DATABASE MANAGEMENT SYSTEM

Sweta Singh

Assistant Professor, Faculty of Management Studies, BHU, Varanasi, India
E-mail: sweta.v.singh27@gmail.com

ABSTRACT

Today, more than at any previous time, the success of an organization depends on its ability to acquire accurate and timely data about its operations, to manage this data effectively, and to use it to analyse and guide its activities. Phrases such as the information superhighway have become ubiquitous, and information processing is a rapidly growing multibillion dollar industry.

The amount of information available to us is literally exploding, and the value of data as an organizational asset is widely recognized. Yet without the ability to manage this vast amount of data, and to quickly and the information that is relevant to a given question, as the amount of information increases, it tends to become a distraction and a liability, rather than an asset. This paradox drives the need for increasingly powerful and flexible data management systems. To get the most out of their large and complex datasets, users must have tools that simplify the tasks of managing the data and extracting useful information in a timely fashion. Otherwise, data can become a liability, with the cost of acquiring it and managing it far exceeding the value that is derived from it.

INTRODUCTION

A **database** is a collection of data, typically describing the activities of one or more related organizations. For example, a university database might contain information about the following:

- *Entities* such as students, faculty, courses, and classrooms.
- *Relationships* between entities, such as students' enrolment in courses, faculty teaching courses, and the use of rooms for courses.

A **database management system**, or **DBMS**, is software designed to assist in maintaining and utilizing large collections of data, and the need for such systems, as well as their use, is growing rapidly. The alternative to using a DBMS is to use ad hoc approaches that do not carry over from one application to another; for example, to store the data in files and write application-specific code to manage it.

ADVANTAGES OF A DBMS

Using a DBMS to manage data has many advantages:

Data independence:

Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.

Efficient data access:

A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

Data integrity and security:

If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, the DBMS can enforce *access controls* that govern what data is visible to different classes of users.

Data administration:

When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to

minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient.

Concurrent access and crash recovery:

A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.

Reduced application development time:

Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick development of applications. Such applications are also likely to be more robust than applications developed from scratch because many important tasks are handled by the DBMS instead of being implemented by the application.

What Does a DBMS Do?

Database management systems provide several functions in addition to simple file management:
• allow concurrency
• control security
• maintain data integrity
• provide for backup and recovery
• control redundancy
• allow data independence
• provide non-procedural query language
• perform automatic query optimization

View of Data

A database is a collection of interrelated files and a set of programs that allow users to access and modify these files. A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

1. Data Abstraction: For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database – systems users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users interactions with the system:

- Physical level: The lowest level of abstraction describes how the data are actually stored.
- Logical level: The next higher level of abstraction describes what data are stored in the database, and what data are stored in the database, and what relationship exist among those data.
- View level: The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database.

2. Instances and Schemas: Database change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called instance of the database. The overall design of the database is called the database schema. Schemas are changed infrequently, if at all.

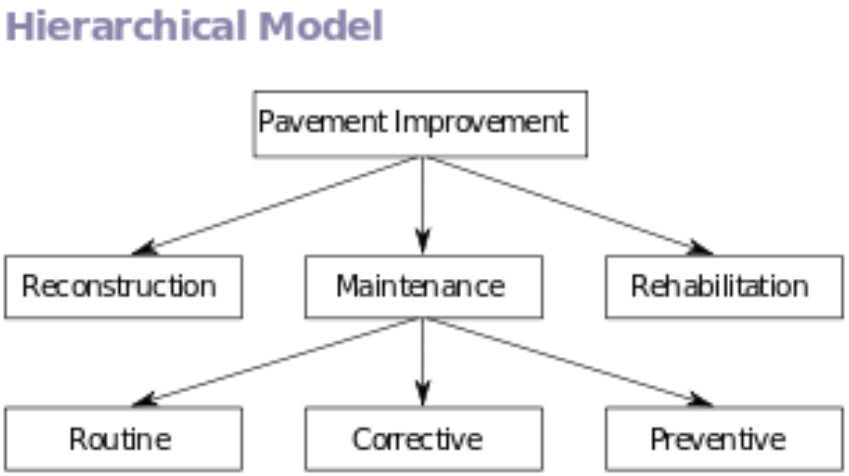
Types of Database

1) The hierarchical structure

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 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.

Hierarchical model redirects here. For the statistics usage, see hierarchical linear modelling. A hierarchical database model is a data model in which the data is organized into a tree-like structure. The structure allows representing information using parent/child

relationships: each parent can have many children, but each child has only one parent (also known as a 1-to-many relationship). All attributes of a specific record are listed under an entity type.



