Chapter 1

Introduction and Overview

Learning Outcomes

After reading this chapter, students will be able to

- explain the concept of the database management system
- differentiate between the file system and DBMS
- explain the three-schema architecture for DBMS
- differentiate between types of database language
- describe the structure of DBMS
- explain the concept of data independence
- explain the types of DBMS users
- describe various types of database system

1.1 INTRODUCTION

A *database* is a collection of data. It contains information about one particular enterprise. Some examples of enterprises and their databases are:

- *Bank* – which stores customers' data
- *Hospital* – which stores patient data
- *University* – which stores student data

DBMS (Data Base Management System) is a collection of programs or it is software that enables the users to create and maintain a database. Also, DBMS allows the users to *insert, update* and *retrieve* the data from the database.

Some examples of DBMS are:

- MS-Access
- D-base
- FoxPro
- Oracle etc.

1.2 APPLICATIONS OF DBMS

Some of the applications areas of DBMS are listed below:

- Banking
- Airlines
- Universities
- Credit and transaction
- Tele-communication
- Sales
- Manufacturing etc.

1.3 NEED FOR DBMS

Before the DBMS came into existence, '*file processing system*' was used to handle the data of various organizations. The file system needs a set of application programs to add information to files, extract information from files and update the existing information.

There were several other problems with the file system and they are explained as shown overleaf.

- **Data Redundancy and Inconsistency**

  In a file system, same information is stored in multiple places. This duplication of data is called *data redundancy*. Because of redundancy, the file system suffers from data *inconsistency* problems during updates. Inconsistency means that the value of an attribute is different in different places.

- **Difficulty in Accessing Data**

  A conventional file processing system does not allow the needed data to be retrieved in a convenient and efficient manner. For example, consider a data file named as *saving-account* with fields named as *acc-no, name, address*, and *balance.* Application programs to access the data are also written. But, if the user wants to display only those records for which the balance is greater than Rs.10,000 and if that program is not written then, it is very difficult to access that data.

- **Data Isolation**

  Because data are scattered in various files and files may be in different formats so, it is difficult to write new application programs to retrieve the appropriate data.

- **Difficulty in Enforcing Integrity Constraints**

  The data values stored in the database must satisfy certain consistency constraints. A constraint is a restriction that we want to impose on some data values. Application programs enforce these consistency constraints by adding appropriate code to the various application programs. However, when a new constraint is to be added, it is difficult to change the programs to enforce the new constraint.

- **Atomicity Problem**

  A computer system can fail at any time. In many applications, it is needed to ensure that once a failure has occurred and has been detected, the data are stored in a consistent state that existed prior to the failure. It is very difficult to ensure this property in a conventional file processing system.

  For example, consider a program to transfer Rs. 600 from account *X* to account *Y*. If a failure occurs after removing Rs.600 from account *X* but before adding Rs.600 to account *Y* then, it is very difficult to maintain the atomicity property.

- **Difficulty in Concurrency Control**

  In the case of the file processing system, data is not centralized. Sometimes, two or more users want to access the database at the same time but it is very difficult to build the concurrency control feature at the application programs level.

- **Security Problems**

  Since, application programs are added to the system in an *ad-hoc* manner and information does not have a centralized access path thus, it is difficult to enforce security constraints.

1.4 ADVANTAGES OF DBMS

DBMS was developed to overcome the limitation of a file processing system.

Following are some of the advantages of DBMS:

- **Controlling Redundancy**

  In file processing systems, every user group maintains its own file for handling data. For example, in a *University*, two groups of users might be *course registration* and the *accounting office*. The accounting office keeps data on registration and related billing information whereas the registration office keeps data on student courses and grades. Most of the data is stored twice, once in each user file. This results in *redundancy* of data which leads to several problems. Firstly, there is a wastage of storage space as data is duplicated. Another problem is *inconsistency*. This may happen because an update is applied to some of the files but not to others. For example, if the address of a student is changed and the student informs this change to the course-registration user but does not inform the accounting office. The changed address is updated by the course registration user but it is not updated by the accounting office. Thus, the value of the address is different in two different places. This results in data inconsistency. For consistency, we should have a database design that stores each logical data item such as *name* or *birth date* in only one place in the database. This design does not permit inconsistency and also saves storage space.

