Database

A **database** is an organized collection of data.[1] It is the collection of schemes, tables, queries, reports, views and other objects. The data is typically organized to model aspects of reality in a way that supports processes requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

A **database management system** (**DBMS**) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS. Database management systems are often classified according to the database model that they support; the most popular database systems since the 1980s have all supported the relational model as represented by the SQL language. Sometimes a DBMS is loosely referred to as a 'database'.

1 Terminology and overview

Formally, a “database” refers to a set of related data and the way it is organized. Access to this data is usually provided by a “database management system” (DBMS) consisting of an integrated set of computer software that allows users to interact with one or more databases and provides access to all of the data contained in the database (although restrictions may exist that limit access to particular data). The DBMS provides various functions that allow entry, storage and retrieval of large quantities of information as well as provides ways to manage how that information is organized.

Because of the close relationship between them, the term “database” is often used casually to refer to both a database and the DBMS used to manipulate it.

Outside the world of professional information technology, the term *database* is often used to refer to any collection of related data (such as a spreadsheet or a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.[2]

Existing DBMSs provide various functions that allow management of a database and its data which can be classified into four main functional groups:

- **Data definition** – Creation, modification and removal of definitions that define the organization of the data.
- **Update** – Insertion, modification, and deletion of the actual data.[3]
- **Retrieval** – Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form basically the same as it is stored in the database or in a new form obtained by altering or combining existing data from the database.[4]
- **Administration** – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information that has been corrupted by some event such as an unexpected system failure.[5]

Both a database and its DBMS conform to the principles of a particular database model.[6] “Database system” refers collectively to the database model, database management system, and database.[7]

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. Since DBMSs comprise a significant economical market, computer and storage vendors often take into account DBMS requirements in their own development plans.

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

2 Applications

Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers (see Enterprise software).

Databases are used to hold administrative information and more specialized data, such as engineering data or economic models. Examples of database applications include computerized library systems, flight reservation systems and computerized parts inventory systems.

Application areas of DBMS

1. Banking: For customer information, accounts, and loans, and banking transactions.

2. Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner - terminals situated around the world accessed the central database system through phone lines and other data networks.

3. Universities: For student information, course registrations, and grades.

4. Credit card transactions: For purchases on credit cards and generation of monthly statements.

5. Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

6. Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.

7. Sales: For customer, product, and purchase information.

8. Manufacturing: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses / stores, and orders for items.

9. Human resources: For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.[8]

3 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.[9] 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 that performs many of the functions of a general-purpose DBMS such as the insertion and deletion of messages composed of various items of data or associating messages with a particular email address; but these functions are limited to what is required to handle email and don't provide the user with the all of the functionality that would be available using a general-purpose DBMS.

Many other 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.

4 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,[10] 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 become 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 2015 they remain dominant : IBM DB2, Oracle, mySQL and SQL server are the top DBMS.[11] The dominant database language, standardised SQL for the relational model, has influenced database languages for other data models.

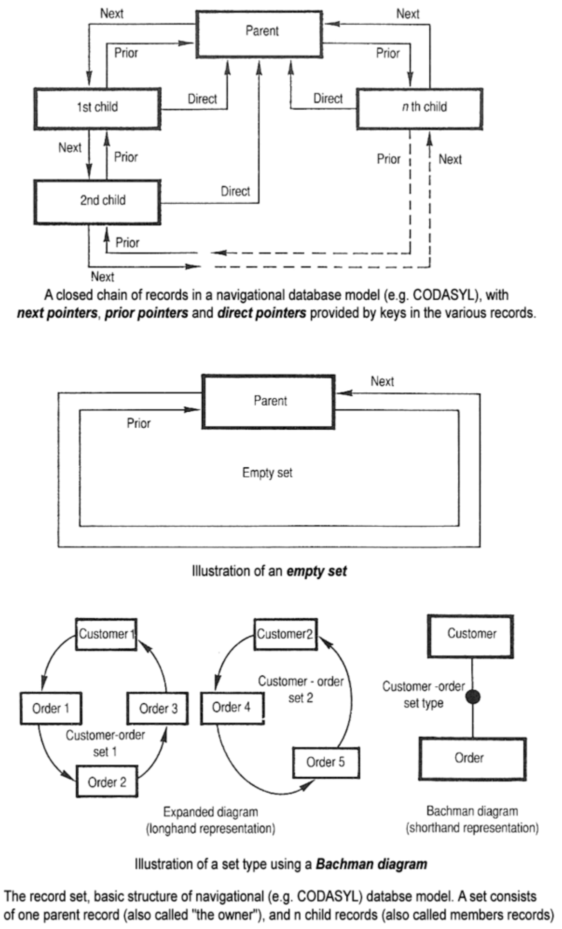
Object databases were developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.

