UNIT -1

UNIT -1
The Worlds of Database Systems

INTRODUCTION TO BASIC CONCEPTS OF DATABASE SYSTEMS:

What is Data?

The raw facts are called as data. The word “raw” indicates that they have not been processed.

Ex: For example 89 is the data.

What is information?

The processed data is known as information.

Ex: Marks: 89; then it becomes information.

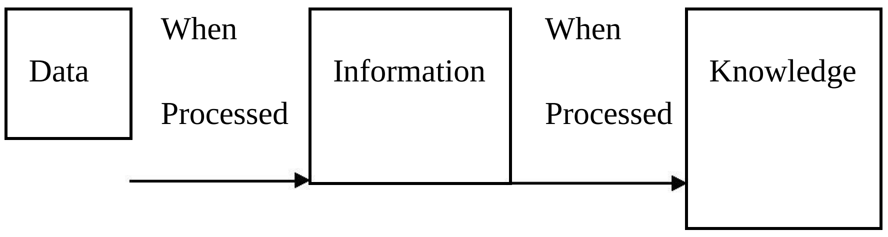
What is Knowledge?

1. Knowledge refers to the practical use of information.

2. Knowledge necessarily involves a personal experience.

DATA/INFORMATION PROCESSING:

The process of converting the data (raw facts) into meaningful information is called as data/information processing.



Note: In business processing knowledge is more useful to make decisions for any organization.

DIFFERENCE BETWEEN DATA AND INFORMATION:

DATA	INFORMATION
1.Raw facts	1.Processed data
2. It is in unorganized form	2. It is in organized form
3. Data doesn't help in decision making process	3. Information helps in decision making process

FILE ORIENTED APPROACH:

The earliest business computer systems were used to process business records and produce information. They were generally faster and more accurate than equivalent manual systems. These systems stored groups of records in separate files, and so they were called **file processing systems.**

1. File system is a collection of data. Any management with the file system, user has to write the procedures

2. File system gives the details of the data representation and Storage of data.

3. In File system storing and retrieving of data cannot be done efficiently.

4. Concurrent access to the data in the file system has many problems like a Reading the file while other deleting some information, updating some information

5. File system doesn't provide crash recovery mechanism.
Eg. While we are entering some data into the file if System crashes then content of the file is lost.

6. Protecting a file under file system is very difficult.

The typical file-oriented system is supported by a conventional operating system. Permanent records are stored in various files and a number of different application programs are written to extract records from and add records to the appropriate files.

DISADVANTAGES OF FILE-ORIENTED SYSTEM:

The following are the disadvantages of File-Oriented System:

Data Redundancy and Inconsistency:

Since files and application programs are created by different programmers over a long period of time, the files are likely to be having different formats and the programs may be written in several programming languages. Moreover, the same piece of information may be duplicated in several places. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency.

Difficulty in Accessing Data:

The conventional file processing environments do not allow needed data to be retrieved in a convenient and efficient manner. Better data retrieval system must be developed for general use.

Data Isolation:

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

Concurrent Access Anomalies:

In order to improve the overall performance of the system and obtain a faster response time, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data.

Security Problems:

Not every user of the database system should be able to access all the data. For example, in banking system, payroll personnel need only that part of the database that has information about various bank employees. They do not need access to information about customer accounts. It is difficult to enforce such security constraints.

Integrity Problems:

The data values stored in the database must satisfy certain types of consistency constraints. For example, the balance of a bank account may never fall below a prescribed amount. These constraints are enforced in the system by adding appropriate code in the various application programs. When new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items for different files.

Atomicity Problem:

A computer system like any other mechanical or electrical device is subject to failure. In many applications, it is crucial to ensure that once a failure has occurred and has been detected, the data are restored to the consistent state existed prior to the failure

Example:

Consider part of a savings-bank enterprise that keeps information about all customers and savings accounts. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including:

- A program to debit or credit an account
- A program to add a new account
- A program to find the balance of an account
- A program to generate monthly statements

Programmers wrote these application programs to meet the needs of the bank. New application programs are added to the system as the need arises. For example, suppose that the savings bank decides to offer checking accounts.