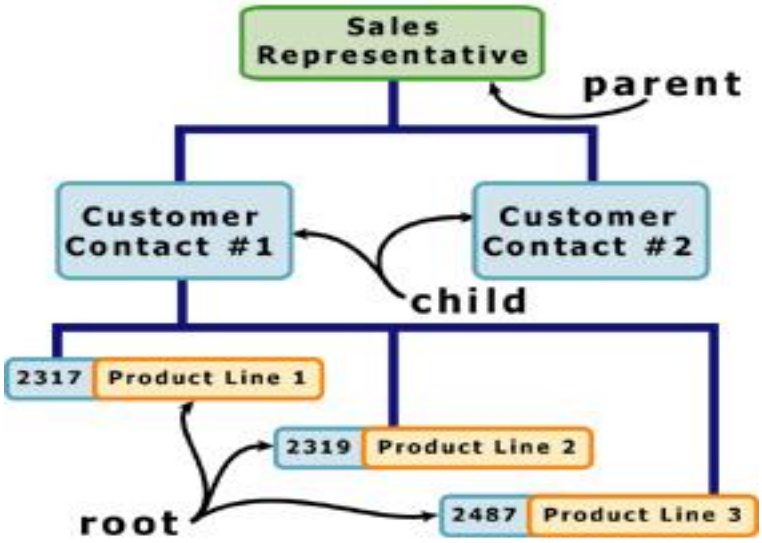
Example of a hierarchical model

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

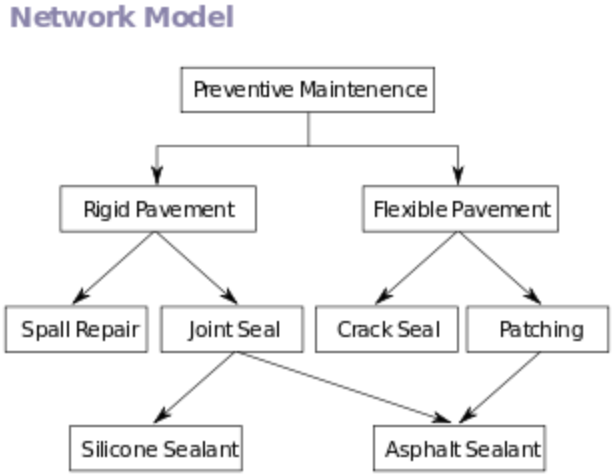
Hierarchical Databases (DBMS), commonly used on mainframe computers, have been around for a long time. It is one of the oldest methods of organizing and storing data, and it is still used by some organizations for making travel reservations. A hierarchical database is organized in pyramid fashion, like the branches of a tree extending downwards. Related fields or records are grouped together so that there are higher-level records and lower-level records, just like the parents in a family tree sit above the subordinated children.

Based on this analogy, the parent record at the top of the pyramid is called the **root record**. A child record always has only one parent record to which it is linked, just like in a normal family tree. In contrast, a parent record may have more than one child record linked to it. Hierarchical databases work by moving from the top down. A record search is conducted by starting at the top of the pyramid and working down through the tree from parent to child until the appropriate child record is found. Furthermore, each child can also be a parent with children underneath it.

The advantage of hierarchical databases is that they can be accessed and updated rapidly because the tree-like structure and the relationships between records are defined in advance. However, this feature is a two-edged sword. The disadvantage of this type of database structure is that each child in the tree may have only one parent, and relationships or linkages between children are not permitted, even if they make sense from a logical standpoint. Hierarchical databases are so rigid in their design that adding a new field or record requires that the entire database be redefined.



2) The Network Structure: 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.



Example of a Network Model.

The network model's original inventor was Charles Bachman, and it was developed into a standard specification published in 1969 by the CODASYL Consortium.