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.

4.1 1960s, navigational DBMS

Further information: Navigational database

The introduction of the term *database* coincided with


Next
Parent
Next
Prior
Prior
1st child
Direct
Direct
n th child
Next
Prior
Prior
Next
2nd child
Direct
Prior
Next
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.
Parent
Next
Prior
Empty set
Illustration of an empty set
Customer 1
Customer2
Customer
Order 1
Order 3
Order 4
Customer - order set 2
Customer -order set type
Customer-order set 1
Order 5
Order
Order 2
Expanded diagram (longhand representation)
Bachman diagram (shorthand representation)
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

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[12] a 1962 report by the System Development Corporation of California as the first to use the term "data-base" in a specific technical sense.

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 the Database 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:

1. Use of a primary key (known as a CALC key, typically implemented by hashing)
2. Navigating relationships (called sets) from one record to another
3. Scanning all the records in a sequential order

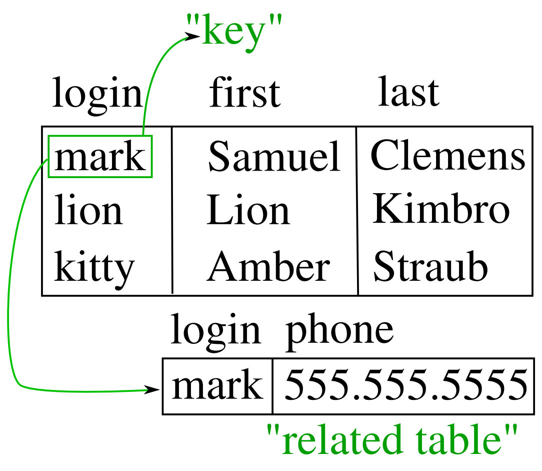
Later systems added B-Trees to provide alternate access paths. Many CODASYL databases also added a very straightforward query language. However, in the final tally, CODASYL was very complex and required significant training and effort to produce useful applications.

IBM also had their own DBMS in 1968, known as Information Management System (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 classified as a hierarchical database. IDMS and Cincom Systems' TOTAL database are classified as network databases. IMS remains in use as of 2014.[13]

4.2 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*.[14]

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.



In the relational model, records are "linked" using virtual keys not stored in the database but defined as needed between the data contained in the records.

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.

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[15] based on D.L. Childs' Set-Theoretic Data model.[16][17][18] 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.[19] The system remained in production until 1998.

4.3 Integrated approach

Main article: Database machine

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).

4.4 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 – 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.

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.

4.5 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."[20] dBASE was one of the top selling software titles in the 1980s and early 1990s.

4.6 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.[21] 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.

4.7 2000s, NoSQL and NewSQL

Main articles: 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. XML databases are mostly used in enterprise database management, where XML is being used as the machine-to-machine data interoperability standard. XML database management systems include commercial software MarkLogic and Oracle Berkeley DB XML, and a free use software Clusterpoint

Distributed XML/JSON Database. All are enterprise software database platforms and support industry standard ACID-compliant transaction processing with strong database consistency characteristics and high level of database security.[22][23][24]

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. The most popular NoSQL systems include MongoDB, Couchbase, Riak, Memcached, Redis, CouchDB, Hazelcast, Apache Cassandra and HBase,[25] which are all open-source software products.

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 reduced level of data consistency.

NewSQL is a class of modern relational databases that aims to provide the same scalable performance of NoSQL systems for online transaction processing (read-write) workloads while still using SQL and maintaining the ACID guarantees of a traditional database system. Such databases include ScaleBase, Clustrix, EnterpriseDB, MemSQL, NuoDB[26] and VoltDB.

5 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).

6 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.[27] 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 extracting, 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.[28]

- **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.

- An array DBMS is a kind of NoSQL DBMS that allows to model, store, and retrieve (usually large) multi-dimensional arrays such as satellite images and climate simulation output.

- 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[29][30]) 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.

7 Design and modeling

Main article: Database design

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.