- **Restricting Unauthorized Access**

  When multiple users share a database, some users will not be authorized to access all the information in the database. For example, *financial* data is often confidential and hence, only authorized persons are allowed to access such data. Also, some users may be permitted only to retrieve data, others are allowed both to retrieve and update the data. For this purpose, a *password* is given to all the users by DBA (Database administrator). Thus, only the authorized person can perform a particular operation on the database for which the authority is given to him.

- **Centralized Control**

  DBA (Database administrator) is the person who has centralized control over the database. Several of the drawbacks of the file processing system are eliminated in DBMS because of centralized control.

- **Backup and Recovery**

  Hardware and software can fail at any time. So, there should be some mechanism for recovering from such failures. The backup and recovery subsystem of DBMS is responsible for the recovery of such failures. For example, if a computer's system fails in the middle of a complex update program then, the recovery subsystem ensures that the database is restored to the state it was in before the program started executing.

- **Enforcing Integrity Constraints**

  For some applications, integrity constraints must hold for the data values. For example, data values for a column of numeric type must be an integer between 1 and 7, the value of the name must be a string no more than 20 characters, etc. Another constraint is that a data value should not be null. DBMS provides all these constraints.

- **Providing Multiple User Interface**

  Many users of different levels use the database. DBMS provides a variety of interfaces for these users. These include query languages for casual users, programming language interfaces for application programmers, etc.

- **Shared Data**

  DBMS allows the multiple users to share the database. It means that more than one user can use the same database.

- **Representing Complex Relationships among Data**

  DBMS provides not only the simple relationships among the data but, also provides the complex relationships.

1.5 DISADVANTAGES OF DBMS

Along with several advantages, DBMS also has some disadvantages. But, these disadvantages are negligible as compared to the advantages offered by DBMS.

Some of the disadvantages of DBMS are as follows:

- Numbers of problems are associating with centralized data.
- Cost of hardware and software.
- Complexity of backup and recovery mechanism.

1.6 DBMS ARCHITECTURE

DBMS architecture is also called *three-level* architecture. The goal of three-level architecture is to separate the user applications and the physical database. Three-level architecture is shown in figure 1.1.

A major purpose of DBMS is to provide users with an abstract view of data. Many database systems users are not computer-trained hence the complexity is hidden from them through several levels of abstraction.

Abstraction means hiding certain details of how the data is stored and maintained. These levels of abstraction can be explained with the help of three-level architecture.

The following are the different levels of abstraction:

- **Physical Level**

  It is the lowest level of abstraction. It describes how data is actually stored and also described the data structures and access methods to be used by the database. The physical level has an internal schema that describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

- **Conceptual Level**

  It is the next higher level of the abstraction. The conceptual level has a conceptual schema that describes the structure of the whole database for a community of users. This schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, and constraints.

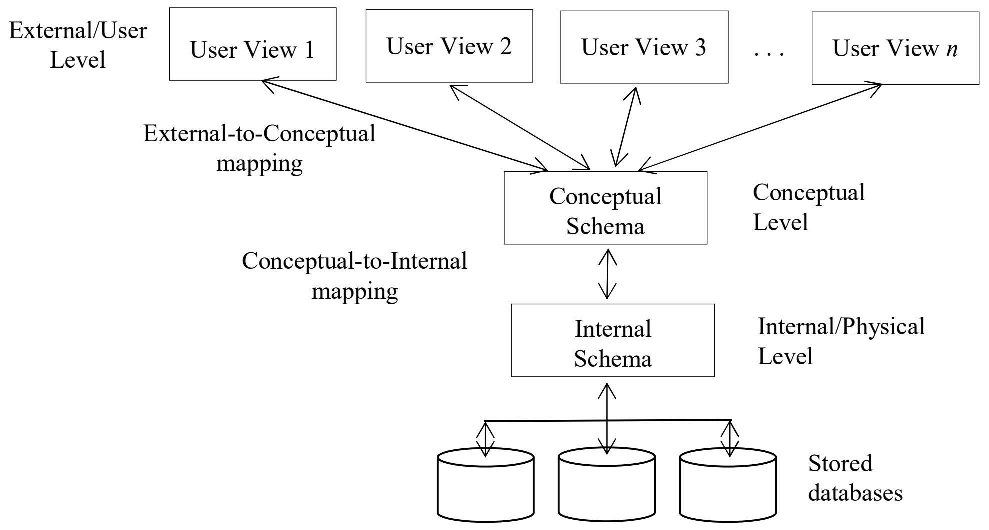


Figure 1.1: Three-Level Architecture

- **View Level**

  It is the highest level of abstraction. The view level includes a number of user views. It describes only a part of the entire database. Many users will not be concerned with all of the information. Different users may need only a part of the entire database. To simplify their interaction with the database, view level is defined. A high-level data model can be used at this level.

