Introduction Database System

Contents

- Database
- Database management system
- Database components
- Categories of Data Model
- Data Independence

1- Database

A ***database*** is a collection of data, typically describing the activities of one or more related organizations.

A ***database*** is a collection of interrelated data store together without harmful or unnecessary redundancy data to serve multiple application. The data is stored so as to be independent from the programs.

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' enrollment in courses, faculty-teaching courses, and the use of rooms for courses.

Database Advantages

- ✓ Reduction in data redundancy.
- ✓ The ability to operate on deferent data structure.
- ✓ Independent of data from the program.
- ✓ High speed of retrieval and fast on line.
- ✓ High degree of flexibility in handling data format.
- ✓ Minimum cost.
- ✓ Inconsistent can be avoided.
- ✓ Integrity can be maintained.
- ✓ Standard parameter can be enforced.
- ✓ Security restriction can be applied

Why high speed of retrieval data in database?

- ✓ Data base using direct access.
- ✓ using Index keys.
- ✓ Data linking together one with other.
- ✓ Redundancy is existed in some time.

2-Database management system

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 issues addressed and the techniques used span a wide spectrum, including languages, object-orientation and other programming paradigms, compilation, operating systems, concurrent programming, data structures, algorithms, theory, parallel and distributed systems, user interfaces, expert systems and artificial intelligence, statistical techniques, and dynamic programming.

A ***database management system***, or ***DBMS*** is a software package designed to store and manage database to gets:

- ✓ Data independence and efficient access.
- ✓ Reduced application development time.
- ✓ Data integrity and security.
- ✓ Uniform data administration.
- ✓ Concurrent access and recovery

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. These Users of DBMS can perform (or request the system to perform rather) a variety of operation involving such files for example Adding new files to the database

- ✓ Inserting data into existing files
- ✓ Retrieving data from existing files
- ✓ Deleting data from existing files
- ✓ Changing data in existing files
- ✓ Removing existing files from the database

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.

3- Database components:

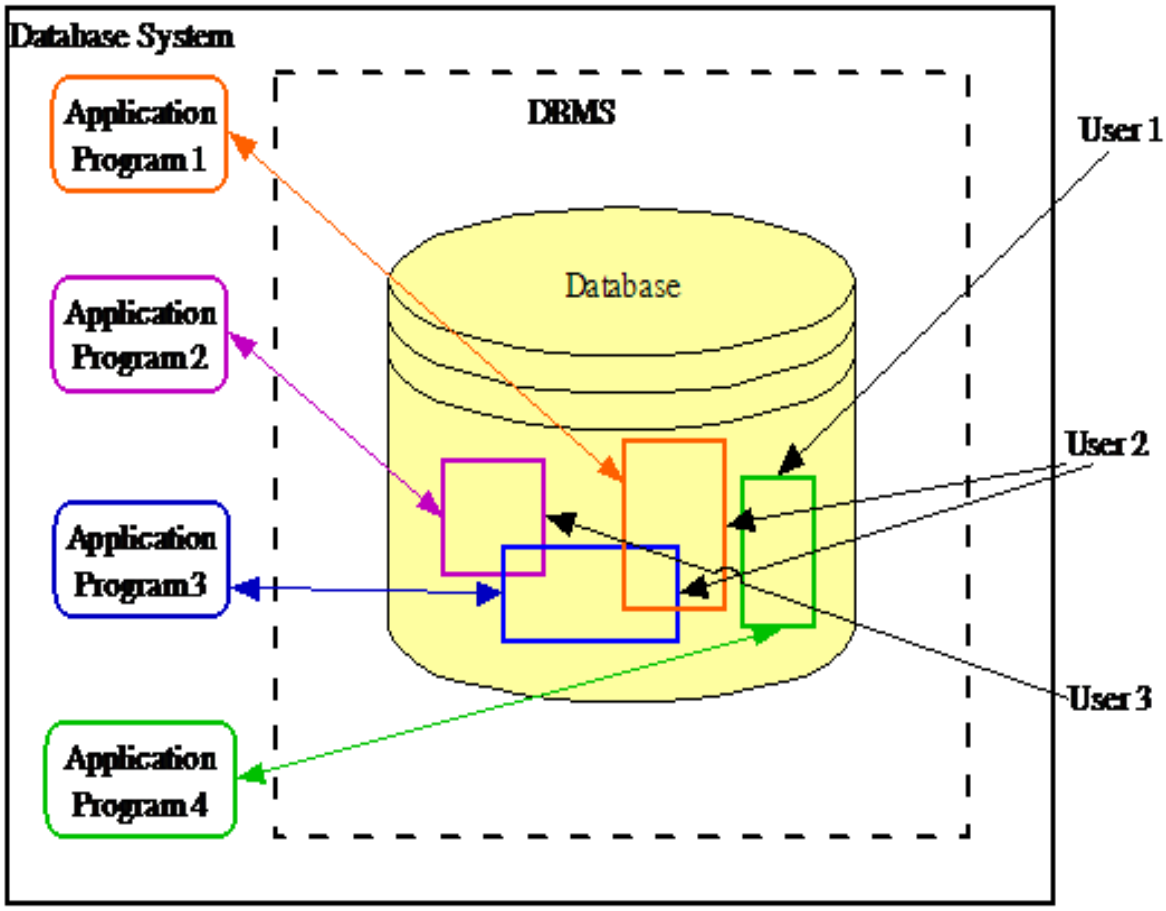


Fig 1-1 Simplified picture of a database system

The fig 1-1 is a simplified picture of database system. It is meant to show that a database system involves four components

- ☒ Data
- ☒ Hardware
- ☒ Software
- ☒ Users

☒ Data

Database system are available on machines that range all the way from the smallest personal computers to the largest mainframes. Needless to say the facilities provided by any given system are determined to some extent by the size and power of the underlying machine. In particular systems on large machines tend to be ***mulit-user***, whereas those on smaller machines tend to be ***single-user.***

☒ Hardware

The hardware comports of the system consists of :

- ✓ The secondary storage volumes mostly magnetic disks that are used to hold the stored data together with associated I/O device(disk drives ,etc.), device controllers, I/O channels, and forth.
- ✓ The hardware processors(s) and associated main memory that used to support the execution of the database system software.

☒ Software

the data as physically stored and the users of the system is a layer of software, known variously as the database manager or database server or , most commonly , the database management system(DBMS). all requests for access to the database are handled by the DBMS; the facilities sketched in section 2 for adding and removing files(tables),retrieving data from and updating data in such files or table and so forth, are all facilities provided by the DBMS.

☒ Users

We consider three broad(and somewhat overlapping)classes of users:

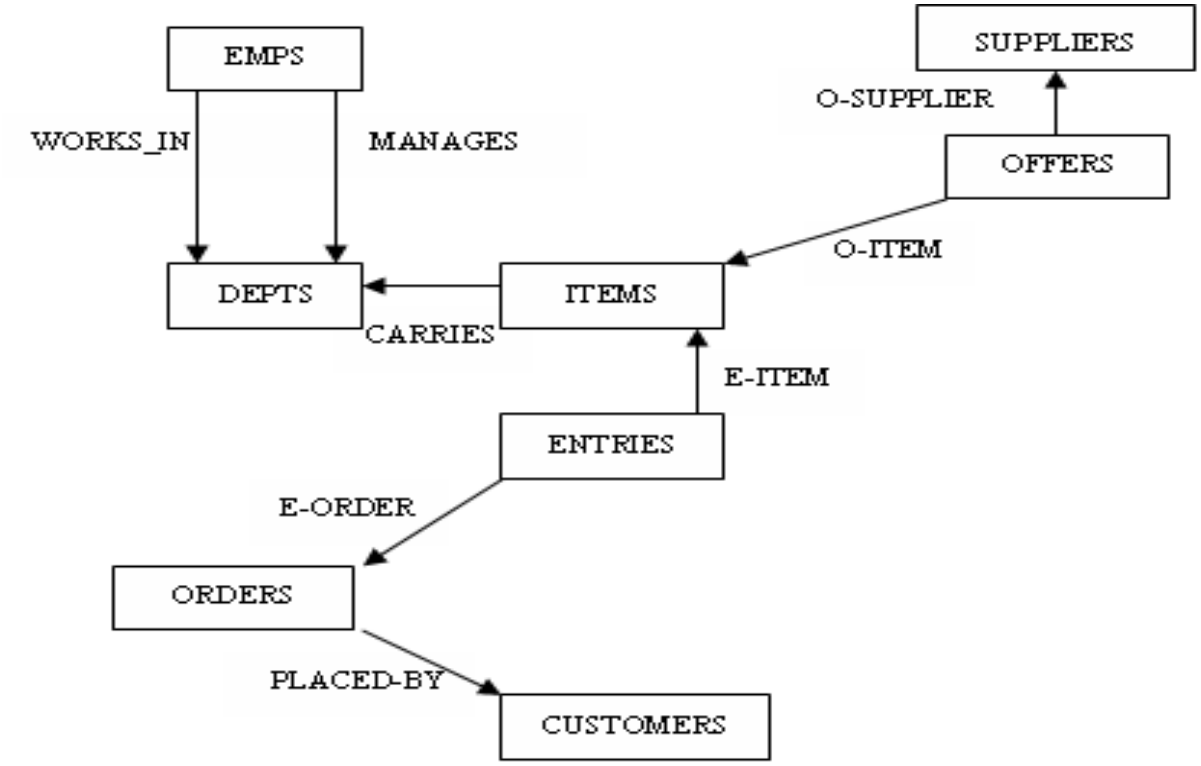
- ✓ First there are application programmers, responsible for writing database application programs in some programming language such as COBOL,PL/L,C++,JAVA, or some higher-level "fourth generation" language. Such programs access the database by issuing the appropriate request typically an SQL statement to the DBMS.
- ✓ The second class of user, then is ***end user*** who interact with the system from online workstations or terminals

4- Categories of Data Model

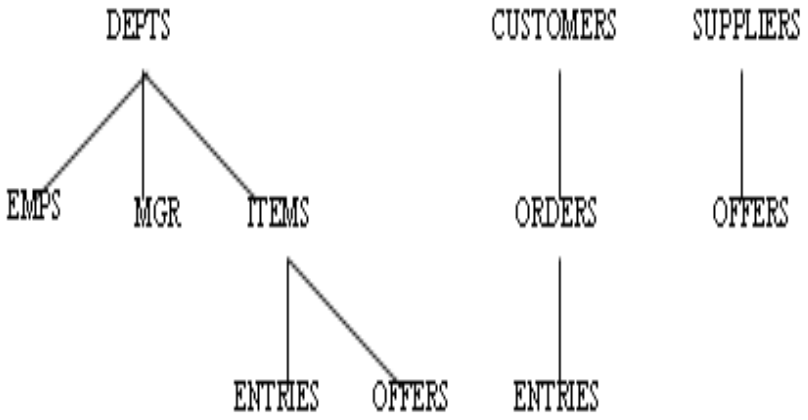
- ☒ High-level Conceptual Data models:
  Provide concepts that are close to the way people perceive data to present the data. Typical example of this type is entity – relationship model which use main concepts like entities, attributes, relationships. An entity represents real-world object such as an employee, a project. An entity has some attributes which represents properties of entity such as employee's name, address, birthdate. A relationship represents association among entities for example a works on relationships between employee and project.
- ☒ Record-based Logical Data models:
  Provide concepts that can be understood by the user but not too far from the way data is stored in the computer. Three well-known data models of this type are relational data model, network data model and hierarchical data model.
  - ✓ The Relational model represents data as relations. Here is an example of relational schema for the SUPERMARKET database

Example :- Table	
MPS (ENAME, SALARY)	MANAGERS (ENAME)
DEPTS (DNAME, DEPT#)	SUPPLIERS (SNAME, SADDR)
ITEMS (INAME, ITEM#)	ORDERS (O#, DATE)
WORKS_IN (ENAME, DNAME)	MANAGES (ENAME, DNAME)
CARRIES (INAME, DNAME)	PLACED_BY (O#, CNAME)
CUSTOMERS(CNAME,CADDR,BALANCE)	INCLUDES (O#, INAME, QUANTITY)
SUPPLIES (SNAME, INAME, PRICES)	

- ✓ The Network model represents data as record types and also represents a limited type of one to many relationship, called set type. The figure below shows a schema in network model notation



- ✓ The Hierarchical model represent data as hierarchical tree structures. Each hierarchy represents a number of related records. Here is the schema in hierarchical model notation



- ✓ Physical Data models: Provide concepts that describe how data is actually stored in the computer.

