Pjacklam, Polluks, Qpniteflyqp, R'n'B, Random832, RandomXYZb, RaviKartheek, RealKenny, Rogerbj, Roznicki, Ruslan zasukhin, Rwalker, Rybec, Sasha.sheinberg, SchreiberBike, Serge.rider, Shdwsclan, Simon04, Skittleys, Slaweks, Sobchakw, Starovd, Starsonh, Stranger578, Sunnyok, TJRC, Tassedethe, Tedder, Tim baroon, Trialsanderrors, U2perkunas, Ugeeeen, V.vayer, Vaclav Frolik, Valeriy kh, Vrenator, Welsh, Wongha, Woohookitty, Zaher kadour, ZigArt, Zoonosis, Zundark, 347 anonymous edits

Comparison of object-relational database management systems *Source*: http://en.wikipedia.org/w/index.php?oldid=588854729 *Contributors*: Akagel, Alexandre.Morgaut, Anas2048, Beland, Beta m, Calabrese, Chikako, Chris the speller, Christian75, Cigano, Connor Behan, Cubridorg, DRady, Donhalcon, Garyzx, Ghp, Gudeldar, JJay, Jeff3000, Jerome Charles Potts, Karnesky, KingsleyIdehen, Leotohill, Lotje, MER-C, Mark Renier, Minghong, Palosirkka, Pamri, Piano non troppo, Pwnzor.ak, Reedy, Requestion, Ruud Koot, Rwwww, Salix alba, Skyezx, Squids and Chips, Versus22, Wmahan, 19 anonymous edits

Transactions per second *Source*: http://en.wikipedia.org/w/index.php?oldid=590602217 *Contributors*: Cae prince, Danim, Dawynn, Frap, Hexamon, JCLately, Pascal.Tesson, Petri Krohn, Steventee, Whitejay251, XPumpkinPi, Zhenqinli

Microsoft SQL Server *Source*: http://en.wikipedia.org/w/index.php?oldid=599132477 *Contributors*: *Kat*, A Softer Answer, Aaronsteers, Abatishchev, Accurizer, AdRiley, Aeonx, Agent Conundrum, Ahoerstemeier, Aim Here, Alcmaeonid, Alexf, AlistairMcMillan, AndrewHZ, Andrewman327, Andrewpmk, Andy Dingley, AndyLawton, AnonymousV2013, Antandrus, Anteallach, Aoidh, Ap, Apau98, Apavlo, Atanamir, Atropin06, Avsharath, AxelBoldt, Axeltroike, BBCWatcher, BBird, BD2412, BW, Bahder, Banaticus, Beland, Bender235, Bevo, Bezenek, Bgwhite, BioStu, Bluemoose, Bobblewik, Bocercus, Bonadea, BonsaiViking, Borgx, Boson, Bradmcgehee, BuckWoody, Buryshane, C+C, CALR, CRGreathouse, CWii, Cadr, Caffm8, Can't sleep, clown will eat me, CanadianLinuxUser, Cannolis, Cbommm, Cgillhoolley, ChazBeckett, Chealer, Clausen, CommonsDelinker, Comp.arch, Compfreak7, Cornflake pirate, Cowb0y, Cphi, Cpl Syx, Cwolfsheep, DARTH SIDIOUS 2, Daniel.Cardenas, Dave Braunschweig, DeadEyeArrow, Debresser, DerHexer, Dewritech, Diberri, Discreetlogic, Dkt5488, Dmsar, Doc0tis, Dr.Soft, Duke Ganote, Dumbledad, Dylan620, ED-tech, Eddiet19, Edgar181, Editore99, Eeekster, Ejdzej, ElTyrant, Eliavg9, Emurphy42, Eraserhead1, ErinRea, Ewebxml, Faizan, Fennec, Fishnet37222, FleetCommand, Forces91, ForestsWing, Fox2030, Fraggle81, Fred Bradstadt, Frei sein, Futurix, Garion96, Genejohannas72, Georgeby, Gianna ch, Glenn4pr, Gogo Dodo, Gogoplata1234, GoingBatty, GraemeL, GrahamDo, Grand ua, GregorB, Gwernol, Haakon, Hallows AG, Halluck, Happysailor, Heirpixel, Herbythyme, Heron, Hitesh84.patel, Hmains, Hoo man, HorseShoe, Hu12, IO Device, Idjles, IlyaHaykinson, Imsaguy, Iridescent, Ivan sus77, J.delanoy, J.swapnil28, JAZAM007, JForget, JLaTondre, JNW, Ja 62, JaGa, Jackdirks, Jago84, Jakec, Jam02, JamesBWatson, JamesMohr, Jammycakes, Jan1nad, Jasper Deng, Jayrascoe, Jcmorin, Jdlambert, JeepdaySock, Jeffryfisher, Jehochman, Jerome Charles Potts, Jfchickimon, Jim1138, Jlin, Joe Schmedley, Johnthomasmorton, Jojalozzo, Jonathonblocker, Jorge Stolfi, Jrowlandjones, JustWinBaby, Jutiphan, Jzylstra, Karimfayazi, Katmairock, KellyCoinGuy, Ketiltrout, KevinMadsen, Kevinliu173, Khenell, Kjramesh, Kkm010, KnowledgeOfSelf, Konstable, Kprobst, Krishnareddyn73, Kypr8, Laganojunior, Lain OTN, Larrymcp, Larsinio, Leifnsn, Leiterfluid, Lfstevens, LiDaobing, Limideen, Ling.Nut, Little Mountain 5, Loser137, M7, MER-C, Macduff, MahaDave, MainFrame, Marc Kupper, Marcusogden, Marina Nastenko, Markhurd, Materialscientist, MattPenner, Matthew Yeager, MayaSimFan, Mazin07, Mboverload, Mbwallace, Mckaysalisbury, Mdchachi, Mentifisto, Merlinsorca, Metasquares, Mg79, Michael Devore, Mike Rosoft, Mikeblas, Mikejpoole, Mindmatrix, Minghong, MinorContributor, Mjb, Mkoval, Mlasaj, Mmmichelle a, Modster, Moki80, MoorTnelis, Mormegil, Mortense, MosYassin, MrOllie, Mscantland, Msp0, Mwtoews, Mysorian, Nakon, Natalie Erin, Naviguessor, NawlinWiki, Nealw1590, Neil.moorthy, Neil9327, NewEnglandYankee, Nicholasethier, NickGarvey, Nixdorf, Norm mit, Noxia, Noypi380, OLAWRawhide, ObserverToSee, Ocaasi, Odie5533, Ognjen k, Olathe, Ondertitel, Outriggr, OwenBlacker, Oxymoron83, Pajz, Pangloss, Pats1, Paul Foxworthy, Paulajayi, Pbb, Pelister, Peter Blackburn, Petr.koc, Pillefj, Pinaldave, PittSammy, Pop1corn, Poss, Proxyma, Pstavroulis, Pugglewuggle, Quadrature, Quantix, QuiteUnusual, Qwyrxian, Rajeearul, Rajpaj, Ratarsed, Ravedave, Raysonho, Rboatright, RedWolf, Rednblu, Redrose64, Redsixfannorth, Reedy, Revision17, Rex2001to2004, Rhobite, Rich Farmbrough, Riki, Rjwilmsi, Roberto.icaza, Romualdo Juan Caruso, Ronwarshawsky, RoyBoy, Royhandy, Rrburke, Rsocol, Ryulong, S.Örvarr.S, SP-KP, Samuel Pepys, Sanjaykaul2000, Sbrockway, ScratzNutz, Scs, Seaphoto, Seb35, Sega381, Serge MS, Sesu Prime, Shannara, Shoeofdeath, SiliconCerebrate, Skwuent, Slwri, Smalljim, Sneha.kumar, Snori, Snoyes, Socialservice, Solarra, Sophus Bie, Soumyasch, Sql mvp, Stankrom, Starionwolf, Steeknek, StefanPapp, SteinbDJ, Stick109, Stickee, Strangnet, Superzohar, Suprotim, Surturz, Svick, TJJFV, Ta bu shi da yu, TakuyaMurata, Tarquin, TastyPoutine, Tbsdy lives, TeaDrinker, Tedickey, Thalter, The Epopt, The Herald, The Letter J, The Thing That Should Not Be, TheFeds, Thingg, Thumperward, Tiger.Soon, TigerShark, Tijfo098, Timc, Timotheus Canens, Toddst1, TommyG, Tonyfaull, Topbanana, Torc2, Tpbradbury, Troels Arvin, Twilight (renamed), Ugur Basak, Ulric1313, Uzume, Vacio, Vanished user 5zariu3jisj0j4irj, Vegaswikian, Verticalsearch, Vladislav Artukov, Voidxor, Vrenator, W3bbo, WOSlinker, Wadamja, Warren, Wesley, West.andrew.g, Widefox, Widr, Wiki fanatic, Wikiolap, WikipedianMarlith, Will381796, Winterst, Wiretse, Wpedzich, Xmm0, Xpclient, Yamamoto Ichiro, YordanGeorgiev, ZacBowling, Zbjornson, ZooFari, Zzombie, 1040 anonymous edits

Microsoft SQL Server Master Data Services *Source*: http://en.wikipedia.org/w/index.php?oldid=594214777 *Contributors*: CoolingGibbon, Ensslen, FleetCommand, Gary, Giraffedata, GregorB, Jlin, Johhart, Kshivakumar, Matthewmilad, OsamaK, R w morris, RadioFan, Sadads, Soumyasch, Suzay12, Technical 13, Thumperward, Yellowdesk, 12 anonymous edits

Transact-SQL *Source*: http://en.wikipedia.org/w/index.php?oldid=595075516 *Contributors*: Ajcomeau, Apokrif, Arinamaz, Ashmoo, Bibescu, Big Smooth, Bongwarrior, Boshomi, Brick Thrower, ChessKnight, Chrisahn, Cory Donnelly, Donarreiskoffer, EdwardH, Fotek, Greystork, Hbent, Hu12, IDX, IceKarma, Jamelan, Jan.hasller, Jogloran, Klapi, Leonren, MIT Trekkie, Marius, Masgatotkaca, Mdchachi, Mikeblas, MrOllie, Mrzeinali, MusikAnimal, Mwtoews, Norm mit, Nricardo, Osklil, Pascal666, PatrickFisher, PatrikR, Paulineward, Perfecto, Phil Boswell, Pxma, Quilokos, Qxz, Rhobite, Rsocol, Rturnham, Ruud Koot, Ryoga Godai, S.Örvarr.S, Sarabagwa, Shanes, Spolster, SummerWithMorons, The Thing That Should Not Be, Tijfo098, Troels Arvin, Warren, William Avery, உமாபதி, 153 anonymous edits

WCF Data Services *Source*: http://en.wikipedia.org/w/index.php?oldid=572447661 *Contributors*: Alex LE, Cherrythomas, Damaster98, Eaefremov, Emurphy42, Law, Mohamed El-Deeb, Niceguyedc, PavlosDaGreek, Pointillist, Rich Farmbrough, Root4(one), Senfo, Soumyasch, Wikiwind, Xpclient, 23 anonymous edits

Windows Internal Database *Source*: http://en.wikipedia.org/w/index.php?oldid=598107538 *Contributors*: BenStrauss, Evice, FleetCommand, Fraggle81, Po8crg, Soumyasch, Warren, 20 anonymous edits

SQL Server Agent *Source*: http://en.wikipedia.org/w/index.php?oldid=570861297 *Contributors*: Ajoiner, Cander0000, Johnvargheseliu4m, Mogism, Pjoef, Redrose64, SMC, Visik, Yutsi, 8 anonymous edits

SQL Server Compact *Source*: http://en.wikipedia.org/w/index.php?oldid=592089841 *Contributors*: Beland, Brianreading, Cbogart2, Chris the speller, Djmckee1, FleetCommand, Fratrep, Hu12, Igloobone, Ikrshnan, Lfstevens, Mao59, Northgrove, Ptoniolo, Rockfang, Ronark, Ruijoel, Soumyasch, Surturz, TastyPoutine, Txt.file, Ubik70, Uzume, Vroo, Vsmith, Ward99, 42 anonymous edits

SQL CLR *Source*: http://en.wikipedia.org/w/index.php?oldid=544344469 *Contributors*: Alai, Amalas, Colonies Chris, Dkent600, Dojikami, Echuck215, Jmkehayias, KathrynLybarger, Malcolma, Rabin06, Rich Farmbrough, Rider.cz, Rogerd, Soumyasch, Torc2, Uzume, Warren, Xpclient, 8 anonymous edits

Microsoft Transaction Server *Source*: http://en.wikipedia.org/w/index.php?oldid=542432468 *Contributors*: Alansohn, Bovineone, Bunnyhop11, Cwolfsheep, Cybercobra, Firsfron, Gaius Cornelius, JCLately, Kku, Paul Foxworthy, Pearle, Sanspeur, SimonEast, Smallman12q, Sun Creator, The Illusive Man, Winterst, Xpclient, ZooFari, 19 anonymous edits

Database schema *Source*: http://en.wikipedia.org/w/index.php?oldid=597403471 *Contributors*: Ageonix, Bunnyhop11, CRGreathouse, CWenger, Capricorn42, Cptrwizd1990, Crisis, David Gerard, DePiep, Deflective, Donose.mihai, Dprutean, Epicgenius, Ewlyahoocom, Findling67, Fluffernutter, Ginsuloft, Handstandwarrior, Hanifbbz, Hcovitz, Isheden, Island Monkey, Ivan Pozdeev, Jandalhandler, Jarble, Jason Quinn, Jerome Charles Potts, Jpfagerback, Marius, Mark Renier, Mdd, Mindmatrix, NemoNFE, Omnipaedista, Pfctdayelise, Pnm, Sae1962, Santoshijd, Shadowjams, Sugarbat, Terkaal, Vacation9, WayKurat, William-NJITWILL, Филатов Алексей, 77 anonymous edits

Oracle Database *Source*: http://en.wikipedia.org/w/index.php?oldid=598386172 *Contributors*: 2over0, ABCdba, AVM, Aarem, Abb615, Achromatic, Aditreeslime, Aednichols, Afabbro, Afed, Ahoerstemeier, Ahunt, AlanM1, Alaugt, Aleksandr Grigoryev, Alexis0509, Alexius08, Andy Dingley, AndyHassall, Annaaren74, Antandrus, Apparition11, Apyule, ArdentPerf, Arthena, Autumn Wind, AutumnSnow, Axeltroike, BBCWatcher, Basib malik, Beland, Ben Ben, BesigedB, Bezik, Bgwhite, Bhadani, Boly38, Bovineone, Brick Thrower, Brisvegas, Bronger, Bruce1ee, Brulath, Btwied, Buryshane, CTF83!, Calimo, Camw, Can't sleep, clown will eat me, Cander0000, CanisRufus, Canterbury Tail, Caphrim007, Capricorn42, Cavu2, Cbuckley, Cenarium, Chenzw, Chowbok, Chris the speller, ChrisCork, Christopher.widdowson, Chuq, ClementSeveillac, CliffC, Clowess, Coffee, Coffeehood, Comindico, Conan, ConstNT, Coolboy1234, Craig Stuntz, Crazycomputers, Cyfal, DARTH SIDIOUS 2, DFS454, DJPhazer, DMCer, DStoykov, DVdm, Damorgan, Dan100, Dancter, DanilaV, Danilo.Piazzalunga, Danmichaelo, Danroy10, Davelapo555, DavidLeighEllis, Dawnseeker2000, Deejayk, Deleteme42, Denisarona, Derek R Bullamore, Dewritech, Dexp, Dfordhiraj, Dharris, Dialectric, Dinomite, Dl2000, Dll99, Dmitry Skavish, Dp76764, DragonHawk, Dtrebbien, EagleOne, Eastsidehastings, Edward, ElTyrant, Elayaraja, Elb2000, Eli lilly, Epbr123, Excirial, Fabrictramp, Faisal.akeel, FatalError, Favonian, Fishnet37222, Frap, Frencheigh, GLaDOS, GabeIglesia, Gail, Geoff97, Gfoley4, Gilliam, Gimboid13, GnuDoyng, GoShow, Gogo Dodo, GoingBatty, GrandPoohBah, GregorB, Gwizard, Gypsum Fantastic, H a m m o, Hadal, Hadlock, Hasan uiu, Henryboltsman82, Hffmgb899, Hnapel, IcedNut, Intgr, IronGargoyle, Itai, Itbeme, J.delanoy, JLaTondre, Jack007, Jagdeepbisht89, JamesRegan, Jan.hasller, Jan1nad, Jandalhandler, Jasper Deng, Jay, Jbeacontec, Jcin617, JeremyA, Jerome Charles Potts, Jjs, Jksteiner, Jmejedi, Jnw93, John Reaves, John Vandenberg, Josh Parris, Jsaylor3, Jujutacular, Jusdafax, Justforasecond, Justin W Smith, Kadin2048, Kcordina, Keegan, KingDominik, Kishore234, Kittyhawk2, Kkm010, Kmorozov, Knowledge Seeker, Kornemuz, Kubanczyk, Kuru, Kyorosuke, Lightmouse, Linkspamremover, Linxj, Lisa Seelye, Listmeister, Loren.wilton, Lotje, Lped999, LynnRohrer, Lzer, MER-C, MGralike, Magioladitis, Mandarax, Mani1, Mannafredo, MarekMahut, Mark Renier, Martarius, Masgatotkaca, Matthew V Ball, Mattsearle, Mbobak, Mbruggen, McGeddon, MeekMark, Mehdi ab, Melissavaes, Mellissa.mcconnell, Melvynadam, MentalForager, Mfenniak, Michig, MightyWarrior, Mindmatrix, Minghong, Minimac's Clone, Minority Report, Mintleaf, Mkspies, Molas, Moreati, Mortense, MrOllie, Mschlindwein, Mwtoews, Nakon, NapoliRoma, Naudefj, Neilc, Neilrieck, Ngpd, NixonB, Nn123645, Nnemo, Norm, Now3d, Ohnoitsjamie, Ohyoko, Oicumayberight, Oleg Alexandrov, Olivier Debre, Omidbarbod, OnePt618, Oracall, OracleDBGuru, Oramoss, Oxymoron83, Oyauguru, Pacifistpanda, PanagosTheOther, Patrickjolliffe, Paul Foxworthy, Paulscrawl,

Paulvallee, Pedant17, Perftune, Peterl, Philip Trueman, Pmsyyz, Pnm, Pogson, Prabinprakash, Prathap91, Psb777, PseudoSudo, Pshuff, Ptoniolo, Puckly, Purplepiano, RJHall, Radiant chains, Raraghav, Rash009, Raysonho, Reedy, Retired username, Rhobite, Rich Farmbrough, Rich257, Riki, Rjwilmsi, Rm mainframe, Ronz, Rror, Rurik, Rv77ax, Rybec, S.K., SF007, SSTwinrova, Sagaciousuk, Sandman1142, Scha-Yorkshire, Schumi555, Seraphim, Shadowjams, Silverferns, Simon Brady, Sippsin, Skarkkai, Smyth, Sogle, Soumyasch, SpyMagician, Squash Racket, Srpnor, SteinbDJ, Stephan Leeds, Sun Creator, Suraj.kawade, SwisterTwister, TPierce, TRAINS GO CHOO CHOO, Takveen, TastyPoutine, Tate Poon, Tbsdy lives, Tcanty, Thaarkika, The Herald, The Thing That Should Not Be, The wub, Thuja, Thumperward, Tiluslaa, TimHall, Tingles, TomCerul, TommyG, Tomvenning, Tonyfaull, Touranaut, TrioteXNeo, Troels Arvin, Trojo, True Pagan Warrior, Ttwaring, TucsonDavid, Turnstep, Txomin, Useight, Vejvančický, Versageek, Viames, ViperSnake151, Vrenator, WOSlinker, WadeSimMiser, Wdchk, Welsh, Where next Columbus?, Widr, Wikieditor06, WikipedianMarlith, WikipedianYknOK, Winterst, Woohookitty, WorldAsWill, Wwwdjtoniofr, Xaje, Ylee, Yonkie, Yunus Simsek, Yzzr, ZuluPapa5, Zvar, Δ, 849 anonymous edits

Oracle Exadata *Source*: http://en.wikipedia.org/w/index.php?oldid=590658407 *Contributors*: 069952497a, Aman oracleace, Bezik, Bzorro, CCC2012, CeciliaPang, Chowbok, DePiep, Dewritech, Dpickens, Eddyleesinti, Erreid, GoingBatty, Ironholds, Jean.julius, Leendert123, LenzGr, Mild Bill Hiccup, Millim66, Mrn3, Nadimshaikh123, Netmonger, Raysonho, Sbscottw, Skamecrazy123, Sosthenes12, W Nowicki, 32 anonymous edits

Oracle Coherence *Source*: http://en.wikipedia.org/w/index.php?oldid=597669298 *Contributors*: Amniarix, Asaganich, Beland, Compfreak7, DStoykov, DixonD, Earthlyreason, Edward, Enes1177, Falsodar, Guy Van Hooveld, Hyang04, Lmatt, M4gnum0n, Mark Arsten, Mn185015, Mogism, Palosirkka, Paul Foxworthy, Paulcolmer, Rtview, Sfan00 IMG, Winterst, 5 anonymous edits

NVL *Source*: http://en.wikipedia.org/w/index.php?oldid=579702693 *Contributors*: BlueNovember, Fsmatovu, JeepdaySock, Kylebrennan1, NickelShoe, Random832, Sasawat, Tijfo098, Weedwhacker128, 12 anonymous edits

Pro*C *Source*: http://en.wikipedia.org/w/index.php?oldid=596304965 *Contributors*: Closedmouth, EagleOne, Gareng13, Joseph A. Spadaro, Kingturtle, Shamatt, 7 anonymous edits

Toad Data Modeler *Source*: http://en.wikipedia.org/w/index.php?oldid=580425549 *Contributors*: ALloydFlanagan, Berean Hunter, Claygate, Danim, Gadfium, JnRouvignac, Lukasmacek, SheepNotGoats, Stifle, Vaclav Frolik, 7 anonymous edits

SQuirreL SQL Client *Source*: http://en.wikipedia.org/w/index.php?oldid=565298506 *Contributors*: 1984, EagleOne, Ftiercel, HJ Mitchell, Kl4m-AWB, Lowellian, MRqtH2, Manningr, MickScott, Patriotic dissent, Rich Farmbrough, Satya.sid, Shdwsclan, Tgabor, TreasuryTag, 17 anonymous edits

SQuirreL SQL Client Plugin API *Source*: http://en.wikipedia.org/w/index.php?oldid=581978711 *Contributors*: Cander0000, Cedar101, Danim, DoctorKubla, Eglmainz, Elkman, Jerry, Manningr, Mild Bill Hiccup, PigFlu Oink, Plasticspork, Sir Nicholas de Mimsy-Porpington, Somewherepurple

SQLPro SQL Client *Source*: http://en.wikipedia.org/w/index.php?oldid=588520428 *Contributors*: Glenn, JLaTondre, Jesse V., Nice999, R'n'B, Woohookitty, 2 anonymous edits

Navicat *Source*: http://en.wikipedia.org/w/index.php?oldid=599257798 *Contributors*: 16@r, Anakin101, Asfreeas, Bbatsell, Bob barker rox my sox, Bunnyhop11, Cander0000, CeciliaPang, Closedmouth, Dreftymac, FSElias, Fiftyquid, Frap, JLaTondre, Jak123, Jonny-mt, LilHelpa, Manycat, Nkocharh, PatrickFisher, PhnomPencil, Premiumsoft, Qpniteflyqp, Raysonho, Redvers, Rwwww, Scarian, Sfan00 IMG, Simple Bob, The Evil Spartan, ThomasG-gPM, TommyG, Wanion, Winterst, Wongha, Смерека Михайло, 84 anonymous edits

ModelRight *Source*: http://en.wikipedia.org/w/index.php?oldid=508770958 *Contributors*: Bearcat, Drbreznjev, Magioladitis, Timguinther, Wilhelmina Will, 6 anonymous edits

MySQL *Source*: http://en.wikipedia.org/w/index.php?oldid=598593801 *Contributors*: 142.177.80.xxx, 16@r, 16x9, 1exec1, 2mcm, 4v4l0n42, A bit iffy, Aaboelela, Aaron Ray, Aaron44126, Acdx, Action potential, Adamryanlee, Addshore, Aforencich, Ahsanmani, Akopts, AlbertBickford, Aldie, Aleskva, Alexius08, AlistairMcMillan, Allencheung, Altenmann, Altered Walter, Altonbr, Amgc56, Amsfaithsoftware, Amux, Anbu121, Andrarias, Andre Engels, Andrewferrier, AndriuZ, Aneah, Ann Stouter, Antilived, Applejack1234, April Arcus, Ariel., Arite, ArmitageAmy, Artagnon, Aruton, Asenine, Astronautics, Atlantia, Attlas, Aviv007, Awfief, Axan.bulut, Axecution, AxelBoldt, Ayeroxor, Az1568, BCube, BMT, BW, BaldPark, Barefootguru, Bazzargh, Beetstra, Beetstra public, Beland, Beno1000, Bentogoa, Bevo, Bg, Bgwhite, Bkarwin, Blacktrance, Blindmatrix, Blocked!, Blue Em, Bluemoose, Boatman, Bobblewik, Bobo192, Bobsawyer20, Bobzchemist, Bofoc Tagar, Bogdangiusca, Bookbrad, Borgx, Brainsnorkel, BraneJ, Bravissimo594, BrianAker, Brianski, Briantw, Brick Thrower, Brion VIBBER, Britton ohl, Brockert, Bruggeri, Bryan, Bryan Derksen, Bunnyhop11, Burgundavia, Buryshane, Bwisey, Byte, CDV, CLW, CWenger, Callmeonnet, Calltech, CambridgeBayWeather, Camilo Sanchez, Can't sleep, clown will eat me, Cander0000, Capricorn42, CaptTofu, Captain Conundrum, Carl T, Carpetsmoker, Cbdorsett, Cburnett, Centrx, ChadMiller, ChandraSukiman, Charles Gaudette, CharlotteWebb, Chealer, Cherkash, Cheyer, Chiclops, Chiragpinjar, Chris Mitchell, Chris Wood, Cleared as filed, ClementSeveillac, Cliffashford, ClockworkLunch, Closedmouth, Codename Lisa, Coffee, Cokoli, Colindolly, Colinstu, Comp.arch, Compfreak7, Conti, Conversion script, CoolKid1993, Coolboy1234, Copito42, Coredumperror, Cosmotron, Coyote376, Craigm71, Crazyman6721, Cst17, Cybercobra, DBigXray, DStoykov, DVdm, DaBest1, Dalahäst, Dan100, Daniel.Cardenas, Danielluyo, Dank, Danyaddita, Darkfate, Darkride, David Biddulph, David H Braun (1964), Davidcopperfield123, Davidelit, Davidhorman, Db the dba, Deadcorpse, Deflective, Dejvid, Den fjättrade ankan, Dennis714, Derek Ross, Designdroide, Designer1993, Devileo, Dexp, Dfoxvog, Dfreeman@akebulan.com, Dgw, Diannaa, Diberri, DigitalSorceress, Dinnerface, Dinomite, Dittaeva, Djg2006, Dl2000, Dmillman, Dmitry Cherniachenko, DocteurCosmos, Dontdeletecontent, Doradus, DouglasCalvert, Downtown dan seattle, Dreemteem, Dsimic, Dustinhess, E0steven, Ebelular, Ed g2s, Editore99, Edward, ElTyrant, Electriccatfish2, Elwikipedista, Emwave, Engleman, Eomund, Erget2005, Ericnadler, Erilong, Erp Erpington, Exert, Eyreland, Faganp, Faisal.akeel, Fchoong, Feder raz, Fhussain, Finbarr Saunders, Flamingspinach, Fokat, FootholdTechnology, Forderud, Formulax, Frap, Freyk, Frungi, Fubar Obfusco, Furrykef, Fvw, GFHandel, GSK, Gadfium, Garo, Gary, Gavinatkinson, Gbeeker, Ghepeu, Gilad.maayan, Gilliam, Gimmetoo, Glenn, Goa103, Goffrie, Gogo Dodo, GoingBatty, Graham87, Grantmx, Grath, GreenReaper, Greenman, Greg Lindahl, GregorB, Gregoroq, Grog.Wilson, Gronky, Groogle, GroveGuy, Grymwulf, Gtg264f, Gutworth, Guycalledryan, Gwern, H@r@ld, HDCase, Haakon, Hannes Röst, Hao2lian, HarrisonFisk, Havermayer, Hcobb, HectorAE, HexaChord, HiDrNick, Holsen32, Hoover789, Hossein.ir, Hu12, Hulagutten, Hydrargyrum, Hydrogen Iodide, I already forgot, ILYA INDIGO, Imroy, Intgr, Iridescent, IronGargoyle, Irrevenant, Isotopp, IvanStepaniuk, Ixfd64, J JMesserly, JHunterJ, JLaTondre, JM.Beaubourg, Jaberwocky6669, Jakub85, James Crippen, James Plummer, Jamesday, Jandalhandler, Jarble, Jasper Chua, Jasper Deng, Jawnsy, JayW, Jayron32, JeLuF, Jeffwang, Jepafi, Jeremy Visser, JeremyCole, Jerome Charles Potts, Jeysaba, Jim1138, Jj137, Jkelly, Jmmorris, Joelrwilliams, Joelwest, Johansosa, John Vandenberg, JohnMGarrison, Jojalozzo, Jon787, Jonabbey, Jonkaz, Jonsafari, JorgeGG, Josephers, Josve05a, Juancnuno, Juiceentertainment, Julekmen, Julyscripter, JzG, KGasso, Kainaw, Kalakatha, Kalkadoon, Karan.102, Karimarie, Karl-Henner, Kbdank71, Kesla, Kgoarany, Khazar2, KillerLegend, Kipitis, Kiwi128, Kkm010, Kl4m, Kl4m-AWB, Kristof vt, Ksn, Kuru, Kvdveer, Kwamikagami, Kwiki, LA2, Larsinio, Ldonna, LenzGr, LeslieBD, Lichtgestalt, LimoWreck, LinguistAtLarge, Linkspamremover, LittleDan, Lmxspice, Locos epraix, LokiClock, Lotje, Lrusso99, Lyf2002, M4gnum0n, MK8, ML, Mac Davis, Macutty, Madduck, Magnus.de, Maheshpatil.blr, Male1979, Manning Bartlett, Manorhill, Marianolu, Marina Nastenko, MarkAtwood, MarkPilgrim, Marqmike2, Martarius, Materialscientist, Matthew-New, Matthuxtable, Mattworld, Mausy5043, Maxim, McGeddon, Mchl, Mckaysalisbury, Medovina, Mhillyer, Mhopeng, Midom, Miha Ulanov, Mike2ykme, Mikeblas, Mindmatrix, Minesweeper, Minghong, Miszobi, Mjb, Mjhoogev, Mmmready, Moeron, Mogism, Montybarker, Moondyne, MoreNet, Morgankevinj huggle, Morte, Mountaingoat, Mpopp75, Mr Minchin, Mr. Credible, MrOllie, Msjaiswal, Mushroom, Mwindrim, Mxn, MySchizoBuddy, Mz2000, N3hima, N5iln, Nanshu, NapoliRoma, Nattee, NeilSantos, Neilc, Neilrieck, Nemolan, Neocodesoftware, Neustradamus, New Thought, NicM, NickCatal, Nickdc, Nickshanks, Nicoglop, Ninestrokes, Nixeagle, Nmacu, Noerrorsfound, Nohat, Noleander, Norandav, Norm mit, Nurg, OAC, Oberiko, Ochado, Odalcet, Oegly, Ohyoko, Oleg Alexandrov, Oli Filth, Omegamormegil, Oneiros, Opticyclic, Orenburg1, Orzelf, OsamaK, Oshadmon, Palmbeachguy, Palosirkka, Parthian Scribe, Patrias, Patrick, Patsw, Pauli133, Paulvallee, Pcbene, Pedant17, Pedro, Pegasos2, Peter Hitchmough, Peter Winnberg, PeterMoulding.com, Peterl, Petri Krohn, Pgan002, Pgillman, Philip Trueman, PhilipMW, Piano non troppo, Pillefj, Pmalkowski, Pnm, Pointillist, Preeeemo, Prius 2, Qpniteflyqp, Qu1j0t3, Quadra23, Quasipalm, Qwertyus, R'n'B, Rabarberski, Radnam, Rafert, Rajasekaran Deepak, Ramu50, Raysonho, Raztus, Rbuj, Rdnk, Rdsmith4, Realkyhick, Red banana3, RedWolf, RedYeti, Reedy, Reinderien, Returnkey, Rgrof, Rich Farmbrough, RichF, Rjwilmsi, Rochkind, Rockfang, Rodhullandemu, Ronz, Rubicon, Russelljtdyer, S-n-ushakov, S.Örvarr.S, SF007, Safety Cap, Saiswa, Sajmure, Sandb, Sandeep74, Sander Säde, Sarcasting the hallucination, Satanuke, Satertek, Scarfy, ScotXW, Sdht, Seb26, Sebleblanc, Sebquantic, Secretlondon, Semi, Senator2029, Serj.by, ShakataGaNai, Shareme, ShaunMacPherson, Sherbrooke, Shlomital, Shoaler, Shortride, Sibyllaviki, Silverfish70, Simple Bob, Simplytaty, Sisutcliffe, Sjö, Skamecrazy123, SkyWalker, Slezak, Sligocki, SloppyG, Smileyborg, Smyth, Snoyes, Sobchakw, Someslowly, Sonez1113, Spikey, Spock of Vulcan, Squash Racket, Stassats, Staticmain, StefanHinz, Stephen Gilbert, SteveChervitzTrutane, SteveSims, Stevenjgarner, Stevertigo, Stevietheman, Strcat, StuffOfInterest, Subversive.sound, SummerWithMorons, Super3boy, SuperHamster, SuperMidget, Superm401, Suruena, Suwatest, Sverdrup, TJRC, Tadman, Taw, Tcnuk, Techietim, Technopat, Tedickey, Tesssla, Texture, Tgeairn, ThePlaz, Thebritt, Thecheesykid, Theone00, Theopolisme, Thesmug, Thesocialdude, Thesquaregroot, Thetorpedodog, Thomas Willerich, Thommym, Thowland, Thumperward, Tigerbomb8, Tim Starling, TimTay, Timl, Timwi, Tisane, Tkbwik, Tkynerd, TobiasPersson, Toby Douglass, Toffeepot, Tom Jenkins, TommyG, Tonusamuel, Topbanana, Travelbird, Tripathisoft, Troels Arvin, Turnstep, Twinxor, Tóraí, UKAmerican, UU, Ullika, Ultra 11, Uncle G, UnixMan45, Unknown W. Brackets, Utility Knife, VOGELLA, Vbgamer45, Vegaswikian, Veinor, Vespristiano, ViktorHaag, VincentV, ViperSnake151, Vipuser, Vivek.pandey, Vlad, Voidxor, Wdflake, Wereon, Weylinp, Whale plane, Where, Wikieditor06, Wikipelli, William Avery, Winashwin24, Winterst, Wk muriithi, Wlievens, Wolfc01, Wolkykim, Wykypydya, Xieqi200, Xiong, Xmm0, Xompanthy, XtinaS, Xumxum, Yamla, Ydriuf, YellowLeftHand, Youcantryreachingme, Yourbane, Yworo, Zacchiro, Zafar142003, Zaiken, Zero0w, Zeus, ZigArt, Zippanova, Zondor, Zzuuzz, ^demon, Ævar Arnfjörð Bjarmason, 1218 anonymous edits

SQL Anywhere *Source*: http://en.wikipedia.org/w/index.php?oldid=592519043 *Contributors*: AVM, Altruism, Audiolight, Beland, Brick Thrower, Cander0000, CesarB, ChrisGualtieri, ChrisRuvolo, Czarkoff, Diannaa, Drmahesh 06, EricGiguere, Frecklefoot, IronJohnSr, JLaTondre, Jan1nad, Jetixx93, Jkrishnaswamy, Lfstevens, Margin1522, Moontube, MrBoo, NSR, Nainawalli, PrimeHunter, Quietn, S.K., SqlPac, Stevenwagner, Ularx, Weeks1956, Weirdo12, Whimsley, 26 anonymous edits

SQLite *Source*: http://en.wikipedia.org/w/index.php?oldid=598390780 *Contributors*: *drew, 0x54097DAA, A. Parrot, AVRS, Abdull, Alex LE, Alex degarate, AlistairMcMillan, Alvin-cs, Amanda hoic, AndrewHowse, Anybody, Aquatopia, Arcturis, Arite, Arulprasad, Aulis2003, AxelBoldt, Baest, Barefootguru, Bbb2007, Beethoven05, Beeveer, Bgwhite, Bigown, Bkkbrad, Boleslav Bobcik, Bro4, Brockert, Bryan Derksen, C. A. Russell, CWenger, Canadianacademic, Cander0000, Captain Conundrum, CeciliaPang, Chealer, Choas, Chris Caven, ChrisNoe, Claw of

Slime, Cmbeelby, Compctech, Cooldude7273, DG, DStoykov, Damian Yerrick, DanBishop, Danim, Darkzen.bps, Dave2, David Gerard, Daxx wp, Dchestnykh, Deflective, Delpino, Devin Asay, Dexp, DivideByZero14, Djbriddock, DmitTrix, Dnas, Drano, Drhipp, Dsimpson dcsi, Ebraminio, Eequor, Egrabczewski, Eugenwpg, FFMG, FatalError, Fenke, Fnielsen, Frap, Fred Bradstadt, Fubar Obfusco, Furrykef, Gabrielepx, Gerardo cabero, Gilesmorant, Gioto, Glenn, Glider87, Graue, Greensburger, GregorB, Grshiplett, Gudeldar, Gwern, Götz, H@r@ld, Haakon, Hankwang, Hareesh.t, Havarhen, Heelmijnlevenlang, Hif, Hoo man, Hu12, Huji, Inverse.chi, Irishguy, Isilanes, J Milburn, JLaTondre, Jaguar83, JamesHaigh, Jandalhandler, Javier Odom, Jaxelrod, Jayapalchandran, Jeanfarias, Jmkim dot com, Johnuniq, JonathanWakely, Jorel314, Jrouquie, Jstaniek, KTC, Kate, Kimchi.sg, Kirbylover4000, Kl4m-AWB, Korval, Kriplozoik, Kwamikagami, Kwiki, Leonelhs, Lethalmonk, Lfstevens, LilHelpa, Lmatt, LordArtemis, Lotje, Luk, MBParker, MBlakley, MVelliste, Madooo12, Magicmike, Magister Mathematicae, Mainstreetmark, Makyen, Many delicious meats, Marco1317, Marco1713, Mark Arsten, Marvinella, Materialscientist, Mattisgoo, Mchirico, Mdwh, Mecanismo, Melizg, Mindmatrix, Minghong, Mipadi, Molyneux.peter, Moxfyre, MrOllie, Mrinal.kant, Mschumacher69, Mwtoews, Najeeb1010, Nate Silva, Ned Scott, Neilc, Neurolysis, Niceguyedc, Nikai, Nnemo, Noncoercive, Nuno Brito, Od1n, Ohconfucius, Optikos, Otterfan, OzOle, PabloCastellano, Palosirkka, Paolosupernova, Pascal sa, Patuck, Paul Magnussen, Peak, Permafrost46, Philipolson, Philippe.petrinko, Pmsyyz, Pnm, Polar, PotatoEater, Qu1j0t3, Quadunit404, Qwyrxian, RadioActive, RandalSchwartz, Rapha222, Reedy, Rfl, Rgbea, Rich Farmbrough, Risk one, RobertCailliau, Roberta F., Ronark, Rorx, RossPatterson, Rrjanbiah, Ruakh, Rursus, SF007, Sae1962, Saltlakejohn, Sanxiyn, Sappy, Sarathbs, Scartboy, Scjessey, Scullder, Sdfisher, Sekelsenmat, SexyBern, Shello, SimonEast, Simple Bob, Sleske, Slicky, Smyth, Soumyasch, SpacePacket, Specious, SqlPac, Srperez, Stachelfisch, Staticmain, Steph1978, Stephan Leeds, Streambag, Superm401, Svick, Syntaxerrorz, TOR, Tablizer, TankMiche, Temblast, Tgwena, Theone00, Therealgeeves, Theta682, Thumperward, ThurnerRupert, Tim baroon, Timwi, Tobias Conradi, Toehead2001, Tom Jenkins, TripleF, Turnstep, Two Bananas, U1024, Ungzd, Vdvluc, Visor, Vlops, Volkris, Wael Ellithy, Waldir, Where, Who.am.i, Wikimercenary, Wikitiki89, Wikiwikiwu, Winterst, Wjgilmore, Wwwwolf, Wykypydya, Xephyrous, Xtremejames183, Ybungalobill, Yoghurt, Zaxn1234, Zbig Lebowsky, ZeaForUs, Zero0w, Zhongchen, Zo8shong, 469 anonymous edits

SESAM (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=477031909 *Contributors*: Halloleo, Jimmy Pitt, Magioladitis, Malcolmxl5, Rangoon11, Shdwsclan, Woohookitty

LAMP *Source*: http://en.wikipedia.org/w/index.php?oldid=597392846 *Contributors*: 2mcm, 5alam83, AJR, Abtin, Abune, Accrisoft, Acjelen, Ahunt, AlexKarpman, AlistairMcMillan, Amolshah, Andreas Kaufmann, Anthonypants, Apnicsolutions, Arthur Rubin, Bill.D Nguyen, Bkkbrad, Blanford robinson, Bomazi, Brian Kendig, BrianJones, Bunnyhop11, Bytebear, Cander0000, Captain Conundrum, ChaosR, Chimin 07, Cjcollier, Codename Lisa, ColdFusion650, Cradam, Csabo, Cybercobra, Davidjcmorris, Davidkazuhiro, Dcirovic, Deepakr, Den fjättrade ankan, DexDor, Djg2006, DoohanOK, DoorsAjar, Dragon8Fire, Dsimic, Durrantm, DwayneP, El Cubano, Electrolite, Elephant in a tornado, Eptin, Eyu100, Faisal.akeel, Farooqakbarkhan, Fences and windows, Filceolaire, ForumJoiner, FrYGuY, Frap, Frbe, Fred Bradstadt, Freefox, G.armellin, Gardar Rurak, Gareth Griffith-Jones, GargoyleMT, Generalmiaow, Geordiecoder, George Leung, Ghettoblaster, Gigs, Gracehoper, GraemeL, Graham87, Gronky, Gryllida, Haakon, Haidut, Haysead, Htmlapps, Issinoho, Ivansanchez, JHunterJ, JLaTondre, Jamse, Jasper Deng, Jeffrey O. Gustafson, Jerryobject, Jimcreate, Jlee1973, Jmg157, Johan D'Hondt, John.szucs, Joncojonathan, Jwarhol, K7.india, KTC, Kanags, Karnesky, Kbrose, Kent zacharias, Kingflamz, Kl4m-AWB, Komarov om, Korny O'Near, Korte, Kozuch, Kuukunen, Kylex3y, Lethe, LicenseFee, Liftarn, LilHelpa, Løde, MZMcBride, Mahanga, Mariuz, Martarius, Marudubshinki, Materialscientist, Mayazcherquoi, Melody Lavender, Mfrisk, Mhoenig, Michael Slone, Michael.Urban, Minghong, Mirokado, Morn, MrOllie, MusikAnimal, Mzajac, Najoj, NapoliRoma, Nate1481, Nbarth, Nealmcb, Neustradamus, Nicblais, Nick Wallis, Nigelj, NoClutter, Nono64, Northernhenge, Nuwewsco, Patrick, Peaceray, Pearle, Pengo, Pepve, Peterdjones, Peteresch, PhilKnight, Pinecar, Pjacklam, Pmc, Pne, Pnm, Prophile, Quantum Burrito, QuiteUnusual, RadaVarshavskaya, Raffaele Megabyte, Raggiskula, RandalSchwartz, Reedy, Requiem18th, Rich Farmbrough, Rjmars97, Rjwilmsi, Rkarlsba, Robert K S, Ronz, Rosettesor, Ryandsmith, S Carpenter, Salgueiro, SamJohnston, Samkass, Samwaltz, ScotXW, Sfermigier, Shanti subr, Sherbrooke, Sir Nicholas de Mimsy-Porpington, Skarebo, Slady, Sligocki, Smalljim, Snowolf, Spartanicus, Staalmannen, Stickee, Stu42j, Superm401, Supernerd, Suruena, Szlampy, TJRC, Taejo, The wub, Thine Antique Pen, Thumperward, ThunderPeel2001, TiTsaPopolous, Timosa, Tobias Conradi, Tomhannen, Toussaint, Trungie, Txomin, UU, Utcursch, Vasiliy Faronov, Vdegroot, Veinor, Vicsar, Visor, Vorik111, Waltloc, Wangi, Welshmike, WhiteEcho, Wikidrone, Wrinklygrandma, Wshato, Xaje, Yosofun, ZacBowling, Zalmoxe, Zcbeaton, Znmeb, 419 anonymous edits

Watcom SQL *Source*: http://en.wikipedia.org/w/index.php?oldid=499225042 *Contributors*: CanisRufus, Gaius Cornelius, GregorB, Mark Renier, Mdchachi, NatusRoma, Rettetast, Rich Farmbrough, TheParanoidOne, Thecheesykid, Weirdo12, 1 anonymous edits

Microsoft Access *Source*: http://en.wikipedia.org/w/index.php?oldid=598125180 *Contributors*: 144.132.70.xxx, 1stContact, 24.15.135.xxx, 28bytes, A. B., Aamfk, Aaronhill, Acebulf, After Midnight, AgentCDE, Ahoerstemeier, Ajcomeau, Alansohn, Alaudo, Albert Kallal, Alcmaeonid, Aldaron, AlexDybenko, Alexander.hugh.george, Alexdyb, AlistairMcMillan, Alzarian16, Amaljoseph24, Amelia Hunt, Amicron, Amiodarone, Anonymoues, Antalas, Antandrus, Arctic Kangaroo, Arosa, Aruton, Atkinsdc, Axeltroike, BWCNY, Barcelona Wiki, Belovedfreak, Bevo, Binary TSO, Bobo192, Boggie, Borgx, BraneJ, Bug42, CC90, Cae prince, Cander0000, Chad01, Charles Baynham, Chillum, Chuunen Baka, Cinnamon42, Ckredo, Cleared as filed, ClickRick, Closedmouth, Cmc87, Cmdrjameson, Cobraman156, Cocoma, Codename Lisa, Coffeehood, Coldacid, Colinstu, Comatmebro, CommonsDelinker, Conversion script, Corn cheese, Courcelles, Cp111, Ctkeene, CutOffTies, Cwolfsheep, Cyclonenim, D0762, DanielRigal, DataAnalyzer, David Jordan, Davidshq, Dcirovic, Deathtreath93, Decltype, Dekisugi, Delpino, Deon, Deor, Derumi, Diamondland, Dina, Discospinster, Djm1279, Djmckee1, DocWatson42, Dogloverxyz, Donner60, Doradus, Dpritchett0320, Draconxanathos, Dreammaker182, Dureo, Dylan Lake, Dysepsion, Dysprosia, ERcheck, ESkog, ETomeny, Ed Poor, ElKevbo, Ellmist, Elm-39, Endtoend, Epbr123, Erik Zachte, Excirial, ExpertDBDev, FayssalF, FleetCommand, Fluffernutter, Flyer22, Forenti, Fraggle81, Frap, Fred Bradstadt, Fredrick day, FrenchIsAwesome, FreplySpang, Fyrael, GL1zdA, GNRY09, Gadfium, Gary, Gaudoine, Gavin Lisburn, Genesispc, Georgeryp, Gerald, Ghettoblaster, Gilliam, Ginsuloft, Glenn, Goa103, Grafen, Greenrd, GreyCat, Gscshoyru, HMman, Hahaha07, Hasek is the best, Hawaiian Eskimo, Hellknowz, Hj3973, Hu, Hydrogen Iodide, I dream of horses, IGod, II MusLiM HyBRiD II, IW.HG, Immunize, Intgr, Iridescent, J. Finkelstein, J.delanoy, J04n, JHunterJ, JLaTondre, JackWilfred, Jackmcbarn, Jackz, Jaeger5432, Jaksmata, JamesBWatson, Jamesmorrison, Jasper Deng, Javierluraschi, Jay, Jevmonteron98, Jhenderson777, Jinkinson, Jj0909jj, Jnl001, JoanneB, John Quincy Adding Machine, John of Reading, Johnleemk, Jojhutton, Josh the Nerd, Jstaniek, Juicycat, Jwoodger, Jykosu11, KD5TVI, Kainaw, Kanags, Kassie, Kataluveno, Katharineamy, Katieh5584, Kawasemi, Kerotan, Ketiltrout, Kevyn82, Khazar2, Khlo, King of Hearts, Konstable, Krich, Kuru, Kwiki, Kyle Rayner, Latha P Nair, LauraFarina, LeaveSleaves, LedgendGamer, Lfstevens, Libcub, Liftarn, Limbojones, Linkspamremover, Linuxerist, Lofote, Logan, Lollol07, Lou.weird, Lowellian, Luci Sandor, Luna Santin, Lupin, M4bwav, M7, MER-C, Ma'ame Michu, Mal Browncoat, Manzell, Mark Arsten, MarkGallagher, MartinTallett, Mary Spencer 20, Marygabq, Materialscientist, MattieTK, Mcornelius, Melaen, MetaManFromTomorrow, Metatinara, MetsFan76, Mewkalewk, Mggpublishing, Michig, Mika1h, Mike12345678, Mikebrand, Millennas, Mindenn123, Minghong, Modster, Mopza, Morwen, Moverton, Mr. Ash, Mr. James, Mr. Stradivarius, MrJones, MrOllie, Mushroom, Mwtoews, My name is not dave, Myanw, Nae'blis, Nainawalli, NamelessDude49, NawlinWiki, Necessary Evil, Neilc, Neko-chan, Nexusdb, Nico5038, Njaelkies Lea, Norm, North8000, Nwt, Oberiko, Ohei, Olivier Mengué, Omicronpersei8, Ophir, Orrc, Paul Magnussen, Peter Delmonte, Peterl, Ph.eyes, Phlegat, Piano non troppo, Pinkadelica, Pixi, Pizza1016, Pjb007, Poor Yorick, Pvercello, R'n'B, Rabbit994, Rajpaj, RandomLittleHelper, RandomStuff, Razimantv, Rdrs, Reach Out to the Truth, Rebroad, RedKnight7, Reedy, Revmachine21, Reyhaanbinny, Reywas92, Rezonansowy, Rfdparker, Rhobite, Riki, Rjwilmsi, Rklawton, Robbie andrew, RobertG, Rodgerbeard, RogerJennings, Rprpr, Rrabins, Rsocol, Ryan Norton, Ryanmay123455, SDSWIKI, SJP, Sa145, Saintyboy666, SaltyBoatr, Sam Hocevar, Sam Korn, Samuel Blanning, Sandstein, Santiagoagentino, Schmiteye, Schultkl, SchuminWeb, Scottmcd9999, Sfgiants1995, Shadowjams, Shinjiman, ShotoSensei, Silvergoat, SimonEast, Sirgshar, Sjakkalle, Skittleys, Sligocki, Soumyasch, SqueakBox, Stephan Leeds, Stephenb, Stephenchou0722, Stinerman, Stolsvik, Suffusion of Yellow, Surturz, Surv1v4l1st, Sushi500, TAIntedCHInese, THEN WHO WAS PHONE?, Ta bu shi da yu, Tagus, Tanaats, Taruz, Tbhotch, Ted.macneil, Template namespace initialisation script, Teneriff, Texture, ThaProjects THA, The Anonymouse, The Dark Melon, The High Fin Sperm Whale, The Thing That Should Not Be, The Witty Warrior, The stuart, TheProject, Themodernizer, Theslick50, Thinknirmal, This, that and the other, Thnder, Tide rolls, Timneu22, Tkbwik, Tmopkisn, Todorojo, Tohd8BohaithuGh1, Tom harrison, TommyG, Troels Arvin, Tuanese, Tyomitch, Tyw7, Ubruni, Uhai, Uildriks, Ulric1313, Uncle Dick, Uncle Gizmo, UnitedStatesian, Useight, Utcursch, Valdez123, Vanished user 9i39j3, Versus22, Vicenarian, VictorReinhart, Violetriga, W3stfa11, WDavis1911, Warren, Wheat, Whiskerz, Widr, Wiki alf, Wiki fanatic, Wikinator2008, Winterst, Witharebelyell, Wknight94, Wmahan, Wugzr, Wylve, WysardOfInformation, Wysprgr2005, Xpclient, Yidisheryid, YuTanaka, ZacBowling, Zackaback, Zimbabwer, Zippy1981, Zwikipenguin, Zzuuzz, Ævar Arnfjörð Bjarmason, Тиверополник, أحمد مصطفى السيد, , 1065 anonymous edits