As a result, the bank creates new permanent files that contain information about all the checking accounts maintained in the bank, and it may have to write new application programs to deal with situations that do not arise in savings accounts, such as overdrafts. Thus, as time goes by, the system acquires more files and more application programs. The system stores permanent records in various files, and it needs different

Application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMS) came along, organizations usually stored information in such systems. Organizational information in a file-processing system has a number of major disadvantages:

1. Data Redundancy and Inconsistency:

The address and telephone number of a particular customer may appear in a file that consists of savings-account records and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost. In, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed customer address may be reflected in savings-account records but not elsewhere in the system.

2. **Difficulty in Accessing Data:**

Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer asks the data-processing department to generate such a list. Because there is no application program to generate that. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory.

3. **Data Isolation:**

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

4. **Integrity Problems:**

The balance of a bank account may never fall below a prescribed amount (say, $25). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

5. **Atomicity Problems:**

A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a program to transfer $50 from account *A* to account *B*. If a system failure occurs during the execution of the program, it is possible that the $50 was removed from account *A* but was not credited to account *B*, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be *atomic*—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

6. **Concurrent-Access Anomalies:**

For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data. Consider bank account *A*, containing $500. If two customers withdraw funds (say $50 and $100 respectively) from account *A* at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $500, and write back $450 and $400, respectively. Depending on which one writes the value last, the account may contain $450 or $400, rather than the correct value of $350. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

7. **Security Problems:**

Not every user of the database system should be able to access all the data. For example, in a banking system, payroll personnel need to see only that part of the database that has information about the various bank employees. They do not need access to information about customer accounts. But, since application programs are added to the system in an ad hoc manner, enforcing such security constraints is difficult. These difficulties, among others, prompted the development of database systems.

INTRODUCTION TO DATABASES:

History of Database Systems:

1950s and early 1960s:

- Magnetic tapes were developed for data storage
- Data processing tasks such as payroll were automated, with data stored on tapes.
- Data could also be input from punched card decks, and output to printers.
- Late 1960s and 1970s: The use of hard disks in the late 1960s changed the scenario for data processing greatly, since hard disks allowed direct access to data.
- With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.
- With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.
- In the 1970's the EF CODD defined the **Relational Model.**

In the 1980's:

- Initial commercial relational database systems, such as IBM DB2, Oracle, Ingress, and DEC Rdb, played a major role in advancing techniques for efficient processing of declarative queries.
- In the early 1980s, relational databases had become competitive with network and hierarchical database systems even in the area of performance.
- The 1980s also saw much research on parallel and distributed databases, as well as initial work on object-oriented databases.

Early 1990s:

- The SQL language was designed primarily in the 1990's.
- And this is used for the transaction processing applications.
- Decision support and querying re-emerged as a major application area for databases.

- Database vendors also began to add object-relational support to their databases.

Late 1990s:

- The major event was the explosive growth of the World Wide Web.
- Databases were deployed much more extensively than ever before. Database systems now had to support very high transaction processing rates, as well as very high reliability and 24 * 7 availability (availability 24 hours a day, 7 days a week, meaning no downtime for scheduled maintenance activities).
- Database systems also had to support Web interfaces to data.

The Evolution of Database systems:

The Evolution of Database systems are as follows:

1. File Management System
2. Hierarchical database System
3. Network Database System
4. Relational Database System

File Management System:

The file management system also called as FMS in short is one in which all data is stored on a single large file. The main disadvantage in this system is searching a record or data takes a long time. This lead to the introduction of the concept, of indexing in this system. Then also the FMS system had lot of drawbacks to name a few like updating or modifications to the data cannot be handled easily, sorting the records took long time and so on. All these drawbacks led to the introduction of the Hierarchical Database System.