Between the two levels, there is *mapping* as shown in figure 1.1. The process of transforming the requests and results between the different levels is called *mapping*.

Example 1.1

To differentiate between different levels, consider a record defined using structure in *C* language as follows:

```
struct client
{
    char client-name [20];
    char client-street [20];
    char client-city [20];
} clnt;
```

In this example, *client* is a structure name having three fields. Each of the field has a name and its data type.

Now, let us compare three levels with the above example.

- **Physical Level**

  In the case of programming language, at physical level, the client structure can be described as a block of consecutive storage locations (say 250 bytes). The compiler hides this level of detail from programmers. Similarly, the database system hides many of the lowest level storage details from database programmers. Database administrators may be aware of certain details of the physical organization of data.

- **Conceptual Level**

  At the conceptual level, programmers of programming languages work at this level of abstraction and define structure by a type definition, and also the interrelationship among these structures is defined. Similarly, database administrators usually work at this level of abstraction.

- **View Level**

  At the view level, users can see the final results of their programs by giving different inputs. Similarly, at the view level, several views of the database are defined and database users can get the required output.

1.7 INSTANCES AND SCHEMAS

A database is a collection of related data. Database changes over time as the information is inserted and deleted from the database. The collection of the information stored in the database at a particular moment of time is called an *instance* of the database.

The overall design of the database is called as a database *schema*.

Consider the following C program statements:

```
int a, b, c;      // variable declaration
c = 5;            // assigning 5 to variable c
```

A database schema corresponds to the variable declaration in any programming language (i.e., int a, b, c;). Each variable has a particular value at a given instant. Thus, the value of the variables in a program at a given moment corresponds to an instance of database schema (i.e., c = 5).

According to the levels of abstraction, the database system has several schemas and they are as follows:

- **Physical Schema:** The physical schema describes the database design at the physical level.
- **Logical Schema:** The logical schema describes the database design at logical level.
- **Subschema:** A database may have several schemas at the view level and these schemas are called as sub-schemas that describe different views of the database.

Once defined, the database *schema* rarely changes whereas *instance* may change as a result of insertion, deletion and updation commands.

1.8 DATABASE LANGUAGES

A *database* is a collection of inter-related data. To perform various operations on schema and the data, we need some kind of languages using which commands can be issued. Database languages act as an interface between the database and the user. For example, *SQL* (Structured query language) is a database language. SQL has a set of commands. Some of these commands are used to specify the database schema whereas others are used to express database queries and updates. SQL is classified into following three parts:

- Date definition language (DDL)
- Data manipulation language (DML)
- Data control language (DCL)

1.8.1 Data Definition Language

A database schema is specified by a set of definitions expressed by a special language called *data definition language*. DDL includes the commands that are used to *create, alter* and *drop* the structure of the tables. In simple words, we can say that DDL deals with the structure of the database.

1.8.2 Data Manipulation Language

DML is a language that enables the user to access or manipulate the data. Data manipulation language includes the commands:

- To retrieve the information from the database.
- To insert new information into the database.
- To delete information from the database.
- To modify the information stored in the database.

DML is classified into following types:

- **Procedural DML:** It requires the user to specify what data are required and also how to get those data.
- **Non-procedural DML:** It requires the user to specify only what data are required without specifying how to get those data.

The DML component of the SQL is *non-procedural*.

1.8.3 Data Control Language

DCL includes the statements which control access to data and database. A privilege can be granted to a user by using *GRANT* statement. The privileges assigned can be *select, alter, delete, execute, insert* and *index* etc. Privileges are assigned to the users because we do not want every user to perform all the operations on the database. In addition, we can also revoke (take back) the privileges by using *REVOKE* command.