VMDS *Source*: http://en.wikipedia.org/w/index.php?oldid=544182184 *Contributors*: Anticipation of a New Lover's Arrival, The, Beland, Borgx, Cmdrjameson, David6.jenkins@ge.com, Edward, Gaius Cornelius, GreatWhiteNortherner, LilHelpa, Uthbrian, Woohookitty, Wuub, 11 anonymous edits

Superbase database *Source*: http://en.wikipedia.org/w/index.php?oldid=577116980 *Contributors*: Danim, Databaseg, Dthomsen8, Jpbowen, Nono64, Ntsimp, Polluks, Raffaele Megabyte, RufusG, Saadahmad, Soilcreep, Superbaseguru, SymlynX, Tabletop, Targaryen, UnitedStatesian, Wikiwriter80132, 16 anonymous edits

Rocket U2 *Source*: http://en.wikipedia.org/w/index.php?oldid=579631123 *Contributors*: Cander0000, Charleca, CharlesBarouch, Cybercobra, Danim, Danmcg.au, Discospinster, Dpetersco, Edd Porter, Elegy, Gareth Griffith-Jones, Hlala, JackieBurhans, Jerryobject, Kemipa, Klemen Kocjancic, Kuru, Mannyneira, Mcclarke, MichaelPreece, Mikaey, Mikeyboysan, Nonnb, Pcb21, Rich Farmbrough, Rory096, RoyGoldsmith, Rwwww, Sasquatch, Shelvin, Some Wiki Editor, Stuboy, Trausche, Vrenator, Wagesj45, Wjhonson, Xdamr, Zouf, 36 anonymous edits

PointBase *Source*: http://en.wikipedia.org/w/index.php?oldid=482257626 *Contributors*: Cander0000, DinosaursLoveExistence, Guoguo12, Polaroidforever

R:BASE System *Source*: http://en.wikipedia.org/w/index.php?oldid=581862606 *Contributors*: Boothy443, Cander0000, Choster, Chris the speller, DGG, Dank, David Jordan, Eronbo, Glenn, HEAdrian, Jesse V., Matthiaspaul, Mogism, Pichpich, ProveIt, Razzakmemon, Reiknir, Remember the dot, Rich Farmbrough, Squeakyfrommage, TJSwoboda, Tassedethe, Twp, Zanglazor, Zodon, 42 anonymous edits

REAL Server *Source*: http://en.wikipedia.org/w/index.php?oldid=530578738 *Contributors*: Afoleyreal, Azylber, Cander0000, CommonsDelinker, Danamay29, J04n, LarryJeff, Lightlowemon, Mdanabrown, Sqlabs, Thorncrag, Trilobyte fossil

MaxDB *Source*: http://en.wikipedia.org/w/index.php?oldid=592457926 *Contributors*: Ahunt, AlistairMcMillan, Andareed, Bobprime, Cander0000, Chillywillycd, Cjcollier, Clowess, Cool Hand Luke, Craesh, Deeahbz, DocendoDiscimus, Doco, Dreadstar, Edward nz, Gurch, JasonLax, Jerome Charles Potts, Jlundell, Joel Alcalay, John Vandenberg, Jon vs, Jonah.harris, Jtdirl, K1Bond007, Karnesky, Kent Wang, Kingdon, Kkm010, Knaveofhearts, Minghong, Mogism, Mwarren us, Nainawalli, Neilc, Nihiltres, Nono64, Odie5533, Pnm, Reedy, Rjwilmsi, Ryan Norton, Seb35, Sharcho, Skychutw, Sleske, Technobadger, Tilman, Tim baroon, Unyoyega, WatchAndObserve, Who, Yayabobi, Zero0w, Zondor, 40 anonymous edits

Adaptive Server Enterprise *Source*: http://en.wikipedia.org/w/index.php?oldid=583241110 *Contributors*: Aednichols, Ary29, AutumnSnow, Bobthecow, Bovineone, Cander0000, Chlor, ChrisGualtieri, Corti, Curmi, Echion2, Edward, FlorencePS, Highspam, Himan0110, Jerome Charles Potts, John of Reading, Kgf0, Khalid hassani, Leranedo, Lohi8, LrdChaos, Mdchachi, Michael Hardy, Minghong, Nyseans, Ohedland, Opticyclic, Osmeier, Phatom87, Samwb123, Timo Honkasalo, Troels Arvin, W Nowicki, Will henderson, Wlh197, Xezbeth, Y work, 49 anonymous edits

Advantage Database Server *Source*: http://en.wikipedia.org/w/index.php?oldid=574284030 *Contributors*: Cander0000, DMacks, Databaseguy, Decompiled, Doctorfluffy, Kbrumback, Mdd, SchreiberBike, 9 anonymous edits

Ingres (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=596269236 *Contributors*: Abhijitpai, Anastrophe, AutumnSnow, AxelBoldt, Bobo192, Bruce1ee, Cander0000, ChrisGualtieri, Codwangler, Coherers, Cosmiquemuffin, Craig Stuntz, Cxbrx, Cybercobra, Deflective, Dmsar, Eduemoni, Elf, Frap, Frecklefoot, G Russellis, Grantc, Guilhem06, HoundsOfSpring, Hu12, Hydrargyrum, ISC PB, Isnow, J mareeswaran, Jahowell, Jamelan, Jan Hidders, Jnc, Joy, Jpzuate, KB7FUN, Kbjo, Kwamikagami, Kwiki, Larry.Rowe, Leandrod, Leandrpf, Levin, Lfstevens, LjL, Lowellian, Lox, MacTed, Mahalath8, Mandarax, Marasmusine, Mark Arsten, Marqueed, Maury Markowitz, Mhkay, Mikeubell, Mild Bill Hiccup, Minesweeper, Minghong, MuffledThud, Mullen.rob, Natalya, Neilc, Norm mit, Osbootcamp, Palosirkka, Pete.rabjohns, Philu, PigFlu Oink, Pvercello, Quelgeek, Qwertyus, Raysonho, RedWolf, Reedy, Rich Farmbrough, Rjwilmsi, Rojomoke, Rursus, SF007, Sargdub, SchreiberBike, Seashorewiki, Shusseina, Skizzik, Spatialguru, Sydbarrett74, Tabletrack53, Tedder, The Anome, The Thing That Should Not Be, The sock that should not be, Thebrid, Thomas.uhl, Thumperward, TimeTravellingMonkey, Tin, Todd Vierling, TommyG, Troels Arvin, Tuhl, Vdo2000, W Nowicki, Wetman, Where, Wik, Wikid77, William Avery, Yvesnimmo, Zero0w, 145 anonymous edits

Data redundancy *Source*: http://en.wikipedia.org/w/index.php?oldid=594981161 *Contributors*: AgadaUrbanit, Ashley thomas80, Bearcat, Boli1107, Captain-tucker, Chriskl, DanDanRevolution, Dave Braunschweig, Dingar, Donsaves, Egydarceyes, Ettrig, Fiftytwo thirty, Garion96, GregorB, Hu12, Informedbanker, Ja 62, JaGa, Ksbrown, Malcolma, MilerWhite, MilfordBoy1991, Nelson Nanataktuk, Patrick, Paul August, Pnm, RC Master, Rd4ever4u, Rebroad, Tim Goodwyn, Tommy Kronkvist, Uršul, Yelloeyes, 65 anonymous edits

Database normalization *Source*: http://en.wikipedia.org/w/index.php?oldid=595742341 *Contributors*: 1exec1, 4pq1injbok, A3 nm, ARPIT SRIVASTAV, Ahoerstemeier, Akamad, Akhristov, Alai, Alasdair, Alest, Alexey.kudinkin, Alpha 4615, Amr40, AndrewWTaylor, Antonielly, Anwar saadat, Apapadop, Arakunem, Arashium, Archer3, Arcturus, Arthena, Arthur Schnabel, Ascend, AstroWiki, AubreyEllenShomo, Autocracy, AutumnSnow, Azhar600-1, BMF81, Babbling.Brook, Bernard François, Bewildebeast, Bgwhite, Billben74, Billpennock, BillyPreset, Black Eagle, Blade44, Blakewest, Blanchardb, Bloodshedder, Blowdart, BlueNovember, BlueWanderer, Bongwarrior, Boson, Bovineone, BradBeattie, Brick Thrower, BrokenSegue, Bruceshining, Bschmidt, Bugsbunny1611, BuzCo, CLW, Callavinash1, Can't sleep, clown will eat me, Chairboy, Chrislk02, Citral, Cl22333, CodeNaked, Combatentropy, Conversion script, Creature, Crenner, Crosbiesmith, DARTH SIDIOUS 2, Damian Yerrick, DanMS, Dancraggs, Danim, Danlev, Datasmid, David Colbourn, DavidConrad, DavidHOzAu, Davidhorman, Dean001, Decrease789, Demosta, Denisarona, DerHexer, Dfass, Dflock, Discospinster, DistributorScientiae, Doc vogt, DocRuby, Docu, Don Hammond, Doud101, Dqmiller, Dreftymac, Drowne, Dthomsen8, Duke Ganote, Ed Poor, Edward Z. Yang, Eghanvat, Elcool83, Electricmuffin11, Elwikipedista, EmmetCaulfield, Emperorbma, Emw, Encognito, Enric Naval, Epepke, Eric Burnett, Escape Orbit, Ethan, Evilyuffie, Ewebxml, Falcon8765, Farquaadhnchmn, Fathergod, FauxFaux, Fieldday-sunday, Fireman biff, Flewellyn, Fluffernutter, Fmjohnson, Fraggle81, Fred Bradstadt, Furrykef, Gadfium, GateKeeper, Gilliam, Gimboid13, Ginsuloft, Gk5885, Gogo Dodo, Gottabekd, Gregbard, GregorB, Groganus, Gustavb, Guybrush, HMSSolent, Hadal, Hairy Dude, Hanifbbz, Hapsiainen, HbJ, Hbf, Heracles31, HiDrNick, Hoo man, Hu12, Hydrogen Iodide, Hz.tiang, Ianblanes, IceUnshattered, Imre Fabian, Inquam, Intgr, Jadvinia, Jakew, James086, JamesBWatson, Jamesday, Jamesjusty, Jan Hidders, Japo, Jarble, Jason Quinn, Javert16, Jdlambert, Jgro, Jjjjjjjjjj, Jklin, Joness59, Joseph Dwayne, Jpatokal, Jpo, Justin W Smith, KAtremer, KathrynLybarger, Keane2007, Keegan, KevinOwen, KeyStroke, Keyvez, Kgwikipedian, Kingpin13, Klausness, Kushalbiswas777, L Kensington, L'Aquatique, LOL, Larsinio, Lawrence Cohen, Leandrod, Lee J Haywood, Legless the oaf, Leleutd, Leotohill, Lerdthenerd, Les boys, Lethe, Libcub, Lifeweaver, Linhvn88, LittleOldMe, Longhair, Lssilva, Lujianxiong, Lulu of the Lotus-Eaters, Lumingz, Luna Santin, M4gnum0n, MER-C, Magantygk, Manavkataria, Mark Renier, Marknew, MarownIOM, MartinHarper, Masterstupid, Materialscientist, Matmota, Matthew 1130, Mckaysalisbury, Metaeducation, Michael Hardy, Michalis Famelis, Michealt, Microtony, Mike Rosoft, Mikeblas, Mikeo, Mindmatrix, Miss Madeline, Mjhorrell, Mo0, Modeha, Mooredc, Mpd, Mr Stephen, MrDarcy, MrOllie, Nabav, NawlinWiki, Nick1nildram, NickCT, NoahWolfe, Nocat50, Noisy, Northamerica1000, Nsaa, NubKnacker, Obradovic Goran, Ocrow, OliverMay, Olof nord, Opes, Oxymoron83, Pagh, Peachey88, Pearle, Perfectblue97, Pete142, Pharaoh of the Wizards, Phil Boswell, Philip Trueman, Pie Man 360, Pinethicket, Plastic rat, Polluxian, Prakicov, ProveIt, Purplepiano, Quarl, RB972, RBarryYoung, RadioFan, Railgun, Rathgemz, Rdsmith4, Rdummarf, RealityApologist, Reedy, Regancy42, Reinyday, Remy B, Reofi, RichF, Rjwilmsi, Robert McClenon, Robomaeyhem, Rockcool19, Rodasmith, Romke, Ronfagin, Rp, Rumplefish, Ruud Koot, Ryulong, Sam Hocevar, Sasha.sheinberg, SchuminWeb, ScottJ, Scwlong, Seaphoto, Sfnhltb, Shadowjams, Shakinglord, Shawn wiki, Shreyasjoshis, Shyamal, Silpi, Simeon, Simetrical, Sixpence, Skritek, Smjg, Smurfix, Snezzy, Snigbrook, Socialservice, Sonett72, Soulpatch, Soumyasch, Spacesoon, Sstrader, Stacyshaelo, Stannered, Starwiz, Stephen e nelson, Stephenb, SteveHL, Stifle, Stolkin, Strike Eagle, Sue Rangell, Superjaws, Sydneyw, Sylvain Mielot, Szathmar, Taw, Tbhotch, Tcamacho, Tedickey, Teknic, Tgantos, Thane, The Thing That Should Not Be, The undertow, The1physicist, Tide rolls, Titofhr, Tobias Bergemann, Toddst1, Tom Lougheed, Tom Morris, Tommy2010, Toxicwaste288, Traxs7, Troels Arvin, Turnstep, Twinney12, Tyc20, Unforgettableid, Upholder, Utcursch, Vald, Valdor65, Vampyrium, VanishedUserABC, Velella, VinceBowdren, Vladsinger, Vodak, Voidxor, Waggers, Wakimakirolls, Wammes Waggel, Wavelength, Wexcan, WikiPuppies, WikipedianYknOK, Wildheat, Wilfordbrimley, Wilsondavidc, Winterst, Wjhonson, Woohookitty, WookieInHeat, Xiong Chiamiov, Xiroth, Yong-Yeol Ahn, Zedla, Zeyn1, Zhenqinli, Zzuuzz, 石庭豐, 1355 anonymous edits

Functional dependency *Source*: http://en.wikipedia.org/w/index.php?oldid=598420072 *Contributors*: 1ForTheMoney, Armagedescu, AutumnSnow, BartVB, Cdrdata, ChrisGualtieri, Citral, Crosbiesmith, D6bmg, EagleFan, Edward, Ejrh, Eric22, Favonian, FireyFly, Fragment, Ganesh121292, GeorgeBills, Graham87, Grassnbread, HenningThielemann, Hut 8.5, Igor Yalovecky, Iridiumcao, J. M., Jan Hidders, Jleedev, Jwp, KelvSYC, KeyStroke, Kushalbiswas777, LOL, LilHelpa, MaBoehm, Mark Renier, MartinWaite, Matthiaspaul, MaxDel, Michael Hardy, Michaelcomella, Mroberts297, Notheruser, Odsh, Pdfpdf, Porges, Poromenos, Prathik Rajendran M, Qetuth, Ruud Koot, Saravu2k, Slavy13, Spamduck, Thebulbs, Tijfo098, ValuableAppendage, VinceBowdren, Waitak, Wilson44691, Yintan, Zero0000, Zrisher, 107 anonymous edits

Armstrong's axioms *Source*: http://en.wikipedia.org/w/index.php?oldid=599225019 *Contributors*: A3 nm, Aednichols, Andonic, Arosa, CBM, Can't sleep, clown will eat me, Charles Matthews, ChrisGualtieri, Cogentleman, Cornellcloud, Entropeter, Inklein, Jh559, Jonemerson, Joseph Dwayne, LouI, Mark Renier, Meng6, Mentoz86, Paolo Serafino, Q-lio, Telofy, Tijfo098, Vegpuff, Wavelength, 65 anonymous edits

Transitive dependency *Source*: http://en.wikipedia.org/w/index.php?oldid=579621987 *Contributors*: Boson, Equendil, Jeff Wheeler, Jiaoziren, Jwalantsoneji, Malcolma, Nabav, ProcerusDecor, Thine Antique Pen, Woohookitty, 14 anonymous edits

Superkey *Source*: http://en.wikipedia.org/w/index.php?oldid=587816569 *Contributors*: AndrewWarden, Anog, AutumnSnow, Boson, CeleronNutcase, CharlotteWebb, ChrisGualtieri, ColinFine, Crosbiesmith, Dawynn, Fatherlinux, Fimp, Igor Yalovecky, IronGargoyle, James Crippen, Jan Hidders, Jorge Stolfi, Jusdafax, Jwulff, Katieh5584, KeyStroke, Kranix, LOL, Larsinio, M. Frederick, Magioladitis, Mark Renier, Metron4, Michaelcomella, Mikeblas, Millermk, Mindmatrix, Nabav, Pimlottc, ProcerusDecor, Reedy, Rhoerbe, SpuriousQ, Sss41, Stbrob, The Thing That Should Not Be, TheParanoidOne, Tobias Bergemann, Torzsmokus, Twarther, Voidxor, Welsh, Wikitanvir, Yay unto the Chicken, Zzuuzz, Ox, Δ, 石庭豐, 80 anonymous edits

First normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=598716684 *Contributors*: Aeonx, Alansohn, Alphama, Alxndr, Ambuj.Saxena, Arctic Kangaroo, Bernard Ladenthin, BillyPreset, Boson, Brianga, Brick Thrower, Burner0718, Closedmouth, Crosbiesmith, DanielLemire, Davidhorman, Dfass, Dixtosa, Dougher, Dreftymac, Eallik, Ebraminio, Eibcga, Flyer22, Funnyfarmofdoom, General Wesc, GermanX, GregorB, Gwernol, Hamidrizeh, Heathcliff, Hobsonlane, Isnow, Jacobolus, Jason Quinn, Jerome Charles Potts, Jgzheng, John of Reading, Jordan Brown, Klilidiplomus, Kwetal, LarRan, Lordmwesh, M.r santosh kumar., Mahlon, Mark Renier, Materialscientist, Mfpinhal, Montchav, Morra, Mystagogue, Nabav, NawlinWiki, RBarryYoung, ReformatMe, Rhododendrites, Saravu2k, SilverbackNet, Vegpuff, VictorAnyakin, VinceBowdren, Whitmerj, 파괸, 205 anonymous edits

Second normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=597540945 *Contributors*: Ak786, Allens, Apugazh, Benjamin.Cramphorn, Bernard Ladenthin, Boson, Btilm, Carlhoerberg, Chrislk02, Crosbiesmith, DARTH SIDIOUS 2, DVdm, Don Hammond, Dougher, ESkog, Ebraminio, Fraggle81, GermanX, Glane23, GregorB, Haffasoul, Ijliao, IronGargoyle, Jason Quinn, Javert16, Jerome Charles Potts, JianzhouZhou, JimpsEd, Jprg1966, Jsharpminor, Mark Renier, Materialscientist, Mike Rosoft, Mordashov, Nabav, RBarryYoung, Sanchitideas, Saravu2k, Shnako, Shreyasjoshis, Snehasapte, SqlPac, Svick, Uncle Dick, Velella, VinceBowdren, Whitmerj, 파괸, 106 anonymous edits

Third normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=597541027 *Contributors*: Abe149, Alvin-cs, Amalthea, Anabus, AndrewWarden, Arcturus, Azrich, Bernard Ladenthin, Blahma, Boson, Bxn1358, CapitalR, Centrx, Codeculturist, DVdm, Diego Moya, Don Hammond, Dorfl, Dougher, Ebraminio, Edward Z. Yang, Electriccatfish2, Furrykef, Garde, GermanX, Gingerjoos, Gwen-chan, Ijliao, Jason Quinn, Jcsalterego, Jitse Niesen, Joseph Dwayne, Jswhitten, Kitkatbeard, Leasabp, MeekMark, Michalp, Michealt, Mike Rosoft, MsHyde, Nabav, Natural Cut, Ollie, Pinethicket, Roberticus, Saravu2k, Semaphorite, Shnako, Shreyasjoshis, Sleske, Someusername222, THEN WHO WAS PHONE?, Thingg, Toyota prius 2, USConsLib, Unara, Vegpuff, VinceBowdren, Vlad2000Plus, Wavelength, Whitmerj, Wikimiro, Willking1979, Wyadbb, 101 anonymous edits

Boyce–Codd normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=598514943 *Contributors*: Abe149, Allens, Andy Dingley, Anugrah atreya, Ashwinjacob, Athirubansm, BRW, Bernard Ladenthin, Boson, Briangregory2000, Bulwersator, Cannolis, Chitransh saxena, ChrisGualtieri, Christian75, CiudadanoGlobal, Dotyoyo, Ebraminio, Eggman64, Elizium23, Fctseng, Fieldday-sunday, Hairy Dude, Island Monkey, JForget, Jgzheng, JimpsEd, Jmorwick, Jérôme, Laserpistol, Leflyman, Michealt, Mike Rosoft, Mikeblas, Nabav, Nay Min Thu, NeerajKawathekar, Niddriesteve, Njsg, Obradovic Goran, Oxymoron83, P.kmetski, Pratnala, ProbePLayer, Quanticle, Quarl, Raztus, Saravu2k, Simetrical, Smurfix, Solomon423, SqlPac, Su30, Torzsmokus, Twarther, Uzume, VinceBowdren, Yachtsman1, ZenSaohu, 石庭豐, 93 anonymous edits

Lossless-Join Decomposition *Source*: http://en.wikipedia.org/w/index.php?oldid=598276953 *Contributors*: Aednichols, Bearcat, Bgwhite, ChrisGualtieri, Igor Yalovecky, Jaaap, Katharineamy, Owencm, Reidbaker, Sadads, ToastieIL, Widr, 22 anonymous edits

Join dependency *Source*: http://en.wikipedia.org/w/index.php?oldid=586955929 *Contributors*: A3 nm, Boson, ChrisGualtieri, Hairy Dude, RonaldKunenborg, Signalhead, Tijfo098, VanishedUserABC, 7 anonymous edits

Multivalued dependency *Source*: http://en.wikipedia.org/w/index.php?oldid=574646922 *Contributors*: AutumnSnow, Cdrdata, Citral, Dweller, Emma li mk, Gromuald, Justin W Smith, Kilopi, Mark Renier, Michael Hardy, Michaelcomella, Mkagenius, Poa, ProveIt, R'n'B, Tijfo098, VinceBowdren, 34 anonymous edits

Fourth normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=599220615 *Contributors*: Akerans, Ashimjgec08, Bernard Ladenthin, Boson, Britannica, Dougher, Ebraminio, Fetchcomms, Geeoharee, GermanX, Jason Quinn, Jmabel, Mark Renier, Meng6, Nabav, Northamerica1000, Paramtrivedi, Patrick, RBarryYoung, Savh, Selfworm, Tentinator, Tweenk, Vacation9, VinceBowdren, Vjosullivan, WikHead, Winterst, 36 anonymous edits

Fifth normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=597357005 *Contributors*: Andy M. Wang, Bernard Ladenthin, Boson, Brick Thrower, Cool Blue, Dougher, Dsn.naruka, Dugo, Ebraminio, Faizan, FineganCJ, Flying Panda, Furrykef, GermanX, Igor Yalovecky, Jason Quinn, Libcub, MarcosWozniak, Mark Renier, Nabav, Quarl, RonaldKunenborg, Siryendor, SqlPac, Stamfest, Systemparadox, Tide rolls, Tijfo098, TonyTheTiger, VinceBowdren, Vsagarm, 46 anonymous edits

Sixth normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=593056536 *Contributors*: Boson, DePiep, Dougher, Emurphy42, Esran, Favonian, GregorB, Jason Quinn, Mark Renier, Nabav, Quarl, RBarryYoung, Roenbaeck, RonaldKunenborg, Rp, 14 anonymous edits

Denormalization *Source*: http://en.wikipedia.org/w/index.php?oldid=589830907 *Contributors*: Alan Liefting, Andy Dingley, Bearcat, ChrisGualtieri, Conversion script, Damian Yerrick, Danim, David Gerard, Dreftymac, Furrykef, Fyrael, GermanX, Ghewgill, Gpierre, GregorB, Jay-Jay, Jdlambert, John Coupe, Joncnunn, Jtgerman, Jwolfe, Ketiltrout, Klausness, LOL, Leandrod, MIT Trekkie, Malcolma, Mann jess, Markonen, Matthew0028, Mgt88drcr, Mindmatrix, Netfall, Ninly, Nyttend, Ocrow, PamD, PatrickFisher, Paulo.freire, Pnm, Rich Farmbrough, Robert McClenon, Shunpiker, Skittleys, TechPurism, Tobias Hoevekamp, Topbanana, Troels Arvin, Ver, Verycharpie, VinceBowdren, YetAnotherPseudonym, 38 anonymous edits

Domain/key normal form *Source*: http://en.wikipedia.org/w/index.php?oldid=544531683 *Contributors*: BRW, Bernard Ladenthin, Dcoetzee, Emurphy42, GermanX, Gigs, Gregbard, Jmabel, Mark Renier, Mugaliens, Nabav, Ott2, RonaldKunenborg, SqlPac, VinceBowdren, Widr, 21 anonymous edits

Single Source of Truth *Source*: http://en.wikipedia.org/w/index.php?oldid=552939069 *Contributors*: 4johnny, Arjayay, Balis, Chris the speller, Christian75, Cybercobra, FoxDiamond, Jason Quinn, Johncrab, Lox, MarkBurnard, Markbassett, Minnaert, Pete142, RaviKrishnappa, Rjackson4, Subversive.sound, 4 anonymous edits

Single version of the truth *Source*: http://en.wikipedia.org/w/index.php?oldid=588422595 *Contributors*: 28bytes, DLSieving, FayssalF, Johncrab, Lox, Minnaert, Pumba lt, Quarl, RainbowCrane, RichardVeryard, Tcotco, Tjamesjones, Uncle G, 5 anonymous edits

Principle of Orthogonal Design *Source*: http://en.wikipedia.org/w/index.php?oldid=484397216 *Contributors*: Elwikipedista, GregorB, Neko-chan, Rachel Mosley, RayAYang, Rschoenrank

Database transaction *Source*: http://en.wikipedia.org/w/index.php?oldid=598390179 *Contributors*: 16@r, Adi92, Ajk, Al3ksk, AmandeepJ, AnmaFinotera, Appypani, Babbage, Bgwhite, Billinghurst, Binksternet, Burschik, CharlotteWebb, ChrisGualtieri, Clausen, Comps, Craig Stuntz, DCEdwards1966, Damian Yerrick, Daniel0524, Dauerad, Derbeth, DnetSvg, Forderud, Fratrep, Geniac, Georgeryp, Gerd-HH, Gf uip, Ghettoblaster, GregRobson, Haham hanuka, Hbent, Hede2000, Highguard, HumphreyW, Intgr, JCLately, Jarble, Jason Quinn, Jeltz, Karel Anthonissen, KellyCoinGuy, KeyStroke, Khukri, Kirananils, Larsinio, Leeborkman, Lingliu07, Lubos, Luc4, Lysy, M4gnum0n, Mark Renier, Matiash, MegaHasher, Mike Schwartz, Mikeblas, Mindmatrix, Mintleaf, Neilc, Nixdorf, OMouse, Obradovic Goran, Owen, Paul Foxworthy, Pcap, Pepper, Prakash Nadkarni, RedWolf, RichMorin, Rocketrod1960, Roesser, SAE1962, Sandrarossi, SebastianHelm, Sobia akhtar, SqlPac, Stevag, T0m, Thisismyusername96, Timo, Triwger, Troels Arvin, Turnstep, WeißNix, Zerksis, Zhenqinli, 110 anonymous edits

Transaction processing *Source*: http://en.wikipedia.org/w/index.php?oldid=593892763 *Contributors*: 16@r, Abdull, Adolphus79, Agateller, Akulkis, Alkamins, Andy Dingley, Atlant, Avb, Awolski, BBCWatcher, BD2412, Baiji, Beland, Beve, Bnicolae, Bruvajc, CaroleHenson, Cbwash, Chairman S., Charleyrich, Clausen, Cliffb, Craig Stuntz, CutOffTies, DGG, Danielle009, Danim, DeKXer, Donsez, Download, Ellynwinters, Gf uip, Ghaskins, Gordonjcp, GregRobson, Gutza, JCLately, JHunterJ, Jan1nad, Jmcw37, Jorgenev, Joshua Scott, Kgf0, Khalid hassani, Kubanczyk, Lear's Fool, Luís Felipe Braga, M4gnum0n, MER-C, MONGO, Mandarax, Mark Renier, Maury Markowitz, Mika au, Mikeblas, Mindmatrix, MrOllie, Oo7nets, Oxymoron83, Pcap, Peter Flass, Pratyeka, Radagast83, Rbpasker, Rettetast, Ruud Koot, SEWilco, Stephan Leeds, Stymiee, Suruena, Tobias Bergemann, Tschristoppe, Unimath, Uzume, Wireless friend, Wtmitchell, Zippy, Zzuuzz, 109 anonymous edits

Concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=598451044 *Contributors*: 2GooD, Acdx, Adrianmunyua, Augsod, Bdesham, Brick Thrower, CanisRufus, CarlHewitt, Christian75, Clausen, Comps, Craig Stuntz, Cyberpower678, DavidCary, Donner60, Furrykef, Gdimitr, GeraldH, JCLately, Jesse Viviano, Jirislaby, John of Reading, JonHarder, Jose Icaza, Karada, KeyStroke, Kku, Leibniz, M4gnum0n, Magioladitis, Malbrain, Mark Renier, Mgarcia, Mindmatrix, Miym, N3rV3, Nbarth, Nealcardwell, Nguyen Thanh Quang, Oioisaveloy, PaulMcKenney, Peak, Poor Yorick, Reedy, Rholton, Ruud Koot, Siskus, Smallman12q, Soham, The Anome, Thingg, Thoreaulylazy, Tikuko, TonyW, Touko vk, Tumble, Victor falk, Vincnet, Wbm1058, Wikidrone, Winterspan, YUL89YYZ, 92 anonymous edits

Transaction Control Language *Source*: http://en.wikipedia.org/w/index.php?oldid=556743371 *Contributors*: Asuka Zone, GeorgeLouis, Gyrofrog, Khazar, LegeDoos, RJFJR, SaveTheRbtz, 3 anonymous edits

ACID *Source*: http://en.wikipedia.org/w/index.php?oldid=597872391 *Contributors*: 123Hedgehog456, Accurizer, Acdx, Af648, Agnt9, AlanUS, Amolshah, Andrei S, Anthony Appleyard, AxelBoldt, Barefootguru, Barrylb, Beland, Benandorsqueaks, Bernhard Bauer, Bezenek, BlueNovember, Bluiee, Boing! said Zebedee, BonsaiViking, Bunyk, CPColin, Ceyockey, Christian75, Clausen, Cole2, CorbinSimpson, DAllardyce, DHN, Daniel11, Dave.excira, Decibel, DevonDBA, Download, DragonLord, Drake Redcrest, Duncan.Hull, E2eamon, Edward, Elwikipedista, Endersdouble, Epbr123, Epicgenius, Espoo, Excirial, FatalError, FayssalF, Flewis, Forderud, Fragglet, Fubar Obfusco, Fylbecatulous, Gakusha, Gf uip, Ghostdood, Gioto, Golfguy399, Gorgan almighty, GregRobson, Gurchzilla, HJ Mitchell, Haakon, Harsh 2580, Heiser, Hellknowz, Hmrox, I dream of horses, IMSoP, Inimino, Inter16, Intgr, It Is Me Here, Ivan Pozdeev, J.delanoy, JCLately, Jagun, Jaydlewis, Jeff G., Jessemerriman, Jim1138, Jleedev, Jontomkittredge, Joonasl, Jordonbyers, Jpbowen, Kaell, Kainaw, Kam Solusar, Karada, Kd24911, Kirilldoom16, Kku, Kmorozov, Kristof vt, Larsinio, Lee Carre, Lfstevens, Loren.wilton, LuoShengli, Luís Felipe Braga, MacMog, Maimai009, Makecat, Marek69, Mark Arsten, Mark Renier, Markonen, Martin451, Materialscientist, Matusz, Maury Markowitz, Mcenedella, Mcherm, Mdann52, Michael Hardy, Mindmatrix, Miym, MrRedwood, Mrwojo, Mskfisher, Mynameissskr, Neilc, NewAspen1, Ngpd, Nmfon, Noformation, Noommos, Novusuna, Passargea, Paul Foxworthy, Paul Magnussen, Personman, Petiatil, PierreAbbat, Pip2andahalf, Polupolu890, Ponder, Pontificalibus, Poor Yorick, Prachee.j, Prasannawikis, Premsurya, Puffin, Quuxplusone, R'n'B, RUL3R, Raja99, Raztus, RedWolf, Redrose64, Reelrt, Renku, Rfl, Rich Farmbrough, Rjwilmsi, Rlaager, Rob7139, Rsrikanth05, Sae1962, Saeed Jahed, Safalra, Salix alba, Saucepan, Sean D Martin, SeanAhern, Seaphoto, Sesshomaru, Shenme, Shmuelsamuele, Siggimund, Siskus, Some jerk on the Internet, SpaceFlight89, SqlPac, Stangaa, StephanCom, Strike Eagle, Surturz, Suruena, Svick, Swamp Ig, Synchronism, Tabledhote, Tagus, Tct13, Tgeairn, Thecodysite1, ThinkerFeeler, Thomas Willerich, Throwaway85, Thumperward, Tide rolls, Titodutta, Tolly4bolly, Tommy2010, Trefork, Triesault, Trusilver, Trvth, Turnstep, UFu, Uiteoi, Urhixidur, Verloren, Vertium, Vhabacoreilc, Victor falk, Vina, Viridae, Vrenator, WadeSimMiser, WhiteOak2006, Widr, Wikipelli, Wildrain21, Winston Chuen-Shih Yang, Woo333, Wykypydya, Yazan kokash23, Yourbane, Ysangkok, Ytcracker, Yurik, ZappaOMati, Zhenqinli, Zigger, Zippy, Zoicon5, Σ, 603 anonymous edits

Atomicity (database systems) *Source*: http://en.wikipedia.org/w/index.php?oldid=597881585 *Contributors*: Ancheta Wis, Anutural, Apokrif, ArneBab, Betacommand, Biggins, Bodragon, CesarB, Chris Purcell, DanPope, Danhash, DopefishJustin, Dreftymac, EmmetCaulfield, Enochlau, Ewlyahoocom, Fraggle81, Freshbaked, Gdimitr, Hadal, Hkmaly, Hooperbloob, JCLately, JPG-GR, Jleedev, Jmendez, Jncraton, Julescubtree, Korg, Kvng, LinguistAtLarge, Mark Renier, Marudubshinki, Materialscientist, Michael Hardy, Neelix, Nihiltres, PeterJohnson, Qwertykris, RJFJR, Rfl, Rohan Jayasekera, Sae1962, Smyth, Snailwalker, TimBentley, Vicarious, 42 anonymous edits

Isolation (database systems) *Source*: http://en.wikipedia.org/w/index.php?oldid=597196598 *Contributors*: Alvin-cs, Anonymouslee, Antonio.al.al, Asqueella, AxelBoldt, BYVoid, Beland, Brian.low22, Bunchofgrapes, Cameltrader, Chris Purcell, Closedmouth, Cybercobra, DARTH SIDIOUS 2, Djmitche, Ej, EmmetCaulfield, Enigmaman, Erichero, Ervinn, Ewlyahoocom, Gracenotes, Hadal, Hheimbuerger, Hrishikeshbarua, IMSoP, Ian Clelland, Igoldste, Inovakov, Insanephantom, Irbisgreif, Ivansoto, J18ter, JCLately, JMatthews, Ketiltrout, Kevsteppe, KeyStroke, Kgrittn, Khfan93, Laurent Van Winckel, LonelyBeacon, Magioladitis, Mandarax, Materialscientist, Mattmorgan, Maverick13, Maxal, Mentin, Michal.burda, Mild Bill Hiccup, Mindmatrix, Nczempin, Niceguyedc, NunoFerreira, Olnrao, Olof nord, Paul Clapham, Phlip, Prunesqualer, QLineOrientalist, Ramesh, Rfl, Rsocol, SPKirsch, Sae1962, Sahedin, Sbose7890, Scku, Searcherfinder, Snow Blizzard, SoCalSuperEagle, Stefan Udrea, Swamp Ig, The Thing That Should Not Be, TheJC, Tobias Bergemann, Tommy0605, Tommy2010, Wiki13, Wikidemon, 209 anonymous edits

Durability (database systems) *Source*: http://en.wikipedia.org/w/index.php?oldid=596543787 *Contributors*: Astazi, CesarB, Clausen, D3fault, Edward, Ewlyahoocom, JCLately, LordHz, Mark Renier, Rfl, Saucepan, SqlPac, Tobias Bergemann, Wizardman, Yourbane, 8 anonymous edits

Atomic commit *Source*: http://en.wikipedia.org/w/index.php?oldid=593275084 *Contributors*: Abb615, Abdulqabiz, Acather96, Akumka, Alaerts, Andyzweb, Bmistree, C5st4wr6ch, CesarB, Charles Matthews, Chick Bowen, Chris Q, CoreTechX, Dawynn, Forderud, Frigolit, Gurch, Happysailor, Ideogram, JCLately, Jaksa, John of Reading, Jusdafax, Max Terry, Meatsgains, Mike Schwartz, Miym, PhilipMW, Pne, Radagast83, Randallbsmith, Rjwilmsi, RobJ1981, Rworsnop, SGGH, Segv11, Sortior, SqlPac, Sumanthewhiz, Tassedethe, Tedickey, WurmWoode, 29 anonymous edits

Schedule (computer science) *Source*: http://en.wikipedia.org/w/index.php?oldid=599264886 *Contributors*: ABF, Aaron north, Bmicomp, Bryan Derksen, Cagdasgerede, Comps, Cybercobra, Darthsco, Ehamberg, Flyer22, Grrowl, Hans Adler, Harryboyles, Ivan Kuckir, JCLately, JamesBWatson, KerryVeenstra, M4gnum0n, MZMcBride, Michael Hardy, Michaelcomella, Mild Bill Hiccup, Mkoval, Ruud Koot, Sfan00 IMG, Snow Blizzard, Sprite, Tide rolls, Titodutta, Uncle Dick, Velella, Zipzipzip, 93 anonymous edits

Serializability *Source*: http://en.wikipedia.org/w/index.php?oldid=590700173 *Contributors*: Ahoerstemeier, Alex.mccarthy, Amux, ArnoldReinhold, Arthena, BD2412, Chris the speller, Comps, Craig Pemberton, Cybercobra, Cyberpower678, DRAGON BOOSTER, Darthsco, DavidCary, Deor, Farhikht, Flyguy649, Fyrael, Fæ, Greenrd, JCLately, Jack Greenmaven, John of Reading, Jwoodger, Kgrittn, Klower, Kubanczyk, LilHelpa, M4gnum0n, Mark Renier, MeekMark, Mihai Capotă, Miym, Omnipaedista, Paddles, Paul Foxworthy, R'n'B, Rjwilmsi, Ruud Koot, Rxtreme, Supparluca, Svick, Tbhotch, That Guy, From That Show!, Wavelength, 61 anonymous edits

Precedence graph *Source*: http://en.wikipedia.org/w/index.php?oldid=586669688 *Contributors*: Alai, Ash211, Cdrdata, ChrisGualtieri, Chrisjameskirkham, Comps, Ehamberg, Errikosd, Flyguy649, Kimjoarr, Kubanczyk, Pegship, Provehuman, Ramina66, Rarasha, Requestion, Rjwilmsi, Vandy24, Wikidrone, 18 ,.תומר א anonymous edits

Serializability theory *Source*: http://en.wikipedia.org/w/index.php?oldid=393907966 *Contributors*: Ahoerstemeier, Alex.mccarthy, Amux, ArnoldReinhold, Arthena, BD2412, Chris the speller, Comps, Craig Pemberton, Cybercobra, Cyberpower678, DRAGON BOOSTER, Darthsco, DavidCary, Deor, Farhikht, Flyguy649, Fyrael, Fæ, Greenrd, JCLately, Jack Greenmaven, John of Reading, Jwoodger, Kgrittn, Klower, Kubanczyk, LilHelpa, M4gnum0n, Mark Renier, MeekMark, Mihai Capotă, Miym, Omnipaedista, Paddles, Paul Foxworthy, R'n'B, Rjwilmsi, Ruud Koot, Rxtreme, Supparluca, Svick, Tbhotch, That Guy, From That Show!, Wavelength, 61 anonymous edits

Read–write conflict *Source*: http://en.wikipedia.org/w/index.php?oldid=551039779 *Contributors*: Avocado, Egriffin, Emact, JCLately, Michael Hardy, Paxse, Pnm, Poor Yorick, Vegpuff, Wikibob, 2 anonymous edits

Write–read conflict *Source*: http://en.wikipedia.org/w/index.php?oldid=543696786 *Contributors*: Egriffin, GregorB, JCLately, KeyStroke, Pnm, Poor Yorick, Rettetast, Vegpuff, 2 anonymous edits

Write–write conflict *Source*: http://en.wikipedia.org/w/index.php?oldid=596095175 *Contributors*: Egriffin, Gtrmp, JCLately, Karada, Nabla, Pnm, Poor Yorick, Rettetast, Vegpuff, 3 anonymous edits

Lock (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=598099134 *Contributors*: Belenus, Bruvajc, Caidence, ChrisGualtieri, Danim, FCutic, Greenrd, Gulsig4, InverseHypercube, Jack007, Kidoshisama, LilHelpa, Mauro Bieg, Maxal, Potionism, Sharpshooter4008, Ta bu shi da yu, VinnieCool, Vishnu2011, Wikid77, 23 anonymous edits

Record locking *Source*: http://en.wikipedia.org/w/index.php?oldid=597030970 *Contributors*: Atlant, D6, Finn-Zoltan, JCLately, Michael Hardy, Pol098, Quaddriver, This lousy T-shirt, Umaguna, Waynelwarren, 22 anonymous edits

Multiple granularity locking *Source*: http://en.wikipedia.org/w/index.php?oldid=577057806 *Contributors*: Aednichols, Bwpach, Darkprince117, Fredwert, GPHemsley, Graham87, JingguoYao, JoshRosen, Masterblah777, Materialscientist, Mrtobacco, Poor Yorick, Svick, Tarquin, Zoicon5, 10 anonymous edits

Two-phase locking *Source*: http://en.wikipedia.org/w/index.php?oldid=584668101 *Contributors*: Aaron Schulz, Andreas Kaufmann, Beta16, Chrisjameskirkham, Clausen, Cntras, Comps, Craig Pemberton, Cxz111, Cybercobra, Dtaylor1984, Epbr123, Jeskeca, John of Reading, Materialscientist, Michael Hardy, Nchaimov, Neilc, OrangeDog, Paul20070, Poor Yorick, Rich Farmbrough, Ruud Koot, SeanMon, Seaphoto, Stuart.clayton.22, Svick, Syst3m, Thomas Bjørkan, Touko vk, Wavelength, Woohookitty, 44 anonymous edits

Readers–writer lock *Source*: http://en.wikipedia.org/w/index.php?oldid=599054676 *Contributors*: Andreas Kaufmann, Babobibo, Beland, Ber, Darth Panda, Download, Dublet, Dvyukov, Forderud, Fuzzbox, Greensburger, JC Chu, Jakarr, Jeenuv, KindDragon33, Loopy48, Malbrain, Msnicki, Ohconfucius, Peepeedia, Pnm, Quuxplusone, RandyFischer, Ronklein, Spoonboy42, Thornrag, Vald, ZeroOne, 24 anonymous edits

Blind write *Source*: http://en.wikipedia.org/w/index.php?oldid=576700913 *Contributors*: Alynna Kasmira, Andreas Kaufmann, Greenrd, KylieTastic, Michael Hardy, 5 anonymous edits

Conservative two-phase locking *Source*: http://en.wikipedia.org/w/index.php?oldid=525975025 *Contributors*: Bergsten, Poor Yorick, Svick, 7 anonymous edits

Strong strict two-phase locking *Source*: http://en.wikipedia.org/w/index.php?oldid=324138798 *Contributors*: Aaron Schulz, Andreas Kaufmann, Beta16, Chrisjameskirkham, Clausen, Cntras, Comps, Craig Pemberton, Cxz111, Cybercobra, Dtaylor1984, Epbr123, Jeskeca, John of Reading, Materialscientist, Michael Hardy, Nchaimov, Neilc, OrangeDog, Paul20070, Poor Yorick, Rich Farmbrough, Ruud Koot, SeanMon, Seaphoto, Stuart.clayton.22, Svick, Syst3m, Thomas Bjørkan, Touko vk, Wavelength, Woohookitty, 44 anonymous edits

Index locking *Source*: http://en.wikipedia.org/w/index.php?oldid=595151627 *Contributors*: Cander0000, Closedmouth, Comps, Cybercobra, Dennis Bratland, E946, JCLately, Mcoupal, Pearle, Pinktulip, Rich Farmbrough, Robth, Wikibofh, 6 anonymous edits

Snapshot isolation *Source*: http://en.wikipedia.org/w/index.php?oldid=597867348 *Contributors*: AnnHarrison, Blowdart, Chowbok, Chris Purcell, Comps, Craig Stuntz, David Eppstein, Ej, Elkman, Idleloop, Isidore, JCLately, Johndburger, Kgrittn, Neilc, Pcap, Ruud Koot, Searcherfinder, Tuntable, VanishedUserABC, Woohookitty, Ysangkok, 17 anonymous edits

Non-lock concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=532096805 *Contributors*: Comps, DavidCary, Drbreznjev, Dtremenak, Greenrd, GregorB, JCLately, Jwoodger, Mark Renier, Poor Yorick, R. S. Shaw, RJFJR, Smyth, The Anome, 2 anonymous edits

Commitment ordering *Source*: http://en.wikipedia.org/w/index.php?oldid=596925106 *Contributors*: Chris the speller, ChrisGualtieri, Comps, Cybercobra, Dicklyon, Dthomsen8, Duffbeerforme, Gu1dry, HJ Mitchell, I JethroBT, JCLately, JHunterJ, JaGa, JohnI, Jpmonroe, Jwoodger, Khazar, Miym, Oioisaveloy, Pat111, Paul Foxworthy, Phil Boswell, Pinkadelica, R'n'B, Rich Farmbrough, SarekOfVulcan, Sun Creator, Thatcher, Thumperward, Tony1, Torchiest, 15 anonymous edits

Long-running transaction *Source*: http://en.wikipedia.org/w/index.php?oldid=551388140 *Contributors*: AManWithNoPlan, Aednichols, Cgwaters, Charlton, D33pInside, Darkwind, Finlay McWalter, Intgr, JCLately, Michael Hardy, 6 anonymous edits

Timestamp-based concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=595991570 *Contributors*: Antonio.al.al, BasEI, Conrad.Irwin, DavidCary, Derek Parnell, GregorB, Hertzsprung, JCLately, Jncraton, Lambart, Michael Hardy, Miym, Nils Grimsmo, Nohat, Poor Yorick, PrincessofLlyr, Ruud Koot, Turnstep, VanishedUserABC, Woohookitty, 22 anonymous edits

Pseudoconversational transaction *Source*: http://en.wikipedia.org/w/index.php?oldid=554543528 *Contributors*: Beao, Dekart, Download, Kdakin, Mild Bill Hiccup, Miyagawa, PamD, WikHead, 3 anonymous edits

Thomas write rule *Source*: http://en.wikipedia.org/w/index.php?oldid=551812353 *Contributors*: Avocado, Berlinetta1492, Bsdlogical, Charles Matthews, CiSerg, Conrad.Irwin, Dbmcclain, Fjzpwq1385, JCLately, KathrynLybarger, Krishna.91, Mark T, Michael Hardy, Molly-in-md, Poor Yorick, Ssd, Totiproti, Xthemage, 8 anonymous edits

Global concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=544873458 *Contributors*: Comps, Miym, Ruud Koot, 1 anonymous edits

Global serializability *Source*: http://en.wikipedia.org/w/index.php?oldid=552636158 *Contributors*: BD2412, Comps, Eugene-elgato, Kithira, Mild Bill Hiccup, Miym, Pdbne, RHaworth, Rjwilmsi, Ruud Koot, Sun Creator, The Anome, The Thing That Should Not Be, Underpants, Vegaswikian, Woohookitty, 1 anonymous edits

Modular concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=323027606 *Contributors*: Comps, Miym, Ruud Koot, 1 anonymous edits

Multiversion concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=598367311 *Contributors*: Ahodgkinson, Arleach, Basil.bourque, Bdempsey64, Bill.zopf, Blowdart, Breinbaas, CYCC, Cbbrowne, Chowbok, Chris Purcell, ChrisCork, Compfreak7, Comps, Craig Stuntz, Cwhii, DanielWeinreb, Danilo.Piazzalunga, DavidCary, Dcoetzee, Dexp, Dfetter, Dllahr, Doug4j, Dougher, Drachmae, Drbreznjev, Elendal, EricBloch, Fecund, Franl, Gaius Cornelius, Ggaughan, Giloki, GregorB, Gritzko, Hga, Highlandsun, Hu12, JCLately, JLaTondre, Jamesday, Jerryobject, JingguoYao, John of Reading, Johnkarva, Jonathanstray, Jordan Brown, Julien2512, Kalotus, Kedawa, Kevin 71984, Kevinroy09, Kweetal, KyleJ1, LilHelpa, M4gnum0n, Marcuscalabresus, Martpol, Mbautin, MrChupon, Naasking, Neilc, Nowhere man, Nthiery, Obankston, Palfrey, Pcap, Piet Delport, Plothridge, Poor Yorick, ProfessorBaltasar, Pvjohnson, R. S. Shaw, Rawlife, Raysonho, RickBeton, Rjwilmsi, Rsfinn, Seancribbs, Siskus, Snnn, Tedickey, Terrycojones, That Guy, From That Show!, ThomasTomMueller, ThurnerRupert, Troels Arvin, Tsunanet, Tuhl, Tuntable, Turnstep, Unordained, Visik, Wasbeer, Waynelwarren, Whimsley, Will Faught, Yannick56, Yoonforh, Ysangkok, 124 anonymous edits

Optimistic concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=590968429 *Contributors*: Ace of Spades, Ahunt, Algotr, Allan McInnes, Beland, Cybercobra, D6, DavidCary, Dmeranda, DraX3D, Drewnoakes, GregorB, HenryLi, Homer Landskirty, Intgr, Iridescent, JCLately, JeromeJerome, Jfromcanada, Karada, Lfstevens, M4gnum0n, Magioladitis, Methossant, Mnot, Mydoghasworms, NYKevin, Nczempin, Neilc, Nikai, Noazark, Poor Yorick, R. S. Shaw, Randomalious, Rich Farmbrough, SAE1962, Sim IJskes, Simetrical, Slamb, SmartGuy Old, Smyth, Suruena, Svick, Therealmatbrown, Timwi, Zero sharp, Zoicon5, 48 anonymous edits

Autocommit *Source*: http://en.wikipedia.org/w/index.php?oldid=597222178 *Contributors*: Al3ksk, Faizan, Malcolma, Nibios, Prakash Nadkarni, Ysangkok, 4 anonymous edits

Transaction log *Source*: http://en.wikipedia.org/w/index.php?oldid=596225819 *Contributors*: Clausen, DGG, Damian Yerrick, Gustronico, Intgr, JCLately, JLaTondre, KeyStroke, Larsinio, Lupin, Mark Renier, Mikeblas, Mindmatrix, Neoconfederate, Pelister, Poor Yorick, SJP, Sleske, SoledadKabocha, Stolze, Twimoki, 39 anonymous edits

Savepoint *Source*: http://en.wikipedia.org/w/index.php?oldid=594073946 *Contributors*: Abdull, Asfreeas, Cotevertu, Damian Yerrick, Derbeth, Frze, JCLately, Neilc, Robofish, Sameer78613, Turnstep, Weregerbil, 11 anonymous edits

No-force *Source*: http://en.wikipedia.org/w/index.php?oldid=254061429 *Contributors*: -Midorihana-, Alai, Ehamberg, Janm-tw, Mandy003, Panyd, WikHead, 1 anonymous edits

Non-blocking algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=599171807 *Contributors*: A.Ou, Aldinuc, Andreas Kaufmann, Bdongol, Betacommand, Bovineone, Bryan Derksen, Choess, Chris Purcell, Chris the speller, Daira Hopwood, DavidCary, Discospinster, Dougher, Dvyukov, Elnyka, Ewlyahoocom, Fantr, Gadfium, Greg Ward, Helwr, IngerAlHaosului, Ivan Pozdeev, Jay, JonHarder, Joy, Keithathaide, Khizmax, M4gnum0n, Mathiasl26, Mblumber, Miym, MrOllie, Neilc, Neo-Jay, Ohiostandard, Parsecboy, Phoe6, Raanoo, Radagast83, Rattusdatorum, Runtime, Salix alba, Silverrocker, TheRedPenOfDoom, TimBentley, Timlevin, Tjdw, Wapawlo, Wikip rhyre, Zigger, Zvar, 62 anonymous edits

