

# Database management system

From Wikipedia, the free encyclopedia

A **database management system (DBMS)** is computer software designed for the purpose of managing databases. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems.

## Contents

- 1 Description
- 2 Features and Abilities Of DBMS
  - 2.1 Query ability
  - 2.2 Backup and replication
  - 2.3 Rule enforcement
  - 2.4 Security
  - 2.5 Computation
  - 2.6 Change and access logging
  - 2.7 Automated optimization
  - 2.8 Meta-data repository
  - 2.9 History
  - 2.10 Navigational DBMS
  - 2.11 Relational DBMS
  - 2.12 SQL DBMS
- 3 See also
- 4 References
- 5 External references

## Description

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

1. A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
  - The four most common types of organizations are the hierarchical, network, relational and object models. Inverted lists and other methods are also used. A given database management system may provide one or more of the four 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).

- The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.
2. Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device (which implies relatively slow access compared to volatile main memory).
  3. A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.
    - It also controls the security of the database.
    - 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. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.
  4. A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).
    - It also maintains the integrity of the data in the database.
    - The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. New categories of data can be added to the database without disruption to the existing system.

Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

Database servers are specially designed computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with RAID disk arrays used for stable storage. Connected to one or more servers via a high-speed channel, hardware database accelerators are also used in large volume transaction processing environments.

DBMSs are found at the heart of most database applications. Sometimes DBMSs are built around a private multitasking kernel with built-in networking support although nowadays these functions are left to the operating system.

## Features and Abilities Of DBMS

One can characterize a DBMS as an "attribute management system" where attributes are small chunks of information that describe something. For example, "color" is an attribute of a car. The value of the attribute may be a color such as "red", "blue", "silver", etc. Lately databases have been modified to accept large or unstructured (pre-digested or pre-categorized) information as well, such as images and text documents. However, the main focus is still on descriptive attributes.

DBMS roll together frequently-needed services or features of attribute management. This allows one to get powerful functionality "out of the box" rather than program each from scratch or add and integrate them incrementally. Such features include:

### Query ability

Querying is the process of requesting attribute information from various perspectives and combinations of factors. Example: "How many 2-door cars in Texas are green?"

A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data. It also controls the security of the database. 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. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.

### Backup and replication

Copies of attributes need to be made regularly in case primary disks or other equipment fails. A periodic copy of attributes may also be created for a distant organization that cannot readily access the original. DBMS usually provide utilities to facilitate the process of extracting and disseminating attribute sets.

When data is replicated between database servers, so that the information remains consistent throughout the database system and users cannot tell or even know which server in the DBMS

they are using, the system is said to exhibit replication transparency.

## Rule enforcement

Often one wants to apply rules to attributes so that the attributes are clean and reliable. For example, we may have a rule that says each car can have only one engine associated with it (identified by Engine Number). If somebody tries to associate a second engine with a given car, we want the DBMS to deny such a request and display an error message. However, with changes in the model specification such as, in this example, hybrid gas-electric cars, rules may need to change. Ideally such rules should be able to be added and removed as needed without significant data layout redesign.

## Security

Often it is desirable to limit who can see or change which attributes or groups of attributes. This may be managed directly by individual, 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.

## Computation

There are common computations requested on attributes such as counting, summing, averaging, sorting, grouping, cross-referencing, etc. Rather than have each computer application implement these from scratch, they can rely on the DBMS to supply such calculations.

## Change and access logging

Often one wants to know who accessed what attributes, what was changed, and when it was changed. Logging services allow this by keeping a record of access occurrences and changes.

## Automated optimization

If there are frequently occurring usage patterns or requests, some DBMS can adjust themselves to improve the speed of those interactions. In some cases the DBMS will merely provide tools to monitor performance, allowing a human expert to make the necessary adjustments after reviewing the statistics collected.....

## Meta-data repository

Metadata (also spelled **meta-data**) is data describing data. For example, a listing that describes what attributes are allowed to be in data sets is called "meta-information".

## History

Databases have been in use since the earliest days of electronic computing. Unlike modern systems which can be applied to widely different databases and needs, the vast majority of older systems were tightly linked to the custom databases in order to gain speed at the expense of flexibility. Originally DBMSs were found only in large organizations with the computer hardware needed to support large data sets.

## Navigational DBMS

As computers grew in capability, this trade-off became increasingly unnecessary and a number of general-purpose database systems emerged; by the mid-1960s there were a number of such systems in commercial use. Interest in a standard began to grow, and Charles Bachman, author of one such product, **IDS**, founded the *Database Task Group* within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971 they delivered their standard, which generally became known as the **CodasyI approach**, and soon there were a number of commercial products based on it available.

The CodasyI approach was based on the "manual" navigation of a linked data set which was formed into a large network. When the database was first opened, the program was handed back a link to the first record in the database, which also contained pointers to other pieces of data. To find any particular record the programmer had to step through these pointers one at a time until the required record was returned. Simple queries like "find all the people in Sweden" required the program to walk the entire data set and collect the matching results. There was, essentially, no concept of "find" or "search". This might sound like a serious limitation today, but in an era when the data was most often stored on magnetic tape such operations were too expensive to contemplate anyway.

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 CodasyI, but used a strict hierarchy for its model of data navigation instead of CodasyI's network model.

Both concepts later became known as **navigational databases** due to the way data was accessed, and Bachman's 1973 Turing Award award presentation was *The Programmer as Navigator*.

IMS is classified as a hierarchical database. IDS and IDMS (both CODASYL databases) as well as CINCOMs TOTAL database are classified as network databases.