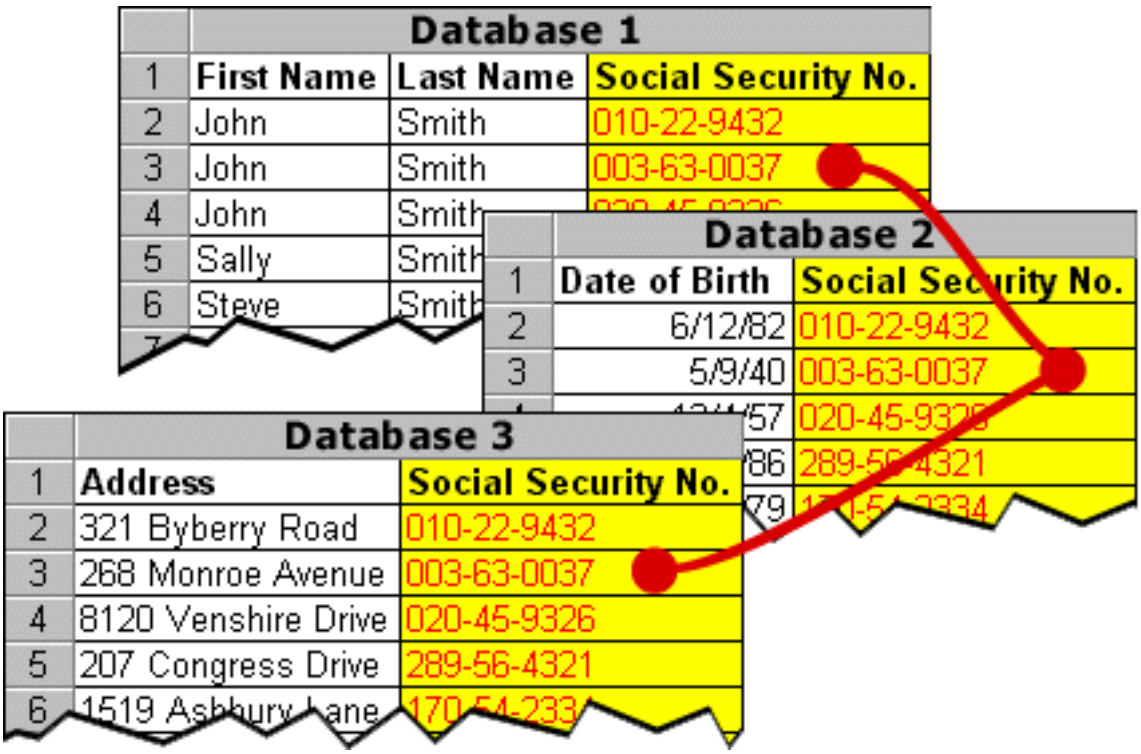
Network databases are similar to hierarchical databases by also having a hierarchical structure. There are a few key differences, however. Instead of looking like an upside-down tree, a network database looks more like a cobweb or interconnected network of records. In network databases, children are called **members** and parents are called **owners**. The most important difference is that each child or member can have more than one parent (or owner). Like hierarchical databases, network databases are principally used on mainframe computers. Since more connections can be made between different types of data, network databases are considered more flexible. However, two limitations must be considered when using this kind of database. Similar to hierarchical databases, network databases must be defined in advance. There is also a limit to the number of connections that can be made between records.

3) The relational structure: 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 1972.The model is not easy for the end user to run queries with because it may require a complex combination of many tables.

In **relational databases**, the relationship between data files is relational, not hierarchical. Hierarchical and network databases require the user to pass down through a hierarchy in order to access needed data. Relational databases connect data in different files by using common data elements or a key field. Data in relational databases is stored in different tables, each having a key field that uniquely identifies each row. Relational databases are more flexible than either the hierarchical or network database structures. In relational databases, tables or files filled with data are called **relations**, **tuples** designates a row or record, and columns are referred to as **attributes** or fields.

Relational databases work on the principle that each table has a key field that uniquely identifies each row, and that these key fields can be used to connect one table of data to another. Thus, one table might have a row consisting of a customer account number as the key field along with address and telephone number. The customer account number in this table could be linked to another table of data that also includes customer account number (a key field), but in this case, contains information about product returns, including an item number (another key field). This key field can be linked to another table that contains item numbers and other product information such as production location, color, quality control person, and other data. Therefore, using this database, customer information can be linked to specific product information.

The relational database has become quite popular for two major reasons. First, relational databases can be used with little or no training. Second, database entries can be modified without redefining the entire structure. The downside of using a relational database is that searching for data can take more time than if other methods are used.



Database 1

	First Name	Last Name	Social Security No.
2	John	Smith	010-22-9432
3	John	Smith	003-63-0037
4	John	Smith	
5	Sally	Smith	
6	Steve	Smith	

Database 2

	Date of Birth	Social Security No.
2	6/12/82	010-22-9432
3	5/9/40	003-63-0037

Database 3

	Address	Social Security No.
2	321 Byberry Road	010-22-9432
3	268 Monroe Avenue	003-63-0037
4	8120 Venshire Drive	020-45-9326
5	207 Congress Drive	289-56-4321
6	1519 Ashbury Lane	

4) The multidimensional structure: The *multidimensional structure* is similar to the relational model. The dimensions of the cube-like 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—in the same way they are 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).