Data recovery *Source*: http://en.wikipedia.org/w/index.php?oldid=595124903 *Contributors*: 16@r, AKA MBG, Abhijith.nath, Adam7117, Admindrorg, Admiralthrawn999, Afcyrus, Airplaneman, Ajmishra1989, Aladdin Sane, Alansohn, Alexius08, Alfredsmithseo, AlistairMcMillan, Altay8, Amitpandey21, Amontero, Anna Frodesiak, Antimatt, Aoidh, Arbustoo, Arjayay, ArroLu, Arthena, Astral, Atnalta, Aura707, Ausnomz, Austinmurphy, Awolfe58, AxelBoldt, Az1568, Badgernet, Bakanov, Baseball Watcher, Batareikin, Bbubbabronco, Beenakuliyal, Betacommand, Bgwhite, Bhopkins, Bluebusy, Bluemoose, Bmikel, Bonadea, Borisbaran, Brett hunter, Brian Cometa, Broc, CONFLICTGOD, Calton, Caper13, Captin Shmit, Cassacawn, Catjumpjohn, Cavrdg, Cgrenier, Charlene.fic, CharmlessCoin, Chetanprakashjpr, Chris Architect, Chris the speller, Chris16514, ChrisCotton, ClementSeveillac, Cnruntu, Computerbeast, Coolslko, Cuvtixo, DOSGuy, DanielCD, Darth Mike, Data990, Datarecov, Dauja, Dawnseeker2000, Dcirovic, Diagramma Della Verita, Dina, Disklabs, Divakar01, Doniago, Download, Dragomiloff, Drilnoth, Dtidata, Dvehrs, Dwoodsy90, Dzubint, EASEUS, ESkog, East718, Eazyrecover, Edm-man, Eds147eds, El C, Eleassar, Electron9, Eleven even, Elfriede12, Elsendero, Emre D., Epolk, Erwin, Esanchez7587, Evolutionbusiness, Excirial, Feinoha, Fightingboy, Fivealiveprize, Fixentries, Flewis, Forever Dusk, Fortunateone, Frank, FreelanceWizard, Freewol, Fvw, Gancell12, Garo, Gbaor, Gdh77777, Gillwaredotcom, Gogo Dodo, GraemeL, Greco8523, Griffinity, Gwalla, Götz, Haikupoet, Hammer Raccoon, Hm2k, Holme053, Honglake, Hu12, Hut 8.5, Imran22, Inferno, Lord of Penguins, Iridescent, J.delanoy, JCLately, JHP, Ja 62, Jaciverson, Jack Robins, Jackster, JakobVoss, JamesMio, Jennifer456, Jim1138, Jimmierude, Jj137, Johnchristopher, Johnnyd96, Johnpitterson, Joshnpowell, Jusdafax, KLLvr283, Kagnie, KeyStroke, Khatru2, Kiara621, King of Hearts, Kispvtltd, Konman72, Kotao, Kozuch, Kurtrosenfeld, Kuru, Lctech, Leafyplant, LeaveSleaves, Lectonar, Leebert, Leemason, Linkspamremover, LittleBenW, Luckynumbers, M4gnum0n, MER-C, Mac, Macbookexpert, Maheshcharjan, Marketyung, McSly, Methodicalj, Mfisherkirshner, Mike Stang, Mike.lifeguard, MikeWDavis, Mikem22, Minimac, Mipadi, Mireth, Mitusoft, Mohbed, Molicasolis, Morenoodles, Mr Gronk, MrArt, MsMeasures, Music Sorter, Mvidata, NailPuppy, Nakon, Nancy, Netkinetic, Newportm, Nexus501, Nick Majors, Nick Number, Nikkolla, Nil Einne, Nnp, Northamerica1000, Ohnoitsjamie, Oiler, Onlyfact, Onorem, Pankaj.nucleus, ParticleMan, PashaPal, Pdinhofer, Pearle, Petrb, Petter.smith, Phoogenb, Pinethicket, Pnm, Pol098, Posix memalign, Pp16514, Prari, PrometheusX303, Public Menace, Qiongeramber, Quantumor, RHaworth, Rancelan, Rchandra, Reccorx, RenamedUser01302013, Renesis, Reswobslc, RexNL, Rich Farmbrough, Rjcable, Rmikel11, Roadrunners123, Robert Bond, Romal, RomanSpa, RroccoMaroc, Rubybarett, Rurik, Ryansccs, SF007, Sallyte, SamWiltshire, Sc4074100, Schewek, SchuminWeb, Sclm, Scriberius, Sebastian Stadil, Semsi Paco Virchow, Seobeglobal, SergeShirobokov, SexyBern, Shalajack, Smileydotone, Sobreira, Softlogica, Spoon!, StaticVision, Stellarinfo, Stephenb, Subinhommer, Suech, Sugreev2001, Suthaar, Swaq, Sylvainbruley, Takeaway, TampaDataTech, Taruntyagiji, Tarus2, Taxman, Tedickey, The PIPE, TheManLindsey, TheStarman, TheWeasel, Themfromspace, Theroadislong, Thomas Larsen, Tide rolls, Tim1357, Tins128, Toolingu, Tregoweth, Trustreliance, Tschild, Tuankiet65, Tugboat100, Untrue Believer, Uršul, Versageek, Vineetrajput, Warut, Wayne Slam, Webgrunt, WikEditor, WikiDan61, Wikiwikikid, Wikkedit, Wonderstruck, Wz0911, Xp54321, Yhgdnow, Ykhwong, Yoguy888, Yosri, Z007007, Zahid Abdassabur, Zelda, Zian, Zzptichka, Zzuuzz, 655 anonymous edits

Point-in-time recovery *Source*: http://en.wikipedia.org/w/index.php?oldid=549318053 *Contributors*: Grrowl, Guoguo12, Malcolma, Mike Rosoft, Plinehan, Stillnotelf, Tinucherian, Zenaan, 7 anonymous edits

Redo log *Source*: http://en.wikipedia.org/w/index.php?oldid=592035539 *Contributors*: Avenue X at Cicero, Cgpug306, Dialectric, Ghane, I dream of horses, JCLately, Jsaylor3, LilHelpa, Lkesteloot, Manualph, Pedant17, Pluti, Shadowjams, Wikielwikingo, Winterst, 22 anonymous edits

Extreme transaction processing *Source*: http://en.wikipedia.org/w/index.php?oldid=573366435 *Contributors*: Arunvgchat, Eastlaw, GreatWhiteNortherner, Kirananils, SarahStierch, Sfan00 IMG, Shar1R, 3 anonymous edits

In-database processing *Source*: http://en.wikipedia.org/w/index.php?oldid=564699387 *Contributors*: AGK, Alan Liefting, Bunnyhop11, Chire, LittleWink, MC Wapiti, MrOllie, Soni, SpecMode, Vanhoosear, W Nowicki, WOSlinker, Zodon, 11 anonymous edits

Locks with ordered sharing *Source*: http://en.wikipedia.org/w/index.php?oldid=505912493 *Contributors*: Comps, Sun Creator

Nested transaction *Source*: http://en.wikipedia.org/w/index.php?oldid=515928838 *Contributors*: Alynna Kasmira, Gerald Zincke, Imc, Intgr, JCLately, Jack Wester, Jeffrey O. Gustafson, JonHarder, Kbdank71, Nczempin, Pcap, Pedronet, Rl, Tinucherian, Xhienne, 12 anonymous edits

Transaction processing system *Source*: http://en.wikipedia.org/w/index.php?oldid=596476873 *Contributors*: Adrian J. Hunter, Alansohn, Aleksd, Andrew1111111, Ary29, Bellenion, Capitalismojo, Captain panda, CaroleHenson, Chris the speller, Cobi, Cory Wilkie, CosineKitty, DJ Bungi, Dacrow, Dialectric, Discospinster, Dreadstar, Edward, Electriccatfish2, Ewlyahoocom, Falcon8765, Flewis, Garion96, GregorB, Himayrami, Imzadi1979, JCLately, JediMooCow, Jim1138, Jwestbrook, KNHaw, Kawasemi, Keilana, Killian441, Klausness, Kubanczyk, Leszek Jańczuk, Lmatt, MaBoehm, Magioladitis, Materialscientist, Miracleworker5263, Music Sorter, O.Koslowski, OS2Warp, Oliver Lineham, Pankaj.unexpected, Pcap, Pearle, Peter Flass, Pharos, Philip Trueman, Pinkadelica, Pratyeka, R. S. Shaw, Radagast83, Randomguy17, Rbakels, Regregex, Reinhardtw, Rich257, Rnb, RobertG, Robvanvee, Rocketrod1960, Ryan19891, SEWilco, Sandylawn86, ShakingSpirit, Shirik, SirIsaacBrock, Smtchahal, Steven Zhang, Sujit kumar, Sukari, Taxman, The Thing That Should Not Be, Thekinkycookie, Tide rolls, Tsunhimtse, Ularevalo98, Velvetsmog, Vikalp123123, W Nowicki, Wackywace, Wbm1058, Wickey-nl, Wikipelli, Woohookitty, Wptoler, Zundark, 166 anonymous edits

Transaction processing systems *Source*: http://en.wikipedia.org/w/index.php?oldid=476361631 *Contributors*: Adrian J. Hunter, Alansohn, Aleksd, Andrew1111111, Ary29, Bellenion, Capitalismojo, Captain panda, CaroleHenson, Chris the speller, Cobi, Cory Wilkie, CosineKitty, DJ Bungi, Dacrow, Dialectric, Discospinster, Dreadstar, Edward, Electriccatfish2, Ewlyahoocom, Falcon8765, Flewis, Garion96, GregorB, Himayrami, Imzadi1979, JCLately, JediMooCow, Jim1138, Jwestbrook, KNHaw, Kawasemi, Keilana, Killian441, Klausness, Kubanczyk, Leszek Jańczuk, Lmatt, MaBoehm, Magioladitis, Materialscientist, Miracleworker5263, Music Sorter, O.Koslowski, OS2Warp, Oliver Lineham, Pankaj.unexpected, Pcap, Pearle, Peter Flass, Pharos, Philip Trueman, Pinkadelica, Pratyeka, R. S. Shaw, Radagast83, Randomguy17, Rbakels, Regregex, Reinhardtw, Rich257, Rnb, RobertG, Robvanvee, Rocketrod1960, Ryan19891, SEWilco, Sandylawn86, ShakingSpirit, Shirik, SirIsaacBrock, Smtchahal, Steven Zhang, Sujit kumar, Sukari, Taxman, The Thing That Should Not Be, Thekinkycookie, Tide rolls, Tsunhimtse, Ularevalo98, Velvetsmog, Vikalp123123, W Nowicki, Wackywace, Wbm1058, Wickey-nl, Wikipelli, Woohookitty, Wptoler, Zundark, 166 anonymous edits

Transaction server *Source*: http://en.wikipedia.org/w/index.php?oldid=523364427 *Contributors*: Adrian J. Hunter, Alansohn, Aleksd, Andrew1111111, Ary29, Bellenion, Capitalismojo, Captain panda, CaroleHenson, Chris the speller, Cobi, Cory Wilkie, CosineKitty, DJ Bungi, Dacrow, Dialectric, Discospinster, Dreadstar, Edward, Electriccatfish2, Ewlyahoocom, Falcon8765, Flewis, Garion96, GregorB, Himayrami, Imzadi1979, JCLately, JediMooCow, Jim1138, Jwestbrook, KNHaw, Kawasemi, Keilana, Killian441, Klausness, Kubanczyk, Leszek Jańczuk, Lmatt, MaBoehm, Magioladitis, Materialscientist, Miracleworker5263, Music Sorter, O.Koslowski, OS2Warp, Oliver Lineham, Pankaj.unexpected, Pcap, Pearle, Peter Flass, Pharos, Philip Trueman, Pinkadelica, Pratyeka, R. S. Shaw, Radagast83, Randomguy17, Rbakels, Regregex, Reinhardtw, Rich257, Rnb, RobertG, Robvanvee, Rocketrod1960, Ryan19891, SEWilco, Sandylawn86, ShakingSpirit, Shirik, SirIsaacBrock, Smtchahal, Steven Zhang, Sujit kumar, Sukari, Taxman, The Thing That Should Not Be, Thekinkycookie, Tide rolls, Tsunhimtse, Ularevalo98, Velvetsmog, Vikalp123123, W Nowicki, Wackywace, Wbm1058, Wickey-nl, Wikipelli, Woohookitty, Wptoler, Zundark, 166 anonymous edits

Priority inversion *Source*: http://en.wikipedia.org/w/index.php?oldid=592852401 *Contributors*: @modi, Active Banana, AndrewGarber, Archer3, Artoonie, AstroPig7, B4hand, Brucer42, Charles Matthews, CiaPan, Cmccormick8, DJ Clayworth, DenisHowe, Dicklyon, Dori, Dougluce, FloydRTurbo, Frap, Frappucino, Fuzzbox, Gadfium, Heyjebbo, JLaTondre, Jakew, Jfmantis, Joel Saks, Kbdank71, Kbube, Lambiam, Lorryboy, Lpgeffen, M4gnum0n, Marswalker, Math Teacher, McNeight, Mr Vholes, Msnicki, Neilc, Nfm, OleksandrOmelchuk, Paddles, PaulMcKenney, Pmod, Raanoo, Sgeo, SimonP, Slaniel, SolarSauna, TheMandarin, Tobias Bergemann, Topbanana, Vinsci, Vocaro, Wernher, Xanzzibar, ZeVlad, 85 anonymous edits

Priority ceiling protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=592852385 *Contributors*: Abhijithtk, Axeoth, Bachrach44, Blathnaid, Esmehwiki, Fantr, Hmmmmike, InTheCastle, Mayur, MichaK, RedWordSmith, SchreiberBike, Suruena, Ultrarob, 13 anonymous edits

Priority inheritance *Source*: http://en.wikipedia.org/w/index.php?oldid=597317634 *Contributors*: AmosWolfe, Billhuey, CesarB, Fennec, Fratrep, Jliberatore, JonHarder, Kennithng, Mt.Restivo, Pear10155, Pearle, Quenla Runbrik, RJFJR, Rotlink, Rupeshsk, SEI Publications, Suruena, That Guy, From That Show!, TheMandarin, United States Man, 23 anonymous edits

Query optimization *Source*: http://en.wikipedia.org/w/index.php?oldid=592778662 *Contributors*: Abdull, Andreas Kaufmann, Andy Dingley, Avalon, Bearcat, Beland, Cadvga, Cedar101, Danim, Edward, Ginsuloft, Glux, GregorB, Gzuckier, Isulica, JoshRosen, Less Than Free, MBisanz, Mild Bill Hiccup, Mouchoir le Souris, MrOllie, Mrmatiko, Nadeemhussain, Neilc, Owl3638, Paige Master, Pascal.Tesson, Ronwarshawsky, Sct72, Sudhir h, TechPurism, Vitriden, Walter Görlitz, 29 anonymous edits

Query optimizer *Source*: http://en.wikipedia.org/w/index.php?oldid=543977698 *Contributors*: Abdull, Andreas Kaufmann, Andy Dingley, Avalon, Bearcat, Beland, Cadvga, Cedar101, Danim, Edward, Ginsuloft, Glux, GregorB, Gzuckier, Isulica, JoshRosen, Less Than Free, MBisanz, Mild Bill Hiccup, Mouchoir le Souris, MrOllie, Mrmatiko, Nadeemhussain, Neilc, Owl3638, Paige Master, Pascal.Tesson, Ronwarshawsky, Sct72, Sudhir h, TechPurism, Vitriden, Walter Görlitz, 29 anonymous edits

Query plan *Source*: http://en.wikipedia.org/w/index.php?oldid=574422744 *Contributors*: Aaronbrick, Alpha Quadrant, Ammar.w, Ancheta Wis, Arcann, Bevo, Cedar101, Cww, Freezegravity, Grace Note, Hardeeps, James barton, Larsinio, Mark Renier, Mbarbier, Mdesmet, Mikeblas, Mindmatrix, Neilc, Nikola Smolenski, Reedy, Rl, Ronwarshawsky, SimonP, Sippsin, Slaniel, TheParanoidOne, UnitedStatesian, Walter Görlitz, Woodshed, ZoBlitz, 26 anonymous edits

Index (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=470874797 *Contributors*: 16@r, 31stCenturyMatt, Abolen, Afriza, Antandrus, Apavlo, Arcann, Arleyl, Arny, Atree, Aurlee, Bezenek, Brian Tvedt, Cander0000, Carmichael, Ccare, Ceyockey, Chamoquemas, Chire, ChrisGualtieri, CloudNine, ColinFrayn, Comps, Cybercobra, DJPohly, Dainis, Danlev, Deon Steyn, Dewritech, Dionyziz, Dominiktesla, Dougher, Drewnoakes, Dvik, Echawkes, Ercanyuz, ErikHaugen, Euryalus, Excirial, Flewis, Flyer22, Flyrev, Focus22, Furrykef, Gaur1982, Gergie, Glacialfox, Gnaaye, GordonFindlay, Groffg, Groves.w, Gwyant, InShaneee, Interiot, Intgr, JCLately, Jadecristal, Jamesjiao, Jasimab, Jerryji1976, Jfroelich, Jim Carnicelli, Jivadent, Jlehew, JohnF1980, Jon Awbrey, Jschnur, Jspashett, Jwchong, Kayau, KnightRider, Kuru, Larsinio, Leuko, Lfstevens, Lsschwar, Mabuali, Machadoman, MahSim, Manishkarwa, Mark Renier, MarkusWinand, Mereman, Mets501, Microchip2013, Mike Rosoft, Mindmatrix, Morfeuz, Movses, MrOllie, Mxcatania, Müslimix, NGPriest, Nahoo, NellieBly, Nepenthes, NicDumZ, Nicolas1981, Norm mit, Oxymoron83, P21n7, Pawanjain19, Pietrow, Planetneutral, Ppntori, Radagast83, Raypereda, Rich Farmbrough, Rl, RobSimpson, Ruzihm, S.K., Salvio giuliano, Samroar, Samson ayalew, Sandgem Addict, Sbisolo, Searcherfinder, Sideswipe091976, SimonP, Sippsin, Sleske, SpaniardGR, SpeedyGonsales, Stefan Udrea, Stegop, Sumathi.M, THEN WHO WAS PHONE?, Taka, The Thing That Should Not Be, Thesquaregroot, Tide rolls, Tommy2010, Triddle, Turnstep, TutterMouse, Vacio, Wbm1058, Wikiwikithe3rd, William Avery, Woohookitty, X7q, Yalckram, Yamaguchi先生, Zhenqinli, 484 anonymous edits

Partial index *Source*: http://en.wikipedia.org/w/index.php?oldid=593949135 *Contributors*: Adono, Can't sleep, clown will eat me, Cedar101, Decibel, EvanCarroll, Intgr, Marokwitz, Ted, Turnstep, 8 anonymous edits

Expression index *Source*: http://en.wikipedia.org/w/index.php?oldid=580598515 *Contributors*: Cornellrockey, Decibel, Edward, GregorB, Infrangible, Marokwitz, Oakley77

Reverse index *Source*: http://en.wikipedia.org/w/index.php?oldid=543647686 *Contributors*: Andrewpmk, Beland, Donthom, Kku, Lfstevens, Miquonranger03, MuZemike, Owlmonkey, RainbowCrane, Riverpa, Snarius, Tolly4bolly, Unhingedloon, 7 anonymous edits

Bitmap index *Source*: http://en.wikipedia.org/w/index.php?oldid=598767975 *Contributors*: Abolen, Alex Muscar, Bgwhite, Bonustracks, Cactus26, Ceyockey, Conortodd, DanBealeCocks, David Eppstein, Deon Steyn, Doubleyouyou, Fanwaken, Fdeliege, Fivelittlemonkeys, Fred Bradstadt, Gazpacho, Geekchick77, GregorB, Headbomb, Intgr, Jerome Charles Potts, Kcarnold, LeilaniLad, Lfstevens, Lfstevens.us, Mike929t, N0xin, Neilc, Niels Grundtvig Nielsen, Oaf2, Peter.thejackos, Prakash Nadkarni, Radagast83, Ragib, Renata3, Rich Farmbrough, Riki, RobertG, SJ.van.Schaik, ShadowRangerRIT, Snodnipper, Ta bu shi da yu, Thekmc, TomHug, Touko vk, Turnstep, Vadmium, Whywhenwhohow, Zanetu, 60 anonymous edits

Inverted index *Source*: http://en.wikipedia.org/w/index.php?oldid=591814302 *Contributors*: AKA MBG, AlanUS, Andreas Kaufmann, Badgettrg, Beland, Cadillac, Chetvorno, Colonies Chris, Dcoetzee, EdwardLas, Fmccown, GregorB, Hyad, Jerome Charles Potts, Jfroelich, Jogloran, KJS77, Kku, Kolenskim, Marudubshinki, Mdomig, Mohamedadaly, Nigel V Thomas, Nils Grimsmo, Nivix, Pegacat, Ruud Koot, Short Circuit, Siriusvector, SqlPac, Svick, TheCois, Tinku99, Todorojo, X7q, Yurivict, 46 anonymous edits

Sargable *Source*: http://en.wikipedia.org/w/index.php?oldid=573806804 *Contributors*: ArmenInSeattle, Cadvga, Danim, Gertjanstrik, JeepdaySock, JustinLin, Rsocol, Scodeswiki, TubularWorld, 15 anonymous edits

V-optimal histograms *Source*: http://en.wikipedia.org/w/index.php?oldid=596384640 *Contributors*: AnAj, Davewild, Electriccatfish2, Fabrictramp, Garde, GregorB, JamesBWatson, John of Reading, Lavaka, LilHelpa, Melcombe, Nardling, Peni, Penwhale, Phonemic, Pnm, SMC, Sho222, Whpq, 6 anonymous edits

Cardinality (SQL statements) *Source*: http://en.wikipedia.org/w/index.php?oldid=595710037 *Contributors*: BostonRed, Closedmouth, DaBest1, DerekAsirvadem, Eric Burnett, GregorB, HornedKavu, Little Mountain 5, Mark Renier, Mathman1550, R'n'B, Rjwilmsi, Thal3s, Troels Arvin, 9 anonymous edits

Online aggregation *Source*: http://en.wikipedia.org/w/index.php?oldid=594792937 *Contributors*: Headbomb, Melchoir, Neilc, Np6

Very large database *Source*: http://en.wikipedia.org/w/index.php?oldid=591168098 *Contributors*: Akwatve, Beefyt, Beland, CapitalR, Crysb, Danim, Elwikipedista, Fkfe, Garygoh884, Kaihsu, Miym, NawlinWiki, Neilc, RJHall, Raysonho, Sderose, Y10k, 28 anonymous edits

Big data *Source*: http://en.wikipedia.org/w/index.php?oldid=599186097 *Contributors*: Abhishek1605, Accountdp, Aeusoes1, Almaz, Ampersandian, Analytics ireland, And Adoil Descended, Andrewman327, Anirudhrata, Anna Frodesiak, Apptrain, Arbitrarily0, Arcamacho, ArnoldReinhold, Asplanchna, Atlasowa, AtmosNews, Auntof6, AuthorAnil, Axeman89, Azra2013, B3t, Bar David, Behrad3d, Beland, Benboy00, Bgold12, BigDataGuru1, Bigdatavomit, BobGourley, BrighterTomorrow, Broeni, Camberleybates, Caraconan, Casieg, Checkingfax, Chengying10, Chire, Chris the speller, ChrisGualtieri, Cirt, Cowb0y, Dabramsdt, Danielg922, DavidKSchneider, Davidogm, Dawn Bard, DigitalDev, Dilaila, Dilaila123, Dimensionsix, Discospinster, Download, Drevicko, Edwinboothnyc, Epicgenius, Ethansdad, Evaluatorgroup, F3meyer, Faalagorn, ForumOxford Online, Fraggle81, Fvillanustre, Fylbecatulous, Gary Simon, Bsc, FCA, FBCS, CITP, Giftlite, Gilliam, Grantbow, Grinq, Haroldpolo, Helenellis, Henryyan, Hessmike, I dream of horses, I42, IjonTichyIjonTichy, Indianbusiness, IndustrialAutomationGuru, InfoCmplx, Jac16888, Jacoblarsen net, Jandalhandler, Jantana, Jarble, Jazzwang, Jean.julius, JeanneHolm, Jehochman, JenniferAndy, Jeremy Kolb, Jeremykemp, Jfmantis, Jim1138, Jj1236, Jkofron4, Joaquin008, Joe204, JohnBlackburne, Jojikiba, Jonathanchaitow, Jonesey95, Jordanzhang, Josephmarty, JuanCarlosBrandt, Katieh5584, Kdammers, Kforeman1, Khazar2, Kilopi, Kku, Krexer, Kuru, Lamp90, Langec, Laurawilber, Lawsonstu, Lmusher, Lspin011, Lumin, MMeTrew, MPH007, Madman2001, Magioladitis, MainFrame, Malleus Fatuorum, Manivannan pk, MarkTraceur, Marko Grobelnik, Martarius, Materialscientist, McSly, Mcioffi, Melcombe, Mgualtieri, Mherradora, Mhiji, Miakeay, Mild Bill Hiccup, Mingminchi, Miranche, Mjvaugh2, Mm479flarok, Moosehadley, Morganmissen, Morrisjd1, MrOllie, Msalganik, Mymallandnews, Noelwclarke, Northamerica1000, Nyq, Od Mishehu, Ohnoitsjamie, OnTheNet21, Ost316, Ottawahitech, Ottb19, P.r.newman, Pablomendes, Palosirkka, Parasdoshiblog, Pchackal, Pegua, Petermcelwee, Philip Trueman, Pinar, Pol098, Pramanicks, Prussonyc, Quibik, Qwertyus, RDBrown, Relyk, Resoru, Rich Farmbrough, Richard asr, Rick jens, Rjwilmsi, Rmyeid, Ryguyrg, Ryuch, Rzicari, Samw, Saturdayswiki, SchreiberBike, Scottishweather, Scottywong, Sean Quixote, Seherrell, Seppemans123, Shahbazali101, Shirishnetke, Sideways713, Siskus, Smallman12q, Socratesplato9, SteveLoughran, Stevebillings, Steven Walling, Sunray, Syscøp, TJLaher123, Tedder, The Letter J, TheJJJunk, Thevoid00, Thomas888b, Timtempleton, Tobych, Tomwsulcer, Topbanana, Tothwolf, Untioencolonia, Utcursch, Vaibhav017, ViaJFK, Viriditas, WH98, Warrenpd86, Widr, WikiMSL, Wikientg, Willymomo, Winchetan, Woohookitty, Xtzou, Yintan, Yourconnotation, Yragha, Yzerman123, 328 בן נרשון, anonymous edits

XLDB *Source*: http://en.wikipedia.org/w/index.php?oldid=572226439 *Contributors*: Danim, Delusion23, Drbreznjev, Jbecla, Kdborne, Khazar2, MZMcBride, PirateMink, 17 anonymous edits

Secondary database server *Source*: http://en.wikipedia.org/w/index.php?oldid=474234709 *Contributors*: Alvin Seville, BSTRhino, Dialectric, Reyk, 2 anonymous edits

Centralized database *Source*: http://en.wikipedia.org/w/index.php?oldid=543323916 *Contributors*: Alvin Seville, Bearcat, Bearian, Beland, Chrisportelli, Contagious2142, Danim, Hamtechperson, Katharineamy, Malcolma, Pjoef, Sole Soul, Spinoff, 7 anonymous edits

Distributed database *Source*: http://en.wikipedia.org/w/index.php?oldid=599206987 *Contributors*: Alansohn, Ammubhave, Anthony, Arthur Rubin, Beland, Bomazi, Bporopat, CanisRufus, Centrx, Compfreak7, Danim, Derbeth, Dewritech, Donhalcon, Dpkade, ENeville, Eastlaw, Eliz81, Gary, Gensanders, GeorgeBills, Gregbard, Hooperbloob, Hu, Intelligentfool, Intgr, JCLately, Jamelan, Jandalhandler, Jason.yosinski, Jim1138, KeyStroke, Kku, Knowlengr, Kuteni, Lguzenda, LilHelpa, M4gnum0n, Magioladitis, MelRobinson, Mere Mortal, Michaellacorte, Miym, Mogism, Mschlindwein, Nikhil search, Nivix, Nonnompow, Owenja, Ozsu, Passport90, Pebkac, Perfecto, Peruvianllama, PigFlu Oink, Prasanna8585, Ramaksoud2000, Satellizer, Sboehringer, Shibaji.paul, Sparky132, Squiddy, Sun Creator, Tempodivalse, Terry1944, TheThomas, Uncle Dick, Vektor330, Wbigger, Wizgha, 167 anonymous edits

Distributed database management system *Source*: http://en.wikipedia.org/w/index.php?oldid=548877651 *Contributors*: Alansohn, Ammubhave, Anthony, Arthur Rubin, Beland, Bomazi, Bporopat, CanisRufus, Centrx, Compfreak7, Danim, Derbeth, Dewritech, Donhalcon, Dpkade, ENeville, Eastlaw, Eliz81, Gary, Gensanders, GeorgeBills, Gregbard, Hooperbloob, Hu, Intelligentfool, Intgr, JCLately, Jamelan, Jandalhandler, Jason.yosinski, Jim1138, KeyStroke, Kku, Knowlengr, Kuteni, Lguzenda, LilHelpa, M4gnum0n, Magioladitis, MelRobinson, Mere Mortal, Michaellacorte, Miym, Mogism, Mschlindwein, Nikhil search, Nivix, Nonnompow, Owenja, Ozsu, Passport90, Pebkac, Perfecto, Peruvianllama, PigFlu Oink, Prasanna8585, Ramaksoud2000, Satellizer, Sboehringer, Shibaji.paul, Sparky132, Squiddy, Sun Creator, Tempodivalse, Terry1944, TheThomas, Uncle Dick, Vektor330, Wbigger, Wizgha, 167 anonymous edits

Distributed file system *Source*: http://en.wikipedia.org/w/index.php?oldid=564830376 *Contributors*: Amannm, AmolPatil99, Amux, Andy.Cowley, Beland, Bilbo1507, Cgomersall, Cloachland, Compfreak7, Dsimic, Eyrian, Goodzili, Int21h, Intgr, Lightmouse, Magioladitis, Obdurodon, Orlando.richards, Oxide94, Pnm, Polluks, Q4U, Roland Bavington, Schw3rt, Someone not using his real name, Thunderbritches, ToddDeLuca, UU, VanishedUserABC, Walter.Arrighetti, WhiteDragon, William Avery, Winterst, 62 מרול רפאים, anonymous edits

Distributed data store *Source*: http://en.wikipedia.org/w/index.php?oldid=595148269 *Contributors*: 4th-otaku, Anthony, Arikan, Badon, Bearcat, ChrisGualtieri, Crashie, Dadu, DataWraith, DavidCary, Elauminri, Frap, Gronky, Harryboyles, Hu12, Intgr, Jamelan, Jason Quinn, Jeff G., KeyStroke, Khiladi 2010, Kku, ManuSporny, Matspca, Muhammad Ikramulhaq Khawaja, Omegatron, Pichpich, RGripper, Remuel, Rhs98, Ringbang, Rjwilmsi, Robofish, Rogerdpack, Sae1962, ShaunMacPherson, Sureshperspecte, Tarret, Textractor, Tsuchiya Hikaru, Windmost, Wknight94, 17 anonymous edits

Heterogeneous Database System *Source*: http://en.wikipedia.org/w/index.php?oldid=573440154 *Contributors*: AnAj, Beland, Charlesdrakew, Haeinous, Jacobko, Minnaert, Moe Epsilon, Myasuda, NewEnglandYankee, RAFPeterM, Rob cowie, TheAMmollusc, Welsh, Wikilaurita, 12 anonymous edits

Simple Sloppy Semantic Database *Source*: http://en.wikipedia.org/w/index.php?oldid=593052542 *Contributors*: Auntof6, Bgwhite, Cander0000, Charivari, CommonsDelinker, Danim, Hdeus, Helenadeus, Hubpedia, Illia Connell, Jodi.a.schneider, Jonasalmeida, R'n'B, RDBrown, Rich Farmbrough, Rjwilmsi, Sfan00 IMG, Zundark, 4 anonymous edits

Distributed transaction *Source*: http://en.wikipedia.org/w/index.php?oldid=543572332 *Contributors*: Brick Thrower, Comps, Darklilac, Exceeder, GL1zdA, Gaius Cornelius, Gtrmp, JCLately, Jeskeca, John of Reading, MartinBiely, Mediran, Pcap, Pmerson, Ruud Koot, Saucepan, ShinyKnows, X42bn6, 16 anonymous edits

Network transparency *Source*: http://en.wikipedia.org/w/index.php?oldid=561198918 *Contributors*: Amcfreely, BD2412, Bevo, Charles Matthews, Duckbill, Frap, Holtzlpeter, Imediadisha, Jfmantis, MZMcBride, Mild Bill Hiccup, Mion, Mohamed-Ahmed-FG, Neelix, PSIplus, Remy B, Robert Brockway, Stassats, Teapeat, Valermos, VikasOjha, Xcntryrunna518, 14 anonymous edits

Long-lived transaction *Source*: http://en.wikipedia.org/w/index.php?oldid=341431432 *Contributors*: Ara vartanian, Woohookitty, 1 anonymous edits

Distributed concurrency control *Source*: http://en.wikipedia.org/w/index.php?oldid=562997414 *Contributors*: Comps, M4gnum0n, Miym, Ruud Koot, Squiddy, 2 anonymous edits

Consistency model *Source*: http://en.wikipedia.org/w/index.php?oldid=593847989 *Contributors*: Adrruiz, Aij, AllanBz, Ashuang, Beland, Blaisorblade, Chris Bainbridge, David Gerard, Dodiad, Dougher, Fresheneesz, Gregbard, Headbomb, Hu12, JCLately, Jesse Viviano, JonHarder, Karada, Krlis1337, LuisVeiga, MedeaMelana, Mo ainm, Momo54, Pbailis, Pnm, Rfl, Rscottbailey, Sae1962, Strait, Suruena, Szopen, Therealgandalf, Tothwolf, VanishedUserABC, Zaxius, 20 anonymous edits

Distributed Transaction Coordinator *Source*: http://en.wikipedia.org/w/index.php?oldid=541845494 *Contributors*: Bodnotbod, Bswilson, Darmanious, EdJohnston, JCLately, Jimmyzimms, PurpleHz, Pvss, S.riccardelli, Warren, Xiaomao123, Xpclient, 8 anonymous edits

Two-phase commit protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=590939648 *Contributors*: Bayle Shanks, Bestchai, Braincricket, CanisRufus, Choas, Ciphergoth, Comps, Coredesat, Cyberjoac, Daleh, Dubwai, Electriccatfish2, Emilong, Gdr, Gtrmp, JCLately, Ja 62, Jdeisenh, Jerome Baum, JingguoYao, Jwoodger, Killian441, LenzGr, Liao, Lkesteloot, Materialscientist, Mbloore, Neilc, PleaseStand, Pmerson, Pnm, R. S. Shaw, Rdsmith4, ReubenGarrett, Rich Farmbrough, Richard KAL, Ruud Koot, Rworsnop, SLipRaTi, Segv11, Stephanwehner, Suruena, Svick, Touko vk, Twimoki, Uglybugger, WikHead, Wktsugue, Yagibear, Zero sharp, 97 anonymous edits

Three-phase commit protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=593186422 *Contributors*: Aaron Schulz, Ampledata, Bayle Shanks, Ceyockey, Choas, Chris Q, Emilong, GreyCat, Guy Harris, IdishK, JCLately, Jsnx, Junche, Levin, MarkKampe, Megatronium, N.o.bouvin, Pnm, Remcgrath, Rjwilmsi, Rworsnop, Segv11, Trevor Johns, Zero sharp, 32 anonymous edits

Xeround *Source*: http://en.wikipedia.org/w/index.php?oldid=583505730 *Contributors*: Altered Walter, BD2412, Bearcat, Beland, Bunnyhop11, Cander0000, Compfreak7, Danim, Friism, Gilad.maayan, Hondel1970, Intgr, Isaac Rabinovitch, Jojalozzo, Katharineamy, MPH007, MrOllie, PhnomPencil, Rojer 31, Wavelength, Wilhelmina Will, 13 anonymous edits

Vector-field consistency *Source*: http://en.wikipedia.org/w/index.php?oldid=532923942 *Contributors*: Blaisorblade, Dougher, Fabrictramp, Katharineamy, LuisVeiga, Misarxist, Moorsmur, 4 anonymous edits

Storage area network *Source*: http://en.wikipedia.org/w/index.php?oldid=595107019 *Contributors*: Aaaidan, Aaronmsmith, Abune, Abznak, Academic Challenger, Agathoclea, Aghar1, Ahoerstemeier, Albertktau, Ale jrb, AlistairMcMillan, AnatolyVilchinsky, Annsilverthorn, ArielGold, Austinmurphy, Beland, Bezenek, Billca42, Bingoeleven, Blambore, Blaxthos, Blowdart, Bobo192, BoomerAB, Bovineone, BradBeattie, Brainsik, Brianhe, Brickmack, Brim, BrunoF, Butzmann, C5st4wr6ch, Cameron Scott, Cander0000, CapitalR, Ccordray, Chenzw, Chippi, Chrisbolt, Clawson, ClementSeveillac, CommonsDelinker, Compfreak7, Conquerist, Corti, Cuziyam, DeadlyAssassin, DeweyQ, Dfxdeimos, Dipankan001, Disavian, Disdero, Dispenser, Djcoolmax, Dmeranda, DokReggar, Dru of Id, Dullfig, Dyl, DéRahier, Ecemaml, Electriccatfish2, Emilyjetnexus, Epbr123, Evercat, Evomr05, Fabriciodosanjossilva, Faradayplank, Fastilysock, Fchams, Feline Hymnic, Flewis, FlyHigh, GKelley, Gardar Rurak, Gary09202000, Geokills, Gexxer72, Ghostreveries, Giftlite, Gigeoff, Gilliam, Gilo1969, Giraffedata, GoodwinC, Gracefool, Gregpavlov, Grenavitar, Gudeldar, HalfShadow, Hauhr, Hughcharlesparker, Ibmstorage, Imjustmatthew, Intgr, Issactang2, J36miles, JCLately, JDG, JWaters, Jacob.jose, JarlaxleArtemis, Jdforrester, Jdowland, JeLuF, Jeroenr, Jim1138, Jimbojw, Jimbreed, Jliv, Jmgonzalez, John Fader, JonHarder, Jondelac, JoshuaGrainger, Jrereeves, Jschnur, Jsteenhagen, Julesd, JzG, K mahesh85, Kace7, Kbdank71, Kbolino, Kbrose, Kevin B12, Kfor, Khalnath, Killcmd, Kl4m, Klaus100, Kornfan71, Kozuch, Ksacilotto, Kubanczyk, Kuthup, Kvedulv, Kvng, Kyle van der Meer, LCP, Lars, LeilaniLad, Levin, Lightmouse, Lingliu07, Loosecannon93, Lord Pistachio, Lore uni, Loren.wilton, Lradrama, Luk, Luspari, MC MasterChef, MaGa, Maderiaboy, Maikel, Makele-90, Markhoney, Marx01, Materialscientist, Matt Britt, Matt Crypto, Mclayto, Mediran, Mel Etitis, Mendel, Mennis, Merope, MiNombreDeGuerra, Michael Hardy, Mikeblas, Mikecron, Minna Sora no Shita, Modster, Mohanpatamata, Moppet65535, Mrholybrain, Mvdhout, Mwtoews, N328KF, NapoliRoma, Neilc, New World Tech Girl, Nicholsr, Nick5768, Nneonneo, Oben, Oceco, Oconnorrory, Ohconfucius, Paul Koning, Pavel Vozenilek, Pelago, Perfecto, Perhelion, Phatom87, PhilHibbs, Pip11, Piperfect, Pnm, Polarscribe, Portsean, Possum, Preetso19, ProfessorEmc2, Quotemstr, R. S. Shaw, Radagast, Radagast83, Raysonho, Red Director, Relativitydrive, Rhobite, Rich Janis, Rickjpelleg, Rjwilmsi, Rluchs, Robofish, Rogerio25, Ronz, Roux-HG, SJP, Sae1962, Scienceguy8m, Seashorewiki, Sfoskett, Shshme, Sigmundur, Silverrage, Slambo, Snori, Soliver Golden, Soumyasch, Squids and Chips, Srleffler, Straif, Strait, Suffusion of Yellow, Suruena, T-storm, TMC1221, TastyPoutine, Tbhotch, Technopat, TheBlueFox, Thomas.uhl, Thunderbritches, Tig528, Tjofras, Tombohannon, Touch Of Light, Tsaylor, Ugog Nizdast, Ularre, VampWillow, Vico311, Vijayram900, Vrenator, Wernher, Wetodid, Whirlwindcircleflat, WikiLeon, Wknight94, Wmahan, YUL89YYZ, Youcangetabhi, Zedmelon, Érico Júnior Wouters, Тивеполник, 676 anonymous edits

Partition (database) *Source*: http://en.wikipedia.org/w/index.php?oldid=598419914 *Contributors*: Alai, Andrew.rose, Andy Dingley, Angusmca, Beaddy1238, Brian1975, Ccubedd, Ceva, Chrisarnesen, Doubleplusjeff, Drrngrvy, Ehn, Fholahan, Foonly, Geoffmcgrath, Georgewilliamherbert, Habitmelon, Highflyerjl, Isheden, Jamelan, Jan.hasller, Jonstephens, Lurkfest, Mark Renier, Materialscientist, Mdfst13, Mikeblas, Mindmatrix, MrOllie, Peak, Pinkadelica, S.K., Salobaas, Saravu2k, Semmerich, SmallRepair, Stevelihn, Vishnava, Wordstext, Yahya Abdal-Aziz, 43 anonymous edits

Shared nothing architecture *Source*: http://en.wikipedia.org/w/index.php?oldid=595892622 *Contributors*: Aaron Brenneman, Akibalogh, Andy Dingley, Banej, Booler80, Bulwersator, Cedders, Compfreak7, Craig Stuntz, DatabaseDiva, Dpm64, Edward, Epbr123, Eve Teschlemacher, Frap, GregorB, GreysAnatomy, HelloAnnyong, Intgr, JnRouvignac, Johnsantaferraro, JonHarder, Julesd, Kbdank71, Kognitio, LenzGr, Luxxor, MickScott, MrOllie, Neilc, Nixdorf, Ore4444, Oshadmon, Peterdjones, Peterjmerelise, Peterl, Petri Krohn, Pigsonthewing, Rednblu, Robert Brockway, Sole Soul, Soulmerge, Stephan Leeds, TechPurism, TheTito, Thesaturnine, TimBentley, Tohd8BohaithuGh1, VMryder, W Nowicki, 49 anonymous edits

Shard (database architecture) *Source*: http://en.wikipedia.org/w/index.php?oldid=599211736 *Contributors*: Andy Dingley, Angusmca, Bezenek, Cc4fire, Cfupdate, Crysb, Cswpride, Cybercobra, David9911, Delicious carbuncle, Dougher, Drewandersonnz, Dthomsen8, Eleschinski2000, Eric-vrcl, Fche, FontOfSomeKnowledge, Gautamsomani, Haeinous, Hairy Dude, Isheden, Jacosi, Jadahl, Jaybear, Jdlambert, Kenfar, Liran.zelkha, MeganShield, Mopashinov, MusikAnimal, Noq, Otisg, Rfl, Rfportilla, Sae1962, Sanspeur, SmallRepair, Steipe, Tmalone22, Underpants, Visvadinu, Wainstead, Winterst, Winterstein, X7q, 95 anonymous edits

Quorum (distributed computing) *Source*: http://en.wikipedia.org/w/index.php?oldid=542897992 *Contributors*: Ashishgandhe, Balabiot, Barrettcook, Carpenter aka, Dex, Fratrep, Hcobb, Jeandré du Toit, Pnm, Singaporian, 9 anonymous edits

Physical data model *Source*: http://en.wikipedia.org/w/index.php?oldid=542129661 *Contributors*: Beland, Colinb007, DickieRose, Half-Blood Auror, JLaTondre, Jandalhandler, John.mickey, Kbrose, KeyStroke, MZMcBride, Mark Renier, Mdd, MountainSplash, Oxymoron83, PamD, Pratyeka, Rockfang, Walter Görlitz, Wittzi, 29 anonymous edits

Physical schema *Source*: http://en.wikipedia.org/w/index.php?oldid=542129607 *Contributors*: Avalon, Beland, ChrisGualtieri, Closedmouth, Danim, DavidCHay, Dawynn, Greenrd, Jb-adder, Jhabib, Johan Natt och Dag, KeyStroke, Mark Renier, Tinucherian, Tribaal, Varlaam, 12 anonymous edits

Storage model *Source*: http://en.wikipedia.org/w/index.php?oldid=415292239 *Contributors*: Bearcat, Can't sleep, clown will eat me, Egrabczewski, Malcolma, Nixeagle, Rich Farmbrough, Tikiwont

Storage block *Source*: http://en.wikipedia.org/w/index.php?oldid=475436853 *Contributors*: Aspects, EnriqueVillar, Kubanczyk, M-le-mot-dit, Syp, Thumperward, XLerate

Tablespace *Source*: http://en.wikipedia.org/w/index.php?oldid=597155643 *Contributors*: Anjum Vear, Briefstyle, Bruce1ee, D6, DARTH SIDIOUS 2, JLaTondre, Jandalhandler, MarcM1098, MightyWarrior, Rich Farmbrough, Rjwilmsi, Seanohagan, Srbauer, Thumperward, Warthog32, Zhenqinli, 40 anonymous edits

Database tuning *Source*: http://en.wikipedia.org/w/index.php?oldid=595557589 *Contributors*: Abdull, Aednichols, Ben D., Chronomaster5779, Czarkoff, Danim, DerHexer, Elkman, Eproegler, ErrantX, Flat Out, Frehley, Fuhghettaboutit, Gamsbart, Gilliam, GravRidr, Intgr, J.delanoy, J36miles, Jeepday, Jitse Niesen, JonHarder, Kingturtle, Kukini, Mhhza, Morris7200, Pearle, Referencefreak, Rjwilmsi, Spyindiasho, Woohookitty, 17 anonymous edits

Database dump *Source*: http://en.wikipedia.org/w/index.php?oldid=550796625 *Contributors*: AN(Ger), Andreas Kaufmann, Arvindn, Connor Behan, Damian Yerrick, Dannywein, Emj, Filippof, Frap, Greenrd, HLHJ, Ixfd64, Kerowyn, Longhair, Mikeblas, NSR, Pearle, Rev3lati0n, Rjst, Rjstott, Sebastian Wallroth, Swestlake, Teratornis, Tetromino, Versageek, Winashwin24, Ynhockey, 11 anonymous edits

Spindling *Source*: http://en.wikipedia.org/w/index.php?oldid=545131744 *Contributors*: Fratrep, Jll, Jorge Stolfi, Music Sorter, Neelix, Richy, Wolfkeeper

Data mapping *Source*: http://en.wikipedia.org/w/index.php?oldid=586804417 *Contributors*: Adoligno, Appraiser, Bluemoose, Bluerasberry, CWeinmann, Danim, David Eppstein, Dawnseeker2000, Diego Moya, Dmccreary, Elipongo, Femto, Hjebbers, Iworld-media, Jimshilts, Joinarnold, JonHarder, Kku, Krauss, Lindsey Kuper, M4gnum0n, Mark Renier, Mdd, Michael Hardy, MrOllie, Mubde, Nadraabsi, Oktie, PTSE, Prairiesmoke, RHaworth, RayGates, Reinyday, Renatko, Ronz, Spalding, Tgold65, U 06111976, Veyklevar, Welsh, XMLer, Yunshui, 49 anonymous edits

Semantic integration *Source*: http://en.wikipedia.org/w/index.php?oldid=596381944 *Contributors*: Adoligno, Ansell, Artw, Bethmanj, Cainej, Conscious, Daveh1, Dawnseeker2000, Dmccreary, Edward, Fawsy, Fparreiras, Gagsie, Gdm, Hillarivallen, Hu12, HumanCyclist, Jdatsoton, John of Reading, Kevin.cohen, Kimleonard, Kku, Koi.lover, Krlis1337, MER-C, Mdd, Michael Slone, MilerWhite, Minnaert, Morgaladh, Neverclear, Robykiwi, Rogerd, Ronz, Ruud Koot, SchuminWeb, Stfg, The Anome, Triadic2000, Veinor, Yaron K., 25 anonymous edits

Semantic translation *Source*: http://en.wikipedia.org/w/index.php?oldid=524231835 *Contributors*: Andy Dingley, Dmccreary, Kku, Mark Renier, Quiddity, R'n'B, RayGates, Ruud Koot, Suruena, Theespuja, 3 anonymous edits

Record linkage *Source*: http://en.wikipedia.org/w/index.php?oldid=597552478 *Contributors*: 3mta3, Avenue, Barkeep, BrandonBigheart, Bsalvy, Calzakk, Charles Matthews, ChrisGualtieri, Colonies Chris, Cyberwolf UK, DamarisC, David Eppstein, Dawnseeker2000, Demetri1968, Deus Ex, Dheeruyadav, Dongwon, Download, Dupe2012, Edward, Eeekster, Eequor, Everyking, Greggwebster, Gènédic, H@r@ld, Headbomb, Infosolve, Ipeirotis, Isopropyl, Jan Spousta, Joerg Reiher, Kashtabelshata, LarsMarius, Linas, MacTed, Maga, Maurreen, McB, Mean as custard, Myanw, Notecnirp, Ntroutman, Oscmt, Osnetwork, OverlordQ, PGalliadi, Pcullinane, Pfctdayelise, Pjurczy, RJFJR, Ranboii, ReverendSam, Rjwilmsi, Schmke, Simon kue 1305, Skr15081997, Staszek Lem, Thegerf, Thomas Larsen, Tobias Bergemann, Travuun, Tutanchmaat, WBardwin, Will2k, Xuul, 85 anonymous edits

Metadata discovery *Source*: http://en.wikipedia.org/w/index.php?oldid=568652518 *Contributors*: Celticst, Conscious, DEddy, Devarakondar, Dmccreary, Godbyk, GregorB, Henrylam, Kuru, Leszek Jańczuk, MZMcBride, Ruud Koot, Tthheeppaarrttyy, 38 anonymous edits

Schema matching *Source*: http://en.wikipedia.org/w/index.php?oldid=571312663 *Contributors*: Drbreznjev, Giraffedata, Goochelaar, Kku, Krlis1337, Lycurgus, Meena610, Pnm, Rjwilmsi, SwisterTwister, U 06111976, Vinmal, 28 anonymous edits

Schema crosswalk *Source*: http://en.wikipedia.org/w/index.php?oldid=579621445 *Contributors*: AvicAWB, ChrisGualtieri, Felix Folio Secundus, Harald Lordick, JvrJava, KenGoudsward, KowgirlKate, M4gnum0n, Matchups, Morganwallce, Pnm, Rillian, Sephiroth storm, Smartbroad, Stevage, Zigger, 6 anonymous edits

Schema evolution *Source*: http://en.wikipedia.org/w/index.php?oldid=545324594 *Contributors*: Andreas Kaufmann, BabbaQ, D6, Geoffspeicher, Julesd, Krlis1337, Lockley, Malcolma, TakuyaMurata, 14 anonymous edits

Data integration *Source*: http://en.wikipedia.org/w/index.php?oldid=597037364 *Contributors*: 0307Morgan, AJackl, Aasia A, Abduallah mohammed, Adavidb, Adoligno, Alexglaros, Allen0811, AnmaFinotera, Bobdc, Bookofjude, Bradjuhasz, Brandon.degraaf, Brevity, Brezenbene, Byapparov, Carbo1200, Carly805, Cdrdata, Char808, Ckatz, Cskelly68, DEddy, DaleWWay, DamsonDragon, DanPrimus, Daveh1, Dawnseeker2000, Deshkanna, Dibeshshakya, Diego Moya, Dmitry St, Donner60, Dsexpert, Egsan Bacon, Erick.Antezana, Fimelb, Fiorano Software, Fæ, Giftlite, Giraffedata, Gscshoyru, Hanifbbz, Hhenkel, Hk777, Hu12, J mareeswaran, Jbtechwriter, JimmyJL, Jimshilts, Kimchi.sg, Kku, KotetsuKat, Krlis1337, Kuru, Lemnaminor, Leobard, LilHelpa, Lyondif02, MC10, MacTed, MainFrame, Mandarax, Mark Renier, Markcowan, Martarius, Mdd, Meena610, MilerWhite, Nadraabsi, Nav102, Niffweed17, Nposs, Nyttend, Philcoxwb, Pleasetrustme, Poesys, Prashanthmcr, RHaworth, Radagast83, Renatko, Rjd170, Rjwilmsi, Ronz, Rror, Russianspy3, Sandyiit, Schastain, Sfingram, Shaw76, Smueller74, Stcrowley, StephenWeber, SuperKevinP, Tcc619, Tetracube, Tobias Bergemann, Triadic2000, TurningWork, U 06111976, Veinor, Walk&check, Walter Görlitz, Winterst, 98 anonymous edits

