DBMS is a set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMSs are categorized according to their data structures or types. The DBMS accepts requests for data from an application program and instructs the operating system to transfer the appropriate data. The queries and responses must be submitted and received according to a format that conforms to one or more applicable protocols. 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.

Database servers are 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. DBMSs are found at the heart of most database applications.

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.

1960s Navigational DBMS

As computers grew in speed and capability, 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, Integrated Data Store (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 "Codasyl approach", and soon there were a number of commercial products based on it available.

The Codasyl 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 India" 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 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 award presentation was The
Programmer as Navigator. IMS is classified as a hierarchical database. IMS and IDMS, both CODASYL databases, are classified as network databases.

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.*\[1\]

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.

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 (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.

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 (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).

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, and beat IBM to market when the first version was released in 1978.

1980s Object Oriented Databases

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 relationships between data to be relation to objects and their attributes and not to individual fields.[2]

DBMS building blocks

A DBMS includes four main parts: modeling language, data structure, database query language, and transaction mechanisms:

Components of DBMS

- **DBMS Engine** accepts logical request from the various other DBMS subsystems, converts them into physical equivalents, and actually accesses the database and data dictionary as they exist on a storage device.
- **Data Definition Subsystem** helps user to create and maintain the data dictionary and define the structure of the files in a database.
- **Data Manipulation Subsystem** helps user to add, change, and delete information in a database and query it for valuable information. Software tools within the data manipulation subsystem are most often the primary interface between user and the
information contained in a database. It allows user to specify its logical information requirements.

- **Application Generation Subsystem** contains facilities to help users to develop transaction-intensive applications. It usually requires that user perform a detailed series of tasks to process a transaction. It facilitates easy-to-use data entry screens, programming languages, and interfaces.

- **Data Administration Subsystem** helps users to manage the overall database environment by providing facilities for backup and recovery, security management, query optimization, concurrency control, and change management.

**Modeling language**

A data modeling language to define the schema of each database hosted in the DBMS, according to the DBMS database model. The four most common types of models are the:

- hierarchical model,
- network model,
- relational model, and
- object model.

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.

Before the database management approach, organizations relied on file processing systems to organize, store, and process data files. End users became aggravated with file processing because data is stored in many different files and each organized in a different way. Each file was specialized to be used with a specific application. Needless to say, file processing was bulky, costly and nonflexible when it came to supplying needed data accurately and promptly. Data redundancy is an issue with the file processing system because the independent data files produce duplicate data so when updates were needed each separate file would need to be updated. Another issue is the lack of data integration. The data is dependent on other data to organize and store it. Lastly, there was not any consistency or standardization of the data in a file processing system which makes maintenance difficult. For all these reasons, the database management approach was produced. Database management systems (DBMS) are designed to use one of five database structures to provide simplistic access to information stored in databases. The five database structures are hierarchical, network, relational, multidimensional and object-oriented models.

The hierarchical structure was used in early mainframe DBMS. Records’ relationships form a treelike model. This structure is simple but nonflexible because the relationship is confined to a one-to-many relationship. IBM’s IMS system and the RDM Mobile are examples of a
A hierarchical database system with multiple hierarchies over the same data. RDM Mobile is a newly designed embedded database for a mobile computer system. The hierarchical structure is used primarily today for storing geographic information and file systems.

The network structure consists of more complex relationships. Unlike the hierarchical structure, it can relate to many records and accesses them by following one of several paths. In other words, this structure allows for many-to-many relationships.

The relational structure is the most commonly used today. It is used by mainframe, midrange and microcomputer systems. It uses two-dimensional rows and columns to store data. The tables of records can be connected by common key values. While working for IBM, E.F. Codd designed this structure in 1970. The model is not easy for the end user to run queries with because it may require a complex combination of many tables.

The multidimensional structure is similar to the relational model. The dimensions of the cube looking model have data relating to elements in each cell. This structure gives a spreadsheet like view of data. This structure is easy to maintain because records are stored as fundamental attributes, the same way they’re viewed and the structure is easy to understand. Its high performance has made it the most popular database structure when it comes to enabling online analytical processing (OLAP).

The object oriented structure has the ability to handle graphics, pictures, voice and text, types of data, without difficulty unlike the other database structures. This structure is popular for multimedia Web-based applications. It was designed to work with object-oriented programming languages such as Java.

**Data structure**

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**).

**Database query language**

A **database query language** and report writer allows 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.
Transaction mechanism

A database transaction mechanism ideally guarantees 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 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).

DBMS simple definition

Data base management system is the system in which related data is stored in an "efficient" and "compact" manner. Efficient means that the data which is stored in the DBMS is accessed in very quick time and compact means that the data which is stored in DBMS covers very less space in computer's memory. In above definition the phrase "related data" is used which means that the data which is stored in DBMS is about some particular topic.

Throughout recent history specialized databases have existed for scientific, geospatial, imaging, document storage and like uses. Functionality drawn from such applications has lately begun appearing in mainstream DBMSs as well. However, the main focus there, at least when aimed at the commercial data processing market, is still on descriptive attributes on repetitive record structures.

Thus, the DBMSs of today roll together frequently needed services or features of attribute management. By externalizing such functionality to the DBMS, applications effectively share code with each other and are relieved of much internal complexity. Features commonly offered by database management systems 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 allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.

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.

Current trends

In 1998, database management was in need of new style databases to solve current database management problems. Researchers realized that the old trends of database management were becoming too complex and there was a need for automated configuration and management [5]. Surajit Chaudhuri, Gerhard Weikum and Michael Stonebraker, were the pioneers that dramatically affected the thought of database management systems [5]. They believed that database management needed a more modular approach and that there are so many specifications needs for various users [5]. Since this new development process of database management we currently have endless possibilities. Database management is no longer limited to “monolithic entities” [5]. Many solutions have developed to satisfy individual needs of users. Development of numerous database options has created flexible solutions in database management.

Today there are several ways database management has affected the technology world as we know it. Organizations demand for directory services has become an extreme necessity as organizations grow. Businesses are now able to use directory services that provided prompt searches for their company information [5]. Mobile devices are not only able to store contact information of users but have grown to bigger capabilities. Mobile technology is able to cache large information that is used for computers and is able to display it on smaller devices [5]. Web searches have even been affected with database management. Search engine queries are able to locate data within the World Wide Web [5]. Retailers have also benefited from the developments with data warehousing. These companies are able to record customer transactions made within their business [5]. Online transactions have become tremendously popular with the e-business world. Consumers and businesses are able to make payments securely on company websites. None of these current developments would have been possible without the evolution of database management. Even with all the progress and current trends of database management, there will always be a need for new development as specifications and needs grow.