1.9 OVERALL STRUCTURE OF DBMS

A database system is divided into modules and each module is assigned a responsibility of the system.

The main components of a database system are:

- Storage manager
- Query processor

A database can store hundreds of gigabytes. Because, the main memory of computation is small, so it can not store such bulky data. Thus, we use the disks to store this data. Data are moved between disk and main memory as needed. The movement of data to and from the disk is slow. Thus, a lot of time is wasted in moving the data between the main memory and the disk. So, there is a need to minimize the movement of data between the disk and main memory. The *storage-manager* component of the DBMS helps to minimize the data movement between the disk and main memory.

Another component of the database system is the *query-processor*. The query processor transforms the user queries into a series of low-level instructions. It is used to interpret online user queries and convert them into an efficient series of operations in a form capable of being sent to the run-time data manager for execution.

The structure of DBMS is shown in Figure 1.2.

1.9.1 Storage Manager

The storage manager is responsible for storing, retrieving, and updating the data in the database. A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and the queries submitted to the system.

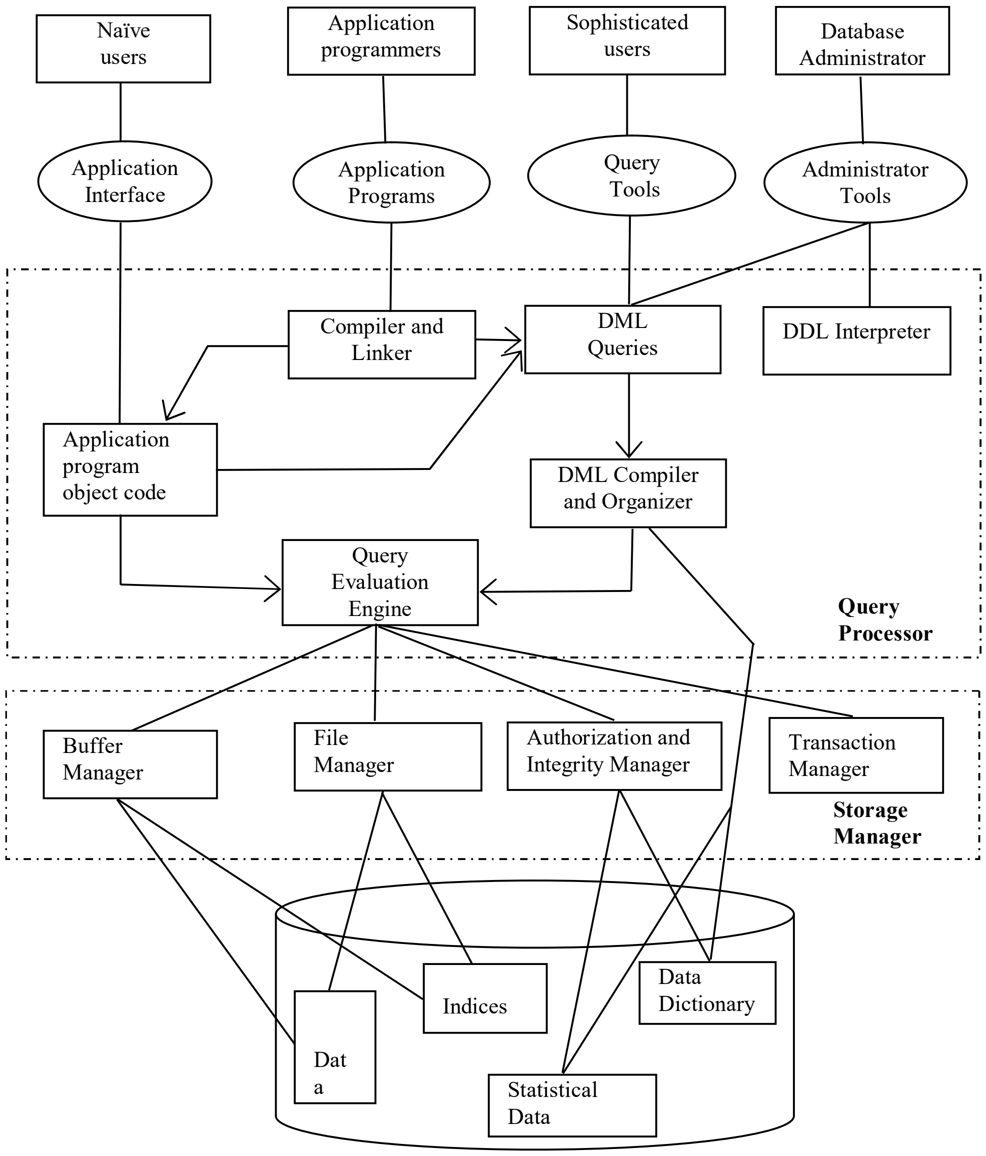


Figure 1.2: Comprehensive Structure of DBMS

The storage manager includes four components and they are explained as follows:

- **Authorization and integrity manager:** It tests for the satisfaction of integrity constraints and also checks the authority of the users to access data.
- **Transaction manager:** It ensures that the database remains in a consistent state despite of system failures.
- **File manager:** It manages the allocation of space on disk storage and the data structures used to represent the information stored on disk.

Buffer manager: It is responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory.

The storage manager implements several data structures. These data structures are explained as follows:

- **Data files:** It stores the database.
- **Data dictionary:** It stores metadata (data about data) about the structure of the database.
- **Indices:** It provides the fast access to data items that hold particular values.

1.9.2 Query Processor

The query processor is responsible for the execution of the query. The components of the query processor are:

- **DDL Interpreter:** It interprets the DDL statements and records them in a set of tables containing metadata or a data dictionary.
- **DML Compiler:** It converts DML statements (in a query language) into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
- **Query Evaluation Engine:** It executes low-level instructions generated by DML compiler.

1.10 DATA INDEPENDENCE

Three-level architecture can be used to explain the concept of data independence. It is defined as the capability to change the schema at one level of the database system without having to change the schema at the next higher level.

The types of data independence are:

- o **Physical data independence:** It is the capability to change the internal schema without having to change the conceptual schema. Change to the internal schema is needed because some physical files have to be reorganized.
- o **Logical data independence:** It is the capability to change the conceptual schema without having to change the external schema. The conceptual schema is changed to expand the database or to reduce the database.

The data independence is accomplished when the schema is changed at some level but, the schema at the next higher level remains unchanged. But, the mapping between the two levels is changed.

1.11 USERS OF DBMS

There are different types of the users who use the database in different ways and they are explained as follows:

- o **Database Administrator (DBA)**

In any organization in which many persons use the same resources, there is the need for chief administrator to oversee and manage those resources. DBA is a person or group of persons who manage these resources. The DBA is responsible for authorizing access to the database. In large organizations, the DBA is assisted by a staff that helps the DBA for carrying out its functions. Also, the centralized control of the database is exerted by the DBA.

Role of DBA: The main role of DBA is explained as follows:

- **Schema definition:** DBA creates the original database schema by writing a set of definitions that is translated by the DDL complier to a set of tables that is stored permanently in the data dictionary.
- **Storage structure and access method definition:** DBA creates the appropriate storage structure and access methods by writing a set of definitions which is translated by the data storage and DDL compiler.
- **Schema and physical organization modification:** It is also the responsibility of the DBA to modify the schema and physical organization of the database by writing a set of definitions that are used either by DDL compiler or data storage and DDL compiler which generates the modifications to the database.
- **Granting authorization for data access:** DBMS allows multiple users to share the data. Not all users are allowed to access all the data because some data is confidential. So, it is the responsibility of DBA to grant authorization to different

users to access the data. A user can access a part of a database if and only if, he is granted authority by the DBA. The authorization information is kept in a special system structure.

- **Integrity constraint specification:** Data values stored in the database must satisfy certain constraints. Such constraints must be specified explicitly by the DBA. The integrity constraints are kept in a special system structure.

o **Database Designer**

Database designers are responsible for identifying the data to be stored in the database and for choosing the appropriate structure to represent and store that data. The database designers communicate with all the database users in order to understand their requirements and to come up with a design that meets these requirements. In many cases, the designers are on the staff of DBA.

o **End Users**

End users are the people whose job requires access to the database for querying, updating, and generating reports.