Ontology-based data integration *Source*: http://en.wikipedia.org/w/index.php?oldid=553554433 *Contributors*: Arnauldvm, Eternal Pink, Frap, Hk777, Hu12, Kku, Krlis1337, Linikujp, Mild Bill Hiccup, Minnaert, Phil Boswell, PhnomPencil, Rjwilmsi, 4 anonymous edits

Ontology merging *Source*: http://en.wikipedia.org/w/index.php?oldid=543030469 *Contributors*: John, Kku, Kuru, Minnaert, Oleg Alexandrov, ReverendSam, 1 anonymous edits

MapReduce *Source*: http://en.wikipedia.org/w/index.php?oldid=598468945 *Contributors*: 900mill, A2Kafir, A5b, Abuyakl, Ade oshineye, AlanUS, Alexwg, Andreas Kaufmann, Audriusa, AxelBoldt, Barry K. Nathan, Beefyt, Bernhard Bauer, Bezenek, Bouktin, Burtonator, Butterwell, Caiyu, Cdiggins, Cedar101, Charles Matthews, Chewie, Chire, Chunbinlin, Clngre, CmdrRickHunter, Coherers, Compfreak7, Corrector623, Crishoj, Dangiankit, Danny Rathjens, Danpovey, Darsenault, David Eppstein, DavidBrooksPokorny, Davidogm, DeC, Dickdock, Diego Moya, Dkf11, Dmccreary, Don.doe, Dougher, Dr.sumeetmalhotra, Dtanders, Dysepsion, Edward, EncMstr, Evolvingjerk, F.ozgur.catak, FatalError, FghIJklm, Frood, Frze, Gajakannan, Gareth Jones, Gilles.Fedak, Giraffedata, Gjnaasaa, Greg 12000, GregorySmith, Gwern, Gwernol, Gyrae, Hairy Dude, Hannes Röst, Heelmijnlevenlang, Hmains, IComputerSaysNo, Ignatzmice, Inimino, Iomartin, J mareeswaran, JCDenton2052, JLaTondre, Jaliyae, Jamitzky, Jandalhandler, Jayron32, Jbo5112, Jenvor, Jfabrizio84, Jhellerstein, Jim1138, Jivecat, Jose Icaza, Joshsteiner, Jswanson3141, Julesd, Kaicarver, Kku, Kocio, Kohistan, Larry V, LilHelpa, Luke Lonergan, LuzGamma, MParaz, Mahbubur-r-aaman, Marudubshinki, Mate Hegyhati, Mauls, McSly, Mdz, Merlinthe, Mindmatrix, Miym, Moe Epsilon, Mortense, Mrflip, Ne vasya, NicDumZ, Nilei81, NotSoAnonymous54, OsamaK, Owen.omalley, PahaOlo, Panoat, Park Flier, PatrickFisher, PaulMEdwards, Pgan002, Phoe6, Pjoef, PlatoCantRepent, Poposhka, Ppgardne, Qsq, Quatloo, Qwertyus, RL0919, RadioYeti, Ravedave, Rcsheets, Rehashed, Rhwawn, Rich Farmbrough, Ripe, Rjwilmsi, Roly Perera, Ronewolf, Ruakh, Ryanreporter, Rybec, Saizai, Samdutton, Seaphoto, SegfaultWizard, Sepreece, Sfermigier, Shenme, Siruguri, SmallRepair, Sprhodes, Starwiz, SteveLoughran, StijnDeWitt, Sumail, Superm401, Sushi Tax, SymlynX, T0ny, Tabletrack53, TedPavlic, ThiagoNegriHunz, Thingg, Timdorr, Tlesher, TraciFaahkahrn, Tree Biting Conspiracy, Twn, Typon, Vrenator, Werdna, Wickethewok, Wonko9, Wujj123456, X7q, Xhienne, YPavan, Yhuai, Yozh, ДмитрОст, 335 anonymous edits

Apache Hadoop *Source*: http://en.wikipedia.org/w/index.php?oldid=599247489 *Contributors*: 900mill, AGraziani, Adam kawa 85, AgniKalpa, Akim Demaille, Alexbrewer, Alexf, Alkini, AngryBear, Angrytoast, Anna Frodesiak, AnonMoos, Aomarks, Arjant, Atree, BaldPark, Banana04131, Beland, BenBildstein, Bilbobaggginz, Billytkid, Blacknosugar, Blaisorblade, Blinkov, Bodzasfanta, Bostonvaulter, Brian Geppert, Bunnyhop11, CJGarner, Caiaffa, Camerojo, Capitalismojo, Captain Screebo, CasualVisitor, CeciliaPang, Chafe66, Chris the speller, Compfreak7, Cosmotron, Craigacp, Curlybraces, Dan100, DaoKaioshin, Davidogm, Den fjättrade ankan, Derek R Bullamore, Dguedo, Diego Moya, Dietcoke3.14, Donner60, Doronve, DougCutting, Drmies, Dskrvk, Dujunping, EIFY, Emocat, EoGuy, Faisal.akeel, Feliciafay, Flyer22, FontOfSomeKnowledge, Frap, Free Software Knight, Fun indra, Fvillanustre, GeneGotimer, Gilliam, Gioto, GoingBatty, GrantIngersoll, GregorB, GriffinGrl, Gsoverby, Gwern, H.E. Hall, HMSSolent, Hadoopor, Hebrides, Hervegirod, His Wikiness, Hu12, HughNo, Hult041956, Hydrox, IjonTichyIjonTichy, IlyaHaykinson, InMktgWeTrust, Intgr, JHunterJ, JLaTondre, Jackson Peebles, Jacobbbc, Jain.anilk, Jan Spousta, Jarble, Jdcourco, Jeremykemp, Jeric14, JerkerNyberg, Jim380, JimJJewett, Joebeone, John Cline, John Vandenberg, John kreisa, John of Reading, Jojalozzo, Jonsafari, Jordanzhang, Josve05a, Jreinsch, Justpossible, Kajaro, Kephir, Kerrick Staley, Kesla, Khalid hassani, Khazar2, Knoche, Kompere, Kotalampi, Krenair, Kuru, Kwertii, Labongo, Leonidas from XIV, LightningDragon, Liorrokach, LittleBenW, LodeRunner, Lollerskates, Lord.Quackstar, Lotje, Lsmll, LucienK, LustreOne, Mahbubur-r-aaman, Maju wiki, Mand Beckett, Marchingknight11, Markpeak, MartinMichlmayr, Marudubshinki, Mblumber, McSly, Mediran, Metroshica, Mg acom, Michael Hardy, Michaeldweir, Michaelmalak, MikeMayer, Mindmatrix, Mingfai, Mjf19, Mlibby, Mlogic, Mortense, MrOllie, Mschober65, Mukesh.khandelwal, Mutt Lunker, Mwtoews, Nafaabout, Nameispete, NapoliRoma, NawlinWiki, Nealmcb, Neilc, Nick Levine, Nmilford, NuclearWizard, Nullwriter, OS2Warp, Ohnoitsjamie, Oleg Alexandrov, Only2sea, Orenfalkowitz, Ost316, Owen.omalley, Parent5446, Paris.butterfield, Patodirahul, Paulboal, Peaceray, Pgan002, Pmj005, PokeYourHeadOff, Quebec99, Ravi.panditsadreena, Raysonho, Rdashore37, Redragon104, Renatokeshet, Research186, Rfc1394, RickFarnell, Rickyphyllis, Ronebofh, Ronnie6657, Rulerofutumno, SColombo, Sae1962, Samataha, SamuelScarano, ScotXW, Sepreece, Serent, Sheilakinsella, Shock an awe, Snorgway, Stennie, SteveLoughran, Straxus, Sumail, Superm401, Svick, Swooledge, TJRC, Tabledhote, Tabletrack53, Tagus, Talbrech, Talkietoaster-nc, TaranSingh, Tas50, Techgirl007, Techlady, TedDunning, Themfromspace, Theneoindian, Thorwald, Thumperward, Toreau, Trappist the monk, Tsasaa12, Tsunanet, Unamigo, Vahid83, Vertium, Vinoduec, W Nowicki, Wadsworth, Wangchuck50, Wavelength, Widr, Wikfr, WikiScrubber, Windmost, Winterst, Y10k, YPavan, Yanngeffrotin, Yintan, Zak.estrada, 446 anonymous edits

Pig (programming language) *Source*: http://en.wikipedia.org/w/index.php?oldid=504263317 *Contributors*: Cedar101, Dougher, Dragan00000, Eugene Pivovarov, Jonsafari, Kku, Kocio, Malleus Fatuorum, Qwertyus, Raysonho, Renegade Lisp, Sgoder, Sorenriise, Spencerk, Timwi, Topbanana, 20 anonymous edits

H-Store *Source*: http://en.wikipedia.org/w/index.php?oldid=592290983 *Contributors*: Apavlo, LeonWrinkles, Maury Markowitz, Msnicki, Nabla, RandyLaheyUK, RedWolf, Rjwilmsi, Stuartyeates, Timotheus Canens, UMD-Database, 4 anonymous edits

Data control language *Source*: http://en.wikipedia.org/w/index.php?oldid=579764462 *Contributors*: Alleborgo, Altenmann, Anna Lincoln, Avicennasis, Chupon, CodeNaked, Dfetter, GregorB, Gwernol, Isotope23, Kayau, Kbrose, Kelti, OMouse, Slipstream, SqlPac, The Thing That Should Not Be, 32 anonymous edits

SQL injection *Source*: http://en.wikipedia.org/w/index.php?oldid=598585539 *Contributors*: -), .anaconda, A bit iffy, Abdurrahman,T, Adward555, Af648, Alansohn, Alerante, Alex Marandon, AllenZh, Americasroof, Aminadav, AndrewWTaylor, AndyDent, AndyHassall, Aneeshjoseph, AnmaFinotera, Anna Lincoln, Antientropic, Apokrif, Arada2112, ArielGold, ArnoldReinhold, Aron.Foster, Arowhun, Ashikali1607, Atheuz, Audriusa, Avinash SoftwareDevloper, Avoided, Axeman89, Ayolucas, Badgernet, BaldPark, Balusc, Banej, Belal qudah, Benjamin Pineau, Bensonwu, Bevnet, Bevo, Bikepunk2, Biskeh, Blake-, BobKeim, Bogdangiusca, Bookbrad, Brianski, Bruce1ee, Btx40, CAN, CKlunck, Caesura, Caim, Calabe1992, Canley, Capitalismojo,

Business intelligence *Source*: http://en.wikipedia.org/w/index.php?oldid=598605531 *Contributors*: A40220, ABCD, AMJBIUser, AVM, Aadeel, Aaron Brenneman, Abhishek191288, Ademkader, AdjustablePliers, Aetheling, Aexis, Alansohn, Alberrosidus, Alem, Alexandra31, Alexf, AlistairMcMillan, Alpha Quadrant (alt), Ananthnats, AndriuZ, Andy Dingley, Ansumang, Ant, Apolitano, Arcann, Arthena, Arthur Rubin, AsceticRose, Ashil04, Aspandyar, Az1568, BIcurious3334, Bansipatel, Batra, Bcarrdba, Beardo, Becky Sayles, Beetstra, Beland, Beroneous, Bevelson, Bgwhite, Bharatcit, BlackLips, Blackstar138, BlaineKohl, Blork-mtl, Bonanjef, Bpm123, Bpmbooks, Brick Thrower, Brookie, Butterwell, Camw, CaveJohnson, Ceranthor, CharlesHoffman, Charlesmnicholls, Chase me ladies, I'm the Cavalry, Chris the speller, Chrisawiki, Chrisvonsimson, Chuckrussell, Chuq, Codeculturist, CommodiCast, Compprof9000, Coolpriyanka10, Corp Vision, Cquels, Cryptblade, Crysb, Czenek, DMG413, DanDoughty, DanMS, Dancter, Danielsmith, DauphineBI, DavidDouthitt, DePiep, Denisarona, Dhavalp453, Discospinster, Dkrapohl, Dnazip, Dnedzel, DouglasGreen, Dr Gangrene, Dr.apostrophe, Drcwright, Dreamrequest, Dwandelt, Edcolins, Edit06, Einsteinlebt, Ejosse1, Eken7, ElixirTechnology, Eliyak, Elvis, Entgroupzd, Ergoodell, Erianna, Ermite, Ethansdad, Etz Haim, Evans1982, Evil Monkey, Extransit, Fauxstar, Folajimi, Force88, Forceblue, Fraggle81, FrancoGG, Frederikton, Fredsmith2, Fæ, Gary, Gary a mason, Genuinedifference, Ginsuloft, Glane23, Glaugh, Golbez, Goyalaishwarya, Grafen, Greenboite, GregorB, Gscshoyru, Guillom, Guppyfinsoup, Gwalarn, HMishkoff, Halfgoku, Hanantaiber, Hans Genten, Happyinmaine, Heirpixel, Helwr, HowardDresner, Hyphen DJW, ITPerforms, ITtalker, IW.HG, Iaantheron, Iamthenewno2, Ianhowlett, Intelligentknowledgeyoudrive, Intergalacticz9, Ionium, Ireas, Islamomt, IvanLanin, Ivytrejo, JEL888, JVRudnick, Jaej, Jahub, Janner210, Jay, Jcarroll, Jeff G., Jeff3000, Jehan21, Jesaisca, Jim380, Jkofron4, Jmkim dot com, Joel7687, Joelm, John Vandenberg, John ellenberger, Joshua.pierce84, Jpnofseattle, Jsmith1108, JukoFF, Julianclark, Just zis Guy, you know?, Jvlock527, Jwcga, Kadambarid, Karl-Henner, Kbeneby, Kellylautt, KeyStroke, Khalid hassani, Khazar2, Kit Berg, Kku, Krich, Kuru, L Kensington, L'Aquatique, Lancet75, Langbk01, Larrymcp, Leandrod, Lfstevens, Liface, Logical Cowboy, Loripiquet, Lovedemon84, Lucky 6.9, Lupo, MER-C, MPerel, Macrakis, Makecat, Man koznar, Mangotron, Manning Bartlett, Manop, Marc Schønwandt, Mark Bosley, Mark Renier, Martarius, Martpol, Materialscientist, Maurreen, McGeddon, Mcclarke, Mdd, Meg2765, Mehtasanjay, Melcombe, Mgt88drcr, Michael Hardy, Michael.h.zimmerman, Mikeblas, Mikevandeneijnden, Mindmatrix, Mirv, Mitrius, Mkoval, Mogism, Momotoshi, Moonriddengirl, Mr.Gaebrial, MrOllie, Muu-karhu, Mydogategodshat, Mymallandnews, Naniwako, Natasha81, Navvod, Ncw0617, Nhgaudreau, Nick Levine, Nixdorf, Nmourfield, Norm, Nposs, Nraden, Nsaa, ObserverToSee, Ohconfucius, Ohnoitsjamie, OnBeyondZebrax, OnTheNet21, Opagecrtr, Openstrings, Outbackprincess, Pacific202, Pamela Haas, PaulHanson, Pedant17, PerfectStorm, Perohanych, Peters72, Peyre, Phani96, Philip Trueman, Piano non troppo, Pmresource, Pravisurabhi, Prazan, Priyank bolia, Prolog, Psb777, Qarakesek, Qqppqqpp, Quantpole, QuiteUnusual, Qwyrxian, R'n'B, RJHall, Racecarradar, Rajashekar iitm, Rayrubenstein, ReclaGroup, Rednblu, Reverend T. R. Malthus, Rhobite, Rich Janis, Rickybarron, Riyadmks, Roberta F., Robiminer, Roc, Rollins83, Rongou, Ronhjones, Ronz, Ruislick0, Ryan Rasmussen, S.K., SEOtools, Sarnholm, Saturnight, Schniider, Seankenalty, SebastianHelm, SethGrimes, Setti, Shadowjams, ShelfSkewed, Sierramadrid, Sinotara, Siryendor, SkyWalker, Slant, Snowolf, Srknet, Srkview, Stationcall, Steelsabre, Stefanomione, Stephen, Stevage, StuffOfInterest, Sualfradique, Supercactus, Superm401, Supertouch, Sutanupaul, Svetovid, Swells65, Technologyvoices, Technopat, TexasAndroid, TheKMan, TheWakeUpFactory, TigerShark, TimMulherin, Tjic, Tjtyrrell, Tomhubbard, Tompana82, Traroth, Travis.a.buckingham, Triplestop, Trusilver, Ukpremier, Urchandru, Vasant Dhar, Vaulttech, Veinor, ViriiK, WLU, Waggers, Warren, Wavelength, Welsh, Wendecover, Wernight, Wiki episteme, Wikidan829, Wikiolap, Wilcoxaj, Wile E. Heresiarch, WinterSpw, Wondigoma, Woohookitty, Woz2, Wperdue, WriterListener, Writerguy71, Wsvlqc, Wxhat1, X0lani, Xjengvyen, Xyzzyplugh, Y.Kondrykava, YK Times, Yanis ahmed, Yasst8, Yorkshiresoul, ZimZalaBim, Zkhall, Zntrip, Zzuuzz, דוד ש', นคเรศ, 965 anonymous edits

Reactive business intelligence *Source*: http://en.wikipedia.org/w/index.php?oldid=479862944 *Contributors*: Bearcat, Brunato, Chire, Leonardo61, Marcuswikipedian, Michael Hardy, Robiminer, Tony1, 1 anonymous edits

Business analytics *Source*: http://en.wikipedia.org/w/index.php?oldid=593443349 *Contributors*: Alvestrand, AmyDenise, B, Boxplot, Chire, Cnbrb, Crmguru2008, Crysb, DeepOpinion, Dnedzel, Dries Debbaut, Earthlyreason, Emcien, Emilygracedell, Ethansdad, Founder DIPM Institute, Fratrep, Gscshoyru, HMSSolent, Hans Genten, Helwr, Huang cynthia, Idea Farm, JLaTondre, Jinij, Kerenb, Kku, Kuru, Mdd, Melcombe, Michael Devore, Michael Hardy, MrOllie, Niceguyedc, Oleg Alexandrov, Photo.iep, Picturepro, Pinethicket, RHaworth, Random user 39849958, Rick lightburn, Rjwilmsi, Rlistou, S.K., Sarnalios, Simonjohnpalmer, Singularit, Smithandteam, Tomas e, Ukpremier, Vlado1, Wbm1058, Wcrosbie, WhartonCAI, Woz2, Writerguy71, XpXiXpY, 93 anonymous edits

Sales intelligence *Source*: http://en.wikipedia.org/w/index.php?oldid=572894627 *Contributors*: AdamSales, Ahunt, Ajaics, Dialectric, Fabrictramp, Gymshaw, Klausness, Kuru, Lightlowemon, Metricschannel, Nwoirhaye, Risraelkloss, Shalom Yechiel, VanishedUserABC, Vecta1, Zaki Usman, 15 anonymous edits

Performance intelligence *Source*: http://en.wikipedia.org/w/index.php?oldid=305187289 *Contributors*: Enviroboy, PFHLai, Relevantsus, Skysmith, Wikiuser315

Data warehouse *Source*: http://en.wikipedia.org/w/index.php?oldid=598146362 *Contributors*: 069952497a, 99magred, A.amitkumar, ABMH, ALargeElk, Aaron Brenneman, Ace of Spades, ActivExpression, Aetherfukz, AgentCDE, Aitias, Allstarecho, Alqayoom, Altenmann, Amit8082, Amorim Parga, Andreid76, Andrewman327, Andy Dingley, Aneah, Angusmca, Ankit Maity, AnmaFinotera, Ansumang, Antonrojo, Apolitano, Arcann, Arch dude, Ari1974, Arielco, Arpabr, Asif earning, Asparagus, Atlantas, Atomician, Atw1996, Augbog, AutumnSnow, Avjoska, BanyanTree, Bdebbarma, Beardo, Berny, Bertport, Betacommand, Bgwhite, BigDunc, Binksternet, Blanchardb, Blue520, Bmotoc, Bobrayner, Bogey97, Brc4, BryantAvey, Btaub, Butros, C960657, CGerrard, Caltas, Camw, Carmichael, Catchsandy, Chaubals, Chowbok, Choyr, Chromatikoma, Chuq, Cobi, ColBatGuano, Comatmebro, Cometstyles, Compfreak7, Conversion script, Coolelle, Corruptcopper, Crysb, Crystalattice, Ctorchia87, Cybercobra, DARTH SIDIOUS 2, DBigXray, DNkYpNcH, DWBIGuru, Dan100, Dancter, Danielg922, Darth Zephyr, DatabaseDiva, Datawarehouseiq, Dawriter, DePiep, Decoy, Denisarona, DhirajGupta, Diannaa, Diego Moya, Dina, Disaas, Dividing, Dmccreary, Doc9871, Donaldpugh, Download, Dpm64, Drbreznjev, Dthvt, Długosz, Edward, Edwardmking, Elonka, Enimer, Er.piyushkp, Ericd, ErinRea, Ethansdad, Evert r, Falcon Kirtaran, Fbooth, Finlay McWalter, Foxj, Foxyliah, Fraggle81, Fred Bradstadt, Frze, Fudoreaper, Funandtrvl, Fyyer, Fæ, G716, Gabrielinux, Gachet, Gaius Cornelius, Gene Fellner, Ghewgill, Gianna ch, Gilliam, Giulio.orru, GreenInker, Greenlightwilly, GregorB, Grjuedes, Gscshoyru, Gskrzypczynski, Gurch, Gzkn, HJ Mitchell, Hadal, Hanifbbz, Hectorpa, Henridv, Hhultgren, Ianemitchell, Imdeadlymon, Intgr, Itmanning, IvanLanin, Ixfd64, J3st, JCLately, JForget, Jackfork, Jagannath6969, Jan.hasller, Jarble, Jaxl, Jay, Jburtenshaw, Jdrlundgren, Je ne détiens pas la vérité universelle, Jelena1, Jenks1987, Jgroove, Jguthaaz, JoeSmack, John Vandenberg, Joinarnold, Jomsviking, Jonwynne, JoyMundy, Joyous!, Jusdafax, Jwissick, Kadambarid, Kellylautt, Kingpin13, Kingpomba, Kiwi137, Kjkolb, Kku, Klausness, Kuru, Kwesterh, L Kensington, Lahiru k, Laurentseries, LcawteHuggle, Leandrod, Les boys, Lightmouse, Lingliu07, Little green rosetta, Llla32, Loonquawl, Lordzilla, Loren.wilton, Lostraven, Luyseyal, MHPSM, MHargraves, MK8, Madman2001, Madmonky, Mahuna2, MainFrame, Makro, Malcolmxl5, Mandarax, Manning Bartlett, Marcika, Mark Renier, Mark94502, Martarius, MartinsB, Mathonius, Maurreen, Mav, McSly, Mdd, MeekMark, Megras, Meyer, Mhooreman, Michael Devore, Michael Hardy, Michal Jurosz, Mike Rosoft, Mikeblas, Mindmatrix, Minervasolutions, Minimac, Minna Sora no Shita, Modify, Mpolichany, MrOllie, Mustafaisonline, Mwaci2, MyWorld, Mydogategodshat, Neilc, NickPenguin, NishithSingh, Nitin ravin, Nixdorf, Nn123645, Noah Salzman, Nraden, Nrahimian, Oaf2, Object01, Ojigiri, OliverKlozoff, Ondertitel, Ordoon, Oxymoron83, PC78, Philip Trueman, Piano non troppo, Pietrow, Pisceswzh, Playmobilonhishorse, Pnm, Propheticone, Qst, Quinsareth, R'n'B, RHaworth, RJaguar3, RScheiber, Ramesh l, Ranjran, RayGates, Raysonho, Rednblu, Reedy, Rhoegg, Ringbang, Rjanag, Rjwilmsi, Rknasc, Robdellis, Robert K S, RocklegendPoker, Roenbaeck, RomanEmbacher, Rrabins, Rschmertz, Rushbugled13, Ryan Rasmussen, S, SJP, ST47, Sabed Aashour, Sae1962, Salvio giuliano, Samroar, Sandeep4tech, Sarnholm, Savan3147754, Schul253, Scurra, Sean D Martin, SebastianHelm, Sesamevoila, Sfingram, Shaded0, Shadowjams, Shortbusmedia, Shubinator, Shwetank99, Sidhekin, SimpleTheory, Siroxo, Sixstone, Sleske, Sloan1919, Slowbro, SlubGlub, Snowolf, SoCalSuperEagle, SoSaysChappy, Soumyasch, Soundmind43, Spoxox, SqlPac, Srperez, Stefmol, Stephenb, Stephenpace, Stevag, Steve G, Steverapaport, Strongsauce, Studerby, Stvltvs, Subash.chandran007, Swatjester, Swintrob, TFinn734, THEN WHO WAS PHONE?, Tamarandom, Tapir Terrific, TechPurism, Techteachermayank, Telugucherla, Tgeller, The Thing That Should Not Be, Tide rolls, Tom Morris, Troels Arvin, Trusilver, Truther2012, Tvarnoe, Two Companions, Uršul, Utcursch, Vaibhaovaibhav, Vald, Van helsing, Vertium, Vincent jonsson, Vinod Alangaram, Vipinhari, Viridae, Volphy, WH98, Walk&check, Wanderingstew, Wavelength, Wedge3rd, WetherMan, WikHead, Wiki Raja, Wikiolap, Wikipelli, Wile E. Heresiarch, Will Beback, Willscraper, Wknight94, WojPob, Writerguy71, Wtmitchell, Xlaran, Yasth, ZimZalaBim, Zundark, 1127 anonymous edits

Data warehouse architectures *Source*: http://en.wikipedia.org/w/index.php?oldid=595100731 *Contributors*: AddWittyNameHere, Laurinavicius, Magioladitis, Mgoj, NDSteve10, Pascal666, Tassedethe, 13 anonymous edits

Data mart *Source*: http://en.wikipedia.org/w/index.php?oldid=586763210 *Contributors*: Adrignola, Andy Dingley, Arcann, Beardo, Betacommand, Bijee, Boli1107, Bonadea, Canadian-Bacon, Chipofbags, Ckw888, Crysb, DARTH SIDIOUS 2, Dally Horton, DePiep, Docu, Doniago, Dustmachine, Electron9, Elsendero, Eptalon, Eyhk, Fearlessunit, Fraggle81, Funnyfarmofdoom, GilCahana, GregorB, Gscshoyru, Gurch, Isomorphic, Jamesx12345, Jay, Joejimbo, Josh5555, KeyStroke, Kingpin13, Kuru, Lallrobe@rahis.com, Ldalle, M7, Madman2001, Mark Renier, Materialscientist, McSly, Michael Hardy, Michel Jansen, MilerWhite, Morpheios Melas, Mpramsey, Mufka, Mwdsmith, Mwtoews, NishithSingh, NortyNort, Nuno Tavares, Opsix, Phileas, Pne, Puckly, Rahulsrivastav2607, Rhillard, Ringbang, Robert K S, Rockbiter, Sadeq, Sarfa, Sarnholm, SqlPac, TFinn734, TechPurism, Turk oğlan, Tzeh, Volphy, Wyatt915, 142 anonymous edits

The Kimball Lifecycle *Source*: http://en.wikipedia.org/w/index.php?oldid=565112471 *Contributors*: Bearcat, Cburge, CommonsDelinker, DoctorKubla, GoingBatty, Malcolmxl5, Thomasaa, WQUlrich, 2 anonymous edits

Time variance *Source*: http://en.wikipedia.org/w/index.php?oldid=539262394 *Contributors*: Elmindreda, Kevinsam, Kvng, Landroni, Markiewp, Rich Farmbrough, Stephenpace, Wyadbb, 6 anonymous edits

Federated database system *Source*: http://en.wikipedia.org/w/index.php?oldid=598714956 *Contributors*: Beland, Bovineone, Cantonnier, Chris the speller, Comps, DBigXray, Dfoxvog, Frap, Gilo1969, Hegedusa, Hu12, Joy, Khazar2, KingsleyIdehen, Kku, MacTed, Mark Renier, Martarius, Meena610, Mgh12, Myasuda, P. Dantressangle, Pmehra5730, Pumba lt, R'n'B, Repentsinner, Rettetast, Ringbang, Rjwilmsi, Sfan00 IMG, Shyamal, Tabletop, Tankiitr, The Thing That Should Not Be, TheParanoidOne, Threazy, Uthbrian, VanishedUserABC, Venullian, Woodshed, Yann Gripay, 64 anonymous edits

Single Source of Truth *Source*: http://en.wikipedia.org/w/index.php?oldid=552939069 *Contributors*: 4johnny, Arjayay, Balis, Chris the speller, Christian75, Cybercobra, FoxDiamond, Jason Quinn, Johncrab, Lox, MarkBurnard, Markbassett, Minnaert, Pete142, RaviKrishnappa, Rjackson4, Subversive.sound, 4 anonymous edits

XLeratorDB *Source*: http://en.wikipedia.org/w/index.php?oldid=574283626 *Contributors*: Belovedfreak, Cander0000, Difu Wu, Drbreznjev, Dthomsen8, J04n, Jpbowen, Magioladitis, Mblumber, R'n'B, Sqltool, 1 anonymous edits

Dimension (data warehouse) *Source*: http://en.wikipedia.org/w/index.php?oldid=599123675 *Contributors*: 1ForTheMoney, Aaronsteers, Andy Dingley, Benanhalt, Bhughes67, Bobprobst, Causa sui, ChrisGualtieri, Coolelle, Crysb, Dannygutters, DePiep, Denaxas, Dmccreary, Elsendero, Ensslen, Frap, GoingBatty, GregorB, Immunize, JMSwtlk, Jakhel, Julianhyde, Lambiam, MN1411, MZMcBride, Mark Renier, Michael Hardy, Mike Rosoft, Mindmatrix, Radagast83, RayGates, Rich Farmbrough, Tbsdy lives, TrulyBlue, Virtuald, Wganesh, Wknight94, Youngamerican, چالاک, 69 anonymous edits

Fact (data warehouse) *Source*: http://en.wikipedia.org/w/index.php?oldid=540931097 *Contributors*: 069952497a, 99magred, A.amitkumar, ABMH, ALargeElk, Aaron Brenneman, Ace of Spades, ActivExpression, Aetherfukz, AgentCDE, Aitias, Allstarecho, Alqayoom, Altenmann, Amit8082, Amorim Parga, Andreid76, Andrewman327, Andy Dingley, Aneah, Angusmca, Ankit Maity, AnmaFinotera, Ansumang, Antonrojo, Apolitano, Arcann, Arch dude, Ari1974, Arielco, Arpabr, Asif earning, Asparagus, Atlantas, Atomician, Atw1996, Augbog, AutumnSnow, Avjoska, BanyanTree, Bdebbarma, Beardo, Berny, Bertport, Betacommand, Bgwhite, BigDunc, Binksternet, Blanchardb, Blue520, Bmotoc, Bobrayner, Bogey97, Brc4, BryantAvey, Btaub, Butros, C960657, CGerrard, Caltas, Camw, Carmichael, Catchsandy, Chaubals, Chowbok, Choyr, Chromatikoma, Chuq, Cobi, ColBatGuano, Comatmebro, Cometstyles, Compfreak7, Conversion script, Coolelle, Corruptcopper, Crysb, Crystalattice, Ctorchia87, Cybercobra, DARTH SIDIOUS 2, DBigXray, DNkYpNcH, DWBIGuru, Dan100, Dancter, Danielg922, Darth Zephyr, DatabaseDiva, Datawarehouseiq, Dawriter, DePiep, Decoy, Denisarona, DhirajGupta, Diannaa, Diego Moya, Dina, Disaas, Dividing, Dmccreary, Doc9871, Donaldpugh, Download, Dpm64, Drbreznjev, Dthvt, Dlugosz, Edward, Edwardmking, Elonka, Enimer, Er.piyushkp, Ericd, ErinRea, Ethansdad, Evert r, Falcon Kirtaran, Fbooth, Finlay McWalter, Foxj, Foxyliah, Fraggle81, Fred Bradstadt, Frze, Fudoreaper, Funandtrvl, Fyyer, Fæ, G716, Gabrielinux, Gachet, Gaius Cornelius, Gene Fellner, Ghewgill, Gianna ch, Gilliam, Giulio.orru, GreenInker, Greenlightwilly, GregorB, Grjuedes, Gscshoyru, Gskrzypczynski, Gurch, Gzkn, HJ Mitchell, Hadal, Hanifbbz, Hectorpa, Henridv, Hhultgren, Ianemitchell, Imdeadlymon, Intgr, Itmanning, IvanLanin, Ixfd64, J3st, JCLately, JForget, Jackfork, Jagannath6969, Jan.hasller, Jarble, Jaxl, Jay, Jburtenshaw, Jdrlundgren, Je ne détiens pas la vérité universelle, Jelena1, Jenks1987, Jgroove, Jguthaaz, JoeSmack, John Vandenberg, Joinarnold, Jomsviking, Jonwynne, JoyMundy, Joyous!, Jusdafax, Jwissick, Kadambarid, Kellylautt, Kingpin13, Kingpomba, Kiwi137, Kjkolb, Kku, Klausness, Kuru, Kwesterh, L Kensington, Lahiru k, Laurentseries, LcawteHuggle, Leandrod, Les boys, Lightmouse, Lingliu07, Little green rosetta, Llla32, Loonquawl, Lordzilla, Loren.wilton, Lostraven, Luyseyal, MHPSM, MHargraves, MK8, Madman2001, Madmonky, Mahuna2, MainFrame, Makro, Malcolmxl5, Mandarax, Manning Bartlett, Marcika, Mark Renier, Mark94502, Martarius, MartinsB, Mathonius, Maurreen, Mav, McSly, Mdd, MeekMark, Megras, Meyer, Mhooreman, Michael Devore, Michael Hardy, Michal Jurosz, Mike Rosoft, Mikeblas, Mindmatrix, Minervasolutions, Minimac, Minna Sora no Shita, Modify, Mpolichany, MrOllie, Mustafaisonline, Mwaci2, MyWorld, Mydogategodshat, Neilc, NickPenguin, NishithSingh, Nitin ravin, Nixdorf, Nn123645, Noah Salzman, Nraden, Nrahimian, Oaf2, Object01, Ojigiri, OliverKlozoff, Ondertitel, Ordoon, Oxymoron83, PC78, Philip Trueman, Piano non troppo, Pietrow, Pisceswzh, Playmobilonhishorse, Pnm, Propheticone, Qst, Quinsareth, R'n'B, RHaworth, RJaguar3, RScheiber, Ramesh l, Ranjran, RayGates, Raysonho, Rednblu, Reedy, Rhoegg, Ringbang, Rjanag, Rjwilmsi, Rknasc, Robdellis, Robert K S, RocklegendPoker, Roenbaeck, RomanEmbacher, Rrabins, Rschmertz, Rushbugled13, Ryan Rasmussen, S, SJP, ST47, Sabed Aashour, Sae1962, Salvio giuliano, Samroar, Sandeep4tech, Sarnholm, Savan3147754, Schul253, Scurra, Sean D Martin, SebastianHelm, Sesamevoila, Sfingram, Shaded0, Shadowjams, Shortbusmedia, Shubinator, Shwetank99, Sidhekin, SimpleTheory, Siroxo, Sixstone, Sleske, Sloan1919, Slowbro, SlubGlub, Snowolf, SoCalSuperEagle, SoSaysChappy, Soumyasch, Soundmind43, Spoxox, SqlPac, Srperez, Stefmol, Stephenb, Stephenpace, Stevag, Steve G, Steverapaport, Strongsauce, Studerby, Stvltvs, Subash.chandran007, Swatjester, Swintrob, TFinn734, THEN WHO WAS PHONE?, Tamarandom, Tapir Terrific, TechPurism, Techteachermayank, Telugucherla, Tgeller, The Thing That Should Not Be, Tide rolls, Tom Morris, Troels Arvin, Trusilver, Truther2012, Tvarnoe, Two Companions, Uršul, Utcursch, Vaibhaovaibhav, Vald, Van helsing, Vertium, Vincent jonsson, Vinod Alangaram, Vipinhari, Viridae, Volphy, WH98, Walk&check, Wanderingstew, Wavelength, Wedge3rd, WetherMan, WikHead, Wiki Raja, Wikiolap, Wikipelli, Wile E. Heresiarch, Will Beback, Willscraper, Wknight94, WojPob, Writerguy71, Wtmitchell, Xlaran, Yasth, ZimZalaBim, Zundark, 1127 anonymous edits

Measure (data warehouse) *Source*: http://en.wikipedia.org/w/index.php?oldid=530965669 *Contributors*: Allens, Andy Dingley, DePiep, Dmccreary, GregorB, JMSwtlk, Michael Hardy, RayGates, Rich Farmbrough, Widr, 6 anonymous edits

Fact table *Source*: http://en.wikipedia.org/w/index.php?oldid=597045820 *Contributors*: Andy Dingley, Aziz talha, Bleveret, Bobprobst, Chris the speller, ChrisGualtieri, CzarB, DePiep, Dmccreary, Elsendero, Frap, GregorB, Hooperbloob, Hug0720, Jtankers, KeenDK, L Kensington, Luís Felipe Braga, Manowar mi77, Mav, Mcclarke, Michael Hardy, Nakul Dhoot, Niteowlneils, Oxymoron83, Peter Hitchmough, Peter.C, SFK2, Saurabhgupta3, Supertouch, TFOWR, TimBentley, Tlaresch, Triona, Umer992, 62 anonymous edits

Degenerate dimension *Source*: http://en.wikipedia.org/w/index.php?oldid=560897315 *Contributors*: Andy Dingley, Bob1960evens, Bobprobst, Coolelle, Danilo.Piazzalunga, DePiep, GregorB, JoyMundy, Mboverload, Rich Farmbrough, Tartan, Wikiolap, Yintan, 11 anonymous edits

Slowly changing dimension *Source*: http://en.wikipedia.org/w/index.php?oldid=591983179 *Contributors*: Alucard (Dr.), Andrewcardno, Atif.t2, Bender235, Cgwaters, Coolelle, DEinspanjer, Dcamp314, DePiep, Djoni Darmawikarta, Dougaljim, Duke Ganote, Epicgenius, Ggaggo, Gouce, GregorB, Iridescent, Jablonov, Jameboy, Jdlambert, Johnsoftek, Legoktm, Lugalde, Mark Renier, PlAnTen, Remy B, Rich Farmbrough, Rob cowie, Sahyogi, Shazin s, Sjmanikt, Skiwi, Stephenpace, Tonyfaull, 198 anonymous edits

Star schema *Source*: http://en.wikipedia.org/w/index.php?oldid=599109132 *Contributors*: .:Ajvol:., 1984, Abergquist, Amattas, Andrew not the saint, AndrewMWebster, Andrewpmk, Anubhab91, Appi, Armchairlinguist, Asqueella, Beland, Bertport, Birger, Budyhead, CWY2190, ChrisGualtieri, Chrissi, Cralize, Crysb, DePiep, Dfrankow, Diego Moya, Electrum, Elsendero, Falcor84, Gahooa, Gilderien, Ginsuloft, Gothick, GregorB, JHunterJ, Jay, Jketola, Jobin RV, Kablammo, KeyStroke, Littldo, Mark Renier, Materialscientist, Michael miceli, Millertimebjm, Mrityu411989, Mselway, Mwarren us, NishithSingh, Nrahimian, Od Mishehu, Ozancan, Panfakes, Pne, Random832, RayGates, Raymondwinn, Remy B, Sharad.sangle, Sitoiganap, SqlPac, Surendra.konathala, Thesuperav, Txnate, Vald, Walk&check, 158 anonymous edits

Dimension table *Source*: http://en.wikipedia.org/w/index.php?oldid=596310237 *Contributors*: Aillile, Andy Dingley, Davideosborne, DePiep, Dmccreary, Dwandelt, EagleFan, Elsendero, Elwikipedista, Ensslen, GregorB, Mark Renier, Michael Hardy, Mindmatrix, Minimac, Mrjavahack, Peter Hitchmough, Pmasiar, Radagast83, Rich Farmbrough, Virtuald, WurmWoode, Xenon325, Yanco, 33 anonymous edits

Dimensional Fact Model *Source*: http://en.wikipedia.org/w/index.php?oldid=594127375 *Contributors*: Cander0000, Lesser Cartographies, Machetebetty, Magioladitis, Matteo.golfarelli, Mcclarke, Wilhelmina Will, 4 anonymous edits

Snowflake schema *Source*: http://en.wikipedia.org/w/index.php?oldid=595178146 *Contributors*: 1984, Allandean, Alxtoth, Aneah, Ashe the Cyborg, CesarB, Coolelle, Cortu01, Crysb, DOSGuy, Damian Yerrick, Danim, Derek Ross, Diego Moya, Edward, Elwood j blues, Falkbjoern, Garymach64, Getramkumar, Ggentleway, GregorB, Helix84, Idib, Jpceayene, Klausness, Manius, Mark Renier, Michael Hardy, Michaelvkim, Mindmatrix, Nakul.vachhrajani, Nikkimaria, Nshuks7, Olivier Debre, Pinethicket, Pne, Prateekbhatia28, Priyank bolia, Rlian, Rykepiji, S.K., Sfermigier, SqlPac, Sweetbluemagic, Tabletop, Teles, That Guy, From That Show!, TomStar81, Trioculite, Walk&check, Wille Raab, Yorrose, Zsero, 93 anonymous edits

Denormalization *Source*: http://en.wikipedia.org/w/index.php?oldid=589830907 *Contributors*: Alan Liefting, Andy Dingley, Bearcat, ChrisGualtieri, Conversion script, Damian Yerrick, Danim, David Gerard, Dreftymac, Furrykef, Fyrael, GermanX, Ghewgill, Gpierre, GregorB, Jay-Jay, Jdlambert, John Coupe, Joncnunn, Jtgerman, Jwolfe, Ketiltrout, Klausness, LOL, Leandrod, MIT Trekkie, Malcolma, Mann jess, Markonen, Matthew0028, Mgt88drcr, Mindmatrix, Netfall, Ninly, Nyttend, Ocrow, PamD, PatrickFisher, Paulo.freire, Pnm, Rich Farmbrough, Robert

McClenon, Shunpiker, Skittleys, TechPurism, Tobias Hoevekamp, Topbanana, Troels Arvin, Ver, Verycharpie, VinceBowdren, YetAnotherPseudonym, 38 anonymous edits

Single version of the truth *Source*: http://en.wikipedia.org/w/index.php?oldid=588422595 *Contributors*: 28bytes, DLSieving, FayssalF, Johncrab, Lox, Minnaert, Pumba lt, Quarl, RainbowCrane, RichardVeryard, Tcotco, Tjamesjones, Uncle G, 5 anonymous edits

Transaction data *Source*: http://en.wikipedia.org/w/index.php?oldid=495282407 *Contributors*: Amillar, CaroleHenson, Edward, Evert r, Hairy poker monster, JCLately, Kubanczyk, Radagast83, Thumperward, 5 anonymous edits

Enterprise bus matrix *Source*: http://en.wikipedia.org/w/index.php?oldid=589706875 *Contributors*: Alqayoom, Bearcat, Dthomsen8, Gene-va, Glane23, Ijha itu, LilHelpa, Santryl, Slasher-fun, Trivialist, 4 anonymous edits

Statistical database *Source*: http://en.wikipedia.org/w/index.php?oldid=556850324 *Contributors*: Adam Bishop, Danim, Farleysm, Gene Nygaard, Gracefool, GregorB, John of Reading, Kaal, Khazar2, KingOfAllTrolls, Lakonislate, Materialscientist, Melcombe, Mindspillage, N d gangadhar, Reinoutr, 14 anonymous edits

Online transaction processing *Source*: http://en.wikipedia.org/w/index.php?oldid=598404421 *Contributors*: 28nebraska, 2openminds, Aapo Laitinen, Abolen, Aeonx, AlonB, Apavlo, Arunsingh16, Ayushi13nov, Bovineone, Bruce1ee, Bulwersator, ChadMeista, Cometstyles, Craig Stuntz, Curtis, DGG, Discospinster, Donner60, Edgar181, Ekey-kbrophy, Electric Wombat, Evert r, Fsiler, Gaur1982, Gogo Dodo, GregorB, Gscshoyru, Insouciance, IronGargoyle, JCLately, Jfredrickson, Jklin, Khandarenb.it, Kubanczyk, Lethe, Macgregorf, Mark Renier, Mcvearry, Modify, Mrt3366, Mwaci11, NapoliRoma, Nickleus, Nom du Clavier, Omar10101971, Peter Flass, Piet Delport, Public Menace, RTNM, Renox, Rmp, SQGibbon, Shashark, Snezzy, Soliloquial, SqlPac, Stephenb, The Thing That Should Not Be, Thestraycat57, TimSwast, Vao, Wiki13, Wimmeljan, Yesbaby, Zhenqinli, Zundark, 130 anonymous edits

Operational system *Source*: http://en.wikipedia.org/w/index.php?oldid=578731355 *Contributors*: DePiep, GregorB, Hanifbbz, Mange01, Mark Renier, Minnaert, RHaworth, Shreevatsa, Villarinho, Writerguy71, 12 anonymous edits

Operational data store *Source*: http://en.wikipedia.org/w/index.php?oldid=595702056 *Contributors*: A2Kafir, AlexInWikiland, ChrisGualtieri, Craig Pemberton, DePiep, Deedub1983, Dwandelt, Fred1026, Garretereis, Greenrd, Guyh92, Joelm, Joshuapaquin, Kku, Kuru, Littldo, Madman2001, Mark Renier, Mblumber, Mindmatrix, Naveen607, Rmenning, Robertbowerman, Satellizer, Sportsguy17, Stephenpace, Stevage, TechPurism, Tee Owe, TheParanoidOne, 34 anonymous edits

Operational database *Source*: http://en.wikipedia.org/w/index.php?oldid=544723649 *Contributors*: Beland, Danim, Drbreznjev, GregorB, Hanifbbz, Ldvaeth, Minnaert, NawlinWiki, RHaworth, Rvgd, 4 anonymous edits

Online analytical processing *Source*: http://en.wikipedia.org/w/index.php?oldid=598361211 *Contributors*: .:Ajvol:., 7, 90 Auto, Afraietta, AlanUS, AlexAnglin, Alfio, Alvarezdebrot, Anubhab91, Anwar saadat, Arcann, Augbog, Awbush, Beardo, Beland, Bitterpeanut, Boson, Bruce1ee, Bunnyhop11, COnFIIcT.sYs, CambridgeBayWeather, Campingcar, Cannolis, Cffrost, ChanciOly, Charleca, Chaubals, Chris lavigne, CommodiCast, Crysb, Csp987654321, Danim, DePiep, Deflective, Degroffdo, Djoni, Dkwebsub, Dll99, Dmsar, Dmuzza, DocendoDiscimus, Dougher, Drewgut, Dturner46, EbenVisher, Elsendero, Elvis, Elwikipedista, Epbr123, Ezrakilty, FatalError, Flankk, Foot, Founder DIPM Institute, Fæ, Gary a mason, Germanseneca, Goethean, Grand ua, Greenrd, Gruay, Gscshoyru, Gspofford, Hanacy, Hobsonlane, Howcheng, I am One of Many, Imhogan, JEB90, Jagyanseni, Jahub, Jan.hasller, Jarda-wien, Jay, Jay-Jay, Jbecher, Jcarroll, JesseHogan, Jgiam, Jherm, Jmcc150, Jmd wiki99, John Vandenberg, John of Reading, Jon Awbrey, JonHarder, Julianhyde, Kerenb, Kgrr, Kher122, Kku, Kronostos, Kuru, Kwamikagami, L Kensington, Larkspurs, Leandrod, Lemmie, Leonbravo, Lesser Cartographies, Lingliu07, Livingthingdan, Lovedemon84, MaGa, Magnabonzo, Mark Renier, Mark T, Markham, Masterpra2002, Maurreen, Mboverload, Mbowen, Mcvearry, Metaeducation, Miaow Miaow, Michael Hardy, Michaell, Mikeblas, Modify, MrDolomite, MrJones, MrOllie, Mydogategodshat, Ndenison, Ningauble, Numbsyd, Oaf2, Object01, Ohnoitsjamie, Op47, Oxymoron83, Pasquale, Paul Magnussen, Pcb21, Pearle, Peter.vanroose, Pgan002, Plasticup, Playmobilonhishorse, Psb777, Qxz, Ramki, Ratarsed, Retired username, Richmaddox, Ringbang, Rjwilmsi, S.K., S1199, Sally Ku, Sam Korn, Sansari13, Sarnholm, Saulat78, Selah28, Sj, Slposey, SqlPac, Sspecter, Stefan, Tbsdy lives, Tcloonan, The Thing That Should Not Be, Theo10011, Tikiwont, Tobycat, Tonyproctor, Tsjustme, TwoOneTwo, Vargabor, Veinor, Wikiolap, Winterst, Woohookitty, WorldsEndGirl, Writerguy71, XKL, Yabdulkarim, Zanaq, Zhenqinli, ZimZalaBim, 426 anonymous edits

OLAP cube *Source*: http://en.wikipedia.org/w/index.php?oldid=588764097 *Contributors*: .:Ajvol:., Andy Dingley, AndyReid, Animum, Anubhab91, Aranel, Asqueella, Astazi, Bertport, Brycen, Carmichael, Charles Matthews, Cma, Crasshopper, Crysb, Cvanhasselt, David Tristram, DePiep, Dll99, EagleFan, Emurphy42, EngineerScotty, Fenice, Flagboy, FluffyPanda, Goethean, Gyrae, Hazmat2, Hiddenfromview, Hooperbloob, Igor Yalovecky, Infopedian, J.delanoy, Jmabel, Jmcc150, John Vandenberg, Jrolston, Julianhyde, Kgrr, Krystof1000, Labreuer, LeoHeska, Lfstevens, Lockley, Lysy, Magioladitis, Materialscientist, MauriceKA, Michael Hardy, Mihirgokani007, Niceguyedc, Ningauble, NinjaCross, OS2Warp, Ohnoitsjamie, Osiris.toth138, PasabaPorAqui, Plasticup, Qseep, Quaeler, Robofish, Rohan Jayasekera, S.K., SEWilco, Sepreece, Soumyasch, SouthLake, SqlPac, Sspecter, T-borg, Technopat, Thumperward, Urantian, Vasile, Wikiolap, Wizzard, Yorrose, Yx7557, 144 anonymous edits

Aggregate (data warehouse) *Source*: http://en.wikipedia.org/w/index.php?oldid=594849346 *Contributors*: Cander0000, DePiep, Gaius Cornelius, GoingBatty, Kintetsubuffalo, Kku, Stefmol, 1 anonymous edits

MOLAP *Source*: http://en.wikipedia.org/w/index.php?oldid=564152586 *Contributors*: Arcann, Billrobo, Dbrukman, DePiep, Dgies, Dusty relic, Elwikipedista, Frvernchanezzz, Getramkumar, Guillom, Hadal, Hicare, Jan.hasller, Jmcc150, Jowa fan, Kaikyro, Kronostos, Leandrod, Mark Renier, MaryEFreeman, MilFlyboy, Mr P Legend, MrOllie, OS2Warp, Porqin, Slovakjoe, SqlPac, The Anome, Toxien, Wikiolap, Wim Leers, Yeshwant altair, 70 anonymous edits

ROLAP *Source*: http://en.wikipedia.org/w/index.php?oldid=579472814 *Contributors*: Andy Dingley, Arcann, Crysb, DePiep, Dgies, Dusty relic, Flagboy, Getramkumar, Iviney, JLaTondre, Kronostos, Kuteni, Lemnaminor, Mark Renier, OS2Warp, Omicronpersei8, Ratarsed, Rhododendrites, SqlPac, Srleffler, Tcloonan, The Anome, Wendecover, Wikiolap, Woohookitty, 77 anonymous edits

HOLAP *Source*: http://en.wikipedia.org/w/index.php?oldid=542563098 *Contributors*: Aecis, Arcann, Ashu14gupta, DanielLemire, DePiep, Jim1138, Mr P Legend, OS2Warp, Ratarsed, Ropez, SqlPac, Wikiolap, 26 anonymous edits

Thomsen Diagrams *Source*: http://en.wikipedia.org/w/index.php?oldid=517251520 *Contributors*: Alvestrand, DoctorKubla, Malcolma, Maximross, Ningauble, 3 anonymous edits

Spreadmart *Source*: http://en.wikipedia.org/w/index.php?oldid=595089947 *Contributors*: 1WizardGuy, Arctic Kangaroo, Cander0000, Crysb, Fortdj33, Gakl, GregorB, Howudooin, JackinTrade, Mr pand, Mwtoews, Nsaa, PamD, RonaldKunenborg, Taliped, YK Times, 23 anonymous edits