## 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 CodasyI approach, notably the lack of a "search" facility which was becoming increasingly useful. 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*.<sup>[1]</sup>

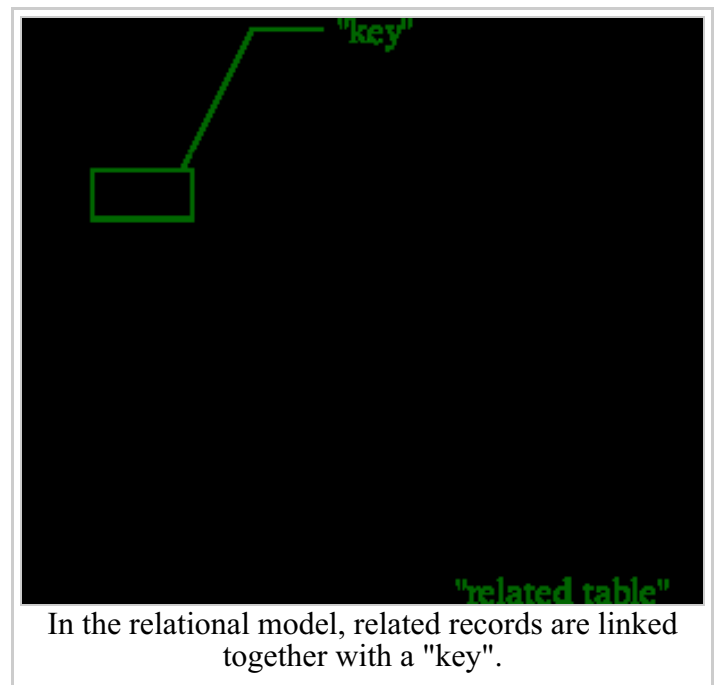
In this paper, he described a new system for storing and working with large databases. Instead of records being stored in some sort of linked list of free-form records as in Codasyl, Codd's idea was to use a "table" of fixed-length records. A linked-list system would be very inefficient when storing "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, with optional elements being moved out of the main table to where they would take up room only if needed.

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 (or *related*) 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 "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, using student programmers to produce code. Beginning in 1973, INGRES delivered its first test products which were generally ready for widespread use in 1979. During this time, a number of people had moved "through" the group — perhaps as many as 30 people worked on the project, about five at a time. INGRES was similar to System R in a number of ways, including the use of a "language" for data access, known as QUEL — QUEL was in fact relational, having been based on Codd's own Alpha language, but has since been corrupted to follow SQL, thus violating much the same concepts of the relational model as SQL itself.

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

In 1968, the [2] (<http://www.eecs.umich.edu/db/>) University of Michigan began development of the Micro DBMS relational database management system. It was used to manage very large data sets by the US Department of Labor, the Environmental Protection Agency and researchers from University of Alberta, the University of Michigan and Wayne State University. It ran on mainframe computers using Michigan Terminal System. The system remained in production until 1996.

## SQL DBMS

IBM started working on a prototype system loosely based on Codd's concepts as **System R** in the early 1970s — unfortunately, System R was conceived as a way of proving Codd's ideas unimplementable, and thus the project was delivered to a group of programmers who were not under Codd's supervision, never understood his ideas fully and ended up violating several fundamentals of the relational model. The first "quickie" version was ready in 1974/5, and work then started on multi-table systems in which the data could be broken down so that all of the data for a record (much of which is often 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).

Many of the people involved with INGRES became convinced of the future commercial success of such systems, and formed their own companies to commercialize the work but with an SQL interface. Sybase, Informix, NonStop SQL and eventually Ingres itself were all being sold as offshoots to the original INGRES product in the 1980s. Even Microsoft SQL Server is actually a re-built version of Sybase, and thus, INGRES. Only Larry Ellison's Oracle started from a different chain, based on IBM's papers on System R, by beating them 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 primarily 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-70s 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 DBMS.

## See also

- Column-oriented DBMS
  - Data warehouse
  - Database-centric architecture
  - Directory service
  - Distributed database management system
  - Hierarchical model
  - Navigational database
  - Network model
  - Object model
  - Object database (OODBMS)
  - Object-relational database (ORDBMS)
  - Relational model (RDBMS model)
  - Relational database management system (RBDMS)
  - Run Book Automation (RBA)
  - Comparison of relational database management systems
- 
- SQL is a language for database management

## References

- <sup>^</sup> [1] (<http://www.acm.org/classics/nov95/toc.html>) Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* 13 (6): 377–387.

## External references

- Databases Information (<http://www.netegle.com/databases.html>)
- [3] (<http://portal.acm.org/citation.cfm?id=1095495.1095500>) Association for Computing Machinery SIGIR Forum archive Volume 7 , Issue 4
- [4] (<http://www.tomandmaria.com/tom/Writing/VeritableBucketOfFactsSIGMOD.pdf>) The origins of the data base concept, early DBMS systems including IDS and IMS, the Data Base Task Group, and the hierarchical, network and relational data models are discussed in Thomas Haigh, "'A Veritable Bucket of Facts:' Origins of the Data Base

- Management System," ACM SIGMOD Record 35:2 (June 2006).
- [5] (<http://www.storage-area-networks.info/san-databases.html>) How Database Systems Share Storage

*This article was originally based on material from the Free On-line Dictionary of Computing, which is licensed under the GFDL.*

Retrieved from "[http://en.wikipedia.org/wiki/Database\\_management\\_system](http://en.wikipedia.org/wiki/Database_management_system)"

Categories: All articles with unsourced statements | Articles with unsourced statements since February 2007 | Databases | Database management systems

- 
- This page was last modified 09:27, 9 December 2007.
  - All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.)  
Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a U.S. registered 501(c)(3) tax-deductible nonprofit charity.