The types of end users are:

- **Casual end users:** These are the users who occasionally access the database but, may need different information each time. They use query language to specify their requests.
- **Naive or parametric end users:** Their main job function revolves around constantly querying and updating the database using standard types of queries and updates called *canned* transactions that have been carefully programmed and tested. For example, *bank-teller* who checks account balance, withdraws and deposits, reservation, *clerks* for airlines, hostels, and railways who make the reservation and canalization.
- **Sophisticated end users:** These include *Engineers, Scientists* and *Analysts* who use database to implement their application to meet their complex requirements.
- **Stand-alone users:** These users maintain personal database by using ready-made program packages. For example, users of *tax* package.

o **System Analyst and Application Programmers**

System analysts determine the requirements of end users, especially *naive* users and develop specifications for *canned* transactions that meet these requirements. *Application programmers* implement these specifications as programs and they test, debug, document, and maintain these *canned* transactions.

1.12 STORAGE MANAGER RESPONSIBILITIES

Storage Manager is also called *Database Manager.* We have seen that the storage manager is an important component of the DBMS structure. It is a program module that provides the interface between low-level data stored in the database and the application programs and queries submitted to the system.

The followings are the responsibilities of the storage manager:

- **Interaction with the file manager:** Actual data is stored in the file system. The database manager is responsible for actual storing, retrieving and updating the data in the database.
- **Integrity constraints enforcement:** Consistency constraints are specified by DBA. But, the responsibility of the database manager is to enforce, implement or check those constraints.
- **Security enforcement:** It is the responsibility of the database manager to enforce the security requirements.
- **Backup and recovery:** It is responsibility of database manager to detect system failures and restore the database to a consistent state.
- **Concurrency control:** Interaction among the concurrent users is controlled by database manager.

1.13 TYPES OF DATABASE SYSTEMS

DBMS can be classified on the basis of number of users and the database site locations. This categorization is shown as in the following figure.

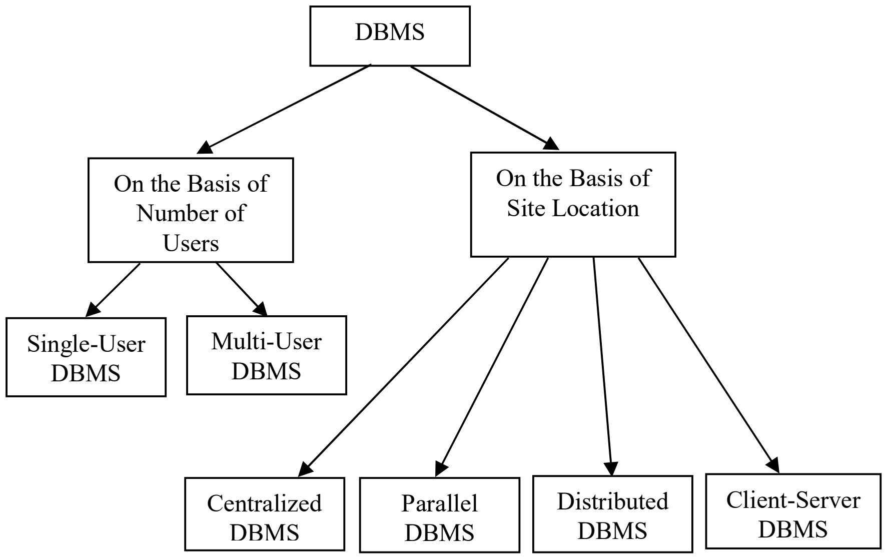


Figure 1.3: Classification of DBMS

1.13.1 On The Basis of Number of Users

Followings are the types of DBMS on the basis of number of users:

- **Single-user DBMS:** In the single-user DBMS, database resides on one computer and it is only accessed by one user at a time.
- **Multi-user DBMS:** In multi-user system, the multiple users access the data from one central storage area so that the database remains in the integrated form. A database is integrated when the same information is not stored in two places. Most of the DBMS are multi-user.

1.13.2 On The Basis of Site Location

Followings are the types of DBMS on the basis of site location:

- **Centralized System**

  In a *centralized* system, the database resides on some single central location. A number of processors can access this central database. These processors are connected to the central database via some computer network. Figure1.4 shows the centralized database arrangement.