Hierarchical Database System:

The previous system FMS drawback of accessing records and sorting records which took a long time was removed in this by the introduction of parent-child relationship between records in database. The origin of the data is called the root from which several branches have data at different levels and the last level is called the leaf. The main drawback in this was if there is any modification or addition made to the structure then the whole structure needed alteration which made the task a tedious one. In order to avoid this next system took its origin which is called as the Network Database System.

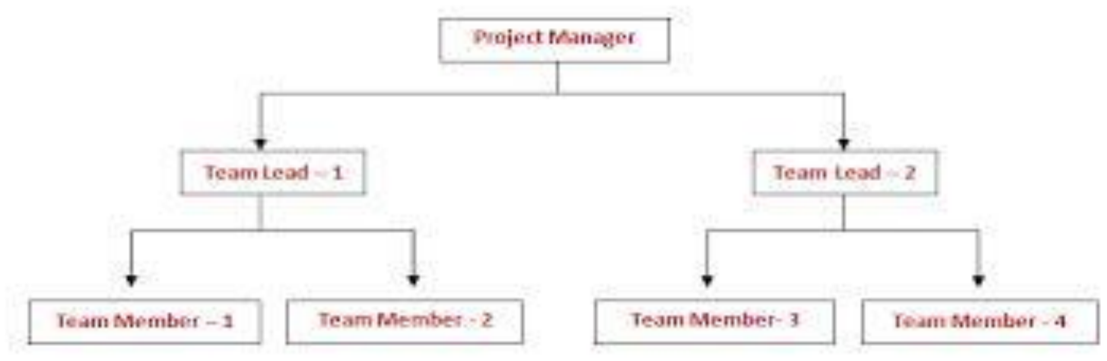
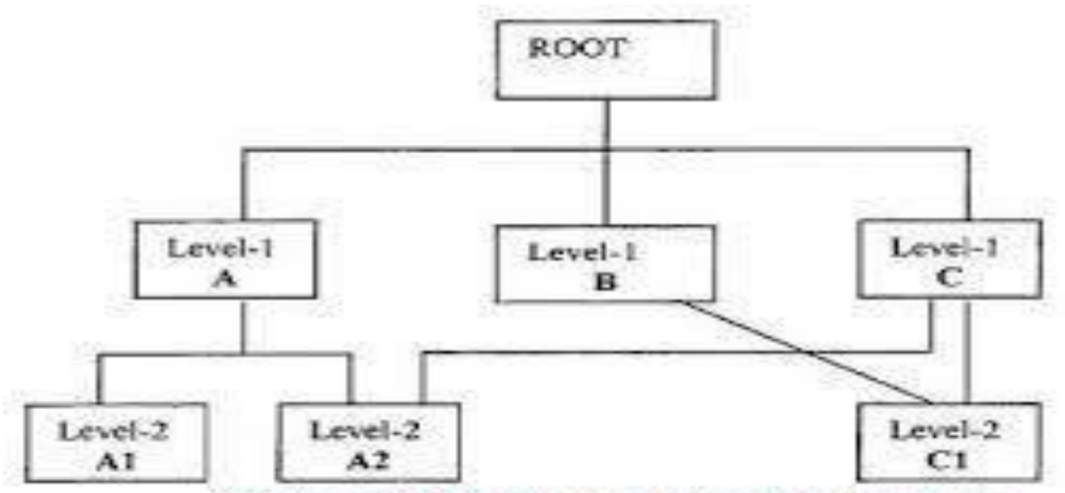


Fig: Hierarchical Database System

<u>Network Database System</u>:

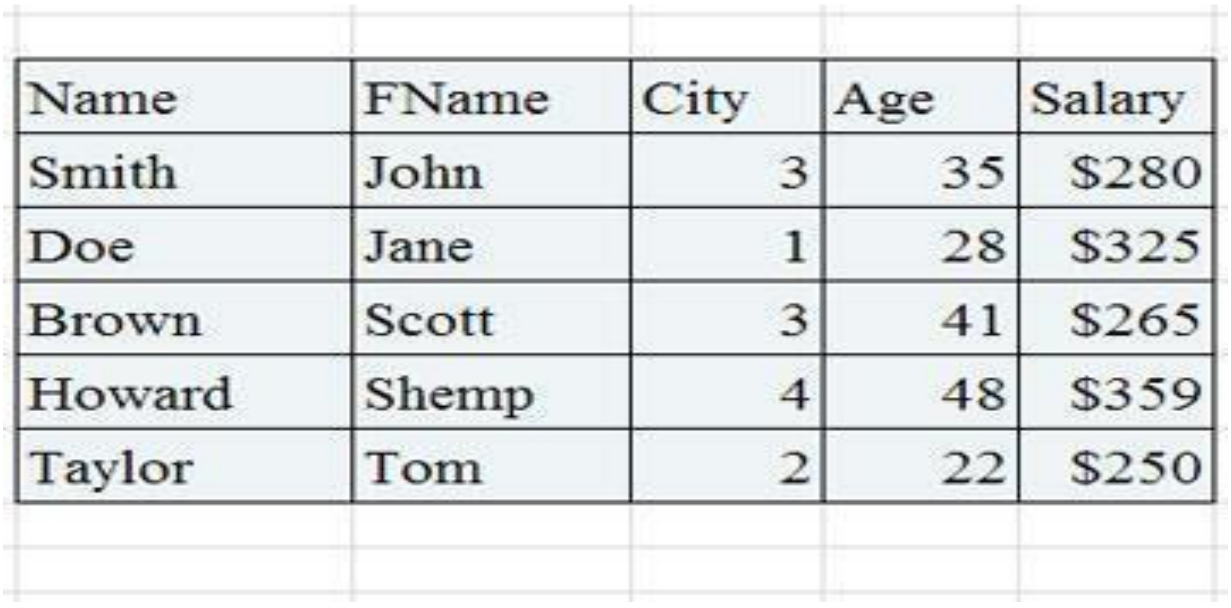
In this the main concept of many-many relationships got introduced. But this also followed the same technology of pointers to define relationships with a difference in this made in the introduction if grouping of data items as sets.



Network Database Model Diagram

Relational Database System:

In order to overcome all the drawbacks of the previous systems, the Relational Database System got introduced in which data get organized as tables and each record forms a row with many fields or attributes in it. Relationships between tables are also formed in this system.



Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

DATABASE:

A database is a collection of related data.

(OR)

A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

Examples / Applications of Database Systems:

The following are the various kinds of applications/organizations uses databases for their business processing activities in their day-to-day life. They are:

1. **Banking:** For customer information, accounts, and loans, and banking transactions.

2. **Airlines:** For reservations and schedule information. Airlines were among the first to use

databases in a geographically distributed manner—terminals situated around the world accessed the central database system through phone lines and other data networks.

3. **Universities:** For student information, course registrations, and grades.

4. **Credit Card Transactions:** For purchases on credit cards and generation of monthly statements.

5. **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

6. **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.

7. **Sales:** For customer, product, and purchase information.

8. **Manufacturing:** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.

9. **Human resources:** For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.

10. **Railway Reservation Systems:** For reservations and schedule information.

11. **Web:** For access the Back accounts and to get the balance amount.

12. **E –Commerce:** For Buying a book or music CD and browse for things like watches, mobiles from the Internet.

CHARACTERISTICS OF DATABASE:

The database approach has some very characteristic features which are discussed in detail below:

Structured and Described Data:

Fundamental feature of the database approach is that the database system does not only contain the data but also the complete definition and description of these data. These descriptions are basically details about the extent, the structure, the type and the format of all data and, additionally, the relationship between the data. This kind of stored data is called metadata ("data about data").

Separation of Data and Applications:

Application software does not need any knowledge about the physical data storage like encoding, format, storage place, etc. It only communicates with the management system of a database (DBMS) via a standardized interface with the help of a standardized language like SQL. The access to the data and the metadata is entirely done by the DBMS. In this way all the applications can be totally separated from the data.