MultiDimensional eXpressions *Source*: http://en.wikipedia.org/w/index.php?oldid=595598213 *Contributors*: A5b, AJackl, Alvarezdebrot, AlyaBasa, Andrew Werdna, Annaaren74, Aredridel, ArglebargleIV, Batareikin, Boing! said Zebedee, Brandongrey, CMSalter, Cedar101, ChrisCork, Crysb, Crystallina, Dagoli, DavidBiesack, Davidjessect, DePiep, DocendoDiscimus, EagleOne, Fryderykfryn, GregorB, GrifM, Gspofford, J04n, Jan.hasller, Jdlambert, John Vandenberg, Kevin Ryde, Kgrr, Leotohill, MTranchant, Map79, Metallion, Michael Hardy, Michael miceli, MrOllie, Offby1, Oudia, Paul Foxworthy, Pewterschmidt Industries, Sarnholm, SchreiberBike, Sgartner, SteinbDJ, Trevie, Wikiolap, Zwiadowca21, 67 anonymous edits

Comparison of OLAP Servers *Source*: http://en.wikipedia.org/w/index.php?oldid=592447135 *Contributors*: 1887Slaughty, Alvarezdebrot, Cander0000, Ceswiki, Chowbok, Crysb, Danim, DePiep, Dewritech, Dougher, Eeekster, Ettebaba, JLaTondre, Jevansen, Julianhyde, Kuru, MI77IVW, Mpolizzi, Mr P Legend, Mtkorb, Muhandes, Nattyspats, Omegas, Pierre1209, Pinnecco, RedWolf, Ryankoski, Steve Shoal, Usbrandon, Wikiolap, YUL89YYZ, 133 anonymous edits

Applix *Source*: http://en.wikipedia.org/w/index.php?oldid=581297271 *Contributors*: BSJWright, Bosticko, Cander0000, Choster, Cirala, Crysb, Davagh, DelaneyTurner, Earthlyreason, Firsfron, Griffinofwales, Jmcc150, Johnpacklambert, Jpusztai, Kaikyro, Lijnema, NapoliRoma, Ningauble, Psb777, Rojomoke, Shal2008, SimonP, Simplysavvy, The Thing That Should Not Be, Wikiolap, Xezbeth, 21 anonymous edits

BusinessObjects OLAP Intelligence *Source*: http://en.wikipedia.org/w/index.php?oldid=349614063 *Contributors*: Ratarsed

Crystal Analysis *Source*: http://en.wikipedia.org/w/index.php?oldid=504508059 *Contributors*: Centrx, Firsfron, Hoekit, John of Reading, MacGyverMagic, Ratarsed, Reedy, SarekOfVulcan, Silverfish70, SueHay, 8 anonymous edits

CubePort *Source*: http://en.wikipedia.org/w/index.php?oldid=578603295 *Contributors*: Cander0000, ChrisGualtieri, Fram, Jac16888, Jfrohm, KylieTastic, Lemnaminor, Mandarax, Rettetast, Wavelength, 17 anonymous edits

Essbase *Source*: http://en.wikipedia.org/w/index.php?oldid=592786213 *Contributors*: Bdiscoe, Ben White, Blaxthos, Brystmar, Chowbok, Chris the speller, Chrislk02, Codename Lisa, Crysb, DDimesby, Denisarona, Dmccreary, Dphillips11, Edward Montgomery Harrington, Enochlau, Gbacskai, Gwen-chan, Ibizarre, Icurtain, Jainnys, JesseW, Jevansen, John Vandenberg, Ketiltrout, Kuru, Lyonsbane, Manoharrana, Maury Markowitz, Mbowen, Modify, NapoliRoma, Neilc, PM Poon, Palox, Paughsw, Pjbeswick, Psiphiorg, Pureapps, R'n'B, Ratarsed, Reedy, Rstackowiak, ST47, Sdc2000, Solidpoint, Surajitdas1979, TFinn734, Tabletop, The Thing That Should Not Be, Timtow, Tsmee, UncleDouggie, Wikiolap, Winterst, Wouterborn, Yen hung, ZWalters, Тихонравов, 125 anonymous edits

Microsoft Analysis Services *Source*: http://en.wikipedia.org/w/index.php?oldid=592010530 *Contributors*: Abutorsam007, Andy Dingley, Angelo.romano, Annaaren74, Colonies Chris, Crysb, Cutley, Cwolfsheep, DePiep, Drewgut, EbenVisher, Edwardmel, ErinRea, Exprexxo, FleetCommand, Grand ua, Hmains, JLaTondre, Jasper Deng, Ketiltrout, Mihart, Mikeblas, Nricardo, Paul Foxworthy, Raimex, Realdanielbyrne, Ropez, SFK2, Sanspeur, Schmiteye, Soumyasch, Tntdj, Warren, Wavelength, Wikiolap, ZMerLynn, Île flottante, 88 anonymous edits

Mondrian OLAP server *Source*: http://en.wikipedia.org/w/index.php?oldid=591653481 *Contributors*: Agentq314, Andre Engels, Anonymous Dissident, Avenue, Beetstra, Brad Dyer, Ciphers, Dinochopins, Dmccreary, Fariel, Feristhia, Fivelittlemonkeys, Fuxx, Gardar Rurak, Gruffi, John Vandenberg, Julianhyde, Kuru, Luca.paoli, Lucboudreau, Magioladitis, Mysticfall, Osiris.toth138, Qseep, Rich257, Sergio Bertele, TheParanoidOne, Ungzd, Wikiolap, Yama, 38 anonymous edits

OLE DB for OLAP *Source*: http://en.wikipedia.org/w/index.php?oldid=548384416 *Contributors*: Drewgut, Hooperbloob, Inetad, Mikeblas, WOSlinker, Wikiolap, 3 anonymous edits

Oracle OLAP *Source*: http://en.wikipedia.org/w/index.php?oldid=567746981 *Contributors*: Chowbok, Crysb, Editrevert, Ibizarre, Pedant17, Rstackowiak, Scooty, Spartan Z84, Tiddly Tom, Wikiolap, Winterst, 4 anonymous edits

Palo (OLAP database) *Source*: http://en.wikipedia.org/w/index.php?oldid=597688405 *Contributors*: Billrobo, CommonsDelinker, DF Jedox AG, Gherson2, Harryboyles, JLaTondre, Jerryobject, Klemen Kocjancic, Kuru, Leifmergener12, Niceguyedc, Orangemike, Rca institute, Riedel, SamJohnston, Sun Creator, Toxien, Ungzd, Utcursch, Wikiolap, Zvar, 24 anonymous edits

Panorama Software *Source*: http://en.wikipedia.org/w/index.php?oldid=567993677 *Contributors*: AbigailAbernathy, Althalitus, Dhartung, Dl2000, Ihcoyc, Jauerback, John Spikowski, Mwtoews, Oudia, Stemonitis, Wikiolap, Y.Kondrykava, 27 anonymous edits

ProClarity *Source*: http://en.wikipedia.org/w/index.php?oldid=578540252 *Contributors*: DGG, Doulos Christos, Eric-Wester, Gary, Ihcoyc, KickahaOta, Kusma, Malcolma, Nricardo, Red Director, Severo, Takamaxa, Universalcosmos, Waded, Wikiolap, 9 anonymous edits

SAP BI Accelerator *Source*: http://en.wikipedia.org/w/index.php?oldid=591144137 *Contributors*: Dialectric, Greenrd, Indieaner, Jechiang, Juliand, Luxxor, Saldinatale, Tinucherian, Uktc896, 6 anonymous edits

SAP NetWeaver Business Intelligence *Source*: http://en.wikipedia.org/w/index.php?oldid=577303302 *Contributors*: Aaron Brenneman, Abhi murali, AdjustShift, Anand2027, Bhargava786, Bigdaddymaj, Bjmullan, Bosa71, Brad.armbrust, Bunnyhop11, ChrisGualtieri, Crysb, Curtcat, Cyde, Deshkapoor, DocendoDiscimus, ESkog, Edubbs, Edward, Gurch, Hometutorials, Hooperbloob, Icairns, Jamestochter, JasonLax, Jbertsch, Jcasarini, Jean Lugan, Jkurgan, Jmchugh, Jncraton, Joel Alcalay, John Vandenberg, Khalid hassani, Killing sparrows, Kjw1969, Kmorozov, Knguyeniii, Kuru, Larrymcp, Lectonar, Lemmie, Merlim taliesin, Mohan sap, Nikhil34, Nmajdan, Paul99, Pravisurabhi, Ratarsed, Rednblu, Rk195555, RossPatterson, S.K., SFK2, SShiralkar27, Sanjaychatwal, Sap-consultant, Sapguru2009, Sapuser1, Sudiptanet, TastyPoutine, Thesuperav, Tommy2010, Treaclecustard, Tristessa de St Ange, Unimedisys, W Nowicki, Wenfengli, Winchelsea, XKL, Yintan, 155 anonymous edits

NEVOD DMB *Source*: http://en.wikipedia.org/w/index.php?oldid=588640838 *Contributors*: CommonsDelinker, Glenn, Jesse V., JimVC3, R'n'B, RJFJR, SimonD, Tassedethe, Wikirosi, 4 anonymous edits

Extract, transform, load *Source*: http://en.wikipedia.org/w/index.php?oldid=594322142 *Contributors*: 16@r, APerson, Ahmedshuhel, Alai, AltiMario, Andy Dingley, Anir1uph, Annaaren74, Arcann, Aremgi, Avnjay, BD2412, Batfly123, Bcrawford, Beardo, Berny68, Besieged, Blanchardb, BoBaH32, Bonadea, Bovineone, Broogg, Bruce1ee, Buddyhutchins, CKlunck, Capstone131, Cenarium, Cgfdmc, ChPietsch, Chris the speller, Cory Donnelly, CosmoDad, Crasshopper, Cst17, Cyber Dog, DamsonDragon, Dawnseeker2000, Dbush, DePiep, Debgup, Dewwalker, DhirajGupta, Diego Moya, Digisus, Dmccreary, Download, Dpavlis, DragonHawk, Dreadstar, Ebersphi, Ecsnp, Edward, Egandrews, Eglobe55, Ehtisham.rasheed, Elf, Emarket, FatalError, Fbdev1988, Ffffloyd, Founder DIPM Institute, Freek Verkerk, FreplySpang, Frtande, Ggoli, Gharvett, Ghhutch, GimliDotNet, Graeme Bartlett, Grandmasterkush, GregorB, Gscshoyru, Gvimalku, Hoplon, Hu12, Iagustin, Ikan dev, Ikansoftware, Ikanweb, Incnis Mrsi, Inter, J mareeswaran, Jamelan, Jan.ahlers, Jay, Jaymishra79, Jesusluisfernando, John Yesberg, Johnbrownsbody, Joinarnold, Jomis, JonHarder, Jtree09, Kadishmal, Karthi acb, Kelbaker, Kennethmac2000, KeyStroke, Kgaughan, Kgobble101, Khalid hassani, Kinghitz, Kjtobo, Kku, Klausness, Kragen, KrakatoaKatie, Kubanczyk, Kuru, Kuteni, Leirith, Levselector, LilHelpa, Limited Atonement, LokiClock, Lotje, Luk, Madman2001, Mandarax, Marek69, Mark Arsten, Mark Renier, Mattpav, MaxSherbinin, Mereman, Michael Hardy, Mikeblas, MonMan, Mr link, MrOllie, Mtfr, Muhandes, NYC sheehy, Naquada, Nbarth, Ndufva, NickW557, Nielst, Nitpicking polisher, Nsaa, Ocarbone, OlenaSherbinin, Oracleguru, Orange Suede Sofa, Pasquale, PeterCanthropus, Phatwest, Philippe, Playsafe, Pmerson, Pne, Pnm, Psb777, Q Chris, RFMack, RHaworth, Randomran, RedHillian, Reddies007, Rjwilmsi, Rmoore080, Robert Will, Rohit labhe, Ruebencampbell, S.K., Sam Korn, Sarnholm, Sathish jo, Sboden, Scootey, Sd-100, SebastianHelm, Secmail, Seth Ilys, Sgoel.engg, Shaw76, Shehzad.kazmi, Shijaz, Siegler, Sjakkalle, Smittio, Sstrader, Stephenpace, Stevage, Subtleguru, Suhasmallya, Sutanupaul, Syaskin, TFinn734, Talkietoaster-nc, Tanvi.p.goel, Tassedethe, That Guy, From That Show!, The Anome, Thegerf, Theo10011, Thumperward, Todfather, TomRobinson, Triadic2000, Uhai, Ups2000, Urgos, Veyklevar, VincentJS, Vmcburney, Voidxor, Waclawiczek, Warminghands, Wdyoung, Wikiolap, Winterst, Wjhonson, Wwfchina, Xmlguru, Yaronf, Your Lord and Master, Zhenqinli, ZimZalaBim, 662 anonymous edits

Staging (data) *Source*: http://en.wikipedia.org/w/index.php?oldid=594525790 *Contributors*: Bender235, ChrisGualtieri, Dark Silver Crow, Finlay McWalter, Hutcher, Kku, Malcolma, Momo.sander, Nsaa, SqlPac, Srperez, 21 anonymous edits

Vocabulary-based transformation *Source*: http://en.wikipedia.org/w/index.php?oldid=593060789 *Contributors*: Chris the speller, Dmccreary, Kku, Malcolma, Mark Renier, Michael Hardy, Niceguyedc, WVhybrid, Wkharrisjr, 6 anonymous edits

Surrogate key *Source*: http://en.wikipedia.org/w/index.php?oldid=598388027 *Contributors*: Barliner, Brick Thrower, Bryant1410, ChrisNoe, Chrisxue815, DVdm, Darinw, Demitsu, Djankowski, Dtuinhof, Egrabczewski, Favonian, Govorun, Groggy Dice, Hairy Dude, Hsauzier, Int19h, Jberkus, Jimgawn, Joeharris76, KeyStroke, Kgaughan, Kjkolb, LachlanA, Leandrod, Lucianosother, M4gnum0n, Mark Renier, Mcbridematt, Mcclarke, Mdchachi, Mdd, Mindmatrix, MyTigers, Neilc, Pearle, PhiLiP, Phil Boswell, Pinkadelica, Raggatt2000, Reddyfire, Reedy, Rich Farmbrough, Rjwilmsi, Robert K S, Shadowjams, Shenme, Simetrical, Sleske, Stewartadcock, Template namespace initialisation script, Tfitzg, Tim.spears, Timhowardriley, Toh, Tomas e, Troels Arvin, Vjosullivan, WikipedianMarlith, Xenan, 167 anonymous edits

Variable data publishing *Source*: http://en.wikipedia.org/w/index.php?oldid=506145406 *Contributors*: Alvin Seville, Bearcat, ChrisGualtieri, Dcowe, Falcon8765, Grafen, Jake Wartenberg, Parhamr, R. S. Shaw, 2 anonymous edits

Semantic warehousing *Source*: http://en.wikipedia.org/w/index.php?oldid=584829998 *Contributors*: 1994miller, Bipachan, Blanchardb, CaroleHenson, Cenarium, Drilnoth, EhJJ, EmanWilm, GoingBatty, KylieTastic, Malcolma, Skittleys, Tabletop, Wozbeauthrozboubleberry, 36 anonymous edits

Scriptella *Source*: http://en.wikipedia.org/w/index.php?oldid=590094998 *Contributors*: 7Volk, Cander0000, Cedar101, Centrx, Chowbok, ChrisGualtieri, DePiep, Dgies, EagleOne, Hangy, Jitix, Khalid hassani, Mean as custard, Mogism, PigFlu Oink, Tkpwns, William Avery, 40 anonymous edits

Bit rot *Source*: http://en.wikipedia.org/w/index.php?oldid=599198412 *Contributors*: 2pem, A. Parrot, ABCD, Adot, Albany NY, AndersJohnson, AtheWeatherman, Benwing, Bhny, Bobdobbs1723, Brambo, CCFreak2K, Chris the speller, Cmdrjameson, Cmlefevre, Daniel Lawrence, David Latapie, Debresser, Drake Wilson, Dsimic, Dysprosia, Enrique r25, Ethanpet113, Fadookie, Frecklefoot, Furrykef, Fuzzbox, Gazpacho, Glst2, Greg searle, GregCutler, Habj, Haslantis, Hobart, Huttarl, Hwttdz, JQF, Jason Quinn, Jhawkinson, KenBailey, Klichka, Kvn8907, Kwns, Locke baron, Mendax666, Mindmatrix, Misund, Morwen, Muslim lo Juheu, Mysid, Nick Number, Nnemo, Ocn169, Peyre, Pgan002, Pjvpjv, ProveIt, Quietust, Qwerty Binary, Rolofft, Rwwww, Sci.templar, SimonTrew, SpectrumDT, Staszek Lem, TenPoundHammer, Thedoubledeuce, Tickle me, Tsidnic Guy, VeryVerily, Wik, X-Bahamut, Ysangkok, Z57N, Zundark, רנדום, 95 anonymous edits

Cleansing and Conforming Data *Source*: http://en.wikipedia.org/w/index.php?oldid=538320417 *Contributors*: Aceldam, Bjacob, Dawnseeker2000, Fram, Malcolma, Muslim lo Juheu, Rich Farmbrough, SchreiberBike, Staszek Lem, Wilhelmina Will, 5 anonymous edits

Data auditing *Source*: http://en.wikipedia.org/w/index.php?oldid=544533350 *Contributors*: AvicAWB, Avicennasis, Dataguru, Dina, GoodDay, Muslim lo Juheu, 3 anonymous edits

Data cleansing *Source*: http://en.wikipedia.org/w/index.php?oldid=597084989 *Contributors*: Anax, Anilkumar 0587, Artem.batkovsky, Avihu, Bbb23, Ben Ben, Binter2, Bomazi, Calzakk, Careful Cowboy, China Crisis, Cnmlimited, Cyberwolf UK, DaCentaur, DaGizza, Databrad, Dataguru, Datatune, David Rolek, Dawnseeker2000, Dheeruyadav, DoorsAjar, Drackaer, DragonHawk, EffeX2, Endreakosven, Florent1024, Founder DIPM Institute, Gmelli, Gmlingus, Gogo Dodo, Haephrati, Hegde Pooja, Ipeirotis, JLaTondre, JaGa, JamesBWatson, Jay1279, Jkosik1, Kku, Kuru, Leannedavidson, Levineps, LittleWink, Logical Cowboy, Lordjohnny, Maximhammoudeh, Melcombe, MikeWazowski, Mikeblas, Mild Bill Hiccup, Mindmatrix, Mitchan, Monikamooorjani, Muslim lo Juheu, Nathkrol, Oliverbaasch, Pcullinane, Peter Flass, Poco a poco, Prakash Nadkarni, Qwfp, SKHyde, Satellizer, Sbdeland, Scottk, Snowded, Staszek Lem, StaticVision, The Anome, Tnxman307, Todd Vierling, Tutanchmaat, Virtualelvis, Vullik, WikiHannibal, Wykoong, Xezbeth, Zundark, Zzuuzz, 142 anonymous edits

Data corruption *Source*: http://en.wikipedia.org/w/index.php?oldid=597152511 *Contributors*: Abhisatya, AnonMoos, Arvindn, Auric, Badon, Bayerischermann, Bearcat, Bloodshedder, Bwrs, C5st4wr6ch, Charles Kozierok, Corn cheese, D1m3b4g, DavidCary, Dreamteamone, Dsimic, El Mayimbe, Indulis.b, Jackson Peebles, Jesster79, JonHarder, Joseph Solis in Australia, Lotje, Mild Bill Hiccup, Mmgareth, Muslim lo Juheu, Natski-asnd8, Nhergert, Nicolas1981, Northamerica1000, Peter L, Public Menace, Reaper Eternal, Ronz, Sarujo, Scriberius, Spike, Squid tamer, Tom94022, Trainspotter, Trappist the monk, Uršul, Wikilolo, Wilfredor, Winterst, X spager, Xezbeth, Z93007, 32 anonymous edits

Data integrity *Source*: http://en.wikipedia.org/w/index.php?oldid=589425089 *Contributors*: Adrory, Alan Liefting, Alansohn, Allens, Amjaabc, Andreas Kaufmann, Andrew nixon, Aniket Pradip Joshi, Arthena, Barkeep, Boli1107, Born2killx, Boson, Bruce1ee, Cerkit, Cntras, Cooldudemcguire, Cybercobra, DARTH SIDIOUS 2, DGtal, Debangadutta, Denisarona, Dreamyshade, Dsimic, Eastlaw, Etu, FF2010, Freemonition, Gene.arboit, Greensburger, Gzkn, Hede2000, HereToHelp, Informatwr, JCLately, Jackfork, John Vandenberg, Joyous!, Kalolguru, Katzell, Kingpin13, Kku, Kyng, LittleDan, Loadmaster, Longhair, M4gnum0n, Madhero88, Makemi, Mark Renier, Mcclurg, Metajohng, Mindmatrix, Mormegil, Morten, Mudeer, MusikAnimal, Muslim lo Juheu, Oddbodz, Offerman, Patrick Denny, Petrb, Philip Trueman, Pmaccabe, Reedy, RelentlessRecusant, Rich Farmbrough, RichardVeryard, Ron Ritzman, Santurwoman, Sb22000, SchreiberBike, Swintrob, Tee Owe, Tentinator, Theserialcomma, Tide rolls, Tobias Bergemann, Tom Jenkins, Trebor, Trusilver, Uršul, Vcfahrenbruck, Vivio Testarossa, Wayfarer, Wbm1058, Windsamurai, Winterst, کیانی نژاد, 187 anonymous edits

Data profiling *Source*: http://en.wikipedia.org/w/index.php?oldid=591729089 *Contributors*: Aaizemberg, AlistairMcMillan, Arragon-uk, AvicAWB, Careful Cowboy, ChrisGualtieri, Countersubject, DEddy, Dayewalker, Derek Ashton, DragonHawk, Dreamyshade, EagleFan, Edward, Fortdj33, Joffeloff, Jrtayloriv, Kasper110382, Kellylautt, KeyStroke, Kku, Kragen, Logical Cowboy, MBisanz, Muslim lo Juheu, Pucciar, RG2, Rabbasher, Shaw76, TurningWork, Waggers, Wavelength, Wyaddow, 71 anonymous edits

Data quality *Source*: http://en.wikipedia.org/w/index.php?oldid=598556997 *Contributors*: Aaron Brenneman, Abbytelleria, Backslash Forwardslash, Bbplugger, BlueMoses, Bookuser, Bsnel, Byapparov, CKlunck, CactusWriter, Careful Cowboy, Cyberwolf UK, Dataguru, Dawnseeker2000, Dhidalgo, Donner60, Dr. Peter Miller, DragonHawk, Duncharris, Edward, Fiordiligi, Founder DIPM Institute, GB fan, Gioto, Glenn Maddox, Gordonwhamilton, GregorB, Gregredman, Hjy3789759, Hoperock, Jonkerz, Jschwa1, Kinu, Kku, Kragen, Kuru, Latiligence, Logical Cowboy, Lvsubram, Mark Renier, Markdhansen, Matthew Yeager, Mdd, Michael Hardy, Mikeblas, Mindmatrix, Minervasolutions, Mjbinfo, Mr. Vernon, Muslim lo Juheu, Nidarreddy, Nsaa, OItaliano, Oliverbaasch, Peter Flass, Phc3719, Piano non troppo, Pkoppenb, Radack, RainbowCrane, Rich Farmbrough, Ronz, Roux-HG, Saalstin, Skysmith, Staszek Lem, StaticVision, Stefanson, Stephenpace, Stlamanda, THE KING, Tdunbar317, Thetan, Thomasgstf, Timbolt, To Fight a Vandal, VanishedUserABC, Viridae, Virtualelvis, ZimZalaBim, 132 anonymous edits

Data quality assessment *Source*: http://en.wikipedia.org/w/index.php?oldid=595547439 *Contributors*: A3 nm, Aka042, Alexf, Gfeeney, GoldenRatioAnomaly, Katharineamy, Malcolma, Muslim lo Juheu, Rupendar, Utopiainc

Data quality assurance *Source*: http://en.wikipedia.org/w/index.php?oldid=591113060 *Contributors*: Aaron Brenneman, Abbytelleria, Backslash Forwardslash, Bbplugger, BlueMoses, Bookuser, Bsnel, Byapparov, CKlunck, CactusWriter, Careful Cowboy, Cyberwolf UK, Dataguru, Dawnseeker2000, Dhidalgo, Donner60, Dr. Peter Miller, DragonHawk, Duncharris, Edward, Fiordiligi, Founder DIPM Institute, GB fan, Gioto, Glenn Maddox, Gordonwhamilton, GregorB, Gregredman, Hjy3789759, Hoperock, Jonkerz, Jschwa1, Kinu, Kku, Kragen, Kuru, Latiligence, Logical Cowboy, Lvsubram, Mark Renier, Markdhansen, Matthew Yeager, Mdd, Michael Hardy, Mikeblas, Mindmatrix, Minervasolutions, Mjbinfo, Mr. Vernon, Muslim lo Juheu, Nidarreddy, Nsaa, OItaliano, Oliverbaasch, Peter Flass, Phc3719, Piano non troppo, Pkoppenb, Radack, RainbowCrane, Rich Farmbrough, Ronz, Roux-HG, Saalstin, Skysmith, Staszek Lem, StaticVision, Stefanson, Stephenpace, Stlamanda, THE KING, Tdunbar317, Thetan, Thomasgstf, Timbolt, To Fight a Vandal, VanishedUserABC, Viridae, Virtualelvis, ZimZalaBim, 132 anonymous edits

Data Quality Firewall *Source*: http://en.wikipedia.org/w/index.php?oldid=544533104 *Contributors*: AltiMario, Dataguru, Elonka, Emeraude, Marasmusine, Mild Bill Hiccup, Muslim lo Juheu, Tikiwont

Data truncation *Source*: http://en.wikipedia.org/w/index.php?oldid=545812649 *Contributors*: Bearcat, Eds147eds, Father Goose, Muslim lo Juheu, Stuartyeates, Uršul, Versageek, Xezbeth, 1 anonymous edits

Data validation *Source*: http://en.wikipedia.org/w/index.php?oldid=598125734 *Contributors*: 511ecso, 534E524B, Adzzzz, Alexf, AltiMario, Amalas, Anantshri, Arlan65, Bananastalktome, Bonadea, Byapparov, CRGreathouse, Ciprian Serbu, DGG, Demi, Dsdalgleish, Ed de Jonge, Edmund372, Epbr123, Father Goose, Fraggle81, Friendlydata, Gaff, Gaius Cornelius, Gorelikalex, Hydrargyrum, JCLately, John of Reading, Jsayre64, Lawrykid, Leszek Jańczuk, Lethaniol, M4gnum0n, MER-C, Mark Renier, Martinkunev, Materialscientist, Mattgirling, Mister Bubbadubba, Mlouns, MrOllie, Muchness, Muslim lo Juheu, Mithrandir, Nbarth, Nigelj, Nistra, Ohnoitsjamie, Oxymoron83, Pegship, Piano non troppo, Pikajedi3, Pontificalibus, Rror, Skizzik, Softtest123, Staszek Lem, The Bushranger, Theemathas, Thnidu, Thomas Larsen, Tobias Bergemann, Tolly4bolly, Vasilyevna, 123 anonymous edits

Data verification *Source*: http://en.wikipedia.org/w/index.php?oldid=582369260 *Contributors*: Abhishekitmbm, JDP90, Kku, MilerWhite, Muslim lo Juheu, Randhirreddy, Shadowjams, Suriel1981, Velella, Wiki13, 36 anonymous edits

Database integrity *Source*: http://en.wikipedia.org/w/index.php?oldid=593337329 *Contributors*: Avicennasis, BirgitteSB, BradBeattie, Brick Thrower, Danim, Dawynn, Flewis, Friendlydata, Greenrd, Igiffin, Jon186, Mikeblas, Muslim lo Juheu, Neverquick, Rich Farmbrough, Saleah, Sheps11, Spydev, This lousy T-shirt, WayKurat, 15 anonymous edits

Database preservation *Source*: http://en.wikipedia.org/w/index.php?oldid=563122273 *Contributors*: Andy Dingley, BD2412, Biscuittin, Danim, Deb, Ilmari Karonen, Muslim lo Juheu, Mycontributions2wiki, Queenmomcat, RichardMcCoy, S.K., Vegaswikian, Wilhelmina Will

Declarative Referential Integrity *Source*: http://en.wikipedia.org/w/index.php?oldid=495141129 *Contributors*: Bertport, Bluemoose, DerekAsirvadem, Emersoni, Kmote, Muslim lo Juheu, PPOST, Remember the dot, Veatch, Wikijafreeman, 7 anonymous edits

Digital continuity *Source*: http://en.wikipedia.org/w/index.php?oldid=532309741 *Contributors*: El grimley, Jimmy Pitt, Mfbear, Muslim lo Juheu, Regent of the Seatopians, Ronz, Tys1961, 1 anonymous edits

Digital preservation *Source*: http://en.wikipedia.org/w/index.php?oldid=596720493 *Contributors*: 84user, Aap3030, Abune, Adam Conover, Agentbla, Ajt814, AndrewNJackson, Andy Dingley, Andypowe11, Archivare, Armando Orleans, Ashenfelder, Aude, BabelStone, Baffle gab1978, Betakate, Bill151, Binksternet, Blotwell, Bugwit, Buridan, Carpenter512, Cffrost, Cmasters26, Conrad Lochner, DMacks, Dancter, Danhash, Daniel Cull, Davissp, Dcoetzee, Debouncle, Digicurator, Disavian, Doneill7576, Dorgolan, Dpv, Dvsok, Edward, Efkeathley, Ejsherry, El grimley, Emerine, Emijrp, Emorm, Emudave, Enkyo2, Farhrm5, FerranJorba, Floating red, Fmccown, Giancarlo.buzzanca, Giraffedata, Glst2, Gordon Ecker, Gostt, Graham87, HabibzadehParham, Henkvanstappen, Hgladney, Hkhorlos, Hs4pratt, HugoD, Hydroxonium, Ian.rutherford, JackyR, JakobVoss, Jamesontai, JeremyRossen, Jmaferreira, Jnadal, John Nesbitt, JohnOwens, Jordaneg, Josh Parris, Jpbowen, Jska313, KLagerquist, Keithonearth, Kff, LaMenta3, LadislavNK, Latona12, Lm0101, Lorem Ip, Lquilter, Lsf7, Lyc. cooperi, Magu2k, Mariaelena8444, Marklclemente, Maurice Carbonaro, Merope, MilenaDobreva, Mindmatrix, Mjb, Mogism, Mpennock, MrOllie, Mseem, Mudheno, Muslim lo Juheu, NapoliRoma, Neparis, Niteshade, Pekinduck, Peyronnin, Pganas, Ramurf, Raphaelle a, Rhonabwy, Rich Farmbrough, RichardMcCoy, Richarddr, Rjwilmsi, Ronz, SarahStierch, Scalkin, SeamusRoss, Shaddim, Shantavira, Shenqing, Simonjkendon, Sodaha, Stewartadcock, TWCarlson, Tabletop, Tchelseat, Tedickey, Tgeller, TheAmazing0and1, Thibgc, Titodutta, Totoro33, Underpants, Vegaswikian, Venache, VitaminDRO, Whattheheckisthat, Wiki-uk, Wujastyk, Xanthar, Xanzzibar, 123 anonymous edits

User:TheAmazing0and1/draftdigipres *Source*: http://en.wikipedia.org/w/index.php?oldid=478149139 *Contributors*: TheAmazing0and1, Wourriezer

Dirty data *Source*: http://en.wikipedia.org/w/index.php?oldid=574294877 *Contributors*: BirgitteSB, Canada Hky, Careful Cowboy, Colonel Warden, DEddy, DragonHawk, Ewlyahoocom, Fram, IanDBailey, Kdmitry, Malcolma, Melaen, Miljoshi, Muslim lo Juheu, Neutrality, Pegship, Qwfp, Rjwilmsi, The Thing That Should Not Be, Timneu22, Welsh, ÆØN, 20 anonymous edits

Entity integrity *Source*: http://en.wikipedia.org/w/index.php?oldid=542592425 *Contributors*: Calpisone, Coelholr, Dawynn, Dubidub, Gaius Cornelius, GregorB, Ibbn, Informatwr, Jvs, Muslim lo Juheu, NishanChakma, RJFJR, Semog, Shriram.mane, Ta bu shi da yu, Tinucherian, Tumble, 20 anonymous edits

Information quality *Source*: http://en.wikipedia.org/w/index.php?oldid=598483597 *Contributors*: Blue Moses, BlueMoses, ChrisGualtieri, Curious88323, David Rowlands, DragonHawk, GB fan, Huzaifa1990, IRP, IvanLanin, J.delanoy, Kjkolb, LilHelpa, Muslim lo Juheu, Pgalaxy, Proborc, Rjwilmsi, Salvar, Satori Son, Simpatico, Snpoj, Staszek Lem, Stefanson, The Thing That Should Not Be, Timbolt, Tinucherian, Transhumanist, 21 anonymous edits

Link rot *Source*: http://en.wikipedia.org/w/index.php?oldid=596962490 *Contributors*: 16x9, Aaron Kauppi, Adaxl, AgapeOO, Alerante, Alvin-cs, Amniarix, Anthony Appleyard, Asat, Axeman89, BalthCat, Barek, Blargh29, Bricaniwi, Capcom1116, Clarenceville Trojan, Coccyx Bloccyx, ColderPalace1925, Curb Safe Charmer, Cybercobra, Dan100, DavRosen, David Rolek, DavidCary, DavidWBrooks, Denelson83, Derek R Bullamore, Dhaluza, Diomidis Spinellis, Drilnoth, Dtobias, Dysprosia, E23, EardleyC, EdGl, Edgarde, Eric119, Eusc, Everything counts, Eysen, Falcon8765, FeralOink, Ferrierd, Fmccown, Fnielsen, Fr33kman, Fruityland34, Fuebar, Furrykef, Fvw, Gary, GoingBatty, Groyolo, Gwern, Here, Htmlcoderexe, Imyourfoot, Ixfd64, JSo9-10, Jarble, Jason Quinn, Jdorje, Jesin, Jesperronn, Jimthing, Joeinwap, John, John Broughton, John of Reading, Joshuapaquin, Kaltenmeyer, Keilana, Kvng, Kzzl, Ladnadruk, Laurusnobilis, Logan, Lotje, LucasVB, Magioladitis, Majorly, Markmiddleton, Marumari, Masharabinovich, Mat-C, Mdotley, Meatsgains, Meelar, Mentifisto, MetaManFromTomorrow, Mewulwe, Michaeldsuarez, Millwall21, Mindmatrix, Mofuggin bob, Mogism, Monkey 32606, Moretz, MrOllie, Mullibok, Muslim lo Juheu, Nvj, Nyttend, Ocolon, OhanaUnited, OttoMäkelä, Pegship, Phil Boswell, PlatanusOccidentalis, Plazmatyk, Pstudier, Pxma, Qr189, Rchandra, Reg porter, SB Johnny, Salamurai, Secretlondon, Shimmin Beg, Sidsel Sørendatter, SimonP, Smile4ever, Spalding, Tcp-ip, The Mysterious El Willstro, TheAllSeeingEye, Tobias Bergemann, UnitedStatesian, Vahagv, VinceFalks, Virag0, Wavelength, Wik, WikiLaurent, WizardDuck, Xyzzyplugh, Zellfaze, Zero1328, 112 anonymous edits

One-for-one checking *Source*: http://en.wikipedia.org/w/index.php?oldid=514469362 *Contributors*: Bearcat, Clarkcj12, Discospinster, Malcolma, Xetxo, 2 anonymous edits

Referential integrity *Source*: http://en.wikipedia.org/w/index.php?oldid=590028145 *Contributors*: A3 nm, Aim Here, Allens, Amux, Andy Dingley, AnubisAscended, AutumnSnow, BL, Bearcat, Brandon, Brick Thrower, Daniel.Cardenas, Darkunor, Daviburg, DavidLevinson, Elwikipedista, Excirial, FatalError, Flon22, Friendlydata, Greentryst, I dream of horses, KeyStroke,

Kmarshba, Lost tourist, Mark Renier, Materialscientist, Michael.Urban, Mindmatrix, Mtking, Muslim lo Juheu, Nburden, Neurolysis, Niceguyedc, Nivix, Obradovic Goran, Omicronpersei8, PatrickJCollins, Penartur, Philip Trueman, Psb777, Reatlas, Reedy, RuM, Sae1962, Sam Hocevar, Sietse Snel, Simtay, Snodnipper, Staszek Lem, Suvs2011, Ta bu shi da yu, Tarquin, Tolly4bolly, Varuna, Wjhonson, Wlievens, 118 anonymous edits

Soft error *Source*: http://en.wikipedia.org/w/index.php?oldid=593377900 *Contributors*: (jarbarf), Alazlogexi, BD2412, Brianhe, CanisRufus, Chemicalinterest, Compellingelegance, Deville, Docu, Download, Dsimic, Edward, Ehn, Emrys2, Erock48210, Euchiasmus, Ewlyahoocom, Extropian314, Giraffedata, Harry, Hazelburr, Hgrosser, Himanic, Hugh Mason, JohnElder, Jokes Free4Me, Jorge Stolfi, Julesd, Killian441, Kolbasz, Merzul, Michael Hardy, MinorContributor, Muslim lo Juheu, Neels, Nick Number, Nneonneo, PDD, Plasticspork, Pnm, Pqrstuv, Ruud Koot, SDS, Shaddack, Silas S. Brown, SlimVirgin, Spike Wilbury, Steinsky, Thinking of England, Timtim.hoque, Veryfoolish, Wernher, Wilkij1, Wyvern, Xanzzibar, Ximalas, Xiong Chiamiov, 84 anonymous edits

Two pass verification *Source*: http://en.wikipedia.org/w/index.php?oldid=564378238 *Contributors*: ArnoldReinhold, Blaxthos, Camw, Furrykef, Ischemia, Materialscientist, Mild Bill Hiccup, Peter Flass, Wfeuer, 13 anonymous edits

Validation rule *Source*: http://en.wikipedia.org/w/index.php?oldid=581833874 *Contributors*: Aitias, AutomaticStrikeout, BD2412, Bgwhite, DGG, Gardar Rurak, Jeepday, Jojalozzo, JonHarder, Mark Renier, Matma Rex, Muslim lo Juheu, PhilKnight, SFK2, Smalljim, William Avery, Yintan, ÆØN, 31 anonymous edits

Semi-structured data *Source*: http://en.wikipedia.org/w/index.php?oldid=596064328 *Contributors*: AKA MBG, Andreas Kaufmann, Anvildoc, D'Artagnol, Download, Fuper, Jengelh, John of Reading, Kandreyev, Khazar, Kku, Lysy, Marcuscalabresus, MilerWhite, Prakash Nadkarni, Triwbe, VLReeder77, Valenciano, 16 anonymous edits

Semi-structured model *Source*: http://en.wikipedia.org/w/index.php?oldid=551699454 *Contributors*: Allens, Danim, Lyondif02, Mwarf, Neilc, Nichtich, Nkayesmith, Ronz, Rosejn, 6 anonymous edits

Standard Generalized Markup Language *Source*: http://en.wikipedia.org/w/index.php?oldid=597309977 *Contributors*: 4th-otaku, Acdx, Adsims2001, Adys, Alexbrn, AlistairMcMillan, Ambarish, AndrewWTaylor, Andy Dingley, Angelpeream, Ashdurbat, Asymmetric, BD2412, Beinsane, Ben-Zin, Betterworld, Blaxthos, BodyTag, Bomazi, Boris Barowski, Burschik, C.M.Sperberg-McQueen, Chalst, Chealer, Chris the speller, Chris55, Chrisahn, Chrislk02, Chuuumus, Conversion script, Dbg0, Dcirovic, Deodar, Discospinster, Dmsar, Dreftymac, Dysprosia, Egil, Ehn, Elwikipedista, FatalError, G74793, George Makepeace, Ghettoblaster, GoingBatty, Graham87, Gurch, HappyDog, Howard McCay, Hubalu, Ike9898, Illegitimate Barrister, Indefatigable, JForget, John Vandenberg, Jordancpeterson, Kazvorpal, Keichwa, KeithTyler, Klingoncowboy4, Klparrot, Kricxjo, Ktdreyer, Kutulu, Kvng, Lafeber, Lambiam, LeeHunter, Lfstevens, Lifefeed, Liftarn, Lockley, LucasVB, Magicbronson, Masgatotkaca, MattiasAndersson, Mecanismo, Meeroslav, Mhazard9, Minghong, Modster, Mratzloff, Msreeharsha, Nanshu, Ninly, Nk, Norm mit, Nuno Tavares, Ohka-, Omnipaedista, Open Book, Optikos, Ott0, Paul Magnussen, Pentap101, Pfrishauf, Przepla, Psychlohexane, Quota, Qutezuce, RedWolf, Rick Jelliffe, Ringbang, Rjgodoy, Rjwilmsi, Robert Hiller, Roo72, RossPatterson, Rtc, Rursus, SamB, Schnolle, Sideshowbarker, Skim, Smyth, Stephen Gilbert, SteveGoodey, Stuartyeates, Suwa, Svick, Tedickey, Tene, The Anome, Thewikimaster17, Tmrtlght, Tomdo08, Toolnut, Traroth, TwoOneTwo, Typhoonhurricane, V&-L.E.E.T, Widefox, Wikivek, ZMughal, Zazpot, 152 anonymous edits

XML *Source*: http://en.wikipedia.org/w/index.php?oldid=599111947 *Contributors*: .:Ajvol:., 207.172.11.xxx, 213.253.39.xxx, 24ten, 28421u2232nfenfcenc, AHMartin, AK382186, AThing, Aadaam, Actam, AdamCarden, Adeio, AdventurousSquirrel, Aeusoes1, Ahabr, Ahkond, Ahoerstemeier, Aitias, Ajcumming, Aklauss, Aksi great, Alan Liefting, Alansohn, Alexbrn, Alison, AlistairMcMillan, Allkeyword, Amakuru, Amire80, AndersFeder, Andrisi, Andy Dingley, Andy Monakov, Angeltoribio, Ani td, Anish9807, Ankitasdeveloper, Anna Lincoln, Anon lynx, AnonMoos, Anti stupidity, Anu-43, Aomarks, Apollo1758, Aselaruwan123, Asqueella, Asteiner, Asymmetric, Atanveer9, Atethnekos, Axd, AzaToth, B4hand, BD2412, Barek, Barticus88, Bdesham, Beetstra, Belamp, Bender235, Bernd in Japan, BertSen, Bevo, Bgwhite, Bhadani, Biezl, BigFatBuddha, Bissinger, Bje2089, Blethering Scot, Blinklmc, Bluemoose, BlurTento, Bobdc, Bobianite, Boehm, Bonbayel, Bonethugnd, Booles, BorgQueen, Borgdylan, Boseko, Brettz9, BrianCully, Brick Thrower, Brighterorange, Brion VIBBER, Bruce1ee, Bryan Derksen, Brz7, Bsadowski1, Bunnyhop11, Burschik, Businessman332211, Bvajet, C.M.Sperberg-McQueen, CLD, CambridgeBayWeather, Cameltrader, Can't sleep, clown will eat me, CanadianLinuxUser, Caomhin, CapitalSasha, Carewolf, CarlHewitt, Caro.d38, Cbdorsett, Cels2, Centrx, Charivari, Chealer, Cherkash, Chininazu12, ChongDae, Chowbok, Chris 73, Chris Roy, ChrisGualtieri, Chrislk02, Chrisnewell, ChristianGruen, ChristopheS, Chzz, Cipherynx, Clayoquot, ClementSeveillac, CoSort2007, Coconut99 99, Cody5, Colonies Chris, Comesuntbob, Compfreak7, Contraverse, Conversion script, CptAnonymous, Crosstowns, Cspan64, Cupid1889, CyberSkull, Cybercobra, Cœur, D6, DARTH SIDIOUS 2, DKEdwards, DVdm, Dan100, DanConnolly, Daniel Olsen, Daniel.Cardenas, DanielVonEhren, DarkFalls, Darkfred, David spector, Davis685, Dcattell, Dcoetzee, DePiep, DeadEyeArrow, Delcnsltmd, Deodar, Derek Ross, Derekread, Dewritech, Dicklyon, Dickpenn, DigitalEnthusiast, Dingbats, Dino72, Dionysia, Dkrms, Dlohcierekim, Dlrohrer2003, Dolcecars, DominiqueHazaelMassieux, Don4of4, Donmay12, DoorsAjar, DopefishJustin, DoriSmith, DougBarry, DouglasHeld, Dpattison2007, Dpbsmith, Dpm64, Dr Headgear, Dr. Zombieman, Dreftymac, Dthvt, Dullhunk, Dwheeler, EastTN, Ebruchez, Edcolins, Edokter, Edward, Edward Z. Yang, Efcavanaugh, Egandrews, Egil, Eidab, Eisnel, ElBenevolente, Electron9, Elharo, Elizarf, Ellmist, Elwikipedista, EngineerScotty, Eob, Eranb, Ericjs, Erik Zachte, Erikdw, Eritain, EternalFlare, Etu, Evaluist, Ewsers, F331491, Fang.zheng, Fantasticfears, FatalError, Feline Hymnic, Ferdinand Pienaar, Figure, Fleminra, FloatingMind, Fnielsen, Folajimi, FootholdTechnology, Forage, Fraggle81, Fragglet, Fran Rogers, Francl, Frankweerasinghe, Frap, Freiddie, Freyr, Frisket, Frze, Fsolda, Funandtrvl, Furrykef, Fvw, GTBacchus, Gaius Cornelius, Gc9580, Gdrori, Geniac, Gennaro Prota, GentlemanGhost, GeoffPurchase, Ghettoblaster, Giftlite, GioeleBarabucci, Gjlubbertsen, Gjs238, Glass of water, Glenn, Goasklaura, Gogo Dodo, Golwengaud, GrEp, GraemeL, Graham, Graham87, Greg Murray, Ground Zero, Grumpycraig, Gudeldar, Guy Macon, Haakon, Hairy Dude, Hannes Hirzel, Harold f, Hashar, Headbomb, Hervegirod, Hicketyhicketyhack, Highwayman65251, Hillbillyholiday, Hires an editor, Hirzel, Hogman500, Hohum, Hoo man, Howard McCay, Hu12, Hurricane111, Hypertrek, Hyuri, IMSoP, Iamjaypatel, Ian Moody, IanBurrell, Iftikhar88hussaini, Ijmorlan, IlanaDavidi, Imars, Imjustmatthew, Int21h, Intgr, Iridescent, IrkBerkeley, Isilanes, Itai, J.delanoy, JForget, JKing, JLaTondre, JPaestpreornJeolhlna, JPalonus, JRocketeer, Jack Greenmaven, Jackacon, Jackmileshunt, Jacobko, Jacobolus, Jah1138, JakobVoss, JamesBrownJr, Jan.Keromnes, Jandalhandler, Jao, Jargon64, Jasper Deng, Jauerback, JavaWoman, Jaxad0127, Jaxsam1, Jay, Jeenuv, Jeff G., Jeff3000, Jehzlau, JeremySmyth, Jerome Charles Potts, Jesin, Jhannah, Jibjibjib, Jilplo Haggins, Jimthing, Jmlipton, Joachim Wuttke, Joanjoc, John Vandenberg, JohnSmith777, JohnWhitlock, Johnmarkh, Johnwcowan, Joku, Jonabbey, Jonkerz, Jonnyamazing, Jor, Jpbowen, Jprg1966, Jshadias, Jzhang2007, Kai.Klesatschke, Kaldosh, Kamalakannanprogrammer, Kanags, Kapoing, Karderio, Karl Dickman, Katalaveno, Kbrose, Kc2idf, Keithgabryelski, Kenmccallum, Kensall, Kevinconroy, Kgaughan, Kha0sK1d, KickAssClown, Kingius, Kjtobo, Kl4m, Klangenfurt, Klaws, Koavf, Kogmaw, Korval, Krauss, Ksaha.pune, KuduIO, Kwi, Kwiki, Kx1186, KylieTastic, LDiracDelta, Lambiam, Larala, Lazynitwit, Leandrogfcdutra, Leeannedy, Lianmei, Liao, Lifefeed, Liftarn, Ligulem, Ling.Nut, LittleDan, Loveenatayal, Lukys, Lumi71, Lycurgus, M.franceschet, M4gnum0n, MER-C, MHLut, MK8, MaBoehm, Madir, Magioladitis, Mah159, MainFrame, Mak Thorpe, Malleus Fatuorum, Manishtomar, Maoj-wsu-sp, Mark Renier, MarkSweep, Martijn faassen, Martin451, Martinp23, MartynDavies, Materialscientist, Mathmo, Matthiaspaul, Matthäus Wander, Mauro Bieg, MaxEnt, Maximaximax, Maximus06, Mayfare, Mbbradford, Mbell, Mcintyem, Mcorazao, Mecanismo, Melab-1, MelbourneStar, Melon039, Meszigues, Mfbubt, Mhkay, Michael Hardy, MichaelJanich, Miguelfms, MikeLynch, Minghong, Mion, Miss Dark, Mjb, Mjpieters, Mmichaelc, Mojo Hand, Mola8sses, Montgomery '39, Moosehadley, Mp, Mr. Shoeless, Mr. Stradivarius, Mr.Z-man, MrJones, MrOllie, Mrjmcneil, Ms2ger, Mthibault, Mustha jm, Muzammal65, Mvulpe, Mwtoews, Mww113, Mxn, NO ACMLM,AND XKEPPER SUCK !, Nannus, Nanshu, Natasha2006, Nations114, NawlinWiki, Neckro, Nemo bis, Netsnipe, NickGarvey, Nicmila, Nigelj, Nikkimaria, Nikosbik, Nile, Ninly, Niteowlneils, Nivaca, Nixeagle, Noldoaran, Nomediga, Norm mit, Nowa, Nsh, Nwbeeson, Ocaasi, Octane, Ogmios, Ohnoitsjamie, Okyea, OliD, OsamaK, Oscar-ja, Osquar F, OverlordQ, Owen Ambur, Oxblood, Oxinabox, Oyster Flute, P3x984, PTSE, Patrick, Paul Foxworthy, PaulXemdli, Pavel Vozenilek, Paxsimius, PeacefulPlanet3, Peashy, Pelle, Pengo, PeteVerdon, Peterl, Pgk, PhiLho, Philip Trueman, Phluid61, Phoenix-forgotten, Phyzome, Pianohacker, Pikiwyn, Pinethicket, Pmberry, Pnm, Poccil, Porges, Pozcircuitboy, Prakash Nadkarni, Quarl, Quasipalm, Quiddity, Quilokos, R'n'B, Ramesses the Great, Rashaunny, Rbonvall, Rbstimers, Rdmsoft, Red660, RedWolf, Redherring, Reinthal, Remy B, RenniePet, Rich Farmbrough, RichMorin, Richalex2010, Rick Block, Rick Jelliffe, RickBeton, Risi, Ritvikbhatnagar1, Rivecoder, Rje, Rjstott, Rjwilmsi, Rklawton, Robert K S, Robert Merkel, Robinjwest, Robomaeyhem, Rodney Boyd, Roger costello, Rory096, RoseParks, Rr2bwreain, Rror, Rvmolen, Ryanrs, Sam Hocevar, SamHathaway, SandiCastle, Sandius, Saqib, Saucepan, Sbvb, Schnolle, Scjessey, Scott MacLean, Scottielad, ScottyBerg, Sderose, Seanhan, Seaphoto, Secret, Seidenstud, Semper discens, Sen Mon, Sfan00 IMG, ShaneCavanaugh, Shanes, Shibboleth, Shii, Shinkolobwe, Shizhao, Shlomital, SickTwist, Signsofstatic, Simetrical, SivaKumar, Sj, Sjc, Sleepyhead81, Smyth, Sosinfo, Sound effx, Spankman, Spe88, Spudstud, SqueakBox, Srflecha, Star767, Stefan.ciobaca, Stephen Gilbert, Steve R Barnes, Stevy76, StewartMH, Stf, Stijn Vermeeren, Stupiddestyredgasd, Stwalkerster, Superm401, Suruena, Suwayya, Svetovid, Syangtar, Sydius, TPK, Tagith, Tags123, Taknik, Talktovalentine, Tassedethe, TastyPoutine, Technopilgrim, Teddyb, Terjen, Terrifictriffid, Terrycojones, Thadius856, The Thing That Should Not Be, TheMightyOrb, TheOldJacobite, Thierryc, Think777, Thnidu, Thumperward, Thunderhead, TimBray, TimR, Timc, Timur.shemsedinov, Tobias Bergemann, Todd Vierling, TommyG, Tony1, ToonArmy, Topbanana, Toussaint, Tpbradbury, Trade2tradewell, Trankin, Traroth, Treekids, Trovatore, Trscavo, Tsunaminoai, Turnstep, TwoOneTwo, Twocs, Tympan, Typhoonhurricane, Typochimp, Tyrol5, UkPaolo, Uncle Milty, Unforgettableid, Unixxx, Unknown W. Brackets, V2580s, Vadmium, Vaganyik, Varlaam, Verdy p, Versageek, Vespristiano, Vigilius, Violetriga, Vladkornea, Vojta, Volphy, Voxii, Vrenator, WSU-AW-AK, Wael Ellithy, Walk Up Trees, Waskage, Wavelength, Wereon, Whale plane, Whkoh, Wickorama, Widefox, Widr, Wiki Leah, Wiki alf, Wiki.Tango.Foxtrot, Wiki0709, Wikilibrarian, Winterst, Wmahan, WojPob, Woohookitty, Wrs1864, Wulfila, Ww, XJaM, Xmltools, Xompanthy, Xpclient, Xxiggy, Yaronf, Ygramul, Yintan, Yonkie, Zhaolei, ZoeB, Zoolium, Zootm, ZzzzBov, Олександр Кравчук, Тиверополник, 1457 anonymous edits