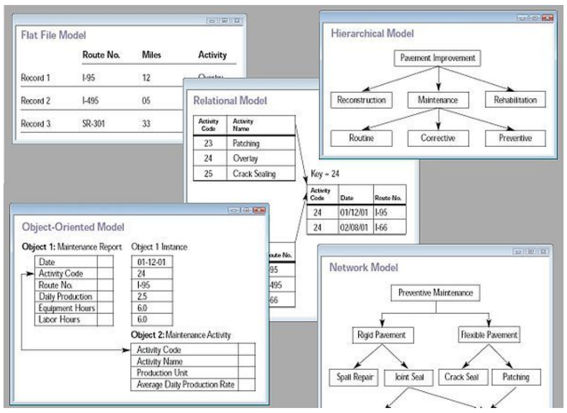
The final stage of database design is to make the decisions that affect performance, scalability, recovery, security, and the like. This is often called *physical database design*. A key goal during this stage is data independence, meaning that the decisions made for performance optimization purposes 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.

7.1 Models

Main article: 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 (or the SQL approximation of relational), which uses a table-based for-



Collage of five types of database models mat.

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

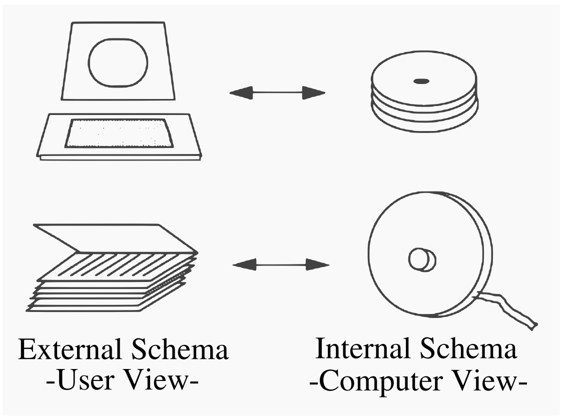
Other models include:

- Associative model
- Multidimensional model
- Array model
- Multivalue model
- Semantic model
- XML database

7.2 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.[32] 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[31]

- 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.[32]

8 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.[33][34]

- 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.[35]

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

9 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.

9.1 Storage

Main articles: Computer data storage and Database engine

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.

9.1.1 Materialized views

Main article: Materialized view

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.

9.1.2 Replication

Main article: Database 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.

9.2 Security

Main article: Database 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.

9.3 Transactions and concurrency

Further information: Concurrency control

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.

9.4 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.

9.5 Building, maintaining, and tuning

Main article: Database 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.

9.6 Backup and restore

Main article: Backup

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.

9.7 Static Analysis

Static analysis techniques for software verification can be applied also in the scenario of query languages. In particular, the *Abstract interpretation framework has been extended to the field of query languages for relational databases as a way to support sound approximation techniques.[36] The semantics of query languages can be tuned according to suitable abstractions of the concrete domain of data. The abstraction of relational database system has many interesting applications, in particular, for security purposes, such as fine grained access control, watermarking, etc.

9.8 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.

10 See also

Main article: Outline of databases

- Comparison of database tools

- Comparison of object database management systems
- Comparison of object-relational database management systems
- Comparison of relational database management systems
- Data hierarchy
- Data bank
- Data store
- Database theory
- Database testing
- Database-centric architecture
- Question-focused dataset

11 References

[1] "Database - Definition of database by Merriam-Webster". *merriam-webster.com*.

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

[3] "Update - Definition of update by Merriam-Webster". *merriam-webster.com*.

[4] "Retrieval - Definition of retrieval by Merriam-Webster". *merriam-webster.com*.

[5] "Administration - Definition of administration by Merriam-Webster". *merriam-webster.com*.

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

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

[8] Garcia-Molina, Hector; Ullman, Jeffrey D.; Widom, Jennifer (2009). "The Worlds of Database Systems" (PDF). *Database systems: the complete book* (2nd ed.). Upper Saddle River, N.J.: Pearson Prentice Hall. ISBN 978-0131873254.

[9] Raul F. Chong, Michael Dang, Dwaine R. Snow, Xiaomei Wang (3 July 2008). "Introduction to DB2". Retrieved 17 March 2013.. This article quotes a development time of 5 years involving 750 people for DB2 release 9 alone

[10] C. W. Bachmann (November 1973), "The Programmer as Navigator" (PDF), *CACM* (Turing Award Lecture 1973)

[11] "TOPDB Top Database index". *pypl.github.io*.