Data Integrity:

Data integrity is a byword for the quality and the reliability of the data of a database system. In a broader sense data integrity includes also the protection of the database from unauthorized access (confidentiality) and unauthorized changes. Data reflect facts of the real world.

Transactions:

A transaction is a bundle of actions which are done within a database to bring it from one consistent state to a new consistent state. In between the data are inevitable inconsistent. *A* transaction is atomic what

means that it cannot be divided up any further. Within a transaction all or none of the actions need to be carried out. Doing only a part of the actions would lead to an inconsistent database state.

Example: One example of a transaction is the transfer of an amount of money from one bank account to another.

Data Persistence:

Data persistence means that in a DBMS all data is maintained as long as it is not deleted explicitly. The life span of data needs to be determined directly or indirectly be the user and must not be dependent on system features. Additionally data once stored in a database must not be lost. Changes of a database which are done by a transaction are persistent. When a transaction is finished even a system crash cannot put the data in danger

TYPES OF DATABASES:

Database can be classified according to the following factors. They are:

1. Number of Users
2. Database Location
3. Expected type
4. Extent of use

1. Based on number of Users:

According to the number of users the databases can be classified into following types. They are :

a). Single user b). Multiuser

Single user database:

- Single user database supports only one user at a time.
- Desktop or personal computer database is an example for single user database.

Multiuser database:

- Multi user database supports multiple users at the same time.
- Workgroup database and enterprise databases are examples for multiuser database.

Workgroup database:

If the multiuser database supports relatively small number of users (fewer than 50) within an organization is called as Workgroup database.

Enterprise database:

If the database is used by the entire organization and supports multiple users (more than 50) across many departments is called as Enterprise database.

2. Based on Location:

According to the location of database the databases can be classified into following types. They are:
a).CentralizedDatabase
b).Distributed Database

<u>Centralized Database</u>:

It is a database that is located, stored, and maintained in a single location. This location is most often a central computer or database system, for example a desktop or server CPU, or a mainframe computer. In most cases, a centralized database would be used by an organization (e.g. a business company) or an institution (e.g. a university.)

Distributed Database:

A distributed database is a database in which storage devices are not all attached to a common CPU. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers.

<u>INTRODUCTION TO DATABASE-MANAGEMENT SYSTEM:</u>

<u>Database Management System:</u>

- A **database-management system** (DBMS) is a collection of interrelated data and a set of programs to access those data.

- The DBMS is a general purpose software system that facilitates the process of defining constructing and manipulating databases for various applications.

<u>Goals of DBMS:</u>

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*

1. Manage large bodies of information

2. Provide convenient and efficient ways to store and access information

3. Secure information against system failure or tampering

4. Permit data to be shared among multiple users

Properties of DBMS:

1. A Database represents some aspect of the real world. Changes to the real world reflected in the database.

2. A Database is a logically coherent collection of data with some inherent meaning.

3. A Database is designed and populated with data for a specific purpose.

Need of DBMS:

1. Before the advent of DBMS, organizations typically stored information using a "File Processing Systems".

Example of such systems is File Handling in High Level Languages like C, Basic and COBOL etc., these systems have Major disadvantages to perform the Data Manipulation. So to overcome those drawbacks now we are using the DBMS.

2. Database systems are designed to manage large bodies of information.

3. In addition to that the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

ADVANTAGES OF A DBMS OVER FILE SYSTEM:

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.

Concurrent Access and Crash Recovery:

A database system allows several users to access the database concurrently. Answering different questions from different users with the same (base) data is a central aspect of an information system. Such concurrent use of data increases the economy of a system.

An example for concurrent use is the travel database of a bigger travel agency. The employees of different branches can access the database concurrently and book journeys for their clients. Each travel agent sees on his interface if there are still seats available for a specific journey or if it is already fully booked.

A DBMS also protects data from failures such as power failures and crashes etc. by the recovery schemes such as backup mechanisms and log files etc.

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 fine-tuning the storage of the data to make retrieval efficient.

Reduced Application Development Time:

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.

DISADVANTAGES OF DBMS:

Danger of a Overkill:

For small and simple applications for single users a database system is often not advisable.