XML database *Source*: http://en.wikipedia.org/w/index.php?oldid=592202755 *Contributors*: 16x9, AJackl, Abukaspar, Adamretter, Adrianwn, AlbertJB, Americanpiggy, Amirfr, Andionita, Arnabdotorg, Avoided, Barefootliam, Beland, Belovedfreak, Bernd vdB old, Bgwhite, Bohumir Zamecnik, Brick Thrower, Bunnyhop11, Bxj, Ccouvrette, Cdicarlo, Chowbok, ChristianGruen, Colonies Chris, CorcaighAbu, Courcelles, Danim, DickieRose, Dilane, Dizzzz, Dmccreary, Doclabyrinth, DoriSmith, Edward C. Zimmermann, Eedeebee, Enric Naval, Epbr123, EricBloch, F331491, GVogeler, Glen Pepicelli, Gonim, Gpallis, Gregburd, Happygiraffe, Hgkamath, Hobartimus, Joerg84, John Vandenberg, Johndbritton, Juansempere, Jzhang2007, Kgfleischmann, Klingon, Kmorozov, Kokotero, Lamdk, Lamp90, Libcub, Linelor, Lugia2453, Mabdul, Mark Renier, Materialscientist, Mauro Bieg, Mdd, Mehran, Metaperl, Michael Slone, MiddleEarth, Mitchumch, Nichtich, Nikkimaria, OlliX, Pearle, Pedant17, Philip Trueman, Playmobilonhishorse, Q Chris, R'n'B, Radim Baca, Rastgoo, Rayngwf, Rjwilmsi, Rtweed1955, Signalhead, Slakr, Snodnipper, Stevertigo, Sykamoore, TRosenbaum, Tags123, Tbradford, Terrifictriffid, Themfromspace, Thumperward, Tide rolls, Touko vk, Vojtechtoman, Xafran, Xmlchamp, Xmlizer, Xpriori, Xshezang, Xxanthippe, 335 anonymous edits

XML Schema Language comparison *Source*: http://en.wikipedia.org/w/index.php?oldid=524481868 *Contributors*: Ahoerstemeier, Bunnyhop11, Cfeet77, ChrisGualtieri, Crystallina, Decrease789, Dongwon, Dreftymac, Drrwebber, Ghettoblaster, Giraffedata, Grumpycraig, Henning Makholm, Hsivonen, Jlowery, Koavf, Korval, Netjeff, Penter ghost, Q Chris, Sloop Jon,

Sześćsetsześćdziesiątsześć, Tijfo098, Tuntable, 37 anonymous edits

XML schema *Source*: http://en.wikipedia.org/w/index.php?oldid=587754230 *Contributors*: ABCD, Acdx, Ahoerstemeier, Alik Kirillovich, AutumnSnow, Beetstra, Bjankuloski06en, Bunnyhop11, Cbdorsett, Choster, Crystallina, Derekread, Dongwon, Doug Bell, Dreftymac, Drrwebber, Ehn, Fryed-peach, Gardenstew, Hervegirod, Hymek, Jamelan, Jaxsam1, Korval, Krauss, Kucing, Mamling, MariahX, Mark Renier, MarkSweep, Mhkay, Minghong, Miracle Pen, Mjb, Ninly, Ottomachin, Pi8ch, Pmerson, Poccil, Pxma, R'n'B, Rich Farmbrough, Runnerupnj, Sae1962, Sahuagin, SheepNotGoats, Slon02, Smyth, Stevage, SteveLoughran, Tijfo098, Tobias Bergemann, Vernanimalcula, Vishrave, Wael Ellithy, Xan 213, Þjóðólfr, 60 anonymous edits

XML validation *Source*: http://en.wikipedia.org/w/index.php?oldid=579089085 *Contributors*: 3nx, Andy Dingley, David Haslam, Dawynn, Dreftymac, Drrwebber, EdJogg, Fnielsen, Hmains, Hymek, Jaxsam1, Korval, LittleBenW, MrOllie, NotinREALITY, Pmerson, Rich Farmbrough, Waacstats, 17 anonymous edits

Xpath data model *Source*: http://en.wikipedia.org/w/index.php?oldid=507748381 *Contributors*: Dawynn, Huttarl, Mauro Bieg, Mdd, Tassedethe, Tinucherian, Velle, 1 anonymous edits

Path expression *Source*: http://en.wikipedia.org/w/index.php?oldid=588011836 *Contributors*: Jesse V., Ligulem, Piet Delport, Raul654, Shoeofdeath, TKD, Twri, Uogl, 1 anonymous edits

XQuery *Source*: http://en.wikipedia.org/w/index.php?oldid=585713280 *Contributors*: A5b, Adamretter, AlchymistX, AlexInWikiland, Angela, ArglebargleIV, Astazi, Baby learns to fly, Bcmoney, BeakerK44, Bovineone, Bunnyhop11, Chrisahn, ChristianGruen, Colonies Chris, Cutlass2009, DagErlingSmørgrav, Danakil, DavidDouthitt, DeirdreGerhardt, Denistorres, Denisutku, Dmccreary, DoriSmith, Doug Bell, Dsewell, Duncan.Hull, Endorf, F331491, FloatingMind, Folajimi, Frap, Gee totes, Ghettoblaster, Gonzobrains, Gregburd, Hgfernan, Hieptl, Housseno, IanTrout, Jan.Sievers, John Vandenberg, Johnpcsnelson, Joshuadfranklin, Juhuyuta, K1Bond007, Kate, Kku, Koavf, Krauss, Lethalmonk, Mark Renier, Mauro Bieg, Maximus Rex, Mhkay, Michael Slone, Mikaey, Minghong, Mjb, Mortense, MrOllie, Mzajac, Nagnatron, Neilc, Nikkimaria, Nikosbik, Paul Foxworthy, Peterl, Reptarx, Rich Farmbrough, Risi, Russianspy3, Salam32, Sathiyamoorthysp, Sbuxton, Scs, Sean.hoyland, Securiger, Smyth, Sstrader, Stanqo, Sykamoore, TXiKi, Tags123, Theo F, Torc2, Toussaint, Typhoonhurricane, Wwmbes, XMLer, Xfranc, Xshezang, Yuvrajpatel, Yzchang, 183 anonymous edits

XSA *Source*: http://en.wikipedia.org/w/index.php?oldid=557773856 *Contributors*: 21655, Bachrach44, Capricorn42, Ckburke, Closedmouth, Darklilac, Erik9, Fahadsadah, Frap, Histolo2, INSOMaNIAC, Ixfd64, Kerotan, Mindmatrix, Mohamed Magdy, MrOllie, Onewhohelps, Sam42, Seantheman444, T3wadna, Techietim, Tide rolls, X2Fusion, 52 anonymous edits

XSIL *Source*: http://en.wikipedia.org/w/index.php?oldid=546476794 *Contributors*: AlistairMcMillan, Altenmann, Bogdangiusca, Decoy, Ebichu63, Gudeldar, Itai, K1Bond007, Kate, Mosca, Pegship, Pjvpjv, Securiger, WolfgangRieger, 2 anonymous edits

SQL/XML *Source*: http://en.wikipedia.org/w/index.php?oldid=598090893 *Contributors*: Danielkec, FloatingMind, Kelti, Klausness, Krauss, Pathoschild, Sinisterstuf, SqlPac, Touko vk, Ysangkok, 19 anonymous edits

Soma File *Source*: http://en.wikipedia.org/w/index.php?oldid=553177924 *Contributors*: Amina.alobaid, Cbdorsett, Dethlock99, Iamjbm, John of Reading, Somainfo2

Regular Language description for XML *Source*: http://en.wikipedia.org/w/index.php?oldid=543728854 *Contributors*: AutumnSnow, Bovineone, Businessman332211, Docu, Evil saltine, Fg2, Kbdank71, Mxn, Phil Boswell, Rorro, Securiger, Wael Ellithy, 5 anonymous edits

PureXML *Source*: http://en.wikipedia.org/w/index.php?oldid=590101575 *Contributors*: Apokrif, Bearcat, CorcaighAbu, Elen of the Roads, Elwikipedista, EmanWilm, Ghettoblaster, Jerryobject, Lpetrazickis, Mabdul, Michael-se, 9 anonymous edits

List of XML schemas *Source*: http://en.wikipedia.org/w/index.php?oldid=595499439 *Contributors*: Asparagus, Beetstra, Chip8888, Cwmhiraeth, DalbS, Dreftymac, Egandrews, Fram, Frap, GVogeler, Gabrielbodard, Gvanecek, JVz, Jarekt, Jgranade, Jnoe1988, Jobin RV, KAtremer, Krsmith, Kvng, Melbmuso, Mfloryan, Nealmcb, Nghawes, Owen Ambur, PrimroseGuy, PrivateWiddle, Ramu50, Rwwww, Simon sprott, Smhdog, Tranale, Tster9306, Utuado, Vanquisher.UA, XMLer, 30 anonymous edits

Object database *Source*: http://en.wikipedia.org/w/index.php?oldid=594627331 *Contributors*: 28421u2232nfenfcenc, Addbc, Alexandre.Morgaut, André Miranda Moreira, Aremith, AutumnSnow, BSTRhino, Bablind, Beardo, Beland, Ben-Zin, BenFrantzDale, Bgwhite, Booler80, Britannica, Bunnyhop11, Cameltrader, Cari.tenang, CesarB, Charwing, Compfreak7, Conversion script, Corpx, DBooth, DallasClarke, Dandv, Danim, Dawn Bard, Dinojc, Dmccreary, DougBarry, Dybdahl, ENeville, EastTN, Edlich, Eduardofeld, Eeekster, Ejrrjs, Elwikipedista, Enric Naval, EoGuy, Ervinn, FatalError, FlyingPhysicist, Forderud, Foxxygirltamara, Fraggle81, Furrykef, Gadfium, Garyaj, Germanviscuso, Gf uip, Grafikm fr, Gwern, Hari, HarlandQPitt, Hofoen, Hu12, Hydrargyrum, ITOntheMind, Icey, Indolering, IronGargoyle, J.delanoy, JCLately, JIP, Jackelfive, Jbom1, Jivecat, John of Reading, Karnesky, Kellen, Khiladi 2010, Ki2010, Kku, Klemen Kocjancic, Kngspook, Kozka, LGuzenda, Larsinio, Leandrod, Lguzenda, Lucpeuvrier, MER-C, Maria Johansson, Mark Renier, Matspca, Maury Markowitz, Mckaysalisbury, Mdd, Mhkay, Mika au, Mindmatrix, Minimac's Clone, Mlibby, Modster, Mvjs, Nick Number, Nkaku, Noldoaran, OEP, Oberst, Osiris, Ott2, Pavel Vozenilek, Pcb21, Phiarc, Pintman, Pradiq009, PsyberS, Pwwamic, RProgrammer, Razorbliss, Rednblu, Reedy, RichardF, Rickyp, Ronz, Ruud Koot, Rzicari, SAE1962, Saifali1, Sam Hocevar, Sandstein, SchuminWeb, Secured128, Sestoft, Shewizzy2005, Shreyasjoshis, SimonArlott, Singerboi22, SmallRepair, Snow Blizzard, Soumyasch, Startswithj, SteinbDJ, Stevertigo, Sun Creator, SvetCo, Tablizer, Tagus, Talyian, The Thing That Should Not Be, Timosa, Torc2, Tordek ar, Usrnme h8er, Voidxor, Vtan, W, Wainstead, Waynenilsen, WikipedianYknOK, Wixardy, Yaris678, Zeliboba7, Zippanova, 335 anonymous edits

Object Definition Language *Source*: http://en.wikipedia.org/w/index.php?oldid=550390547 *Contributors*: AutumnSnow, Dainomite, GMNoel, Hashc0de, Jerome Charles Potts, LOL, Phydeaux, Refaeldakar, Rich Farmbrough, Sadads, Vneiomazza, Wikidrone, 11 anonymous edits

Object Query Language *Source*: http://en.wikipedia.org/w/index.php?oldid=540663885 *Contributors*: AutumnSnow, Bihco, Bmusician, CaptainPinko, DNewhall, Flexx, GMNoel, Gblotter, Gokusandwich, GregorB, Jnothman, Kku, PigFlu Oink, Svick, Valafar, Villarinho, 21 anonymous edits

Object-oriented SQL *Source*: http://en.wikipedia.org/w/index.php?oldid=545121104 *Contributors*: Adolescence, Bearcat, Jeepday, Segv11, 2 anonymous edits

Object Exchange Model *Source*: http://en.wikipedia.org/w/index.php?oldid=544494848 *Contributors*: Egpetersen, Kjkolb, Malcolma, Nichtich, Nils Grimsmo, Ourai, Russianspy3, 1 anonymous edits

Object-relational database *Source*: http://en.wikipedia.org/w/index.php?oldid=598704203 *Contributors*: Acatyes, AndrewWTaylor, Beland, Bohunk, Bryan Derksen, Cbayly, CesarB, ChandraASGI, Chikako, Cybercobra, Danim, Diberri, Dmsar, DougBarry, Dze27, Enric Naval, FatalError, JLaTondre, Jamelan, Jay, Jerome Charles Potts, LedgendGamer, Mark Renier, Maury Markowitz, Mdd, Mindmatrix, Mrwojo, Mwtoews, Nmushov, Nthomas, Oxymoron83, Pedant17, Premil, Razorbliss, Rouenpucelle, Rp, Rursus, SeanTater, Sergsav, Sémaphore, Tedickey, Theking2, Thomas Willerich, Tijfo098, Tom, Turnstep, Vald, 76 anonymous edits

Object-relational impedance mismatch *Source*: http://en.wikipedia.org/w/index.php?oldid=598422839 *Contributors*: Alexadamson, Allister MacLeod, Ambarish, Andy Dingley, Aprock, BazookaJoe, Big Brother 1984, Bkonrad, Brick Thrower, Cantonnier, Chris the speller, Colonies Chris, Craig Stuntz, Creative1985, Cybercobra, Danim, Dbasch, Diego Moya, Dmccreary, Draicone, EngineerScotty, Erik Postma, Esap, Fram, GCarty, GoingBatty, Grafen, Hairy Dude, Hu12, Ideogram, J.delanoy, Jeffq, Jerome Charles Potts, Jiminez, Joshua Davis, Jpp, JubalHarshaw, Kcragin, Leandrod, Lesser Cartographies, Magioladitis, MarSch, Mark Renier, Mentifisto, Merenta, Metafax1, Mike Schwartz, Mojo, Morven, Msiddalingaiah, N8allan, OldTownIT, Ospalh, PKT, Pearle, Pingku, Prakash Nadkarni, Q Chris, Rbraunwa, Rdmil, Rich Farmbrough, Rjwilmsi, Roux-HG, Rursus, Ruud Koot, SAE1962, Salix alba, SarekOfVulcan, Sbhug1, Scope creep, Shaddim, Tablizer, Topbanana, Towopedia, Triddle, Underpants, Warren, Wickethewok, Widr, ZacBowling, Zsvedic, 108 anonymous edits

Object-relational mapping *Source*: http://en.wikipedia.org/w/index.php?oldid=599275749 *Contributors*: ARC Gritt, AWK, Agentq314, Akimov alexey, Albanaco, Alexkon, Alexyakunin, Andre Engels, Andrewmcgregor, Andy Dingley, Ario, Aristedes, Arjayay, AxelBoldt, BSTRhino, Baojia, Beetstra, Binarybits, BlackNeXT, Blarggstar, BlindStriker, Borisyankov, Brian McErlean, Bryan Derksen, Budloveall, Burschik, Cambalachero, Cameltrader, Can't sleep, clown will eat me, Cander0000, Cantonnier, Cennin, Cernatcristi, Chaos5023, Chrifi, Chris G, Clemwang, Cmdrjameson, Crasshopper, Danim, Ddxc, Dhanu1000, Diego.sarmentero, Dionyziz, Dister, Dmccreary, Domesticenginerd, DougBarry, Dougluce, Dperiwal, Dyfrgi, Echartre, EngineerScotty, Eric Le Bigot, Erics, FatalError, Florian Sening, Foxxygirltamara, Frap, Friendlydata, Fthiess, GCarty, Gazpacho, GeorgeLouis, Gilliam, Grahamstewart, Gravbox, Gribeco, Heiko, Hephaestos, Hirzel, Hu12, J.delanoy, JIP, Jamelan, Jarhed, Jeffrey Odell, Jeffreymcmanus, Jim1138, Jk2q3jrklse, Jojalozzo, JonHarder, Joshisachin79, Jpp, Kbg, Kbh3rd, Kevinsparrow, Khalid hassani, Khiladi 2010, Kylehayes, Leandrod, Lesonyrra, Lesser Cartographies, Lesv, LeszekKrupinski, Lgirvin, Lguzenda, Lumingz, MainFrame, MarXidad, Mathieumcguire, Maury Markowitz, Mdd, Mesoderm, Mgnoonan, Mib, Mikealrogers, Mindplay.dk, Mselway, Nate Silva, Nickheidke, Nicolas1981, Nixeagle, Novacatz, Ohnoitsjamie, Okamps, Olinga, Onexdata, Ormapper, PTSE, Patriotic dissent, Pavel Vozenilek, Pedahzur, Pelister, Pengo, Peterbrunner, Pevernagie, Pgan002, Philippe Leybaert, Plugwash, Pricey3000, Project2501a, Q Chris, Quamaretto, Qwertyus, RayKiddy, Rbygrave, Recurve7, RenniePet, Rich Farmbrough, Robsiklos, Roma emu, Ronz, Ruud Koot, SAE1962, Sa'y, Sascha dd, Shimeru, Shizane, Shon, Sietse Snel, Simetrical, Soumyasch, Spsalsm, Techi2ee, The Hokkaido Crow, Thorwald, TimurIzhbulatov, Tnagle, Tonydent, Toppler, Twimoki, Ubernostrum, UkPaolo, Underpants, Unit3, Vaskess, Veinor, Vina, Vinsci, Vortex, Warren, Wdror-wsu-ad, Wikidrone, Williajm, William Avery, William Sisson, Wjhonson, Wolfgang P, Xyzzycoder, Y.P.Y, ZacBowling, Zntrip, 448 anonymous edits

Polymorphic association *Source*: http://en.wikipedia.org/w/index.php?oldid=527222629 *Contributors*: Alex Marandon, Danim, DreamySmurf, JMiall, Raggmopp614, Sinobra, 1 anonymous edits

Polyinstantiation *Source*: http://en.wikipedia.org/w/index.php?oldid=560928068 *Contributors*: Blaufish, Cander0000, Codename Lisa, DanielPharos, Dougher, Greenrd, Jarry1250, Kwamikagami, Malcolma, Mratchford, Thedatastream, Vegaswikian, Waacstats, 13 anonymous edits

Single Table Inheritance *Source*: http://en.wikipedia.org/w/index.php?oldid=532231698 *Contributors*: Berek, Bobcompu, Excirial, Ghost2008, Malcolma, R'n'B, Van der Hoorn, 5 anonymous edits

Versant Object Database *Source*: http://en.wikipedia.org/w/index.php?oldid=589525753 *Contributors*: Bearcat, Bunnyhop11, Cander0000, Danim, Edward, Eugene-elgato, Germanviscuso, HardBoiledEggs, Ipsign, JLaTondre, Ketiltrout, Kku, Koumz, Mandarax, Mild Bill Hiccup, Nick Number, Pmitka, R'n'B, Smallman12q, W Nowicki, WikiDan61, 6 anonymous edits

Terminology-oriented database *Source*: http://en.wikipedia.org/w/index.php?oldid=593139373 *Contributors*: Beland, Fadesga, Gibbja, Michaelbeijer, Odaba, ReinhardK, Wilhelmina Will

Odaba *Source*: http://en.wikipedia.org/w/index.php?oldid=594787259 *Contributors*: Danim, GarbledLecture933, Gibbja, GoingBatty, Intgr, Nadja1960, Pnm, ReinhardK, Wilhelmina Will, 1 anonymous edits

Object Data Management Group *Source*: http://en.wikipedia.org/w/index.php?oldid=554442621 *Contributors*: AutumnSnow, Danim, Dawn Bard, Derbeth, DougBarry, Jerome Charles Potts, Johnny B, Kku, Lengyeltom, Lguzenda, LittleWink, Papajdo, Rzicari, 7 anonymous edits

List of object database management systems *Source*: http://en.wikipedia.org/w/index.php?oldid=571276555 *Contributors*: Alexandre.Morgaut, Bablind, Beland, Brunov, CWuestefeld, ChrisGualtieri, Cristiursachi, D aana, Dmytrob, Espresso999, FatalError, Ftiercel, George A. M., Harryboyles, Hu12, Hyspdrt, JGrosmann, JLaTondre, Jarble, Jerome Charles Potts, Kiore, Knowlengr, Lguzenda, MMSequeira, MacTed, Manfred-jeu, Matspca, Minas.w, Mogism, Nihiltres, Palosirkka, Pwaddles, R'n'B, Radu124, Rjolly, Sadads, Siaqodb, SpeoLeo, Spolnik, Svetoslav.Mateev, Talyian, Tekktura, The Founders Intent, Thegreeneman5, Torc2, Ubergeek3141, Uncommon Sense, Woohookitty, Xiloynaha, 112 anonymous edits

Comparison of object database management systems *Source*: http://en.wikipedia.org/w/index.php?oldid=598616490 *Contributors*: Alexandre.Morgaut, Bablind, Beland, Brunov, CWuestefeld, ChrisGualtieri, Cristiursachi, D aana, Dmytrob, Espresso999, FatalError, Ftiercel, George A. M., Harryboyles, Hu12, Hyspdrt, JGrosmann, JLaTondre, Jarble, Jerome Charles Potts, Kiore, Knowlengr, Lguzenda, MMSequeira, MacTed, Manfred-jeu, Matspca, Minas.w, Mogism, Nihiltres, Palosirkka, Pwaddles, R'n'B, Radu124, Rjolly, Sadads, Siaqodb, SpeoLeo, Spolnik, Svetoslav.Mateev, Talyian, Tekktura, The Founders Intent, Thegreeneman5, Torc2, Ubergeek3141, Uncommon Sense, Woohookitty, Xiloynaha, 112 anonymous edits

PostgreSQL *Source*: http://en.wikipedia.org/w/index.php?oldid=598017402 *Contributors*: (, 12.26.33.xxx, 217.98.151.xxx, A bit iffy, Abolen, AdultSwim, Advorak, Agentq314, Ahoerstemeier, Aldie, Alex.ryazantsev, Allencheung, Analoguedragon, Anas2048, Anastrophe, Andrew.george.hammond, Andrewpmk, Andy318, Andyjsmith, Angela, Angryxpeh, AnmaFinotera, Apostrophyx, Ardonik, Avé, AxelBoldt, B3t, BanzaiSi, Bearheart, Ben.c.roberts, Bender235, Benhoyt, Bernd vdB, Bevo, Bfcase, Billposer, Blindmatrix, Bmomjian, Bovineone, Bowman, Bramschoenmakers, BraneJ, Brian Gunderson, Burgundavia, Cander0000, Carey Evans, Cbraga, Ceejayoz, CesarB, Chealer, Chowbok, Chris the speller, ChrisMiddleton, Christopher Mahan, Cjkporter, Cleduc, Closedmouth, Codename Lisa, Comp.arch, Conversion script, Coolboy1234, Cosmicsoftceo, Craigkerstiens, Crashmatrix, Ctkeene, Cwitty, Cybercobra, Cynical, DStoykov, DagErlingSmørgrav, Dalahäst, Dandv, Dark ixion, Decibel, Deeahbz, Deflective, Deleteme42, Demonkoryu, Denys.Kravchenko, Dfetter, Digoal, Direvus, DisneyG, Distalzou, Dkusnetzky, Doradus, Droll, Dskoll, E rulez, EagleFan, Ebraminio, EdDavies, Edward, Eeekster, Eggyknap, Ehn, Eleven81, Eljope, Emmanuel JARRI, Erilong, FChurca, Faisal.akeel, FatalError, Fchoong, Filiprem, Fleminra, Frap, Frecklefoot, Freerangelibrarian, Fubar Obfusco, Fuzzie, Fvw, Gabriel Kielland, Gaius Cornelius, Gary, Geertivp, Gene s, Georgeryp, Ghen, Gilesmorant, Giraffedata, Glenn, Gmaxwell, Gorgot, Graham87, GreenReaper, Gregben, GregorB, GregorySmith, Grldedwrdbutler, Gronky, Gsherry, Gurch, Gwern, Gz33, HLHJ, Heron, Hu12, Ianb, Imroy, Intgr, JLaTondre, Jabberwoch, Jacobolus, JamesBWatson, Jamesday, Jandalhandler, Jay, Jcarle, Jeltz, Jerome Charles Potts, Jibun, Jnc, Johndburger, Johnsu01, Jojalozzo, Jons2006, Jonsafari, Joy, KAMiKAZOW, Kanzure, Karl-Henner, Keesiewonder, Kindofoctarine, Kl4m, Kl4m-AWB, Klenot, Kmacd, Krauss, KublaChaos, Kukushk, Kwamikagami, Kweetal, Kweetal nl, Kwiki, LX, Lantrix, Larsinio, Lasix, Laurenz albe, Lavagnino, LenzGr, Levin, Liftarn, Lmxspice, Lowellian, Luke Lonergan, Lulu of the Lotus-Eaters, Lvr, M, M4gnum0n, MaNeMeBasat, Majeru, Mangal ratna, Manop, Mark Renier, Marokwitz, Martinkunev, Marudubshinki, Marzalpac, Matchups, MattOates, Maury Markowitz, Mereman, MianZKhurrum, Mikecron, Mindmatrix, Minghong, MinorEdits, Mipadi, Misery, Mlibby, Mortense, Musashi1600, Mwtoews, Nanshu, Nate Silva, Neilc, Nevit, Niceguyedc, Nickdc, Nicolas Barbier, Nikai, Nikolas Stephan, Nixdorf, NoDepositNoReturn, Oberiko, Oska, Palmbeachguy, Palosirkka, Patrias, PaulMEdwards, Pauli133, Peaceray, Pedant17, Peyre, Pgan002, Piano non troppo, PieterDeBruijn, Pmronchi, Poli, Polyglot, Praemonitus, Prell, QuantumEleven, R'n'B, RandalSchwartz, Randolf Richardson, Ravenmewtwo, Raysonho, Redneb33, Reedy, Reetep, Revolus, Rich Farmbrough, Richmeister, Ringerc, Roadrunner, RonaldDuncan, Ronbtni, Room813, Rugh, SF007, Sbmeirow, SchuminWeb, Scottmacpherson, Sesse, Silvestre Zabala, Simple Bob, Slamb, Slipstream, Snarius, Specious, SteinbDJ, Stephen Bain, Stevemidgley, Stevenj, Stewartadcock, Streapadair, Sugarfish, Sven Klemm, Synergy, Tejano, The Anome, The Thing That Should Not Be, Thumperward, Tijfo098, Tim baroon, TimTay, Timsheridan, Tintinobelisk, Tkbwik, TobiasPersson, TomCerul, TommyG, Towsonu2003, Tpbradbury, Trevorbrooks, Troels Arvin, Ttiotsw, Turnstep, Usp, Utcursch, Varnav, Vincenzo.romano, W Nowicki, Web20boom, Weialawaga, Wernher, Wesley, Whale plane, Where, Wikante, WikiNickEN, William.temperley, Winston Chuen-Shih Yang, Wisq, Wolph, Wwwwolf, X-Fi6, Xzilla, Your Lord and Master, Yworo, Zenaan, Zero0w, Zeus, கி. கார்த்திகேயன், 528 anonymous edits

PL/pgSQL *Source*: http://en.wikipedia.org/w/index.php?oldid=597689117 *Contributors*: 5 albert square, AlexKarpman, Ambarshante, Andy Dingley, Avinashm, Boshomi, Brianray, ChrisGualtieri, Farmerboy99, Georgeryp, Intgr, John Vandenberg, Krauss, Martijn Hoekstra, Mrwojo, Mwtoews, Neilc, Nickdc, Northernhenge, Rjwilmsi, The RedBurn, Turnstep, 18 anonymous edits

PL/Perl *Source*: http://en.wikipedia.org/w/index.php?oldid=588743260 *Contributors*: Aardvark92, Chirlu, Gryllida, Hetar, Intgr, SamMason, Tevildo, Turnstep, 5 anonymous edits

JADE (programming language) *Source*: http://en.wikipedia.org/w/index.php?oldid=571428265 *Contributors*: Askjade, BSTRhino, Chopnz, Colonies Chris, Dinojc, Doug Bell, Drable, FatalError, Ghosttie, Grafen, Grantg1207, Greenrd, HappyCamper, Jerryobject, Jjdawson7, John Vandenberg, Jonathanischoice, Jwoodger, KymFarnik, Lod, Martarius, Michael Hardy, Mindspillage, Moloko5, Peter50, Ruud Koot, Sam Pointon, Sfan00 IMG, Sknightly, Snori, Talandor, The Thing That Should Not Be, Topbanana, Torc2, Triona, Victorwss, Yworo, 43 anonymous edits

ObjectDB *Source*: http://en.wikipedia.org/w/index.php?oldid=573945641 *Contributors*: Beland, Bileib, Ki2010, Kku, Magioladitis, Mortense, MrOllie, Palosirkka, 23 anonymous edits

Versant Object Database *Source*: http://en.wikipedia.org/w/index.php?oldid=589525753 *Contributors*: Bearcat, Bunnyhop11, Cander0000, Danim, Edward, Eugene-elgato, Germanviscuso, HardBoiledEggs, Ipsign, JLaTondre, Ketiltrout, Kku, Koumz, Mandarax, Mild Bill Hiccup, Nick Number, Pmitka, R'n'B, Smallman12q, W Nowicki, WikiDan61, 6 anonymous edits

Zope Object Database *Source*: http://en.wikipedia.org/w/index.php?oldid=596022021 *Contributors*: Alynna Kasmira, Bgeron, Brouhaha, Cander0000, Dawynn, Edward, FatalError, Favonian, Ffangs, Garyvdm, Gelma, Gf uip, Huskytreiber, JLaTondre, Jesse V., John Vandenberg, KevinBullock, Mimi.vx, Miohtama, Slinkp, Stevage, Themfromspace, Timosa, TonyLaPatate, Unforgettableid, Zagy, 39 anonymous edits

Strozzi NoSQL (RDBMS) *Source*: http://en.wikipedia.org/w/index.php?oldid=595003833 *Contributors*: Arjayay, Boshomi, Cybercobra, DGG, DavidCary, Gaius Cornelius, Hairy Dude, Mild Bill Hiccup, Morphh, OsamaBinLogin, Palosirkka, Peak, Quadibloc, Rfl, Sae1962, SamJohnston, 14 anonymous edits

NoSQL *Source*: http://en.wikipedia.org/w/index.php?oldid=599008852 *Contributors*: (:Julien:), Adtadt, Al3xpopescu, Alexandre.Morgaut, Alexrakia, AlisonW, Altered Walter, Amire80, Anastrophe, AndrewBass, Angry bee, Anilkumar1129, Anne.naimoli, Anoop K Nayak, Ansh.prat, Anujsahni, Argv0, Arjayay, Arto B, Asafdapper, Ashtango, Ashtango5, Atropos235, AxelBoldt, BD2412, Bbulkow, Bdijkstra, Bearcat, Beland, Benatkin, Benhoyt, Bhaskar, Billinghurst, Biofinderplus, Boshomi, Bovineone, Bramante, Brocsima, Bulwersator, CJGarner, CaptTofu, Ceefour, Cekli829, Charbelgereige, Chenopodiaceous, ChristianGruen, Ciges, Clemwang, Cloud-dev, Cnorvell, Cnwilliams, ColdShine, Coldacid, Colemala, Compfreak7, Corrector623, Craigbeveridge, Crosbiesmith, Crosstantine, Cybercobra, Cyril.wack, DBigXray, Dabron, DallasClarke, DamarisC, Dancrumb, DatabACE, DavidBourguignon, DavidSol, Davidhorman, Dawn Bard, Dericofilho, Dewritech, Dm, Dmccreary, Dmitri.grigoriev, Dredwolff, Drttm, Dshelby, Dstainer, Duncan, Ebalter, Eco schranzer, Edlich, Eedeebee, Ehn, Electricmuffin11, Eno, EricBloch, ErikHaugen, Ertugka, Euphoria, Excirial, Extrovrt101, F331491, Farvartish, Fiskbil, Fitzchak, Fmorstatter, FontOfSomeKnowledge, Fraktalek, FranzKraun, Frap, Freshnfruity, Frze, Furrykef, Fxsjy, Gaborcselle, Gadfium, Germanviscuso, Getmoreatp, GimliDotNet, Ginsuloft, Gkorland, GlobalsDB, GoingBatty, Gonim, Gorman, Gpierre, GraemeL, Griswolf, Gstein, Hairy Dude, Harpreet dandeon, Headbomb, Heelmijnlevenlang, HereToHelp, Hloeung, Hoelzro, Hu12, Inmortalnet, Intgr, Irmatov, Itamar.haber, JLaTondre, Jabawack81, Jandalhandler, Jasonhpang, Javalangstring, Jeffdexter77, Jerome Charles Potts, JnRouvignac, Jnaranjo86, JohnPritchard, Jonasfagundes, Joolean, Jottinger, Jrudisin, Jstplace, Jubalkessler, Justinsheehy, Kamaci, Kbrose, Kgfleischmann, Khiladi 2010, Ki2010, KiloByte, Kkbhumana, Kku, Knudmoeller, Koavf, Komap, Korrawit, LeeAMitchell, Leegee23, Legacypath, Leotohill, Lfstevens, Lguzenda, Linas, Lmxspice, Looris, Luebbert42, Luisramos22, Lyoshenka, MMSequeira, Ma7dy, Mabdul, MacTed, Magnuschr, ManikSurtani, Marasmusine, Mark Arsten, Matspca, Matt Heard, Mauls, Mauro Bieg, Maury Markowitz, Maykurutu, Mbarrenecheajr, Mbonaci, Mbroberg, McSly, Mesut.ayata, Mhegi, Miami33139, Michael Minh, Mikek999, Mitpradeep, Mjresin, Mongochang, Morphh, Mortense, MrOllie, MrWerewolf, Msalvadores, Mshefer, Mtrencseni, Mydoghasworms, Nanolat, Natan.puzis, Natishalom, Nawk, Nawroth, Netmesh, Neustradamus, Nick Number, Nileshbansal, Nosql.analyst, Ntoll, OmerMor, Omidnoorani, Orenfalkowitz, Ostrolphant, PatrickFisher, Pcap, Peak, Pereb, Peter Gulutzan, Petr Kopač, Phillips-Martin, Philu, Phoe6, Phoenix720, Phunehehe, Plustgarten, Pnm, Pointillist, Poohneat, Professionalsql, ProfessorBaltasar, QuiteUnusual, Qwertyus, R39132, RA0808, Rabihnassar, Raysonho, Razorflame, Really Enthusiastic, Rediosoft, Rfl, Robert1947, RobertG, Robhughadams, Ronz, Rossturk, Rpk512, Rtweed1955, Russss, Rzicari, Sae1962, Sagarjhobalia, SamJohnston, Sandy.toast, Sanspeur, Sasindar, ScottConroy, Sdrkyj, Sduplooy, Seancribbs, Seraphimblade, Shadowjams, Shepard, Shijucv, Smyth, Socialuser, Somewherepurple, Sorenriise, Sstrader, StanContributor, Stephen Bain, Stephen E Browne, Steve03Mills, Stevedekorte, Stevenguttman, Stimpy77, Strait, Sugamsha, Svesterli, Syaskin, TJRC, Tabletop, Tagishsimon, Techsaint, Tedder, Tgrall, The-verver, Theandrewdavis, Thegreeneman5, Thomas.uhl, ThomasMueller, Thumperward, ThurnerRupert, Thüringer, Timwi, Tobiasivarsson, Tomdo08, Trbdavies, Tshanky, Tsm32, Tsvljuchsh, Tuvrotya, Tylerskf, Ugurbost, Uhbif19, User db, Vegaswikian, Violaaa, Viper007Bond, Volt42, Voodootikigod, Vychtrle, Walter Görlitz, Wavelength, Webtrill, Weimanm, Whimsley, White gecko, Whoopm, William greenly, Winston Chuen-Shih Yang, Winterst, Woohookitty, Wyverald, Xtremejames183, YPavan, Yasinaktimur, Zapher67, Zaxius, Zond, Милан Јелисавчић, 687 anonymous edits

Graph database *Source*: http://en.wikipedia.org/w/index.php?oldid=599275372 *Contributors*: 0x24a537r9, 4368a, Agavenwurm, Aglnl, Ahzf, Ajmagnifico, AlanUS, Aldonline, Andrearatto, Binshao, Bolerio, Bunnyhop11, Cnorvell, Codename Lisa, Colinniu, Crcsmnky, DamarisC, Danim, Dreamingxk89, E40, Egbert J. van der Haring, Electro rick, Elykahn1, Espeed, Fceller, Ffangs, Fraktalek, Frap, Freshnfruity, Germanviscuso, Giftlite, J12t, JakobVoss, Jmesney, Jncraton, Jni, Jonik, Khayyatzy, Kiryakov ak, Ksinkar, Lambdazen, Lesser Cartographies, Lguzenda, Lillem4n, Luebbert42, Luisbargu, MacTed, Magnuschr, Materialscientist, Miami33139, Michael A. White, MoSarwat, Morphh, MuffledThud, Nawroth, Pelister, Pereb, Pholding, Pointillist, Pombredanne, Ppr15, Praveensripati, ProfessorBaltasar, RecaiAlkan, RichMorin, Sae1962, SamJohnston, Sberriman, Sbrunner, Shengqiyang, Starboy8, Stott.parker, Stybn, Syhuang1988, TTJDenman, Taneltammet, TempestSA, Tgrota, Thinxer, Thomas888b, Thoughtpuzzle, Tsm32, Tuhl, Vonmoltke, Wbeaureg, Yanivby, Zorglub27, 141 anonymous edits

DEX (Graph database) *Source*: http://en.wikipedia.org/w/index.php?oldid=595846779 *Contributors*: DamarisC, Danim, JLaTondre, Jncraton, SchreiberBike, 15 anonymous edits

Neo4j *Source*: http://en.wikipedia.org/w/index.php?oldid=592316184 *Contributors*: Alainr345, Amorpisseur, Bk1 168, Cybercobra, Danim, Dstainer, Fceller, Jonik, Mike Linksvayer, Morphh, MrOllie, Riorben, Sumail, Tobiasivarsson, Ynori7, 8 anonymous edits

Sones GraphDB *Source*: http://en.wikipedia.org/w/index.php?oldid=594019666 *Contributors*: Ahzf, Bearcat, Bunnyhop11, Danim, Edward, ErikHaugen, Frap, Jc37, Jni, Karellstemarie, Morphh, Squids and Chips, Thomas.uhl, Tuhl, 6 anonymous edits

Apache Cassandra *Source*: http://en.wikipedia.org/w/index.php?oldid=598998820 *Contributors*: Aardvark92, Acdx, Al3xpopescu, Alan Liefting, Andrewllavore, Andy Dingley, Anthony Appleyard, ArglebargleIV, Arthurjulian, Bcantoni, Bearcat, Biktora, Billmantisco, CJGarner, Cander0000, Cgraysontx, Chris Chittleborough, ChrisGualtieri, Ciges, Cinderblock63, Cleduc, Clydewylam, Cybercobra, Dabron, Dancraggs, Deansfa, Deineka, Dewritech, Diglio.simoni, Driftx, Drmies, Dstainer, Edward, Ehn, Elisiariocouto, Enchanter, Euphoria, FalconL, FrankTobia, Frap, Frecklefoot, Freenerd, Fyedernoggersnodden, Grafen, GreyTeardrop, Grossenhayn, Gstein, Hashar, Hoestmelankoli, JLaTondre, Jadave234, Jamesx12345, Jbryanscott, Jdorner, Jim1138, Jimtarber, Jmhodges, Jncraton, Jonathanbellis, Jweiss11, Khyryll, Kinglarvae, Kolyma, Krotty, Ksato9700, Kylemurph, Mark Arsten, Materialscientist, Maximgr, Mblumber, McSly, Mercurywoodrose, Mfiguiere, Midinastasurazz, Mmozum, Mortense, MrOllie, Msiebuhr, Mwtoews, Neilc, Nemnkim, Nmilford, OrangeDog, PatrickFisher, Pgan002, Pmiossec, Rich Farmbrough, RichardMills65, RobinUS2, Rollins83, Ronz, Runtime, S4saurabh, Sae1962, SamJohnston, Samarthgahire, Santiagobasulto, Sebastibe, Slebresne, Spiesche, Stather, Stefan, Stuartmccaul, Swaroopch, Tas50, Tdmackey, The-verver, TheJJJunk, Thumperward, Timendum, Timoey, Tommymorgan, Vanger13, Vegaswikian, Viocomnetworks, Wainstead, Wikiuser298, Woodjr, YPavan, Yadavjpr, YogiWanKenobi, 175 anonymous edits

Triplestore *Source*: http://en.wikipedia.org/w/index.php?oldid=598834182 *Contributors*: A3 nm, Andy Dingley, Ansell, Arto B, Beland, Bgwhite, Bomazi, BuZZdEE.BuzZ, Cnorvell, Cybercobra, DBooth, Danim, Dawn Bard, Djlambert, Dreamingxk89, Enric Naval, Erick.Antezana, FrankTobia, Good Olfactory, Inverse.chi, Iæfai, Jaideraf, James.adam.anderson, Jerryobject, Jreast, JustAnotherJoe, Kenkrupa, Kjetil, Linas, LostVagabond, MacTed, Marco74, Mdd, Mhgrove, Mrmariobriggs, Nasa-verve, Nhumfrey, Nicolas1981, Opoirel, PRB, Pholding, Pldms, ProfessorBaltasar, Pumba lt, R'n'B, Rzicari, Sbuxton, Scorlosquet, Scott, Sergioferlo, Sigma0 1, Simnia, Soumyasch, Spencerk, Two Bananas, Wbeaureg, White gecko, WiseWoman, Ysangkok, 66 anonymous edits

Keyspace (distributed data store) *Source*: http://en.wikipedia.org/w/index.php?oldid=592316499 *Contributors*: Bearcat, Jncraton, Khiladi 2010, MrOllie, PurplePiper, RHaworth, Rjwilmsi, Sae1962, Train2104, 1 anonymous edits

Super column *Source*: http://en.wikipedia.org/w/index.php?oldid=577671349 *Contributors*: Bearcat, Lenticel, Sae1962, 1 anonymous edits

BigTable *Source*: http://en.wikipedia.org/w/index.php?oldid=598386228 *Contributors*: A5b, Adpowers, Amux, Andy Dingley, ArlenCuss, Audriusa, Barauswald, Barry K. Nathan, Bearcat, Benstown, Bkonrad, Captin411, Cliffb, Cybercobra, DanielWaterworth, Danim, DataWraith, DeadEyeArrow, Dfarrell07, Discospinster, Drttm, Dstrube, Dwchin, EIFY, Elkman, Erik s paulson, Exerda, FinalRapture, Fintler, FreePeter3000, Georgewilliamherbert, Gnangarra, Goolasso, Gstein, Gwern, Harrigan, Hashar, Heroeswithmetaphors, Hmains, Imroy, Jason Quinn, JennyRad, Jeremykemp, Jeskeca, John Reed Riley, Khalid hassani, Larrymcp, Macrakis, Mark Renier, Markpeak, Marudubshinki, Masharabinovich, Mateuszb, Mgr493, Michael miceli, Moe Epsilon, Nakon, Nearfar, Nileshbansal, Ogendarf, Oo7565, Pgan002, Phoib, Pirroh, Pmsyyz, Quebec99, Raysonho, Rdquay, Replysixty, Rjwilmsi, Sae1962, Sounil, Tom Jenkins, Treekids, Vegaswikian, Wavelength, Whereiswally, Wordstext, Zoz, 131 anonymous edits

Flat file database *Source*: http://en.wikipedia.org/w/index.php?oldid=588904788 *Contributors*: 100110100, 2mcm, ANONYMOUS COWARD0xC0DE, Adrianwn, Aeliav, AeronBuchanan, AlanUS, AltiMario, Andrewsailer, Anonymous3190, Antonrojo, Apokrif, Arcann, ArnoldReinhold, Asfarer, BBuchbinder, Bayerischermann, Begewe, Beland, Brighterorange, Brookie, Brunnock, Byapparov, Capi crimm, Carmichael, Charles Matthews, Chaser, Christophe.billiottet, Clam0p, Computerboy0, CrazyArcher, Cyktsui, DARTH SIDIOUS 2, Damiankelly, Danim, Dannyg007, Darktemplar, Dr.scrath"n"sniff, Dreftymac, Enric Naval, Everyking, Ewlyahoocom, Excirial, Fadookie, Fieldday-sunday, Ghettoblaster, Grstain, Gzornenplatz, Habbie, HaywardRoy, ICberg7, J Di, Jackol, Jamelan, Jerome Charles Potts, JoanneB, Jonin69, Jpvinall, Kevins, Larsinio, Lee J Haywood, Leirith, Luisgarcc, Macaddct1984, Mandarax, Mandra Oleka, Mark Renier, Mark T, Master Lexx, McGeddon, Mdd, Mild Bill Hiccup, Mindmatrix, Minghong, MrOllie, Nbarth, Night Gyr, Notbyworks, OlEnglish, Omicronpersei8, Orpheus, Oxymoron83, Pantergraph, Phantomsteve, PhiLho, Philip Trueman, Piano non troppo, Psb777, Public Menace, Quiddity, Qxz, Rayngwf, RedWolf, Reedy, SDSWIKI, Sam Pointon, Sandcat01, SchnitzelMannGreek, Sedimin, Seidenstud, Shades1Of1Grey, Shanes, Skaryzgik, Slazenger, Sonett72, SteinerMW, Stephen Morley, Stolkin, Suffusion of Yellow, Tannin, Thorwald, Tobias Bergemann, TommyEd, Trolleymusic, WeißNix, Xiong, ZeroEgo, 215 anonymous edits

Terminfo *Source*: http://en.wikipedia.org/w/index.php?oldid=574894163 *Contributors*: Abdull, Beland, Danim, Gryllida, Hertzsprung, Hirsutism, Jfmantis, Jonesey95, Katharineamy, Martarius, MisterLambda, Msnicki, Pjoef, Pne, Rjwilmsi, Schily, Tassedethe, Tedickey, ZyMOS, 9 anonymous edits

Termcap *Source*: http://en.wikipedia.org/w/index.php?oldid=564560589 *Contributors*: Abdull, BiT, Cander0000, Chris the speller, Danim, Guy Harris, Htes.nehoc, Jeenuv, Jfmantis, Joedoedoe, Koko90, Mogigoma, Oswego Palomar, Pjoef, Tedickey, ZyMOS, Пика Пика, 7 anonymous edits

MultiValue *Source*: http://en.wikipedia.org/w/index.php?oldid=596667946 *Contributors*: Adtadt, AvicAWB, CharlesBarouch, Danim, DaveStoner, Dwolt, Gsallis, Hersfold, JIP, JackieBurhans, Jandalhandler, Jncraton, John.Whythe, Khazar2, Klenot, Legacypath, Mark Arsten, Michael Devore, Mpower1multivalue, Niceguyedc, Phillips-Martin, Seanp wiki, TJRC, Tom Morris, Vigneshgautam, Wagesj45, Wjhonson, 26 anonymous edits

OpenInsight *Source*: http://en.wikipedia.org/w/index.php?oldid=531785930 *Contributors*: Cander0000, CharlesBarouch, Danim, Kuru, Mild Bill Hiccup, Mruanejr, Paul Foxworthy, Plastikspork, R'n'B, RHaworth, RasterFaAye, Shadowjams, Softy, 5 anonymous edits

Document-oriented database *Source*: http://en.wikipedia.org/w/index.php?oldid=598589620 *Contributors*: Altered Walter, Antony.stubbs, Argv0, Arthena, Bablind, BrideOfKripkenstein, Bunnyhop11, Bxj, CJGarner, Carleas, Cbuccella, Cedar101, Chris Wood, ChristianGruen, Cobaltbluetony, Compfreak7, Crosstantine, Cybercobra, Danim, Danmcg.au, Dasfrosty, Dm, Dmccreary, Dodilp, Edward, Eedeebee, Ehn, Enric Naval, EricBloch, FatalError, Frap, FreeRangeFrog, Goldzahn, Gwicke, Hzguo, Imroy, Intgr, Iznogoud, JIP, Jerome Charles Potts, Jwoodger, Kingsleyj, Kirt, Lodrian, Luebbert42, Mark Arsten, Mark Renier, Mbroberg, Mdd, Mindmatrix, Mortense, Mqchen, MySchizoBuddy, Neitherk, Niceguyedc, Nikhil Umesh, Nwbeeson, Pcap, Philu, Plamoa, Plasma east, Pointillist, QuiteUnusual, R'n'B, Rachkovsky, RedWolf, Rediosoft, Refactored, Rfl, Rtweed1955, SDSWIKI, Spdegabrielle, Stuartyeates, Superjordo, Thorwald, Thumperward, Toutoune25, Tsm32, Webtrill, Woohookitty, 109 anonymous edits

MongoDB *Source*: http://en.wikipedia.org/w/index.php?oldid=598762423 *Contributors*: Adallas, Adm.Wiggin, Al3xpopescu, AlbinoChocobo, Alexey Feldgendler, Alfre2v, AlisonW, Alobodig, Arkroll, ArlenCuss, ArmitageAmy, Asafdapper, Avleenvig, Awesoham, AxelBoldt, Beaddy1238, Beland, Ben Ben, Ben b, Bernard.szlachta, Bhpdownloads, Blr21000, BookLubber, Bpatrick001, BrianOfRugby, CCC2012, CJGarner, Cababunga, Calorus, Cander0000, Chickencha, Chris Caven, ChrisGualtieri, Ciges, Cnevis, Coldacid, CommonsDelinker, Compfreak7, Crcsmnky, Ctrlfreak13, Dabron, Deineka, Difu Wu, Dm, DrJolo, Dstainer, Dustinrodrigues, Elibarzilay, Ewolff42, Fc07, Flying sheep, Fram, Francium1988, FrankTobia, Frap, Garyvdm, Gian-Pa, Gkorland, Grshiplett, HV, Hairy Dude, Heelmijnlevenlang, Hello71, Hu12, Iapain wiki, Jackson Peebles, Jakuza, JamesBWatson, Jameswahlin, Jamesx12345, Jan.hasller, Jandalhandler, Jason Quinn, Jasper Deng, Jayadevp13, Jim1138, Jinlye, Jocelynp85, Joefromrandb, John Bessa, Johnny99, Jojalozzo, Jonas AGX, Josephgrossberg, KDIsom, KennethJ, KickaXe, Kizar, Kocio, Kolbasz, Koper, LOL, Lan3y, Lateg, Lauciusa, Lmxspice, Lvmetrics, Mackrauss, Magioladitis, Mahbubur-r-aaman, Masterhomer, Materialscientist, Mauls, Mbferg, Mdirolf, Meandtheshell, Mechanical digger, Megaltoid, Mehdi2305, Mike2782, MikeWazowski, Mikewbaca, Mikeyv, Millermk, Mineralè, Miym, Mongochang, Mongodb, MrOllie, Mschneido, Mu Mind, Nabla, Nadavidson, Najeeb1010, Nforbes, Ninja987, Ochbad, Omnipaedista, PCJockey, Pengwynn, Perey, Peridon, Pgan002, Phoe6, Plaes, Quiark, Renatovitolo, Rfl, Rich Farmbrough, SBunce, SamJohnston, Serge925, Shaksharf, Shijucv, Shruti14, Shuitu, Skizzik, Solprovider, Solved009, Sriehl, Stennie, Supa Z, Svick, The Herald, The Illusive Man, The-verver, Theinfo, Thenaberry1, Thepaul0, Thorwald, Tobych, Topbanana, Tuxcantfly, Twimoki, Utku Tanrivere, Valio bg, Versus, ViperSnake151, Vittyvk, WDavis1911, Wavelength, William Avery, Wingedsubmariner, Wjhonson, Xujizhe, Yadavjpr, Ycallaf, Youngtwig, Zachlipton, Zyx, 361 anonymous edits