5) The object-oriented structure: The *object-oriented structure* has the ability to handle graphics, pictures, voice and text, types of data, without difficultly 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.

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 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 criticized file processing because the 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. File processing was bulky, costly and inflexible 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 these reasons, the database management approach was produced.

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 object allows users to interactively interrogate the database, analyse 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 *subschema*. 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. Data structure is a logically representation of relationships between individual elements and data.

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 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.

Implementation: Database management systems

A *database management system* (DBMS) is a system that allows to build and maintain databases, as well as to utilize their data and retrieve information from it. A DBMS defines the database type that it supports, as well as its functionality and operational capabilities. A DBMS provides the internal processes for external applications built on them. The end-users of some such specific application are usually exposed only to that application and do not directly interact with the DBMS. Thus end-users enjoy the effects of the underlying DBMS, but its internals are completely invisible to end-users. 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.

A DBMS consists of software that operates databases, providing storage, access, security, backup and other facilities to meet needed requirements. DBMSs can be categorized according to the database model(s) that they support, such as relational or XML, the type(s) of computer they support, such as a server cluster or a mobile phone, the query language(s) that access the database, such as SQL or XQuery, performance trade-offs, such as maximum scale or maximum speed or others. Some DBMSs cover more than one entry in these categories, e.g., supporting multiple query languages. Database software typically support the Open Database Connectivity (ODBC) standard which allows the database to integrate (to some extent) with other databases.

The development of a mature general-purpose DBMS typically takes several years and many man-years. Developers of DBMS typically update their product to follow and take advantage of progress in computer and storage technologies. Several DBMS products have been in on-going development since the

1970s-1980s. Since DBMSs comprise a significant economical market, computer and storage vendors often take into account DBMS requirements in their own development plans.

DBMS architecture: major DBMS components

DBMS architecture specifies its components (including descriptions of their functions) and their interfaces. DBMS architecture is distinct from database architecture. The following are major DBMS components:

- DBMS external interfaces - They are the means to communicate with the DBMS (both ways, to and from the DBMS) to perform all the operations needed for the DBMS. These can be operations on a database, or operations to operate and manage the DBMS. For example:

  - Direct database operations: defining data types, assigning security levels, updating data, querying the database, etc.
  - Operations related to DBMS operation and management: backup and restore, database recovery, security monitoring, database storage allocation and database layout configuration monitoring, performance monitoring and tuning, etc.

  An external interface can be either a *user interface* (e.g., typically for a database administrator), or an *application programming interface* (API) used for communication between an application program and the DBMS.

- Database language engines (or processors) - Most operations upon databases are performed through expression in Database languages (see above). Languages exist for data definition, data manipulation and queries (e.g., SQL), as well as for specifying various aspects of security, and more. Language expressions are fed into a DBMS through proper interfaces. A language engine processes the language expressions (by a compiler or language interpreter) to extract the intended database operations from the expression in a way that they can be executed by the DBMS.
- 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.
- Database engine - Performs the received database operations on the database objects, typically at their higher-level representation.
- Storage engine - translates the operations to low-level operations on the storage bits. In some references the Storage engine is viewed as part of the Database engine.
- Transaction engine - for correctness and reliability purposes most DBMS internal operations are performed encapsulated in transactions (see below). Transactions can also be specified externally to the DBMS to encapsulate a group of operations. The transaction engine tracks all the transactions and manages their execution according to the transaction rules (e.g., proper concurrency control, and proper *commit* or *abort* for each).
- DBMS management and operation component - Comprises many components that deal with all the DBMS management and operational aspects like performance monitoring and tuning, backup and restore, recovery from failure, security management and monitoring, database storage allocation and database storage layout monitoring, etc.

DATABASE STORAGE

Database storage is the container of the physical materialization of a database. It comprises the *Internal* (physical) *level* in the database architecture. It also contains all the information needed (e.g., metadata, "data about the data", and internal data structures) to reconstruct the *Conceptual level* and *External level* from the Internal level when needed. It is not part of the DBMS but rather manipulated by the DBMS (by its Storage engine; see above) to manage the database that resides in it. 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).

In principle the database storage can be viewed as a linear address space, where every bit of data has its unique address in this address space. Practically only a very small percentage of addresses is kept as initial reference points (which also requires storage), and most of the database data is accessed by indirection using displacement calculations (distance in bits from the reference points) and data structures which define access paths (using pointers) to all needed data in effective manner, optimized for the needed data access operations.