Complexity:

A database system creates additional complexity and requirements. The supply and operation of a database management system with several users and databases is quite costly and demanding.

Qualified Personnel:

`The professional operation of a database system requires appropriately trained staff. Without a qualified database administrator nothing will work for long.

Costs:

Through the use of a database system new costs are generated for the system itself but also for additional hardware and the more complex handling of the system.

Lower Efficiency:

A database system is a multi-use software which is often less efficient than specialized software which is produced and optimized exactly for one problem.

DATABASE USERS & DATABASE ADMINISTRATORS:

People who work with a database can be categorized as database users or database administrators.

Database Users:

There are four different types of database-system users, differentiated by the way they expect to interact with the system.

Naive users:

Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

For example, a bank teller who needs to transfer $50 from account *A* to account *B* invokes a program called *transfer*. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

Application programmers:

Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. **Rapid application development (RAD)** tools are tools that enable an application programmer to construct forms and reports without writing a program.

Sophisticated users:

Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a **query processor**, whose function is to break down DML statements into instructions that the storage manager understands. Analysts who submit queries to explore data in the database fall in this category.

Specialized users:

Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.

Database Administrator:

One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a **database administrator** (**DBA**).

Database Administrator Functions/Roles:

The functions of a DBA include:

Schema definition:

The DBA creates the original database schema by executing a set of data definition statements in the DDL, Storage structure and access-method definition.

Schema and physical-organization modification:

The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

Granting of authorization for data access:

By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.

Routine maintenance:

Examples of the database administrator's routine maintenance activities are:

1. Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.

2. Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.

3. Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

LEVELS OF ABSTRACTION IN A DBMS:

Hiding certain details of how the data are stored and maintained. A major purpose of database system is to provide users with an "Abstract View" of the data. In DBMS there are 3 levels of data abstraction. The goal of the abstraction in the DBMS is to separate the users request and the physical storage of data in the database.

Levels of Abstraction:

Physical Level:

- The lowest Level of Abstraction describes "How" the data are actually stored.
- The physical level describes complex low level data structures in detail.

Logical Level:

- This level of data Abstraction describes “What” data are to be stored in the database and what relationships exist among those data.

- Database Administrators use the logical level of abstraction.

View Level:

- It is the highest level of data Abstracts that describes only part of entire database.
- Different users require different types of data elements from each database.
- The system may provide many views for the some database.

THREE SCHEMA ARCHITECTURE:

Schema:

The overall design of the database is called the “Schema” or “Meta Data”. A database schema corresponds to the programming language type definition. The value of a variable in programming language corresponds to an “Instance” of a database Schema.

Three Schema Architecture:

The goal of this architecture is to separate the user applications and the physical database. In this architecture, schemas can be defined at the following three levels:

1. The **internal level** has an **internal schema,** which 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.

2. The **conceptual level** has a **conceptual schema,** which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. A high-level data model or an implementation data model can be used at this level.

3. The **external** or **view level** includes a number of **external schemas** or **user views.** Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. A high-level data model or an implementation data model can be used at this level.