5- Data Independence

Data Independence is the ability to modify the schema in one level without affecting the schema in the higher level.

There are two levels of data independence:

- ✓ Logical data independence is the ability to make change in the conceptual schema without causing a modify in the user views or application program.
- ✓ Physical data independence is the ability to make change in the internal schema without causing a modify in the conceptual schema or application program.

Physical data independence seem to be easier to achieve since the way the data is organized in the memory affect only the performance of the system. Meanwhile, the application program depends much on the logical structure of the data that they are access.

Example

Suppose we have an application that uses the *EMPLOYEE* file of figure 1-3, and suppose it is decided, for performance reasons, that the file is to be indexed on its "employee name". In an older system, that application in question will typically be aware of the fact the index exists, and aware also of the record sequence as defined by that index, and the internal structure of the application will be built around that knowledge

Figure 1-3 The EMPLOYEE and ENROLLMENT FILES

We state by defining three terms: stored field, stored record and stored file

- ***Stored field*** is, loosely, the smallest unit of stored data. The database will contain many occurrences (or instances) of each of several types of stored field.

For example
A database containing information about different kinds of parts might include a stored field type called "part number" and then there would be one occurrence of that stored field for each kind of part(screw, hinge, lid, etc.).

- ***Stored record*** **is** collection of related stored fields .A stored record occurrence (or instance) consists of group of related stored field occurrence.

For example
A stored record occurrence in the "parts" number, part name, part color and part weight. We say that the database contains many occurrences of the "part" stored record type.

- ***A stored file*** is the collection of all currently existing occurrences of one type a stored.

Stored fields, records, and files

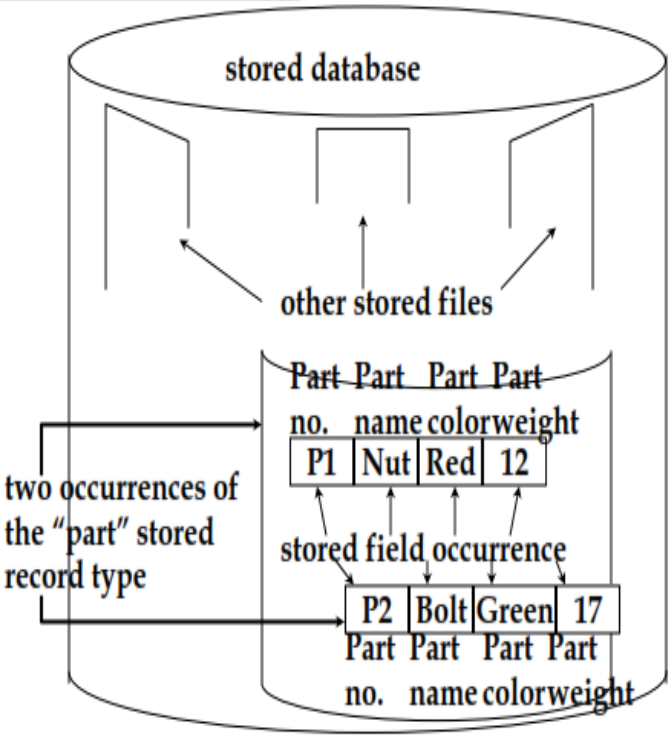


Figure 1.3

Stored fields, records and filed

We present blow a list of aspects of the stored representation that might be subject to change. You should consider in each case what the DBMS would have to do make applications immune to such change:-

- **Representation of numeric data**
- **Representation of character data**
- **Units for numeric data**
- **Data coding**
- **Data materialization**
- **Structure of stored records**
- **Structure of stored files**

- ***Representation of numeric data***

  A number field might be stored in internal arithmetic form (e.g. packed decimal) or as a character string. Either way, the DBA must choose an appropriate ***base*** (e.g. binary or decimal), ***scale*** (fixed or floating point), ***mode*** (real or complex), and ***precision*** (number of digits).