Temporal database *Source*: http://en.wikipedia.org/w/index.php?oldid=585847748 *Contributors*: 99 Willys on Wheels on the wall, 99 Willys on Wheels..., Alan Liefting, AndrewWarden, Apavlo, ArchiSchmedes, Arto B, Beland, Big Brother 1984, Bob-lunney, Charles Matthews, DanDare, Danim, Debresser, Dmeranda, Elwikipedista, Enric Naval, Favonian, Fetterless, Fred Bradstadt, Geoff918, GregorB, H@r@ld, Hatschibratschi, HePal, Helena.Palovská, JCLately, James086, Larsinio, Lugia2453, Mdd, Mortense, Nibblus, Perivamsi, Rathgemz, Ravik, Reedy, Rich257, RonaldKunenborg, Ronz, Roshan baladhanvi, Smcclos, Soap, Tijfo098, Topaz, Twilight (renamed), Vk2010, Who, Wikidrone, 104 anonymous edits

RRDtool *Source*: http://en.wikipedia.org/w/index.php?oldid=593152536 *Contributors*: A2Kafir, Abune, Aclister, Adamathefrog, Arturomartin, BD2412, Benzband, Cander0000, CanisRufus, Captain Conundrum, ChrisJ, CosineKitty, Danim, Dawynn, Dbrukman, Djohnson53, Dogboy756, Eaowens, Edaelon, FabriceB.75, Family Guy Guy, Fish and karate, Fr3d org, Frap, Free

Software Knight, GreyTeardrop, Hashar, Hosszuka, Hubby2debbie, IO Device, Jamesmorrison, Jarfil, Jmabel, Joy, Kenyon, Lzur, Maqs, Mortense, MrOllie, Mrzaius, Myke1829, NickCatal, Octotron, Ondra.lengal, Orenburg1, Ott, Petrb, Phil Boswell, Pmc, Pphaneuf, Prolixium, Ralphy, Roeme, SF007, Schneelocke, ScotXW, SeanFromIT, SiD3WiNDR, Sietse Snel, Snoyes, Ssinyagin, Stas2k, Stolsvik, TheParanoidOne, Tjwagner, Topperfalkon, Vborcan, Wereon, WpediaPaul, Xiong Chiamiov, Xoneca, Zaucker, 64 anonymous edits

Spatial database *Source*: http://en.wikipedia.org/w/index.php?oldid=596941809 *Contributors*: APh, Alksentrs, Amsguc, Antonrojo, Beland, Binksternet, Chire, Cnorvell, Danim, Devashish90, EastTN, Eastlaw, Edward, Ego White Tray, Fetterless, Forkandwait, Funandtrvl, Fæ, Gary a mason, Greghazel, Guaycuru, JamesBWatson, Jason Quinn, John of Reading, Ling.Nut, Lingliu07, Lordsatri, Malcolma, Miami33139, Michael Kümmling, MilerWhite, Mwtoews, Nawroth, Nicolas.fortin, Noonon, Panchitaville, Pascal666, Powerthirst123, Rahst12, Redlands, Robertvan1, ScubbX, Subu690, Sumail, TechnologyTrial, Thinking of England, Tmcw, Trublu, Tschirl, Vijaymgandhi, М И Ф, 47 anonymous edits

Spatial query *Source*: http://en.wikipedia.org/w/index.php?oldid=546573083 *Contributors*: Algomaster, Antonrojo, Chire, Int21h, MeekMark, MilerWhite, Pascal.Tesson, Redlands, Tony Hunter, Viriditas, 4 anonymous edits

Spatiotemporal database *Source*: http://en.wikipedia.org/w/index.php?oldid=578476920 *Contributors*: Arieba, Cchen0, ChrisGualtieri, Danim, Edward, H@r@ld, Ls1g, Niemeyerstein en, Odestiny, SyL64, Tijfo098, Wireless friend, XLerate, 1 anonymous edits

Object-based spatial database *Source*: http://en.wikipedia.org/w/index.php?oldid=549469805 *Contributors*: Avalon, Basio, Bohunk, Chphe, ChrisGualtieri, Colonies Chris, Danim, Darder, EastTN, Ego White Tray, Oh Snap, PBP, Sanspeur, 3 anonymous edits

Tuple-versioning *Source*: http://en.wikipedia.org/w/index.php?oldid=575297735 *Contributors*: Cconnett, Daveroberts, Fyrael, Joelm, Lugalde, Muntfish, R'n'B, SchreiberBike, Widefox, Zvika, 4 anonymous edits

Valid time *Source*: http://en.wikipedia.org/w/index.php?oldid=529976242 *Contributors*: Charles Matthews, Egpetersen, Nibblus, Pegship, RussBlau, Smuckers, TechPurism, Vk2010, 7 anonymous edits

Transaction time *Source*: http://en.wikipedia.org/w/index.php?oldid=575194120 *Contributors*: Charles Matthews, Coccyx Bloccyx, Gurchzilla, JCLately, Jonnabuz, Logan, MBParker, Nibblus, Pegship, TechPurism, Vk2010, 2 anonymous edits

Geospatial metadata *Source*: http://en.wikipedia.org/w/index.php?oldid=582785240 *Contributors*: Addypope, Burkestar, Dark Silver Crow, Dr Shorthair, Flietz, Jacobhsieh, LilHelpa, Mcechini, Neogeography, PeterParslow, Queerbubbles, Rich Farmbrough, The Thing That Should Not Be, Ticheler, Tom Worthington, Tony1212, Vrautenbach, Winterst, 22 anonymous edits

Geographical database *Source*: http://en.wikipedia.org/w/index.php?oldid=535830679 *Contributors*: Bearcat, Comupedi, Danim, Elphion, Katharineamy, The Aviv

Time series database *Source*: http://en.wikipedia.org/w/index.php?oldid=594985405 *Contributors*: Aboriginal Noise, Altered Walter, Andrewcronk, Anjiro, Barticus88, Boxplot, Bsautner, CandaceChang, CassandraToday, Catalystcoder, Ccox5, Cgilchristlim, Clicketyclack, Cowznofski, Danim, Davagh, Derek a adams, Dgoldsmith2, Doctorbigtime, Dwinson, Hmains, Hobsonlane, John of Reading, Justindelay, Jwoodger, Kevin Gorman, Kevin Ryde, Kuru, Lptolik, Manolamancha, MarsRover, Melcombe, Michael Hardy, Mild Bill Hiccup, Nicholsr, Orenfalkowitz, PigFlu Oink, RONNCC, Ricky81682, Ridernyc, RomanSpa, SimonG, Simonemainardi, Slowking Man, 42 anonymous edits

Operational historian *Source*: http://en.wikipedia.org/w/index.php?oldid=597736630 *Contributors*: 859austin, Adrielmichaud, Amyeomans, Arunram, Bearcat, Beland, Boxplot, Celtechm, ChrisGualtieri, Cyfraw, Dawynn, DoctorKubla, Dotab, Jay8bird, Jstplace, Malcolma, My Gussie, Ngpd, Notanotherindustrialblog, RadioFan, Rich Farmbrough, SgrinerHoneywell, ToDeBe, 23 anonymous edits

Real-time database *Source*: http://en.wikipedia.org/w/index.php?oldid=590966493 *Contributors*: Bacchiad, Beland, Danim, Dhartung, Egpetersen, Fabrictramp, Fholahan, Frap, Janm67, Jaydubya93, L Kensington, Malcolma, Marcuscalabresus, Mark Renier, Mindmatrix, Owl3638, Paxse, RJFJR, SmackoVector, Will Pittenger, Wknight94, 29 anonymous edits

In-memory database *Source*: http://en.wikipedia.org/w/index.php?oldid=599284083 *Contributors*: 28nebraska, 4th-otaku, Abernie182, Agentq314, AgniKalpa, AlexandriNo, Altered Walter, ArmitageAmy, Bablind, Bac 66, Beland, Betis76, Broeni, Cander0000, Cheolsoo, Chris the speller, Christophe.billiottet, Cnanney, Codename Lisa, ColinCarberry, Compfreak7, Computermemorynerd, Cpnatwork, Crysb, Darisoft, Darren Duncan, Dbates1999, Dewritech, Dnmurphy, Drachmae, Farialima, Fholahan, Fschupp, Gilad.maayan, Giloki, Greenrd, GregorB, Hcobb, Ingenthr, Intgr, Ittia, Jack Wester, Jatujak, Jeff00seattle, Jerryobject, Jfraima, Jncraton, Jnmoyne, Joel.s.white, Jorjani, Jwoodger, Khazar2, Kunalnitin, Kvramarajus, Longlongago7777, Luxxor, MainFrame, Malcolma, Malikarjun.choudary, Mangotron, Mark Renier, Memorytechguru, Mphnyc, MrOllie, Mstrey, Natishalom, Ne0Freedom, Nealmcb, Neurocod, Ngpd, Oicumayberight, Omnibus3, Os81paul, Pbourke, Pedjaradenkovic, Praba tuty, Ragintec, Rahulsw, RezaeiMostafa, Rfl, Rich Farmbrough, Saldinatale, Sheaffej, Sjkim1971, Stuartyeates, T-002, Tabletop, Techietown, Ted nw, Teqhed, TheParanoidOne, VladSek, Webtrill, Xtremejames183, Yafoor, 127 anonymous edits

Probabilistic database *Source*: http://en.wikipedia.org/w/index.php?oldid=579084692 *Contributors*: Adelpine, Beland, Cdrdata, Danim, Databong, Edward, Pgoyal13, Ssickels, 2 anonymous edits

Deductive database *Source*: http://en.wikipedia.org/w/index.php?oldid=578935761 *Contributors*: Bearcat, Carbo1200, Cdrdata, Cic, Dfinch, Enric Naval, Everyking, GeorgeBills, GoingBatty, Greenrd, Hetar, Hydrogen Iodide, Kku, Larsinio, Leegee23, MCiura, Mark Renier, Mdd, Neurolysis, Reedy, Rjwilmsi, Soulmerge, Steinsky, Trappist the monk, Vzach, Xezbeth, 18 anonymous edits

Deductive language *Source*: http://en.wikipedia.org/w/index.php?oldid=532118312 *Contributors*: Bearcat, Beland, Metacosm, PiMaster3, Some Wiki Editor, The monkeyhate

Mobile database *Source*: http://en.wikipedia.org/w/index.php?oldid=598628882 *Contributors*: AK456, Anaxial, BD2412, Beland, Bunnyhop11, Cgtdk, Chris the speller, Cureden, Czarkoff, DGG, Danim, Dinarphatak, Duyongwen, Fabrictramp, Heqs, Herbythyme, Igloobone, IncrediGeek, J.delanoy, Jeffq, Jessica.couchbase, Khflottorp, LohithMJajee, Lpetrazickis, MBisanz, MobiForms, MrOllie, Paxse, PhilKnight, Ramu50, Raysonho, Rwalker, Sae1962, Secret (renamed), Soalib, Stephan Leeds, Tdietsche, Timmillwood, Toohool, Wavyriver, Wbm1058, 49 anonymous edits

Standard data model *Source*: http://en.wikipedia.org/w/index.php?oldid=546468569 *Contributors*: Ground Zero, IanDBailey, KarenData, Kdc3, KeyStroke, Kuru, Mdd, Patrick, Paul A, Srikant.sharma, Titoxd, Tom Morris, TubularWorld, 9 anonymous edits

Suppliers and Parts database *Source*: http://en.wikipedia.org/w/index.php?oldid=517396769 *Contributors*: Bjelleklang, CBDunkerson, Danim, Drumroll99, Marcelocantos, Melaen, Natalya, ProcureNET, Retired username, Rjwilmsi, SabbeRubbish, That Guy, From That Show!, Xyzzyplugh, 2 anonymous edits

Internet Movie Database *Source*: http://en.wikipedia.org/w/index.php?oldid=598803338 *Contributors*: -ACL-, .I.M.D.b.s.u.c.k.s.I.M.D.b.s.u.c.k.s.I.M.D.b.s.u.c.k.s., 0zero9nine, 12 Noon, 1234r00t, 16@r, 180day, 1966batfan, 1c^ru5, 23skidoo, 45andLP, A8UDI, ABF, ACT1, AHMartin, AKS.9955, APPER, ARC Gritt, Aaron Bowen, Aaron Brenneman, Aaron Schulz, Abcms, Access Denied, Ace Class Shadow, Adrian J. Hunter, Afabbro, Aff123a, Afterwriting, Aghna11, Agüeybaná, Ahawowow, Ahmad.ghamdi.24, Ahoerstemeier, Aishfan, Ajenkins, Akumatatsu61, Al E., Alai, AlbertSM, Alerante, Alex Middleton, Alex43223, Alexdw, Alexf, Alison, Alksub, Alsd2, Altrock78, Amalthea, American Eagle, AmiDaniel, Amxitsa, Anansi00, Anaxial, Anclation, Andre Engels, AndreNatas, Andrevan, Andrew Levine, Andrewman327, Anetode, Angel David, Angel2001, AngelOfMusic, Angela, Anonymous Cow, Another n00b, Anteriorlobe, Antonio La Torre, Antonycarthy, Antrikshy, Antrophica, Aquilosion, Aralvarez, Aristophanes68, Armandtanzarian, Arteyu, Ashrawi, Aspensti, Asteriks, Astronautics, Atenea26, Aude, Auric, Austin512, Austriacus, Ava9494, AxelBoldt, BRG, Bacteria, Badman89, Bagatelle, Barefootguru, Beeblebrox, Beetstra, Beland, Bellhalla, Benandorsqueaks, Bender235, Betacommand, Betterusername, Beyond My Ken, BiddyLady, BigEast55, Bill.matthews, Bilsonius, Bisbis, Bizso, Blake-, Blanchardb, Blaxthos, BlueAmethyst, Bluemask, Bluexar, Bobbyandbeans, Bobo192, Bodnotbod, Bongwarrior, Bony devil, Bourbons3, Bovineboy2008, Br'er Rabbit, Brianga, Brighterorange, Brtkrbzhnv, Bswee, Bubbababubba68, Bumm13, Bushcarrot, Butwhatdoiknow, C.Fred, CaffeinAddict, CakeTerror, Cambam29, CameoAppearance, Can't sleep, clown will eat me, CanadianLinuxUser, Canderson7, CanisRufus, Canterbury Tail, Canuckian89, CapitalR, Capricorn42, CaptainVindaloo, Carbonite, Carbonix, Card, CardinalDan, Cashewbrick, Casius, Casper3142, Catandop7769, Causa sui, Cbbkr, Cbh, Cbm, Cbrown1023, Ccmmyy25, Cdcon, Cdinesh, Cenarium, Cescoby, Chaosduck, Charlene.fic, Charlesviper, Chasingamy, Chatfecter, Cheeselog, Chicago god, Chihuahua0, Chin Man2, Chris G, Chris the speller, ChrisGualtieri, Chuunen Baka, Cigammagicwizard, CityFeedback, CityOfSilver, ClansOfIntrigue, Cliffb, Cm619, Cms, Cnota, Cntras, Coderzombie, Coffeepusher, Cohey, ComedyGuru, Commander Shepard, Connormah, Conversion script, Cookieman123, Cooling tower train walrus, Corpx, Counny, Courcelles, Crashlane, Crazyforreading, Crazymonkey1123, Crimson3000, CrocodileMile, Crohnie, Cromwellt, CrunchySkies, Cryptic, Csanchez, Cvnsfgmg, Cybercobra, Cyy006, DGJM, DJ-Joker16, DRAGON BOOSTER, DSRH, DVdm, Daedalus969, Dancergirl92, Dangherous, Daniel2424, Danntm, Dannybu2001, Dannymears, Danziger100, DarTar, Darguz Parsilvan, DarkFalls, Darrell Greenwood, Dasani, Dave.seidner, DavidLeighEllis, Davidbspalding, Dawn Bard, Dcoetzee, De Katten, DeadEyeArrow, Deansfa, DearPrudence, DeathDealer92, Dekisugi, Deli nk, Deltabeignet, Denisutku, Dennise Delamo, Dep1353, DerHexer, DesdinovaUK, Desk Jockey, Dfoofnik, Dhp1080, Diamond2, Dinesh tt, Discospinster, Dizagaox, Djquackquack39, Dkstamps, Dlabtot, Dnmakid, Dobrevano, Doctor yellow, DoctorHver, Doczilla, Dogman15, Doniago, Doodledoo, DoubleCross, Doug butler, Dp462090, Dpr, Dr. Blofeld, Dream out loud, Drmies, Dtcdthingy, Dtwong, Dubmill, Dude865, Dudestoduds, Dynex436, Dysepsion, E23, EamonnPKeane, Ebbpeg, Editor182, EditorInTheRye, Edlitz36, Edward, Egrove system, El C, El Slameron, ElSaxo, ElinorD, Eliz81, Eljayess, Elvey, Emerson7, Emperorbma, Ennen, Eolas38, Epbr123, Eric-Wester, Ericg33, Erik, Erkman27, Erpert, Esn, Esprit15d, Essjay, Eurofan2005, Evewalker, Evil saltine, Expert China, ExpressingYourself, Eyreland, Ezeu, Ezzex, FCSundae, FF2010, FOX 52, Falcon8765, Fallout boy, Familyguy1234567890, Familyguy1234567890987654321, Fantastic4boy, Fearfulsymmetry, Fieldday-sunday, Filmfancanon, Finlay McWalter, Firebat08, Fishhead2100, Fitch, FlickGeek,

Floatjon, FlorianB, Fordmadoxfraud, Forest-rhino, Fractions, FrankCostanza, Freakofnurture, Frecklefoot, FredR, Fredrik, Freshacconci, Froid, Frostedflame, Func, GBMumby, GPHemsley, GSK, Gardar Rurak, Garo, Gary, Garyzx, Garzo, Gdo01, Generatingfuninfo, GeometryJim, GeorgeLouis, Gerrit, Gertie1999, Getcrunk, Gharonda, Ghosts&empties, Gilliam, Ginkgo100, Ginsengbomb, Ginsuloft, Gioto, GirlyGirl92, Globalsolidarity, Gmudge, Gobonobo, Gogo Dodo, GoldenGoose100, Goldentony111, Good Olfactory, Gourav2711, Graham87, GrahamN, Granpuff, Gratexxxxxxx, Green caterpillar, Greenmonkeys, GregorB, Grendles modor, Grstain, Guanaco, Gurch, Guy M, Guy1423, Gwalla, Gwernol, HaeB, Halibut Thyme, Halsteadk, Hamiltondaniel, Hanif1954, HappyArtichoke, Harro5, Harry491, Hazhk, Heavenhelllord, Herpof, Hexibar, Himasha Wijesurendra, Hiplayboy, Hobbes Goodyear, Holon, Hongooi, Hotdoglives, Howcheng, Hqb, Hungdaddy69, Huseyx2, Husond, I Love Bananas, IAM177, ID4EVER, Iain99, Ianblair23, Ibemonty2000, Idleguy, Idont Havaname, Igordebraga, Iluvcinema, Ilyse Kazar, Imnotminkus, In Transit, Informationmedia21, Iridescent, Irishguy, Irk, Irrypride, Isak, Isfisk, J M Rice, J.delanoy, J3PPiSH, JCarriker, JDoorjam, JHVipond, JHunterJ, JSpung, JaGa, Jack Greenmaven, Jackie Stuntmaster, Jahiegel, Jaianu, Jambo247, James084, JamesAM, Jameshfisher, Jamesx12345, Jampilot, Janeyisfierce, Jared Hunt, Jarhed, Jarjarbinks10, Jason Recliner, Esq., Jasper Deng, Jatebirds, Jaunty mellifluous, Jchiztheniz, Jebba, Jennica, JesseRafe, Jiiimbooh, Jim1138, Jimthing, Jivecat, Jivee Blau, Jmchuff, Jni, Jnorton7558, Jobbus McKnockey, Johaen, John Q. Architext, John254, JohnArmagh, JohnClarknew, Johnpseudo, Jojhutton, Jon hollingsworth, JonErber, Jooler, JordanMussi, Jorgebarrios, Jorgenev, Joseph Solis in Australia, Joshuapaquin, Jqt, Juliancolton, Julius knipl, Jumbolino, Jumping cheese, Justme89, Jvcdude, Jw21, Jweiss11, K igor k, KJBracey, Kap42, Karaboom, Katalaveno, Katimawan2005, Katydidit, Kaygees, Keilana, Kelly Martin, Kenta, Kenyasong, Kevalchacha, Kevin Saff, Keysersoze123456789, Khalid hassani, Khudaheat, Kidlittle, Killbill11, King Curtis Gooden, King Lopez, Kingturtle, Kitch, Kjammer, KnowledgeOfSelf, Koavf, Konniret3, Koyaanis Qatsi, Kpwa gok, Krakatoa, Kramertron, Krasnoludek, Krich, Kristeneaugusta, Kristenq, Kubrick, Kude90, Kvaks, Kyrka, L337dexter, LarRan, Lastorset, Lava20, Leavinthegame, LedgendGamer, Lee J Haywood, LeonWhite, Lessogg, Leszek Jańczuk, Lightmouse, Lignomontanus, Litefantastic, Little Professor, Lmoja, Lochaber, Lokpest, Lolababy, Lomedae, Lone boatman, Longhair, Lord Kestrel, Lord Porchington, Lord of Night, Lordmarchmain, Lordz, Lorentey, Loriannhart, Lotje, Lowellian, LucasVB, Lugnuts, LukeSurl, LukeTheSpook, Lupin, Lving 42day, Lwc, M.O.X, M1ss1ontomars2k4, MER-C, MMAfan2007, MZMcBride, MacTire02, Macca7174, Mackensen, MadenssContinued, Majora4, Manop, Manvi111, Manxwoman, Manymerrymenmakingmuchmoneyinthemonthof May, MarE6, Marcika, Mariomedici707, MarisaGabriella, Mark Arsten, Mark Schierbecker, Mark512, Martarius, Martin villafuerte85, MartinDK, MartinRe, Marx01, Master Bratac, Master of Puppets, Master shepherd, Materialscientist, MattGWeb, Matteh, Matthew1990, Mattnad, Matty j, Mauls, Mazhar Khan Ikram, McGeddon, Mcfly85, Mcsee, Mdd, Mdebets, Mdraffi, Meno25, Mervyn, Metsfreak2121, Mgc0wiki, Mhiji, Michael Devore, Michael Hardy, Micru, Mike Payne, Minaker, Mincus, Minimac, MisfitToys, Mister X, Misterkillboy, Mithent, Mjhutchison, Mkdw, Mkeranat, Moby008, Modemac, Modify, Mogism, Moleculo, Mona, Moncrief, Mongoletsi, Monkeyzpop, Monty845, Moproducer, Movieguru2006, Mpeisenbr, Mpidge, Mr Beale, MrDolomite, Mrmagoo2006, Mrt3366, Muro de Aguas, Mushroom, Mustafaalagoz, Mutant-Lep, NIIRS zero, Nandesuka, Narssarssuaq, Nasmformyzombie, Nataev, Nemo bis, Nescio, Neurillon, New Bully on the Block, Nexusstar, Ngpd, Nhlarry, NickBush24, NickelShoe, Nightscream, Niteowlneils, Noddy1000, NorrYtt, NorthernThunder, Northgrove, NotHugo, Nstarks, Nurg, ONEder Boy, Obli, Odinmetatech, Ohconfucius, OldakQuill, Oliver Pereira, Oliverdl, Olivier, Omicronpersei8, Only, Ontarioboy, Opelio, Opertinicy, Orange Suede Sofa, Oren0, Orlando Rivenstone, Osomec, Otolemur crassicaudatus, Owen, Oxymoron83, PJ Pete, PM800, Pai Walisongo, PainMan, Paine Ellsworth, Pakmahfil, Papercuts Hurt :(, Papushin, Parsifal, Pathollywood, PatriceNeff, Patrick, Pats1, Paul A, Paul1337, Paulusmaximus1983, Pavel Vozenilek, Pawlickir, Petervanwesterloo, Pgk, Phifly, Phil Boswell, Philip Trueman, Phoenixrod, Pie Man 360, Pigman, Pigsonthewing, Pilotguy, Pince Nez, Pinethicket, Poker Player, Poor Yorick, Prateep, Pratikdave, Pratyush, Pruneau, Psantora, Pseudosocrates, Puckly, Punkymonkey987, Puregenious101, Pym98, Qmdb, Quatloo, Qwekiop147, R craycroft, R013, RJ4, RMHED, Racepacket, Ragityman, Rake, Ran4, RandalSchwartz, Raven4x4x, Raymond Cruise, Rdsmith4, Rebochan, Redrocket, Reeveorama, Reflex Reaction, Reisio, Res2216firestar, RevoSE9, ReyBrujo, Rfbarrington, Rich Farmbrough, Richard Arthur Norton (1958-), Ridge Runner, Rikkiprince, Riverstogo, Rjwilmsi, Rlevse, Rmcook4, Roadsworth, Roast chicken, RobbyyOz, Robchurch, Robert 47, Robertgreer, RobyWayne, Rock4arolla, RockMFR, Rocksanddirt, Roeeyaron, RogoPD, RogueJedi86, RolaPL, Roland Kaufmann, Ronhjones, Roux, Rs2360, Rspade, Ruiner2001, Rutherfordjigsaw, RxS, Ryan2807, S h i v a (Visnu), Sacularamacal13, Sajid130, Salamurai, Sam, Samloveman, Samuraimanzero, Sarath9, Sasajid, SatCam, Saul-2, ScaldingHotSoup, Scalhotrod, Schnob Reider, Scotsman1979, Scott Sanchez, ScottSummers84, Screamerific, Scs, Sd-100, SeanMack, Seaphoto, Secfan, Secret of success, Ser Amantio di Nicolao, Seregain, Seth Ilys, Shadowlynk, Shafi4uf, Shakir ali khan, Shanes, Shannernanner, Sharcho, Shark96z, Sheldrake, Shimaspawn, Shishkabob202, Shrigley, ShurTape, Silver Edge, SimonP, SineWave, Sj, Sjakkalle, Skidrow, SkinsFan123, Skizzik, Skomorokh, Skunkboy74, Skyhawk0, Slightsmile, Snaxe920, Snowolf, Soaploaf, Softlavender, Some guy, SonicRacer-MEC, SoundLikeYaFromLANDAN, Space-Wolf, Spartaz, Spearhead, Spicytamale, Spirifil, Splash, SpuriousQ, Sriharsh1234, Sspalfilter, StarbriteQueen, Steeev, Steel1943, Steininnn, Stemoc, Stenun, Stephenb, SteveCrook, Steven Zhang, Stevenrasnick, Stevevance, Stimpy9337, Stormwhisper, StuffOfInterest, Sturmovik, SubSeven, SummerPhD, SuperDude115, Suriel1981, Suzukisue, Svick, SweetNeo85, Switchsonic, Sylocat, T-Dog2000, TEG24601, TJRC, TMC1982, TORR, Tabascoman77, TaerkastUA, Tawker, Tbhotch, Tburroni243, Techsmith, Terminal Freeze, Terriblefish, ThaddeusB, Thane, Thclm, The Ace07, The Giant Puffin, The Halo, The Luggage, The Rambling Man, The Singing Badger, The undertow, TheRealFennShysa, TheSeer, Thehornet, Thehottytoddyhippie, Themovieaddict, Themovieshome, Theoneintraining, Thewolfchild, Thibbs, Thinkgeek, Thisyearsguy, Thivierr, Thomas d stewart, Thorpe, Thorwald, Thread the needle, Thumperward, Thundersnow, Tide rolls, Tim bates, Tim1357, TimTay, Timir2, Tkgd2007, Tktktk, Tlesher, Tobias Hoevekamp, Tom.k, TomPreuss, Tony1, Tonzo, Torchwoodwho, Tpbradbury, Travelbird, Tree Biting Conspiracy, Tregoweth, Trevor MacInnis, Trexpro, Trivialist, Trygfe, Turgan, UB65, Uannis, Uknowwhatisup1, Uncle Dick, UnitedStatesian, Unschool, VG Cats Tipe 2, Vald, Valetude, Vanished user vjhsduheuiui4t5hjri, Vegaswikian, Velella, Venopher, Versageek, Vetes, Vicarious, Vicenarian, Vidor, Viewdrix, ViperSnake151, VirtualDelight, Visor, Vorash2000, Wannabemodel, Ward3001, Wayland, WayneSMT, Waz, Wdfarmer, Wdyoung, Wezzo, WhisperToMe, Whisternefet, White 720, Widefox, Widr, Wifione, Wiki Raja, Wiki alf, WikiMonster, Wikid77, Wikidemon, Wikieditoroftoday, Wikievil666, Wikiloop, Wikipelli, Wimt, Wishmaniac, Woohookitty, Wootonius, Worm That Turned, Wtmitchell, Wuffyz, Wurdnurd, Wwoods, Wygk, X!, XLerate, XXdakill3rzzXx, Xezbeth, Xinoph, Xyzzyplugh, YUL89YYZ, Yamamoto Ichiro, Yanksox, Yannick1995, Yashthe8thwonder, YingYang2, Yintan, Ylem, Yonatan, Yorkmba99, Yunshui, Zachkudrna18@yahoo.com, ZackDouglas, ZappaOMati, Zen-master, Zepheus, Zephyr89, ZeroJanvier, Zoe, Zonder, Zondor, ZorroIII, Zotdragon, Zzuuzz, Александър, Саша Стефановић, Шизомби, Ὁ οἶστρος, 1739 anonymous edits

YAGO (ontology) *Source*: http://en.wikipedia.org/w/index.php?oldid=496856864 *Contributors*: A3 nm, Aellenhicks, Alan Liefting, Baojie, Beland, JosebaAbaitua, LilHelpa, LjL, NikhilKini, Pmj005, Spencerk, 4 anonymous edits

World Wide Molecular Matrix *Source*: http://en.wikipedia.org/w/index.php?oldid=585880147 *Contributors*: 1ForTheMoney, Auric, Bsherr, Download, J04n, Kean1234, Lawsonstu, Michaeldsuarez, Paine Ellsworth, Petermr, S.K., Tomos ANTIGUA Tomos, Wavelength, 3 anonymous edits

Voter database *Source*: http://en.wikipedia.org/w/index.php?oldid=591529279 *Contributors*: Alansing, Antonine, AvicAWB, Biebersbro3, Btball, Ceyockey, Chadlupkes, Danim, Deipnosophista, Electiontechnology, ErinRaub, Evercat, Evil saltine, Flamingspinach, Fuhghettaboutit, Garyrust, Ground Zero, Hughstimson, Idp, Inks.LWC, JRPG, JonHarder, Ketiltrout, Lankiveil, Longhair, MONGO, Marcika, Materialscientist, Mediamezcla, MisterJayEm, Ottawan, Radio15dude, Robofish, SanDiegoPolitico, SimonP, Steviecorey, ThatPhatMan, TheParanoidOne, Welsh, 84 anonymous edits

VIOLIN vaccine database *Source*: http://en.wikipedia.org/w/index.php?oldid=595260990 *Contributors*: Bearcat, Danim, Fallschirmjäger, Katharineamy, Maralia, Rjwilmsi, Yongqunh

Census of Governments *Source*: http://en.wikipedia.org/w/index.php?oldid=531338611 *Contributors*: Blargh29, Closeapple, Cybercobra, Dthomsen8, Gilliam, K4kant, Reywas92, Tom, 1 anonymous edits

Management information base *Source*: http://en.wikipedia.org/w/index.php?oldid=599149214 *Contributors*: 4johnny, 802geek, BD2412, Bomazi, Bruce1ee, Cander0000, Chiz@sympatico.ca, ChrisGualtieri, Coffee, David Legrand, Donald j axel, EagleOne, Escape Orbit, Fleasoft, GabeIglesia, Hardaker, Intgr, Jamelan, Jobymanuel, KelleyCook, Kenyon, KnightRider, Kocio, Kvng, Leandrod, Maiios, Mancini, Michael Hardy, Mmernex, Net, Octavio, Ouatic-7, Pathoschild, Pedant17, Ray Dassen, Razor2988, Rob.desbois, RockMFR, Satellizer, Schoenw, Troels Arvin, UncleBubba, Vlad, WhaleyTim, Wrs1864, Yaronf, Yurik, Zoicon5, 101 anonymous edits

Planetary Data System *Source*: http://en.wikipedia.org/w/index.php?oldid=427355825 *Contributors*: Aducore, Gary, GeorgeLouis, Guillaume2303, LFSBN, MER-C, ONUnicorn, Ohms law, PlanetStar, Rowenrye, Tbhotch, Tking999, Tuvas, Vanderdecken, Viriditas, Wknopf, 22 anonymous edits

Parameter Value Language *Source*: http://en.wikipedia.org/w/index.php?oldid=291732951 *Contributors*: Foxunix, GeorgeLouis, Michael Hardy

National Data Repository *Source*: http://en.wikipedia.org/w/index.php?oldid=591689909 *Contributors*: Beagel, Chris4272, Cmdr Scolan, Cristixav, EnerGistics2012, Erinkadme, Gmonachese, Paul A, Rdgarlick, Rich Farmbrough, Sun Creator, Tinton5, 43 anonymous edits

Data center *Source*: http://en.wikipedia.org/w/index.php?oldid=599112498 *Contributors*: 0goodiegoodie0, 14vivin, 1SockChuck, A.k.a., AMU10, Afaber012, Ahadenfeldt, Akolyth, Aldie, Alistair1978, Ambbes5, Andrewpmk, Aneah, Anthony mitchell, Apparition11, Arch dude, Autumnalmonk, Avoided, Awotter, BBird, BenignEditor, BerlinSight, Bgleatherhead, Bikepunk2, Bill Slawski, Blosoya, Blvrao, Bovineone, CLW, Calvin 1998, CanisRufus, Carribeiro, Cate0012, CeciliaPang, Cenarium, CesarB, Chandlermbing, Checkingfax, Cheese Sandwich, Chowbok, Chris the speller, ChrisGualtieri, Ckatz, CoLocate, Copeland.James.H, Craigwb, CtrlS Datacenter, DARTH SIDIOUS 2, Dan Harkless, Dan100, DanKidger, Darkx1337, Darlock, Dashapiro, DataCenterProfessionals, DatacenterEd, Datacenterfacilities, Datacenterking, Dawnseeker2000, Ddxc, Dekisugi, Derekcslater, Designbystructure, Dgtsyb, Directnet, Dogonsi, Donated, Dosttiey, Dsarokin, Dubious Irony, Dwvisser, Echoray, Eddpayne, Edward1301, Ehn, El Pantera, Epastore, Epicgenius, Esds - Data Center, Eusbls, Everything counts, Ewlyahoocom, EwokiWiki, Fastilysock, FayssalF, Femto, Fred Bauder, G33k84, Gaius Cornelius, Gcsfl, GeoffO, Gidonb, GingerGeek, Giraffedata, Gmaxwell, Go82102, Gogo Dodo, Golfieke, Googol30, GraemeL, Groupbroad, Grover cleveland, Gruver777, Gsmith1of2, Guidddruid, Halcyonforever, Hammersoft, Helix84, Henry Merriam, Hetar, Hu12, Ian Pitchford, Icairns, Ikonos44, Illumynite, Imroy, Invitatious, Iohannes Animosus, IvanLanin, J04n, JNW2, Jackfork, Jadams76, Jelson25, Jenrzzz, Jesse Viviano, Jesster79, Jfreyre, Jiffles1, Jigsaw, Jimthing, Jodypro, Joefrog4, John of Reading, JohnChrysostom, Jojalozzo, JonHarder, Jonathan at DigitalRealty, JonathanBentz, Jordancpeterson, Jruthven, Julesd, Justachild, Karada, Kavutamurthy, Keeganthewelsh, Keilana, Kilmer-san, Knodir, Kumul, Kuru, Kusma, Latiligence, Leifnsn, Lightmouse, LilHelpa, Lmatt, Logan, Londonconsult, LoneWolfJack, Lugia2453, MER-C, MMuzammils, MaXintoshPro, Madhero88, Madman, Maetrix, Makru, Martyvis, Matthew Yeager, Mauls, Maxí, Mcpusc, MichaelBillington, Mike Rosoft, Mindmatrix, Mipadi, Mister Nines, Mmastrosimone, Mr odway, Mrand, Nathgregory, NawlinWiki, Ncws, Ngch89, Niente21, Norizam, Ohnoitsjamie, Oran, PTS Consulting, Pack3406, Paskornc, Pavithran, Pentawing, Peterhgregory, Ph.eyes, PhilKnight, Piano non troppo, Pichpich, Pierre.davodeau, Pigsonthewing, Pinethicket, Poweroid, Priyanshu kumbhare, Ptfmd, Public Menace, Qwfp, Qwyrxian, R. S. Shaw, RJHall, Rahul3r, Randsholman, Raul654, Raysonho, Razakel19, RedWolf, Rees11, RevRagnarok, Rh9923, Rich Farmbrough, Richard001, Riotrocket8676, Rjwilmsi, Robenel, Robert.Harker, RodrigoSampaioPrimo, Ronny Wibowo,

Ronz, Rrburke, RudolfRed, Ryandsmith, Ryemcdonald, SBaker43, Salehigal, SamJohnston, Scjessey, SebastianHelm, Senator2029, Sfan00 IMG, Sfiteditor, Shades97, Shadowjams, Shawnsawb, SimonLyall, Sjakkalle, Sjbirkel, Sladen, Some standardized rigour, SongCoyote, Spleeman, Sst01, StarSaber, Stefan-hp, Stephan Leeds, Stephenb, Stephenodonnell, Stevag, SteveLoughran, Sunny.intel, Susanborton, Symetrix, Teapeat, Technologyarchitect, Techxact, Teddysnook, Telcoterry, Teratornis, Terrivsm, The Anome, Thejetset1, Theresa knott, Thijsniks, Tim1988, Tonkie, Tonkie67, Tork311, Tothwolf, Tungsten, Turretgroup, TwoTwoHello, Ulric1313, Upholder, Uruiamme, VQuakr, Vektor330, VernoWhitney, Versageek, ViggyH, Vinseraluna, Visitor7, Vivek kapur at psu, Vmenkov, Vungvung, WCroslan, WWGB, Wavelength, Wbrameld, Wbunasser, Webwat, Webwonk, Weregerbil, Widr, Wikipediite, Wildsouth57, Winterst, Woohookitty, Wtalarico, Wuciwug, XP, Xhantar, Yamla, Yintan, Yyzrush, ZaffaNE, Zodon, 밝은날에지렁이, 444 anonymous edits

Virtual data room *Source*: http://en.wikipedia.org/w/index.php?oldid=577499021 *Contributors*: Ajbooth, Ansarada, Bonadea, Bruker123, Bsearched, Cjkvolly15, Cloud writer, Datapoint VDR, Drmies, Drsdigital, EllerTO, Gudhkahardik, Hssaha, JamesBWatson, Jeniagorbacheva, John.badger, Jrmandell, Lawrence Dell, Linkedinsheelagh, Merbabu, Merrill virtual data room, Merrillvdr, Mgramegna, Milanpadukka, Nirankar10, Ouser, Paul A, Rens9, Savizizadpanah, Sharefile, Sporcus, Stevenkrauss, Swolf79, Synapse306, Timar91, Ukexpat, Ultraexactzz, 49 anonymous edits

Virtual facility *Source*: http://en.wikipedia.org/w/index.php?oldid=597578169 *Contributors*: Arendedwinter, Cander0000, Stefan hendrickx, Svick, UncleDouggie, Wavelength, 6 anonymous edits

Network-neutral data center *Source*: http://en.wikipedia.org/w/index.php?oldid=535865758 *Contributors*: 336, Arman4711, Cocoaguy, DGG, Favonian, Hairhorn, R'n'B, Srleffler, Stcassociates2009, The Anome, WikHead, 4 anonymous edits

Virtual directory *Source*: http://en.wikipedia.org/w/index.php?oldid=580332668 *Contributors*: Andreas Kaufmann, Antonio navarro cano, Ashwin ambekar, Bigman921, Cyrius, Dewayne.nickerson, EagleOne, Eric Kvaalen, Hu12, Jandalhandler, Jimyangsh, LarryAucoin, LessHeard vanU, LizGere, Mewilcox, MrOllie, Muhandes, Norberthegyaljai, Scoutersig, Tenbaset, Tps777, Uncletravelinmatt, Vscheuber, 56 anonymous edits

Virtuoso Universal Server *Source*: http://en.wikipedia.org/w/index.php?oldid=596472277 *Contributors*: Akadruid, Apavlo, Arichnad, Avicennasis, Bunnyhop11, Cander0000, Charivari, ChrisGualtieri, CyberSkull, DBpedia, DanBri, Danim, DaveBrondsema, DeirdreGerhardt, Den fjättrade ankan, Djmackenzie, Elwikipedista, Erick.Antezana, Firsfron, Fractal91, Funandtrvl, Glenn, Grille Chompa, Götz, Herostratus, Hwilliams62, Intgr, Ipsign, IvanMikhailov, JLaTondre, Jackel, Josh Parris, Kingboyk, KingsleyIdehen, Kuyabribri, MacTed, Magioladitis, Mark Arsten, Mikeblas, Nicolas1981, Niffweed17, OsamaK, Petri Krohn, Rayshade, Reedy, Samuel Blanning, Scott, Sfan00 IMG, Snowolf, Spodzone, Starionwolf, Stefan, Stephen B Streater, Topbanana, Ultimatewisdom, Vadmium, Vegaswikian, William Graham, 34 anonymous edits

Workflow engine *Source*: http://en.wikipedia.org/w/index.php?oldid=598101030 *Contributors*: AManWithNoPlan, Akhilao, Back ache, Cobusve, Deandany, Eric Pignet, Flemobile, Gealion, Happyfish, HeikoHaller, In7sky, Jmettraux, Joanhuang, Joellejojo, Kuru, MartinStrejc, Misteror, MrOllie, Pborenstein, Skalko, U2fanboi, Vanished user 39948282, Vojtech Huser, Vojtech huser, 36 anonymous edits

Metadata *Source*: http://en.wikipedia.org/w/index.php?oldid=598958437 *Contributors*: 0612, 121a0012, 16@r, 210.49.109.xxx, 7, AGK, AGToth, Acdg, Ahoerstemeier, Akhristov, Al Adriance, Alansohn, Albert ip, AlexChurchill, Algocu, AlistairMcMillan, Allen Moore, Allens, Allthingsace, Amr.rs, Andrea Parri, Andrew Gray, Andrew Werdna, AndrewHowse, AndrewRH, Androo, Andy Dingley, AnmaFinotera, Anna Lincoln, Anna Montull, Anniebiogirl, Antandrus, Anthony Borla, Antonielly, Apapadop, Armaced, Arpabr, Artaxiad, Arthena, Avicennasis, BD2412, BackwardsBoy, Balajiveera, Beardo, Beetstra, Belbernard, BenRogers, Benbludget, Beyer, Bhny, Bigpinkthing, Biker JR, BioPupil, Blackphiber, Blitzmut, Boardhead, Btwied, BurntSky, CFilm, CRJO-CRJO, CTZMSC3, Cab88, Canyouhearmenow, Carly805, Carolyn22789, Catgut, Cdc, Ceceliadid, Celticst, Cfwschmidt, Chandan Patel, Charles T. Betz, CharlesC, Chievous, ClaudioFerreira, Cnash11, Coinchon, Cole2, Conversion script, Crimsonmargarine, Crysb, Cureden, DEddy, DGG, DMacks, Dalbon, DanBri, Daniel5127, DarkSaber2k, Daswani.Amit, Dave Hay, David B in Canberra, David Gerard, David Woolley, David.T.Bath, DavidPKendal, Davidryan168, DePiep, Deborah-jl, Dedmonds, Demerara, Denverjeffrey, Devarakondar, Diannaa, Dicklyon, Dmccreary, Do better, Doug Bell, Dramalho, Drrwebber, Dsimic, Dwlegg, ECEstats, Eeheeheeheeheeheehee, El benito, Elf, Elving, Emperorbma, Emre D., EnDumEn, Enfresdezh, Equilibrioception, Eric Blatant, Error, Europrobe, EvenT, Evercat, Everyking, Evil Monkey, Exe, Face.real, Feldermouse, FlamingSilmaril, Flarn2006, Fmccown, Francs2000, FredMBrown, FreplySpang, Fritzpoll, Fuzheado, GRAHAMUK, GVogeler, Gaius Cornelius, Galoubet, Garion96, Gautam3, Gbevin, Genealbert, GhostGirl, Gioto, Girl2k, GoingBatty, Graham87, Graybeal, Green caterpillar, GregLindahl, GregorB, Guffydrawers, Gurch, Gywst, H10130, Halaster, HarryAlffa, Harryzilber, Heron, Hiplibrarianship, Hmains, Hugonius, Hvn0413, Hvs, IRowlands, Iancarter, Iluvcapra, Inkington, Isiaunia, Itai, J Milburn, J04n, JRR Trollkien, Jacobolus, Jamacfarlane, Jarble, Jasongao99, Jdmbellevue, Jeffrey Mall, Jehochman, Jerome Charles Potts, Jewers, Jgjournalist, Jim1138, Jinzhanguwm, Jleedev, Jmundo, Joejoejoejoejoejoejoejoejoe, John Hubbard, John Vandenberg, JohnRonald, JonHarder, Jonasalmeida, Jordgette, JosebaAbaitua, Jschwa1, Juliancolton, Juro2351, Jusdafax, Just plain Bill, K6ka, Kazrak, Keilana, Kenta, Kevin B12, KeyStroke, Khalid hassani, Khym Chanur, Killiondude, Kim Bruning, Kku, Klower, Kraftlos, Kuru, Kvng, Kylemew, LOL, Lando Calrissian, Lazy Techie, Lcme, LeeHunter, Les733, Lethesl, Lheuer, Ligulem, LinaMishima, Ljagerman, Lotje, Lowellian, Luckyz, Lulu of the Lotus-Eaters, M4gnum0n, MDE, MacGyverMagic, Malyctenar, Mamizou, Manfred-jeu, Mani1, Maork, Marchitelli, Mark Renier, Markhurd, Masgatotkaca, Matthewvelie, Matěj Grabovský, Maurice Carbonaro, Mav, Mboverload, Mdd, MetaWorker, Metajohng, Michael Hardy, MikeLynch, Mild Bill Hiccup, Millmoss, Miss Madeline, Mjb, Modify, Mrbradleyt, Mudkipzss, Mushin, Mwestby828, Mxn, Nadimghaznavi, Natalya, NawlinWiki, Neo-Jay, Nerdonpurpose, Newbyguesses, Nf1234, Nichtich, Nickg, Niduzzi, Night Gyr, Nixdorf, Nopetro, Northgrove, Notinasnaid, Nowa, Nscwikiedt, Nyq, Nyttend, Octahedron80, Ojw, Omassey, Omnipaedista, OnceAlpha, Ondertitel, Ottomachin, Pathoschild, Patrick, PatrickFisher, Paul Asman, Peciv, Peepeefigjam123, PerryTachett, Pete Short, Phantomsteve, Phillipgi, Pigsonthewing, PinkAmpersand, Pinkgirl9595, Pinku.nagpal, Poor Yorick, Poornima vijayan, Prince of Strings, Protonk, Public Menace, Purslane, R'n'B, Rabbit67890, Raffaele Megabyte, Ramu50, Rarcher88, RayGates, RedWolf, ResearchRave, Reswobslc, Rexdeaz, Rhagen7, Richard.decal, Rjwilmsi, Rnathanday, Robth, Ronvelig, Ronz, Rougieux, Rspeer, S. Cruz, S.K., SCEhardt, SEWilco, SFK2, Sallyrenee, Santamoly, Schandi, ScienceGolfFanatic, SebastianHelm, Sgb, Shaddim, Shadowjams, Shanebdavis, Sieuthulin, SilkTork, Sk19842, Skizzik, Sky Attacker, Slacka123, Smalljim, Snaxe920, Soler97, Soliloquial, Somewherepurple, Soocom1, Soumyasch, Spalding, Spdegabrielle, Spropis, SqueakBox, Srobak, StaticGull, Steffclarke, Stephen B Streater, Stevertigo, Stevietheman, Stolkin, Strike Eagle, StuartGilbert, Stuartyeates, Sualeh, Superm401, Syaskin, Synctext, TJRC, TYelliot, TaintedMustard, Tajymoid, Tarquin, The Epopt, TheDude813, Thetawave, Thetimperson, Tikiwont, Timtrent, Tipiac, Tkmi.itu.dk, Tnekevets, Tobias Bergemann, Tokailoverock, Tongbram, Tony1212, Tosahilchopra, Tregonsee, Treybien, Trilliumz, Tverbeek, Typhoon, Udayan.warnekar, Uliwitness, UninvitedCompany, UnitedStatesian, Utcursch, Vegard, Vigilius, Vik-Thor, Vincent jonsson, VitalyTarasov, Volphy, Wafulz, Warren, Whatyousage44, WhisperToMe, Wik, Wikky Horse, Wimmeljan, Wingwalker13, Wireless friend, Woodshed, Writ Keeper, WriterHound, Yerpo, Ylvabarker, Yonkie, ZackMartin, Zundark, Zyb5586, 697 anonymous edits

Meta-data management *Source*: http://en.wikipedia.org/w/index.php?oldid=574858303 *Contributors*: AndrewHowse, Arjayay, Barb49r, BirgitteSB, Connich, Delusion23, Edwulf, Fabrictramp, Gabriel Kielland, Infolibrariancorp, Jane023, Libcub, Loadmaster, Midhart90, Mild Bill Hiccup, Minnaert, Mogism, MrMarmite, Obromley, Pxma, Shahbazali101, Steffclarke, UkPaolo, 16 anonymous edits

Metadatabase *Source*: http://en.wikipedia.org/w/index.php?oldid=521009853 *Contributors*: Curtis, Drbreznjev, Dthomsen8, EagleFan, Hodja Nasreddin, Hsuck, Katharineamy, Nurg, 1 anonymous edits

Metadirectory *Source*: http://en.wikipedia.org/w/index.php?oldid=562437156 *Contributors*: Abrahami, DStoykov, DamonG, David.bon, EagleOne, Ediazrod, EurekaLott, Firesta, Furrypop, HFuruseth, Hu12, Jonabbey, JuergenL, KeyStroke, MrOllie, Pnm, Ritchiey, Ronz, Rwinningham, Sony-blitz, Yoni.bartov, 22 anonymous edits

Metadata controller *Source*: http://en.wikipedia.org/w/index.php?oldid=490139835 *Contributors*: Edward, Manghole

Data element *Source*: http://en.wikipedia.org/w/index.php?oldid=585437918 *Contributors*: CryptoDerk, DEddy, Darcy.kirk, Dmccreary, Docu, Eastlaw, Eep², Friendlydata, Haemo, Imran, JeffTan, Kuru, Mark Renier, Mindmatrix, Noel Streatfield, R'n'B, RayGates, SqlPac, The Thing That Should Not Be, Youcantbattle, Филатов Алексей, 22 anonymous edits

Metadata publishing *Source*: http://en.wikipedia.org/w/index.php?oldid=465635639 *Contributors*: Alksentrs, Andy Dingley, Dmccreary, Gaius Cornelius, JonHarder, Jpbowen, Kelly Martin, Kuru, Lheuer, MDE, Michael Hardy, Rabideau, RayGates, S.K., 7 anonymous edits

Metadata registry *Source*: http://en.wikipedia.org/w/index.php?oldid=586843060 *Contributors*: AlainV, Carly805, Cntras, Colonies Chris, Connich, Djembayz, Dmccreary, Dmg va, DonEF, Drrwebber, Farrukh najmi, Irishdad404, John of Reading, Jonphipps, Kuru, Libcub, Linforest, MaryEFreeman, Michael Hardy, Mzajac, Ottomachin, PeterParslow, Pfr138, RayGates, Rprowemdm, Ruud Koot, ShelleyAdams, Sherilabonte, ThatPeskyCommoner, The Grumpy Hacker, Ulric, Velella, Wikipelli, 41 anonymous edits

Metadata facility for Java *Source*: http://en.wikipedia.org/w/index.php?oldid=573014415 *Contributors*: Doug Bell, Manticore55, Mike.aizatsky, Vprajkumar, 3 anonymous edits