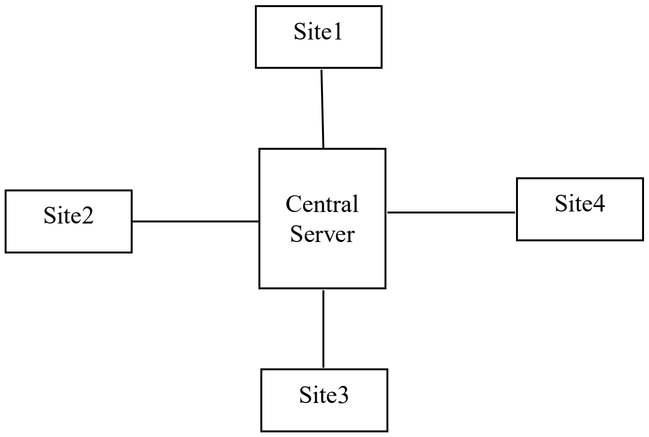


Figure 1.4: Centralized System

The *railway reservation system* is an example of centralized system, where the database is centrally located at New Delhi. A number of reservation counters from all over the country access this database making the reservations and canalizations. The main drawback of the centralized system is that, when the central site computer goes down then, the users are blocked from using system until the system comes back.

- **Parallel System**

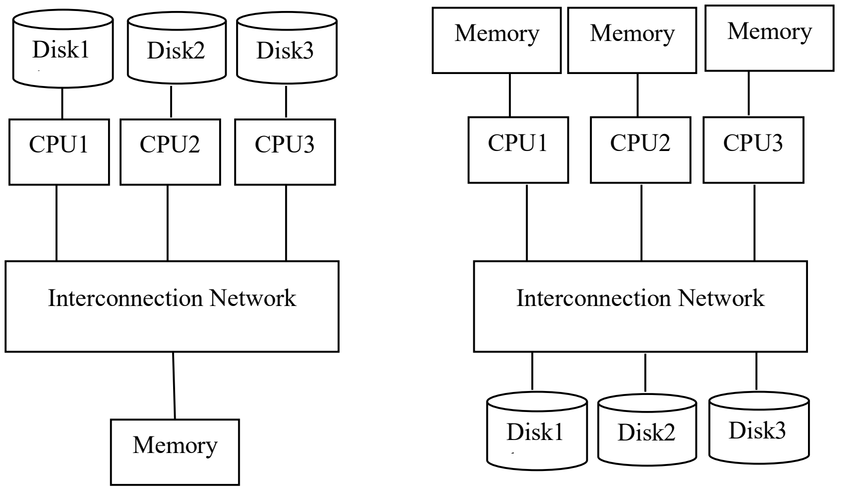
Parallel systems improve the processing and *I/O* speed by using multiple *CPU's* and disks in parallel. In a parallel system, many operations are executed simultaneously.
There are two main measures of performance of the database system:

- **Throughput:** It is the number of tasks that can be completed in a given time interval.
- **Response time:** It is the amount of time the system takes to complete a single task from the time it is submitted to the system.

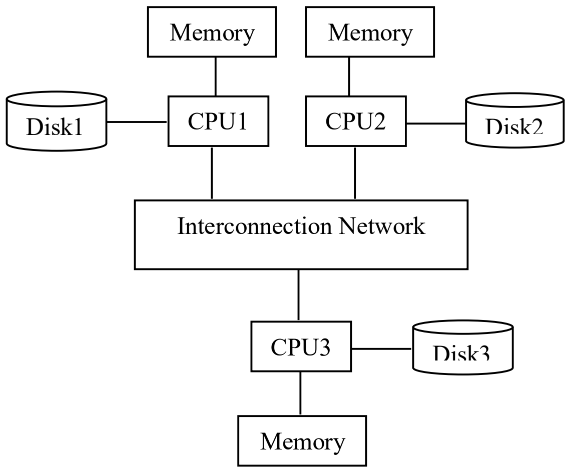
A system that processes a large number of small transactions can improve *throughput* by processing many transactions in parallel.

A system that processes large transactions can improve *response time* as well as *throughput* by performing subtasks of each transaction in parallel. There are three architectures for parallel DBMS as illustrated in Figure 1.5.

- **Shared memory:** In this model, all the processors share a common memory. The communication between the components is through interconnection network. A processor can send massages to other processor using memory writes. This architecture provides high-speed data access for a limited number of processors but, it is not scalable beyond 64 processors since the interconnection network becomes bottleneck. Then, the processors have to spend most of their time waiting for their turn to access the memory.