- **Representation of character data**

  A character string field might be stored using of several distinct coded character sets (e.g. ASCH, EBCDIC, and UNICODE)

- **Units for numeric data**

  The units in a number field might change from inches to centimeters, for example during a process of metrication.

- **Data coding**

  In some situations, it might be desirable to represent data in storage by coded values. For example, the "part color" field, which an application sees as a character string("Red" or "Blue" or "Green"…),might be stored as a single decimal digit, interpreted according to the coding scheme 1="Red , 2="Blue" , 3="Green" and so on .

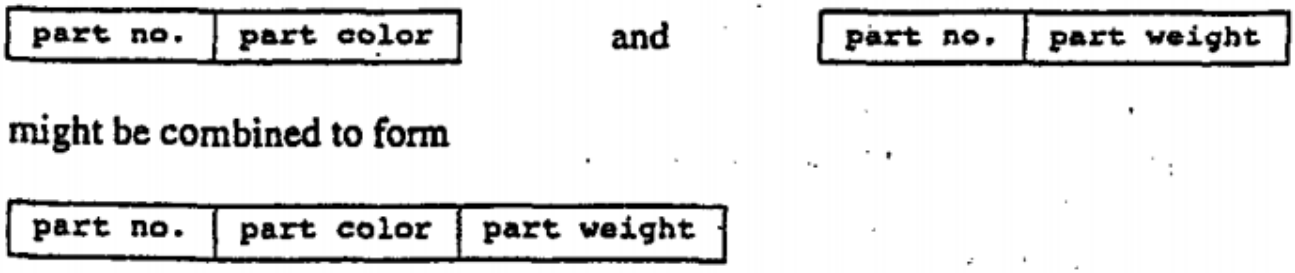
- **Data materialization**

  In practice, the ***logical field*** as seen by an application dose usually correspond to some specific stored field can be said to ***direct***. Sometimes however, a logical field will have no single stored counterpart: instead, its value will be materialized by means of some computation, perhaps on several stored field occurrences. For

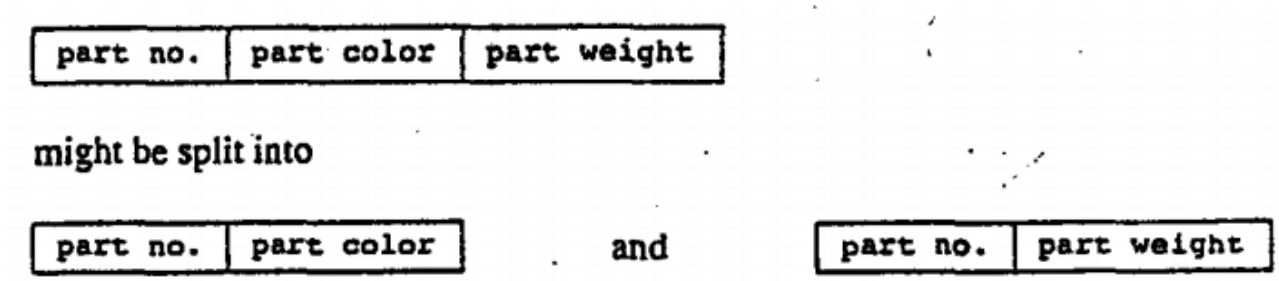
example, values of the logical field "total quantity "might be materialized by summing a number of individual stored quantities "Total quantity " here is an example of a ***virtual field***, and the materialization process is said to be ***indirect***

☒ **Structure of stored records**

Two existing stored records might be combined into one. For example, the stored record



Such a change might occur as existing applications are intergraded into the database system. Alternatively, a single stored record type might be split into two. Reversing the previous example, the stored record



☒ **Structure of stored files**

A given stored file can be physically implemented in storage in a wide variety of ways. For example, it might be entirely contained within a single storage volume