Object Management Group *Source*: http://en.wikipedia.org/w/index.php?oldid=584578797 *Contributors*: 121a0012, 217.162.105.xxx, AbstractBeliefs, Appraiser, Arthena, Bed20, Beland, Bernard.szlachta, Blauerflummi, Briangregory2000, Brick Thrower, Brookie, CALR, CatherineMunro, CesarB, Charwing, Conversion script, Cpl Syx, DJ Clayworth, DamienPo, Dataphile, DerHexer, Deville, Dmsar, Dyhorus, Edward, Elano, Equilibrioception, EvenT, Fatnerdio58, Happysailor, Iamtehpwnagerofl, Icairns, JBrown23, Jayrek, Jesse V., Jishnu, Jmpike11, Jriddell, Jsub, Kesac, Ligulem, LittleDan, Lonestar662p3, Luís Felipe Braga, MDE, MER-C, Mdd, Michael Hardy, Mjchonoles, MrOllie, Niceguyedc, Nikfar, Nixdorf, Noisy, Omgmarketing, Pvosta, Quarl, R'n'B, Raph, Rasmus Faber, Reinhardheydt, Robth, Royalguard11, Siyamed, Spoivre, Suruena, Svilenv, Tawker, Techi2ee, Technopilgrim, Thumperward, Thv, Topbanana, Xactandy, Zigger, 76 anonymous edits

Semantics of Business Vocabulary and Business Rules *Source*: http://en.wikipedia.org/w/index.php?oldid=574577901 *Contributors*: BD2412, Bnorrie, Cerebellum, Chris the speller, Edward, Equilibrioception, Gmelli, Gregbard, Juangelos, Mdd, Pxma, Qteimi, RayGates, Rjwilmsi, Senator2029, Tobias Kuhn, 19 anonymous edits

Business Motivation Model *Source*: http://en.wikipedia.org/w/index.php?oldid=592024888 *Contributors*: Bernard.szlachta, Bszlachta, Cathy Richards, ChrisGualtieri, Davidjcmorris, FlügelRad, Mdd, MuffledThud, RichardVeryard, Usnotes, 5 anonymous edits

Business Process Definition Metamodel *Source*: http://en.wikipedia.org/w/index.php?oldid=551591343 *Contributors*: BPDM, Baudoin1, Diveintobpm, Ehheh, Goflow6206, Jpbowen, Lurp, Niceguyedc, R'n'B, Sisyph, Tomdebevoise, 8 anonymous edits

Knowledge Discovery Metamodel *Source*: http://en.wikipedia.org/w/index.php?oldid=551591528 *Contributors*: 4th-otaku, Andreas Kaufmann, Cander0000, Edward, Equilibrioception, Khalid hassani, Niceguyedc, Softwaresavant, Stephane vaucher, 4 anonymous edits

Resources, events, agents (accounting model) *Source*: http://en.wikipedia.org/w/index.php?oldid=598572789 *Contributors*: Gallitzin, Ggarbellotto, Jmsloderbeck, John Vandenberg, Mild Bill Hiccup, Muhandes, PavelHruby, Rich Farmbrough, Saxifrage, Thiujiac, Tony1, 25 anonymous edits

Learning object metadata *Source*: http://en.wikipedia.org/w/index.php?oldid=594339829 *Contributors*: Abmac, Anton Khorev, Ary29, Asimong, BackwardsBoy, Brezenbene, Brianwc, CQ, DalbS, Danhash, Discospinster, Downes, Dual Freq, Ettrig, Felix Folio Secundus, Frepa, Gaius Cornelius, GermanX, H@r@ld, Hu12, Ipseity, J Di, Jan.pawlowski, Joyous!, Jsweetin, LilHelpa, Local contributor, Loxlie, Lusanaherandraton, Marasmusine, Mcanabalb, Meegs, MichaelaBrighella3, Nadavkav, Nesbit, Nf1234, O Wise 1, Osoneill, PM Poon, Philbarker, RHaworth, Rich Farmbrough, Rjgodoy, Sallyrenee, Slismann, Ssalonso, Stevage, SunAdmin111, Thiseye, Vitello, Wimmuskee, 53 anonymous edits

Learning object *Source*: http://en.wikipedia.org/w/index.php?oldid=590427227 *Contributors*: ARK, AS, AlistairMcMillan, Angela, Arifjinha, Asgara, Ashishtanwer, BWatkins, CQ, Cdnchameleongirl, Chunkthechunk, Conversion script, D Wiley BYU, DGG, Damiens.rf, DanielLemire, Danielcarvalho, Derek Ross, Dmccreary, Dr.Helfrich, EdH, Ericblazek, Eubulides, Ever.anon, Evil saltine, FeatherPluma, Hirzel, Hmcti, Ixfd64, Jamesday, Jbmurray, JiriK, Johnp2hunt, Jose Icaza, Josepant, Justin W Smith, Kabads, Kaychitwood, KennethUrban, Korte, Kurlands2, LeeHunter, Leighblackall, Leszek Jańczuk, Lexor, Lkmorlan, MHV, Manco Capac, Mav, Mbell, Mbonetti, Meadowsa, Michaelsshaw, Micru, Mion, Miss Dark, Mmorrey, Mr.Z-man, Nf1234, Ohnoitsjamie, OlEnglish, Petercoooper, Philbarker, Pierfranco, Reatlas, Redlandrain, Rjgodoy, Robofish, Ronz, Ryulong, Seanashbrook, Steven Walling, Stu 22, The Anome, TheRingess, Thelearningfederation, Tom CLO, Yerpo, 76 anonymous edits

OGML *Source*: http://en.wikipedia.org/w/index.php?oldid=542673713 *Contributors*: Alaarman, DoctorKubla, FoCuSandLeArN, MuffledThud, R'n'B, 1 anonymous edits

Image Sources, Licenses and Contributors

File:CodasylB.png *Source*: http://en.wikipedia.org/w/index.php?title=File:CodasylB.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Jean-Baptiste Waldner, User:Jbw

Image:Relational key SVG.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Relational_key_SVG.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:IkamusumeFan

File:Database models.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Database_models.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Marcel Douwe Dekker

Image:Traditional View of Data SVG.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Traditional_View_of_Data_SVG.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:IkamusumeFan

File:4-3 Data Modelling Today.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:4-3_Data_Modelling_Today.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: 4-3_Data_Modelling_Today.jpg: Matthew West and Julian Fowler derivative work: Razorbliss (talk)

File:3-4 Data model roles.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:3-4_Data_model_roles.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: 3-4_Data_model_roles.jpg: Matthew West and Julian Fowler derivative work: Razorbliss (talk)

File:4-2 ANSI-SPARC three level architecture.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:4-2_ANSI-SPARC_three_level_architecture.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: 4-2_ANSI-SPARC_three_level_architecture.jpg: Matthew West and Julian Fowler derivative work: Razorbliss (talk)

File:Data modeling context.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Data_modeling_context.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Process_and_data_modeling.jpg: Paul R. Smith. Redrawn by Marcel Douwe Dekker Data_modeling_context.jpg: Marcel Douwe Dekker derivative work: Razorbliss (talk)

File:B 5 1 IDEF1X Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:B_5_1_IDEF1X_Diagram.jpg *License*: Public Domain *Contributors*: itl.nist.gov

File:HL7 Reference Information Model.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:HL7_Reference_Information_Model.jpg *License*: Public Domain *Contributors*: Amnon Shabo (Shvo)

File:A2 4 Semantic Data Models.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A2_4_Semantic_Data_Models.jpg *License*: Public Domain *Contributors*: itl.nist.gov

File:PD-icon.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:PD-icon.svg *License*: Public Domain *Contributors*: Alex.muller, Anomie, Anonymous Dissident, CBM, MBisanz, PBS, Quadell, Rocket000, Strangerer, Timotheus Canens, 1 anonymous edits

Image:FigFileConvert000a.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:FigFileConvert000a.svg *License*: Public Domain *Contributors*: Dreftymac

Image:Hierarchisches Datenbankmodell.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Hierarchisches_Datenbankmodell.svg *License*: Public Domain *Contributors*: Sarang

Image:Network DB model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Network_DB_model.svg *License*: Public Domain *Contributors*: Sarang

Image:Relational model concepts.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Relational_model_concepts.png *License*: GNU Free Documentation License *Contributors*: User:AutumnSnow

Image:Company codm.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Company_codm.gif *License*: unknown *Contributors*: en:User:Savinov Original uploader was Savinov at en.wikipedia

Image:Star-schema.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Star-schema.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: SqlPac

File:Aggregate Data Structure Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Aggregate_Data_Structure_Diagram.jpg *License*: Public Domain *Contributors*: NRCS

Image:Groups relate to the process of making a map.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Groups_relate_to_the_process_of_making_a_map.jpg *License*: Public Domain *Contributors*: David R. Soller1 and Thomas M. Berg

Image:NGMDB data model application.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:NGMDB_data_model_application.jpg *License*: Public Domain *Contributors*: David R. Soller1 and Thomas M. Berg

Image:NGMDB databases linked together.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:NGMDB_databases_linked_together.jpg *License*: Public Domain *Contributors*: David R. Soller1 and Thomas M. Berg

Image:Representing three-dimensional map information.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Representing_three-dimensional_map_information.jpg *License*: Public Domain *Contributors*: David R. Soller1 and Thomas M. Berg

File:3-2 Properties of data.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:3-2_Properties_of_data.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: 3-2_Properties_of_data.jpg: Matthew West and Julian Fowler derivative work: Razorbliss (talk)

File:binary tree.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Binary_tree.svg *License*: Public Domain *Contributors*: User:Dcoetzee

Image:Array of array storage.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Array_of_array_storage.svg *License*: Public Domain *Contributors*: User:Dcoetzee

Image:HASHTB08 en.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:HASHTB08_en.svg *License*: Public Domain *Contributors*: HASHTB08_sk.svg:User:Helix84 derivative work: Kotecky (talk)

Image:Singly linked list insert after.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Singly_linked_list_insert_after.png *License*: Public domain *Contributors*: Derrick Coetzee

Image:Data stack.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Data_stack.svg *License*: Public Domain *Contributors*: User:Boivie

File:Data Flow Diagram Example.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Data_Flow_Diagram_Example.jpg *License*: Public Domain *Contributors*: John Azzolini

File:A 01 Audio compact disc collection.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A_01_Audio_compact_disc_collection.jpg *License*: Public Domain *Contributors*: Michael R. McCaleb, National Institute of Standards and Technology

File:JKDOM.SVG *Source*: http://en.wikipedia.org/w/index.php?title=File:JKDOM.SVG *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: John M. Kennedy T.

File:Schema for Geologic Surface.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Schema_for_Geologic_Surface.gif *License*: Public Domain *Contributors*: Stephen M. Richard

Image:Flat File Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flat_File_Model.svg *License*: Public Domain *Contributors*: Wgabrie (talk) 16:48, 13 March 2009 (UTC)

Image:Hierarchical Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Hierarchical_Model.svg *License*: Public Domain *Contributors*: U.S. Department of Transportation vectorization:

Image:Network Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Network_Model.svg *License*: Public Domain *Contributors*: U.S. Department of Transportation vectorization:

File:Emp Tables (Database).PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Emp_Tables_(Database).PNG *License*: Public Domain *Contributors*: Jamesssss

Image:Object-Oriented Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Object-Oriented_Model.svg *License*: Public Domain *Contributors*: U.S. Department of Transportation vectorization:

Image:ER Diagram MMORPG.png *Source*: http://en.wikipedia.org/w/index.php?title=File:ER_Diagram_MMORPG.png *License*: GNU Free Documentation License *Contributors*: Original uploader was TheMattrix at en.wikipedia

Image:Data Structure Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Data_Structure_Diagram.jpg *License*: Public Domain *Contributors*: U.S. Department of Transportation,

File:Data Structure Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Data_Structure_Diagram.jpg *License*: Public Domain *Contributors*: U.S. Department of Transportation,

File:Bachman diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Bachman_diagram.jpg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Jbw, Cropped by Marcel Douwe Dekker

File:Hierarchical Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Hierarchical_Model.svg *License*: Public Domain *Contributors*: U.S. Department of Transportation vectorization:

File:Network Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Network_Model.svg *License*: Public Domain *Contributors*: U.S. Department of Transportation vectorization:

File:6n-graf.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:6n-graf.svg *License*: Public Domain *Contributors*: User:AzaToth

Image:Erd-entity-relationship-example1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Erd-entity-relationship-example1.svg *License*: Creative Commons ShareAlike 1.0 Generic *Contributors*: Chanueting

Image:Erd-entity-with-attribute.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Erd-entity-with-attribute.svg *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was Bigsmoke at en.wikipedia

Image:erd-relationship-with-attribute.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Erd-relationship-with-attribute.png *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was Bigsmoke at en.wikipedia

Image:Erd-id-as-primary-key.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Erd-id-as-primary-key.png *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was Bigsmoke at en.wikipedia

Image:ERD Representation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:ERD_Representation.svg *License*: Public Domain *Contributors*: Benthompson

Image:ERD-artist-performs-song.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:ERD-artist-performs-song.svg *License*: Public Domain *Contributors*: Original uploader was Bignose at en.wikipedia

File:ER Diagram MMORPG.png *Source*: http://en.wikipedia.org/w/index.php?title=File:ER_Diagram_MMORPG.png *License*: GNU Free Documentation License *Contributors*: Original uploader was TheMattrix at en.wikipedia

File:AggregationAndComposition.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:AggregationAndComposition.svg *License*: Public Domain *Contributors*: ACiD2, Mikm, Pne, Mjchonoles, Rjgodoy, NevilleDNZ, Tobias Bergemann

File:CPT-Databases-ManytoMany.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:CPT-Databases-ManytoMany.svg *License*: Creative Commons Zero *Contributors*: Pluke

Image:Weak entity ER-example.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Weak_entity_ER-example.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Mixer

Image:Associate_Entity.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Associate_Entity.png *License*: Public Domain *Contributors*: User:Ahsile

Image:SERM-Symbols.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:SERM-Symbols.JPG *License*: GNU Free Documentation License *Contributors*: Steffen Thomas

Image:SERM-example.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:SERM-example.JPG *License*: GNU Free Documentation License *Contributors*: Steffen Thomas

File:UML Diagrams.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:UML_Diagrams.jpg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Kishorekumar 62

File:OO Modeling languages history.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:OO_Modeling_languages_history.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Mdd

File:UML diagrams overview.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:UML_diagrams_overview.svg *License*: Public Domain *Contributors*: Derfel73; Pmerson

File:BankAccount1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:BankAccount1.svg *License*: GNU Free Documentation License *Contributors*: Donald Bell

File:Policy Admin Component Diagram.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Policy_Admin_Component_Diagram.PNG *License*: GNU Free Documentation License *Contributors*: Kishorekumar 62

File:Composite Structure Diagram.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Composite_Structure_Diagram.png *License*: Public Domain *Contributors*: [:w:User:KenSWebbUser:KenSWebb]

File:Deployment Diagram.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Deployment_Diagram.PNG *License*: GNU Free Documentation License *Contributors*: Kishorekumar 62

File:Object diagram.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Object_diagram.png *License*: Public Domain *Contributors*: Mac9

File:Package Diagram.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Package_Diagram.PNG *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Kishorekumar 62

File:Activity conducting.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Activity_conducting.svg *License*: GNU Free Documentation License *Contributors*: Gwaur

Image:UML state diagram.png *Source*: http://en.wikipedia.org/w/index.php?title=File:UML_state_diagram.png *License*: GNU Free Documentation License *Contributors*: Anakin101, Flappiefh, LeonardoG, Mdd, Pcdude2143, Quibik, Sbwoodside, 3 anonymous edits

Image:UML_Use_Case_diagram.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:UML_Use_Case_diagram.svg *License*: Creative Commons Attribution-Share Alike *Contributors*: Slashme

Image:Kommunikations diagramm-2.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Kommunikations_diagramm-2.png *License*: GNU Free Documentation License *Contributors*: Gubaer

Image:Iau-diagramm-1.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Iau-diagramm-1.png *License*: GNU Free Documentation License *Contributors*: Original uploader was Gubaer at de.wikipedia

Image:CheckEmail.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:CheckEmail.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Sae1962, Skim, Zerodamage

Image:M0-m3.png *Source*: http://en.wikipedia.org/w/index.php?title=File:M0-m3.png *License*: Public Domain *Contributors*: Gubaer

Image:4-2 ANSI-SPARC three level architecture.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:4-2_ANSI-SPARC_three_level_architecture.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: 4-2_ANSI-SPARC_three_level_architecture.jpg: Matthew West and Julian Fowler derivative work: Razorbliss (talk)

Image:A2 2 Traditional View of Data.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A2_2_Traditional_View_of_Data.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A2 3 Three schema approach.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A2_3_Three_schema_approach.jpg *License*: Public Domain *Contributors*: itl.nist.gov

File:ZF What column Data example.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:ZF_What_column_Data_example.jpg *License*: Creative Commons Attribution 3.0 *Contributors*: Stan Locke

File:Anchor Modeling Example.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Anchor_Modeling_Example.svg *License*: unknown *Contributors*: Lars Rönnbäck

Image:B 5 1 IDEF1X Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:B_5_1_IDEF1X_Diagram.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:1 Entity Syntax.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:1_Entity_Syntax.svg *License*: Public Domain *Contributors*: itl.nist.gov

Image:2 Example of a Domain Hierarchy.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:2_Example_of_a_Domain_Hierarchy.svg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 11 Attribute Examples.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_11_Attribute_Examples.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:3 Attribute and Primary Key Syntax.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:3_Attribute_and_Primary_Key_Syntax.svg *License*: Public Domain *Contributors*: itl.nist.gov

Image:4 Relationship Cardinality Syntax.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:4_Relationship_Cardinality_Syntax.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:5 Identifying Relationship Syntax.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:5_Identifying_Relationship_Syntax.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:8 Categorization Relationship Syntax.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:8_Categorization_Relationship_Syntax.svg *License*: Public Domain *Contributors*: itl.nist.gov

Image:9 Non-Specific Relationship Syntax.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:9_Non-Specific_Relationship_Syntax.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 2 Synthesizing an Entity.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_2_Synthesizing_an_Entity.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 4 Entity Relationship Matrix.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_4_Entity_Relationship_Matrix.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 5 Entity Level Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_5_Entity_Level_Diagram.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 6 Phase Two (Entity Level) Diagram Example.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_6_Phase_Two_(Entity_Level)_Diagram_Example.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 7 Reference Diagram (FEO).jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_7_Reference_Diagram_(FEO).jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 8 Example Reference Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_8_Example_Reference_Diagram.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 9 Non-Specific Relationship Refinement.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_9_Non-Specific_Relationship_Refinement.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 10 Scope of a Function View.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_10_Scope_of_a_Function_View.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 16 No-Repeat Rule Refinement.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_16_No-Repeat_Rule_Refinement.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 17 Rule Refinement.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_17_Rule_Refinement.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 19 Path Assertions.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_19_Path_Assertions.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 21 Example of Phase Three Function View Diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_21_Example_of_Phase_Three_Function_View_Diagram.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 23 Phase Four - Applying the No Repeat Rule.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_23_Phase_Four_-_Applying_the_No_Repeat_Rule.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A3 24 Example of Phase Four Function.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A3_24_Example_of_Phase_Four_Function.jpg *License*: Public Domain *Contributors*: itl.nist.gov

Image:A2 4 Semantic Data Models.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:A2_4_Semantic_Data_Models.jpg *License*: Public Domain *Contributors*: itl.nist.gov

File:Relational Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Relational_Model.svg *License*: Public Domain *Contributors*: U.S. Department of Transportation vectorization:

File:Relational key.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Relational_key.png *License*: Public Domain *Contributors*: LionKimbro

File:Relational model concepts.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Relational_model_concepts.png *License*: GNU Free Documentation License *Contributors*: User:AutumnSnow

Image:Relational database terms.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Relational_database_terms.svg *License*: Public Domain *Contributors*: User:Booyabazooka

File:Junction Table.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Junction_Table.svg *License*: Public Domain *Contributors*: Mr.98

File:NestedSetModel.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:NestedSetModel.svg *License*: Public Domain *Contributors*: Nestedsetmodel.jpg: Sherahm derivative work: 0x24a537r9 (talk)

File:Clothing-hierarchy-traversal-2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Clothing-hierarchy-traversal-2.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Clothing-hierarchy-traversal.svg: Rp22 derivative work: Zygmuntjr (talk)

File:SQL ANATOMY wiki.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:SQL_ANATOMY_wiki.svg *License*: unknown *Contributors*: :User:SqlPac, modified by Ferdna. Original uploader was Ferdna at en.wikipedia

File:DBaseLogo BlackWithRed glass 300.png *Source*: http://en.wikipedia.org/w/index.php?title=File:DBaseLogo_BlackWithRed_glass_300.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Mrozlog

Image:FlagShipLogoByMultisoft.png *Source*: http://en.wikipedia.org/w/index.php?title=File:FlagShipLogoByMultisoft.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Balek

Image:FlagShip environment 800px.png *Source*: http://en.wikipedia.org/w/index.php?title=File:FlagShip_environment_800px.png *License*: GNU Free Documentation License *Contributors*: Balek

Image:Address800px.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Address800px.jpg *License*: Creative Commons Attribution-Share Alike *Contributors*: Balek

File:Harbour Sample Code.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Harbour_Sample_Code.png *License*: unknown *Contributors*: Pritoal Bedi

File:HBIDE Editor.png *Source*: http://en.wikipedia.org/w/index.php?title=File:HBIDE_Editor.png *License*: unknown *Contributors*: Pritpal Bedi

File:Animals.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Animals.png *License*: Public Domain *Contributors*: DutiesAtHand

Image:QBESampleTable2.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:QBESampleTable2.jpg *License*: GNU Free Documentation License *Contributors*: SabbeRubbish

Image:DotNet3.0.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:DotNet3.0.svg *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was Surachit at en.wikipedia Later version(s) were uploaded by Soumyasch at en.wikipedia.

File:SQL Server FTS.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:SQL_Server_FTS.svg *License*: GNU Free Documentation License *Contributors*: Soumyasch

Image:Mediawiki database Schema.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Mediawiki_database_Schema.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Donose.mihai

File:Oracle logo.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Oracle_logo.svg *License*: Public Domain *Contributors*: Oracle Corporation. Original uploader was Cristan at en.wikipedia

File:Oracle Exadata X2-2.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Oracle_Exadata_X2-2.jpg *License*: Creative Commons Attribution 2.0 *Contributors*: Bezik, Denniss

File:Mysql-screenshot.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Mysql-screenshot.PNG *License*: Public Domain *Contributors*: Dereckson, Dvorapa, Stephantom

File:Mysqlwb-homepage.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Mysqlwb-homepage.png *License*: GNU General Public License *Contributors*: Official screenshot as published on MySQL.com. Original uploader was Tomjenkins52 at en.wikipedia

File:LAMP software bundle.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:LAMP_software_bundle.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:ScotXW

File:SQLite370.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:SQLite370.svg *License*: Public Domain *Contributors*: Part of the SQLite documentation, which has been released by author D. Richard Hipp to the public domain. SVG conversion by Mike Toews.

Image:LAMPP Architecture.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LAMPP_Architecture.png *License*: Creative Commons Attribution 3.0 *Contributors*: Dsimic

File:Microsoft Access icon.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Microsoft_Access_icon.png *License*: unknown *Contributors*: Closedmouth, Codename Lisa, Rezonansowy, Sfan00 IMG, Tbhotch

File:Access.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Access.PNG *License*: unknown *Contributors*: Codename Lisa, Mark Arsten, Sfan00 IMG, Stefan2, The Dark Melon, 1 anonymous edits

Image:RBASE for CTOS disk photo.png *Source*: http://en.wikipedia.org/w/index.php?title=File:RBASE_for_CTOS_disk_photo.png *License*: unknown *Contributors*: User:David Jordan

File:Update anomaly.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Update_anomaly.svg *License*: Public Domain *Contributors*: Nabav,

File:Insertion anomaly.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Insertion_anomaly.svg *License*: Public domain *Contributors*: en:User:Nabav, User:Stannered

File:Deletion anomaly.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Deletion_anomaly.svg *License*: Public domain *Contributors*: en:User:Nabav, User:Stannered

File:Schedule-serializability.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Schedule-serializability.png *License*: Public Domain *Contributors*: Original uploader was Sprite at en.wikipedia

Image:Precedence graph.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Precedence_graph.svg *License*: Public Domain *Contributors*: Original uploader was Ehamberg at en.wikipedia

Image:CO-ScheduleClasses.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:CO-ScheduleClasses.jpg *License*: unknown *Contributors*: Comps

Image:SCO-VS-SS2PL.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:SCO-VS-SS2PL.jpg *License*: unknown *Contributors*: Comps

File:HD with toasty PCB.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:HD_with_toasty_PCB.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: user:east718

Image:Data loss of image file.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Data_loss_of_image_file.JPG *License*: Public Domain *Contributors*: myself

File:Hierachical-Diagram.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Hierachical-Diagram.PNG *License*: Public Domain *Contributors*: Sukari

File:Network-Diagram.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Network-Diagram.PNG *License*: Public Domain *Contributors*: Sukari

File:Relational-Diagram.PNG *Source*: http://en.wikipedia.org/w/index.php?title=File:Relational-Diagram.PNG *License*: Public Domain *Contributors*: Sukari

File:Backup-DFD.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Backup-DFD.png *License*: Public domain *Contributors*: Stefan4

File:Viegas-UserActivityonWikipedia.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Viegas-UserActivityonWikipedia.gif *License*: Creative Commons Attribution 2.0 *Contributors*: Fernanda B. Viégas

File:Hilbert InfoGrowth.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Hilbert_InfoGrowth.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Myworkforwiki

File:2013-09-11 Bus wrapped with SAP Big Data parked outside IDF13 (9730051783).jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:2013-09-11_Bus_wrapped_with_SAP_Big_Data_parked_outside_IDF13_(9730051783).jpg *License*: Creative Commons Attribution-Sharealike 2.0 *Contributors*: Intel Free Press

File:Big data cartoon t gregorius.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Big_data_cartoon_t_gregorius.jpg *License*: Creative Commons Attribution 2.0 *Contributors*: Thierry Gregorius

Image:Three-phase commit diagram.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Three-phase_commit_diagram.png *License*: Creative Commons Attribution 3.0 *Contributors*: User:Momet, User:Yagibear

Image:Compingles3.GIF *Source*: http://en.wikipedia.org/w/index.php?title=File:Compingles3.GIF *License*: Creative Commons Attribution-Share Alike *Contributors*: Lore uni

Image:ML-QLOGICNFCCONN.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:ML-QLOGICNFCCONN.JPG *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Melee

File:Physical Data Model Options.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Physical_Data_Model_Options.jpg *License*: Public Domain *Contributors*: Department of the Treasury Chief Information Officer Council

File:datawarehouse.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Datawarehouse.png *License*: Public Domain *Contributors*: Leonard^Bloom, Sfingram, 1 anonymous edits

File:dataintegration.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Dataintegration.png *License*: Public Domain *Contributors*: Sfingram

File:GAVLAV.png *Source*: http://en.wikipedia.org/w/index.php?title=File:GAVLAV.png *License*: Public domain *Contributors*: User:Ronhjones

File:Hadoop logo.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Hadoop_logo.svg *License*: Apache *Contributors*: Mwtoews

File:Hadoop 1.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Hadoop_1.png *License*: Public Domain *Contributors*: Paris.butterfield, 1 anonymous edits

File:H-Store-logo.png *Source*: http://en.wikipedia.org/w/index.php?title=File:H-Store-logo.png *License*: Creative Commons Attribution 3.0 *Contributors*: H-Store Project

File:KD SQLIA Classification 2010.png *Source*: http://en.wikipedia.org/w/index.php?title=File:KD_SQLIA_Classification_2010.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: KDeltchev

Image:Reactive Search Optimization Position.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Reactive_Search_Optimization_Position.png *License*: GNU Free Documentation License *Contributors*: Roberto Battiti

File:Data warehouse overview.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Data_warehouse_overview.JPG *License*: Public Domain *Contributors*: Hhultgren

Image:WestClinTech box prod.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:WestClinTech_box_prod.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Sqltool

Image:Star-schema-example.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Star-schema-example.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: SqlPac (talk)

Image:DFMSimpleFactSchema.png *Source*: http://en.wikipedia.org/w/index.php?title=File:DFMSimpleFactSchema.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Matteo.golfarelli

Image:DFMAdvancedFactSchema.png *Source*: http://en.wikipedia.org/w/index.php?title=File:DFMAdvancedFactSchema.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Matteo.golfarelli

Image:DFMMultipleArc.png *Source*: http://en.wikipedia.org/w/index.php?title=File:DFMMultipleArc.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Matteo.golfarelli

Image:Snowflake-schema.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Snowflake-schema.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: SqlPac

Image:Snowflake-schema-example.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Snowflake-schema-example.png *License*: GNU Free Documentation License *Contributors*: SqlPac (talk)

File:OLAP Cube.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:OLAP_Cube.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: OLAP_Cube.png: Konrad Roeder derivative work: Hazmat2 (talk)

File:OLAP slicing.png *Source*: http://en.wikipedia.org/w/index.php?title=File:OLAP_slicing.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Infopedian

File:OLAP dicing.png *Source*: http://en.wikipedia.org/w/index.php?title=File:OLAP_dicing.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Infopedian

File:OLAP drill up&down.png *Source*: http://en.wikipedia.org/w/index.php?title=File:OLAP_drill_up&down.png *License*: Creative Commons Attribution-Share Alike *Contributors*: Infopedian

File:OLAP pivoting.png *Source*: http://en.wikipedia.org/w/index.php?title=File:OLAP_pivoting.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Infopedian

Image:AdvertisementPamphletArchitecturalDiagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:AdvertisementPamphletArchitecturalDiagram.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jfrohm

File:Palo_Logo_2013.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Palo_Logo_2013.png *License*: Logo, Verwendung zu enzyklopädischen Zwecken erlaubt *Contributors*: DF Jedox AG

File:ETL Architecture Pattern.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:ETL_Architecture_Pattern.jpg *License*: Public Domain *Contributors*: ETL

Image:Scriptella logo.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Scriptella_logo.png *License*: Public Domain *Contributors*: Ejboy

File:Dunhuang manuscript digitisation.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Dunhuang_manuscript_digitisation.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Andrew Gray, BabelStone, Vicswift

File:Referential integrity broken.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Referential_integrity_broken.png *License*: GNU Free Documentation License *Contributors*: en:User:Ta bu shi da yu

File:SGML.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:SGML.svg *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Dreftymac

File:OED-LEXX-Bungler.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:OED-LEXX-Bungler.jpg *License*: GNU Free Documentation License *Contributors*: Andrew pmk, Helix84, INeverCry, JuTa, Mfc, Tedickey, WikipediaMaster, 10 anonymous edits

File:XML.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:XML.svg *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: en:User:Dreftymac

File:XQuery and XPath Data Model type hierarchy.png *Source*: http://en.wikipedia.org/w/index.php?title=File:XQuery_and_XPath_Data_Model_type_hierarchy.png *License*: Public Domain *Contributors*: A3 nm, Cwbm (commons), Mauro Bieg

file:Wikibooks-logo-en-noslogan.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Wikibooks-logo-en-noslogan.svg *License*: logo *Contributors*: User:Bastique, User:Ramac et al.

File:Object-Oriented Model.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Object-Oriented_Model.svg *License*: Public Domain *Contributors*: U.S. Department of Transportation vectorization:

Image:Versant logo 2007.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Versant_logo_2007.png *License*: Public Domain *Contributors*: http://www.versant.com/

File:Postgresql elephant.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Postgresql_elephant.svg *License*: BSD *Contributors*: Jeff MacDonald

File:GraphDatabase PropertyGraph.png *Source*: http://en.wikipedia.org/w/index.php?title=File:GraphDatabase_PropertyGraph.png *License*: Creative Commons Zero *Contributors*: User:Obersachse

File:Cassandra logo.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Cassandra_logo.svg *License*: Apache *Contributors*: Mwtoews

Image:Keyspace example (data store).png *Source*: http://en.wikipedia.org/w/index.php?title=File:Keyspace_example_(data_store).png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Bearcat, Sae1962

Image:SuperColumn (data store).png *Source*: http://en.wikipedia.org/w/index.php?title=File:SuperColumn_(data_store).png *License*: Creative Commons Attribution 3.0 *Contributors*: Sae1962

File:Rrddemo.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rrddemo.png *License*: unknown *Contributors*: en:Tobi Oetiker, en:User:Edaelon

File:IMDb logo.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:IMDb_logo.svg *License*: Public Domain *Contributors*: IMDb

File:Decrease Positive.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Decrease_Positive.svg *License*: Public Domain *Contributors*: Decrease2.svg: Sarang derivative work: Dodoïste (talk)

File:NDRs around the world 2011-11.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:NDRs_around_the_world_2011-11.jpg *License*: Creative Commons Attribution 2.0 *Contributors*: User:Cristixav

File:Flag of Algeria.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Algeria.svg *License*: Public Domain *Contributors*: This graphic was originaly drawn by User:SKopp.

File:Flag of Colombia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Colombia.svg *License*: Public Domain *Contributors*: SKopp

File:Flag of Canada.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Canada.svg *License*: Public Domain *Contributors*: Anomie

File:Flag of Australia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Australia.svg *License*: Public Domain *Contributors*: Anomie, Mifter

File:Flag of Western Australia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Western_Australia.svg *License*: Public domain *Contributors*: User:Denelson83

File:Flag of New South Wales.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_New_South_Wales.svg *License*: GNU Free Documentation License *Contributors*: User:Denelson83, User:Greentubing

File:Flag of the Northern Territory.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Northern_Territory.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Vectorized by Froztbyte

File:Flag of Queensland.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Queensland.svg *License*: Public domain *Contributors*: User:Denelson83

File:Flag of South Australia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_South_Australia.svg *License*: Public Domain *Contributors*: User:Denelson83

File:Flag of Tasmania.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Tasmania.svg *License*: Public Domain *Contributors*: User:Denelson83

File:Flag of the People's Republic of China.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_People's_Republic_of_China.svg *License*: Public Domain *Contributors*: Drawn by User:SKopp, redrawn by User:Denelson83 and User:Zscout370 Recode by cs:User:-xfi- (code), User:Shizhao (colors)

File:Flag of Russia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Russia.svg *License*: Public Domain *Contributors*: Anomie, Zscout370

File:Flag of Indonesia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Indonesia.svg *License*: Public Domain *Contributors*: Drawn by User:SKopp, rewritten by User:Gabbe

File:Flag of New Zealand.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_New_Zealand.svg *License*: Public Domain *Contributors*: Achim1999, Adabow, Adambro, Arria Belli, Avenue, Bawolff, Bjankuloski06en, ButterStick, Cycn, Denelson83, Donk, Duduziq, EugeneZelenko, Fred J, Fry1989, George Ho, Hugh Jass, Ibagli, Jusjih, Klemen Kocjancic, MAXXX-309, Mamndassan, Mattes, Nightstallion, O, Ozgurnarin, Peeperman, Poco a poco, Poromiami, Reisio, Rfc1394, Sarang, Shizhao, SiBr4, Tabasco, TintoMeches, Transparent Blue, Väsk, Xufanc, Zscout370, 42 anonymous edits

File:Flag of Jordan.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Jordan.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of Angola.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Angola.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of France.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_France.svg *License*: Public Domain *Contributors*: Anomie

File:Flag of Sao Tome and Principe.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Sao_Tome_and_Principe.svg *License*: Public Domain *Contributors*: User:Gabbe

File:Flag of Tanzania.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Tanzania.svg *License*: Public Domain *Contributors*: User:Alkari, User:Madden, User:SKopp

File:Flag of Oman.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Oman.svg *License*: Public Domain *Contributors*: *drew, Alkari, Bast64, Cycn, Duduziq, Fry1989, Happenstance, Homo lupus, Ittihadawi, Jetijones, Klemen Kocjancic, Liftarn, Mattes, Neq00, Nightstallion, NikNaks, OAlexander, Orange Tuesday, Pumbaa80, Rfc1394, Ricordisamoa, ThomasPusch, Zscout370

File:Flag of the Netherlands.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Netherlands.svg *License*: Public Domain *Contributors*: Zscout370

File:Flag of India.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_India.svg *License*: Public Domain *Contributors*: Anomie, Mifter

File:Flag of Sri Lanka.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Sri_Lanka.svg *License*: Public Domain *Contributors*: Zscout370

File:Flag of Argentina.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Argentina.svg *License*: unknown *Contributors*: Government of Argentina (Vector graphics by Dbenbenn)

File:Flag of Peru.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Peru.svg *License*: Public Domain *Contributors*: User:Dbenbenn

File:Flag of Kazakhstan.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Kazakhstan.svg *License*: unknown *Contributors*: -xfi-

File:Flag of Pakistan.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Pakistan.svg *License*: Public Domain *Contributors*: User:Zscout370

File:Flag of Nigeria.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Nigeria.svg *License*: Public Domain *Contributors*: User:Jhs

File:Flag of Turkey.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Turkey.svg *License*: Public Domain *Contributors*: David Benbennick (original author)

File:Flag of Norway.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Norway.svg *License*: Public Domain *Contributors*: Dbenbenn

File:Flag of the United Kingdom.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_United_Kingdom.svg *License*: Public Domain *Contributors*: Anomie, Good Olfactory, Mifter

File:Flag of Brazil.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Brazil.svg *License*: Public Domain *Contributors*: Anomie

File:Flag of Mexico.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Mexico.svg *License*: Public Domain *Contributors*: Alex Covarrubias, 9 April 2006 Based on the arms by Juan Gabino.

File:Flag of Israel.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Israel.svg *License*: Public Domain *Contributors*: "The Provisional Council of State Proclamation of the Flag of the State of Israel" of 25 Tishrei 5709 (28 October 1948) provides the official specification for the design of the Israeli flag. The color of the Magen David and the stripes of the Israeli flag is not precisely specified by the above legislation. The color depicted in the current version of the image is typical of flags used in Israel today, although individual flags can and do vary. The flag legislation officially specifies dimensions of 220 cm × 160 cm. However, the sizes of actual flags vary (although the aspect ratio is usually retained).

File:Flag of Cyprus.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Cyprus.svg *License*: Public Domain *Contributors*: User:Vzb83

File:Flag of South Africa.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_South_Africa.svg *License*: unknown *Contributors*: Adriaan, Anime Addict AA, AnonMoos, BRUTE, Cathy Richards, Daemonic Kangaroo, Dnik, Duduziq, Dzordzm, Fry1989, Homo lupus, Jappalang, Juliancolton, Kam Solusar, Klemen Kocjancic, Klymene, Lexxyy, MAXXX-309, Mahahahaneapneap, Manuelt15, Moviedefender, NeverDoING, Ninane, Pitke, Poznaniak, Przemub, Ricordisamoa, SKopp, Sarang, SiBr4, ThePCKid, ThomasPusch, Tvdm, Ultratomio, Vzb83, Zscout370, 37 anonymous edits

File:Flag of Kenya.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Kenya.svg *License*: Public Domain *Contributors*: User:Pumbaa80

File:Flag of the United States.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_United_States.svg *License*: Public Domain *Contributors*: Anomie

File:Flag of Cambodia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Cambodia.svg *License*: Public Domain *Contributors*: Draw new flag by User:ទពេ_សុរិចិត្រ

File:Flag of Afghanistan.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Afghanistan.svg *License*: Public Domain *Contributors*: User:Zscout370

File:Flag of Bangladesh.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Bangladesh.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of Ethiopia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Ethiopia.svg *License*: unknown *Contributors*: Aaker, Anime Addict AA, Antemister, Cycn, Djampa, F l a n k e r, Fry1989, GoodMorningEthiopia, Happenstance, Homo lupus, Huhsunqu, Ixfd64, Klemen Kocjancic, Ludger1961, MartinThoma, Mattes, Mozzan, Neq00, OAlexander, Pumbaa80, Rainforest tropicana, Reisio, Ricordisamoa, SKopp, Smooth O, Spiritia, ThomasPusch, Torstein, Wsiegmund, Xoristzatziki, Zscout370, 16 anonymous edits

File:Flag of Cameroon.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Cameroon.svg *License*: Public Domain *Contributors*: (of code) cs:User:-xfi-

File:Flag of Malaysia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Malaysia.svg *License*: Public Domain *Contributors*: , and

File:Flag of Spain.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Spain.svg *License*: Public Domain *Contributors*: Anomie

File:Flag of Morocco.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Morocco.svg *License*: Public Domain *Contributors*: Denelson83, Zscout370

File:Flag of Madagascar.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Madagascar.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of Sudan.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Sudan.svg *License*: Public Domain *Contributors*: Vzb83

File:Flag of Nicaragua.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Nicaragua.svg *License*: Attribution *Contributors*: C records, Ecemaml, Tacsipacsi

File:Flag of Iraq.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Iraq.svg *License*: Public Domain *Contributors*: Unknown, published by Iraqi governemt, vectorized by User:Militaryace based on the work of User:Hoshie

File:Flag of Latvia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Latvia.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of Albania.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Albania.svg *License*: Public Domain *Contributors*: User:Dbenbenn

File:Flag of Uganda.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Uganda.svg *License*: Creative Commons Zero *Contributors*: tobias

File:Flag of Zambia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Zambia.svg *License*: Public Domain *Contributors*: Author: Tobias Jakobs (in the public domain) and User:Zscout370 (Return fire)

File:Flag of Côte d'Ivoire.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Côte_d'Ivoire.svg *License*: Public Domain *Contributors*: User:Jon Harald Søby

File:Flag of Romania.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Romania.svg *License*: Public Domain *Contributors*: AdiJapan

File:Flag of Fiji.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Fiji.svg *License*: Public Domain *Contributors*: Anime Addict AA, Avala, ButterStick, Denelson83, Fred the Oyster, Fry1989, Greentubing, Herbythyme, Homo lupus, Klemen Kocjancic, Krun, Lokal Profil, Ludger1961, Marcus Cyron, Mattes, Multichill, Neq00, Nightstallion, ReconditeRodent, Ricordisamoa, Sam916, Suzuki Auto, Urhixidur, Vzb83, 8 anonymous edits

File:Flag of Papua New Guinea.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Papua_New_Guinea.svg *License*: Public Domain *Contributors*: User:Nightstallion

File:Flag of the Solomon Islands.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Solomon_Islands.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of Tonga.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Tonga.svg *License*: Public Domain *Contributors*: AnonMoos, Badseed, Fry1989, Herbythyme, Homo lupus, Klemen Kocjancic, Krun, Liftarn, Mattes, Neq00, Nightstallion, Pumbaa80, Trockennasenaffe, Wrightbus, 5 anonymous edits

File:Flag of Vanuatu.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Vanuatu.svg *License*: Public Domain *Contributors*: Alkari, Bast64, Cathy Richards, EugeneZelenko, Fry1989, Homo lupus, Klemen Kocjancic, Mattes, Mikiofpersia, Neq00, Nightstallion, OAlexander, PeterSymonds, Vzb83

File:Flag of Guyana.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Guyana.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of Syria.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Syria.svg *License*: Public Domain *Contributors*: see below

File:Flag of Liberia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Liberia.svg *License*: Public Domain *Contributors*: Government of Liberia

File:Flag of Chile.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Chile.svg *License*: Public Domain *Contributors*: Alkari, B1mbo, Cathy Richards, Cycn, David Newton, Dbenbenn, Denelson83, ElmA, Er Komandante, Fibonacci, Fry1989, Fsopolonezcaro, Herbythyme, Huhsunqu, Kallerna, Kanonkas, Klemen Kocjancic, Kyro, MAXXX-309, Mattes, McZusatz, Mozzan, Nagy, Nightstallion, Piastu, Pixeltoo, Pumbaa80, SKopp, Sarang, SiBr4, Srtxg, Sterling.M.Archer, Str4nd, Ultratomio, Vzb83, Xarucoponce, Yakoo, Yonatanh, Zscout370, 50 anonymous edits

File:Flag of Thailand.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Thailand.svg *License*: Public Domain *Contributors*: Zscout370

File:Flag of Venezuela.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Venezuela.svg *License*: Public Domain *Contributors*: Alkari, Bastique, Cesar david rodriguez, Cycn, Denelson83, DerFussi, Fry1989, George McFinnigan, Hedwig in Washington, Herbythyme, Homo lupus, Huhsunqu, Infrogmation, K21edgo, Klemen Kocjancic, Ludger1961, Neq00, Nightstallion, Reisio, Rupert Pupkin, Sarang, SiBr4, ThomasPusch, Unukalhai, Vzb83, Wikisole, Zscout370, 13 anonymous edits

File:Flag of Trinidad and Tobago.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Trinidad_and_Tobago.svg *License*: Public Domain *Contributors*: AnonMoos, Boricuaeddie, Duduziq, Enbéká, Erlenmeyer, Fry1989, Homo lupus, Illegitimate Barrister, Klemen Kocjancic, Madden, Mattes, Nagy, Neq00, Nightstallion, Pumbaa80, S KTT, SKopp, SiBr4, Tomia, 12 anonymous edits

File:Flag of Mozambique.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Mozambique.svg *License*: Public Domain *Contributors*: User:Nightstallion

File:Flag of Denmark.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Denmark.svg *License*: Public Domain *Contributors*: User:Madden

File:Flag of the Dominican Republic.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Dominican_Republic.svg *License*: Public Domain *Contributors*: User:Nightstallion

File:Flag of Equatorial Guinea.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Equatorial_Guinea.svg *License*: Public Domain *Contributors*: Anime Addict AA, Antonsusi, Cycn, Duschgeldrache2, Emc2, Fastily, Fred the Oyster, Fry1989, Homo lupus, Klemen Kocjancic, Maks Stirlitz, Mattes, Neq00, NeverDoING, Nightstallion, OAlexander, Permjak, Pitke, SiBr4, SouthSudan, ThomasPusch, 4 anonymous edits

File:Flag of the Faroe Islands.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Faroe_Islands.svg *License*: Public Domain *Contributors*: User:IceKarma

File:Flag of the Philippines.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Philippines.svg *License*: Public Domain *Contributors*: User:Achim1999

File:Flag of Greenland.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Greenland.svg *License*: Public Domain *Contributors*: Jeffrey Connell (IceKarma)

File:Flag of Iceland.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Iceland.svg *License*: Public Domain *Contributors*: User:Zscout370, User:Ævar Arnfjörð Bjarmason

File:Flag of Myanmar.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Myanmar.svg *License*: Public Domain *Contributors*: *drew, AnonMoos, Cathy Richards, CommonsDelinker, Cycn, Duduziq, Fry1989, Gunkarta, Homo lupus, Idh0854, Josegeographic, Klemen Kocjancic, Legnaw, Mason Decker, Mattes, Neq00, Nightstallion, Pixeltoo, Rfc1394, Rodejong, SeNeKa, SiBr4, Stevanb, Takahara Osaka, Techman224, ThomasPusch, UnreifeKirsche, Vividuppers, WikipediaMaster, Winzipas, Xiengyod, Zscout370, 白布飄扬, 10 anonymous edits

File:Flag of Yemen.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Yemen.svg *License*: Public Domain *Contributors*: Anime Addict AA, AnonMoos, David Levy, Duduziq, Erlenmeyer, F. F. Fjodor, Flad, Fry1989, Homo lupus, Klemen Kocjancic, Krun, Neq00, Nightstallion, Pitke, Reisio, Rodejong, SiBr4, Themadchopper, ThomasPusch, Urmas, Wikiborg, Zscout370, 4 anonymous edits

File:Flag of Tunisia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Tunisia.svg *License*: Public Domain *Contributors*: entraîneur: BEN KHALIFA WISSAM

File:Flag of Gabon.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Gabon.svg *License*: Public Domain *Contributors*: User:Gabbe, User:SKopp

File:Flag of the Republic of the Congo.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Republic_of_the_Congo.svg *License*: Public Domain *Contributors*: Anime Addict AA, Antemister, Courcelles, Denelson83, Erlenmeyer, Estrilda, FischersFritz, Fry1989, Homo lupus, Klemen Kocjancic, LA2, Madden, Mattes, Moyogo, Neq00, Nightstallion, Persiana, Pitke, Ratatosk, Romaine, SiBr4, ThomasPusch, Thuresson, 6 anonymous edits

File:Flag of Mali.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Mali.svg *License*: Public Domain *Contributors*: User:SKopp

File:Flag of Guatemala.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Guatemala.svg *License*: Public Domain *Contributors*: User:K21edgo

File:Flag of Iran.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Iran.svg *License*: Public Domain *Contributors*: Various

File:Flag of Libya.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Libya.svg *License*: Public Domain *Contributors*: Various

File:Flag of the United Arab Emirates.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_United_Arab_Emirates.svg *License*: Public Domain *Contributors*: Anime Addict AA, Avala, Dbenbenn, Duduziq, F l a n k e r, Fry1989, Fukaumi, Gryffindor, Guanaco, Homo lupus, Kacir, Klemen Kocjancic, Krun, Ludger1961, Madden, Neq00, Nightstallion, Piccadilly Circus, Pmsyyz, RamzyAbueita, Ricordisamoa, Zscout370, 5 anonymous edits

File:Flag of Qatar.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Qatar.svg *License*: Public Domain *Contributors*: (of code) cs:User:-xfi-

File:Flag of South Korea.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_South_Korea.svg *License*: Public Domain *Contributors*: Various

File:Flag of the Seychelles.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_Seychelles.svg *License*: unknown *Contributors*: -

File:Flag of Saudi Arabia.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Saudi_Arabia.svg *License*: Public Domain *Contributors*: Alkari, Ancintosh, Anime Addict AA, AnonMoos, Bobika, Brian Ammon, CommonsDelinker, Cycn, Denelson83, Duduziq, Ekabhishek, Er Komandante, Fabioravanelli, Fry1989, Herbythyme, Homo lupus, INeverCry, Jeff G., Klemen Kocjancic, Lokal Profil, Love Krittaya, Love monju, Mattes, Menasim, Mnmazur, Mohammed alkhater, Nard the Bard, Nightstallion, Palosirkka, Pitke, Pmsyyz, Ranveig, Ratatosk, Reisio, Ricordisamoa, Saibo, SiBr4, Wouterhagens, Zscout370, Zyido, 13 anonymous edits

File:Flag of Belarus.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_Belarus.svg *License*: Public Domain *Contributors*: Zscout370

File:Flag of East Timor.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Flag_of_East_Timor.svg *License*: Public Domain *Contributors*: User:SKopp

File:NetworkOperations.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:NetworkOperations.jpg *License*: Public Domain *Contributors*: Original uploader was Gsmith1of2 at en.wikipedia

File:Indiana University Data Center - P1100134.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Indiana_University_Data_Center_-_P1100134.JPG *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: User:Vmenkov

File:Datacenter-telecom.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Datacenter-telecom.jpg *License*: GNU Free Documentation License *Contributors*: User:Gmaxwell

File:Rack001.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Rack001.jpg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: User:Jfreyre

File:CRAC Cabinets 2.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:CRAC_Cabinets_2.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Robert.Harker

File:Under Floor Cable Runs Tee.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Under_Floor_Cable_Runs_Tee.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Robert.Harker

File:Cabinet Asile.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Cabinet_Asile.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Robert.Harker

File:Datacenter Backup Batteries.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Datacenter_Backup_Batteries.jpg *License*: Creative Commons Attribution 3.0 *Contributors*: Jelson25

File:FM200 Three.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:FM200_Three.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Robert.Harker

File:Google Data Center, The Dalles.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Google_Data_Center,_The_Dalles.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Visitor7

File:Paris servers DSC00190.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Paris_servers_DSC00190.jpg *License*: Creative Commons Attribution-ShareAlike 1.0 Generic *Contributors*: David Monniaux. Photo taken by myself with a cellular phone. Copyright © 2005

File:IBMPortableModularDataCenter.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:IBMPortableModularDataCenter.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Raysonho @ Open Grid Scheduler / Grid Engine

Image:VirtualFacility.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:VirtualFacility.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Ben Ben, Stefan hendrickx

File:virtuoso-logo-sm.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Virtuoso-logo-sm.png *License*: GNU Free Documentation License *Contributors*: Bkell, DeirdreGerhardt

File:conductor-sm.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Conductor-sm.png *License*: GNU Free Documentation License *Contributors*: DeirdreGerhardt

File:Linnaeus - Regnum Animale (1735).png *Source*: http://en.wikipedia.org/w/index.php?title=File:Linnaeus_-_Regnum_Animale_(1735).png *License*: Public Domain *Contributors*: User:Fastfission

File:6123034166 card catalog.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:6123034166_card_catalog.jpg *License*: Creative Commons Attribution-Sharealike 2.0 *Contributors*: Maya West from Portland, Oregon, USA

File:Sta-eulalia.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Sta-eulalia.jpg *License*: GNU Free Documentation License *Contributors*: Original uploader was Montrealais at en.wikipedia

File:Article catedral pantalla estreta.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Article_catedral_pantalla_estreta.png *License*: GNU Free Documentation License *Contributors*: Pere López (author of screenshot)

File:OMG Headquarters.jpeg *Source*: http://en.wikipedia.org/w/index.php?title=File:OMG_Headquarters.jpeg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Omgmarketing

File:Business-Motivation-Model-top.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Business-Motivation-Model-top.gif *License*: Creative Commons Attribution 3.0 *Contributors*: David Morris

Image:ADM KDM.png *Source*: http://en.wikipedia.org/w/index.php?title=File:ADM_KDM.png *License*: Public Domain *Contributors*: User:Equilibrioception

Image:LOM base schema.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LOM_base_schema.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: en:User:Rjgodoy

License