[12] "database, n". *OED Online*. Oxford University Press. June 2013. Retrieved July 12, 2013.

[13] IBM Corporation. "IBM Information Management System (IMS) 13 Transaction and Database Servers delivers high performance and low total cost of ownership". Retrieved Feb 20, 2014.

[14] Codd, E.F. (1970)."A Relational Model of Data for Large Shared Data Banks". In: *Communications of the ACM* 13 (6): 377–387.

[15] William Hershey and Carol Easthope, "A set theoretic data structure and retrieval language", Spring Joint Computer Conference, May 1972 in *ACM SIGIR Forum*, Volume 7, Issue 4 (December 1972), pp. 45–55, DOI=10.1145/1095495.1095500

[16] Ken North, "Sets, Data Models and Data Independence", *Dr. Dobb's*, 10 March 2010

[17] *Description of a set-theoretic data structure*, D. L. Childs, 1968, Technical Report 3 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

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

[19] *MICRO Information Management System (Version 5.0) Reference Manual*, 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

[20] Interview with Wayne Ratliff. The FoxPro History. Retrieved on 2013-07-12.

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

[22] "Oracle Berkeley DB XML" (PDF). Retrieved 10 March 2015.

[23] "ACID Transactions, MarkLogic". Retrieved 10 March 2015.

[24] "Clusterpoint Database at a Glance". Retrieved 10 March 2015.

[25] "DB-Engines Ranking". January 2013. Retrieved 22 January 2013.

[26] Proctor, Seth (2013). "Exploring the Architecture of the NuoDB Database, Part 1". Retrieved 2013-07-12.

[27] "TeleCommunication Systems Signs up as a Reseller of TimesTen; Mobile Operators and Carriers Gain Real-Time Platform for Location-Based Services". *Business Wire*. 2002-06-24.

[28] Graves, Steve. "COTS Databases For Embedded Systems", *Embedded Computing Design* magazine, January 2007. Retrieved on August 13, 2008.

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

[30] "OWL DL Semantics". Retrieved 10 December 2010.

[31] itl.nist.gov (1993) *Integration Definition for Information Modeling (IDEFIX)*. 21 December 1993.

[32] Date 1990, pp. 31–32

[33] Chapple, Mike. "SQL Fundamentals". *Databases*. About.com. Retrieved 2009-01-28.

[34] "Structured Query Language (SQL)". International Business Machines. October 27, 2006. Retrieved 2007-06-10.

[35] Wagner, Michael (2010), "1. Auflage", *SQL/XML:2006 – Evaluierung der Standardkonformität ausgewählter Datenbanksysteme*, Diplomica Verlag, ISBN 3-8366-9609-6

[36] R.Halder and A.Cortesi, Abstract Interpretation of Database Query Languages. COMPUTER LANGUAGES, SYSTEMS & STRUCTURES, vol. 38(2), pp. 123-—157, Elsevier Ed. (ISSN 1477-8424)

12 Further reading

- Ling Liu and Tamer M. Özsu (Eds.) (2009). "Encyclopedia of Database Systems, 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*
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts*
- Discussion on database systems,
- 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

13 External links

- Database at DMOZ
- DB File extension – informations about files with DB extension

14 Text and image sources, contributors, and licenses

14.1 Text

14.2 Images

- **File:Wikibooks-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fa/Wikibooks-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.
- **File:Wikinews-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/24/Wikinews-logo.svg *License:* CC BY-SA 3.0 *Contributors:* This is a cropped version of Image:Wikinews-logo-en.png. *Original artist:* Vectorized by Simon 01:05, 2 August 2006 (UTC) Updated by Time3000 17 April 2007 to use official Wikinews colours and appear correctly on dark backgrounds. Originally uploaded by Simon.
- **File:Wikiquote-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fa/Wikiquote-logo.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Wikisource-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/4c/Wikisource-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Rei-artur *Original artist:* Nicholas Moreau
- **File:Wikiversity-logo-Snorky.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/1b/Wikiversity-logo-en.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Snorky
- **File:Wiktionary-logo-en.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f8/Wiktionary-logo-en.svg *License:* Public domain *Contributors:* Vector version of Image:Wiktionary-logo-en.png. *Original artist:* Vectorized by Fvasconcellos (talk · contribs), based on original logo tossed together by Brion Vibber

14.3 Content license