(*a*) Shared Memory (*b*) Shared Disk



(*c*) Shared Nothing

Figure 1.5: Parallel Systems

- **Shared disk:** In this model, all the processors share a common disk. Shared disk models are sometimes called as *clusters*. Here, all the processors can access all the disks directly via an interconnection network. All the processors have their private memories.
- **Shared nothing:** In this model, processors share neither a common memory nor a common disk. Each node of the machine consists of a processor, memory, and one or more disks. The communication between the processors is through the interconnection network. Shared-nothing architecture is more scalable than shared memory and can easily support a large number of processors.

o **Distributed System**

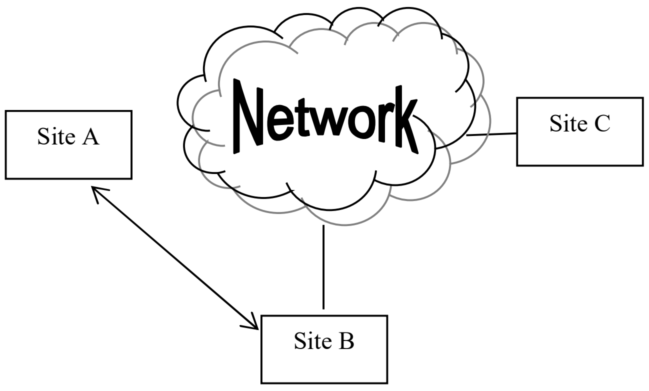


Figure 1.6: Distributed System

In distributed database management system, the database is split into a number of fragments. Each fragment is stored on one or more computers. These computers are connected by a communication network.

The computers in a distributed system may vary in size and function. The computers in a distributed system are referred to by a number of different names such as *sites* or *nodes*. The structure of a distributed system is shown in Figure 1.6.

In a distributed system, a database is geographically separated and is administrated separately, and has slower interconnection. Users access the distributed database via applications. Applications can be *local* or *global*. The local applications do not require data from other sites whereas the global applications require data from other sites.

The *Banking* system is an example of a distributed system, where the database system is implemented on a number of computer systems rather than on a centralized computer. A network connecting the computers will enable the different branches to communicate with each other. Thus, a customer living in one city can also check his account during his stay in another city.

The main advantage of a distributed system is that, if one site fails, the remaining sites may be able to continue operating. The failure of a site does not necessarily imply the shutdown of the system. The drawbacks of this system are software development costs and increased processing overhead.

- **Client-Server System**

The *Client-server* system has two main components namely *client* and *server*. A client is a computer which requests for the service from the server. A server is a computer which provides the service to the client. All data resides at the server site. All the applications execute at the client site. The client and server computers are connected through a network.

A general structure of client-server system is shown in Figure 1.7.

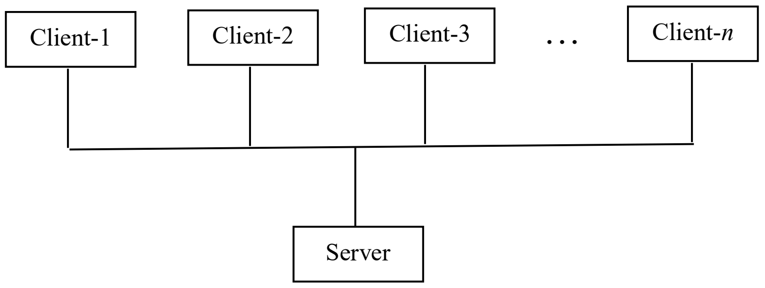


Figure 1.7: Client-Server System.

A client requests a service from the server and the server returns the result to its client. *Internet* is an example of client-server architecture.

EXERCISE . . .

1. What is a file processing system? Explain its disadvantages.
2. What is DBMS? What are its advantages over traditional file systems?
3. Draw and explain the 3-schema architecture of DBMS.
4. What is a data model? Explain its types.
5. Draw and explain the overall structure of DBMS.
6. What is data independence? What are its types?
7. Explain various users of DBMS.
8. What is a DBA? Explain the role of DBA.
9. What are the main responsibilities of storage manager? Explain.
10. Explain various types of database systems.

❑ ❑ ❑