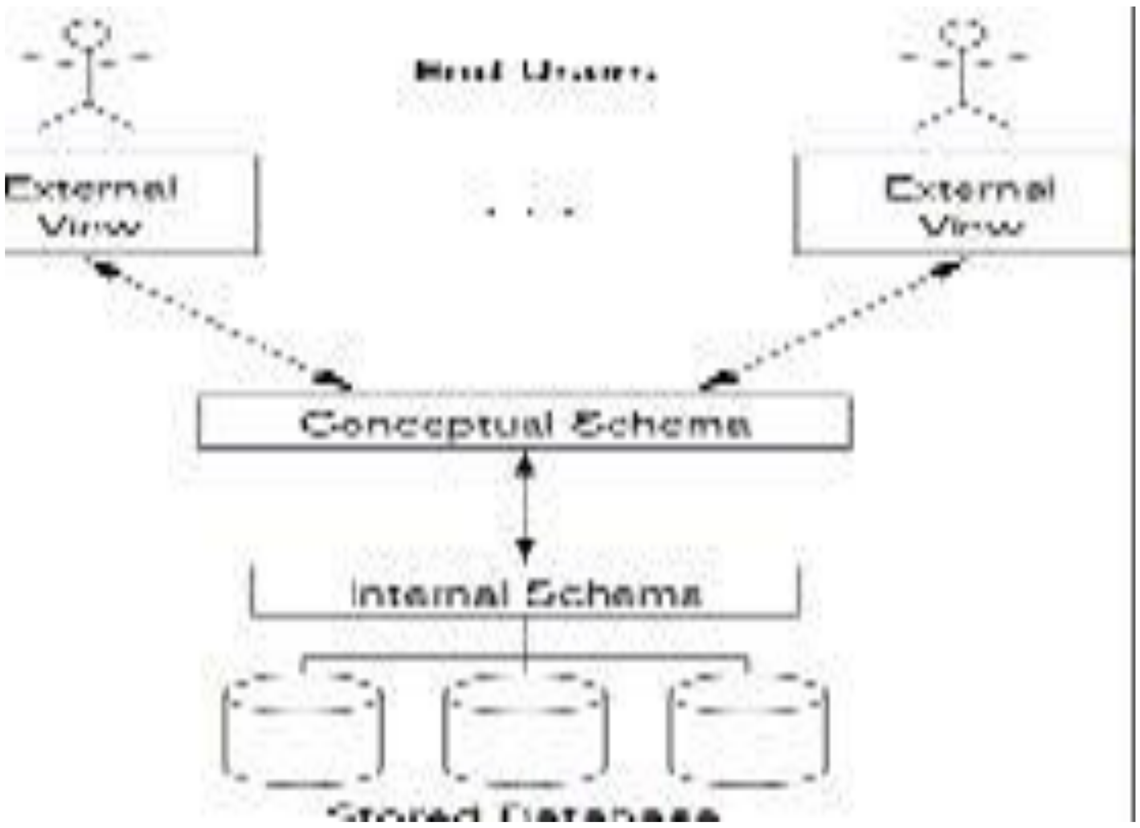


Fig: Three-Schema Architecture

DATA INDEPENDENCE:

- A very important advantage of using DBMS is that it offers Data Independence.
- The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called **data independence**.
- There are two kinds:

1. Physical Data Independence
2. Logical Data Independence

Physical Data Independence:

- The ability to modify the physical schema without causing application programs to be rewritten

- Modifications at this level are usually to improve performance.

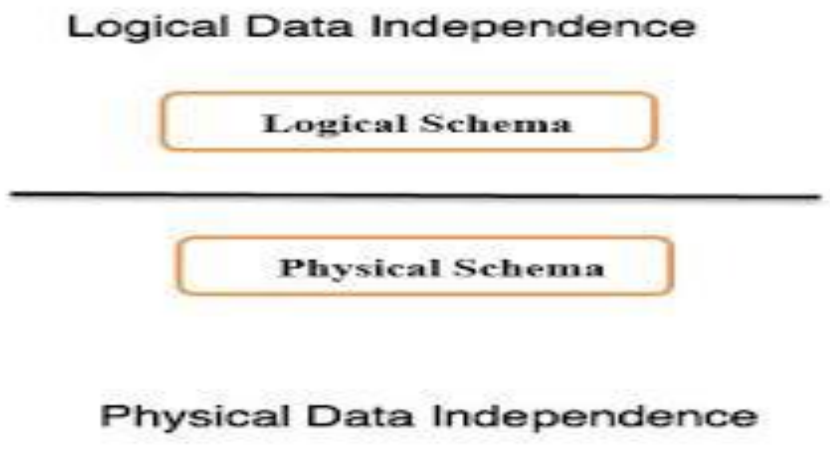


Fig: Data Independence

Logical Data Independence:

- The ability to modify the conceptual schema without causing application programs to be rewritten
- Usually done when logical structure of database is altered

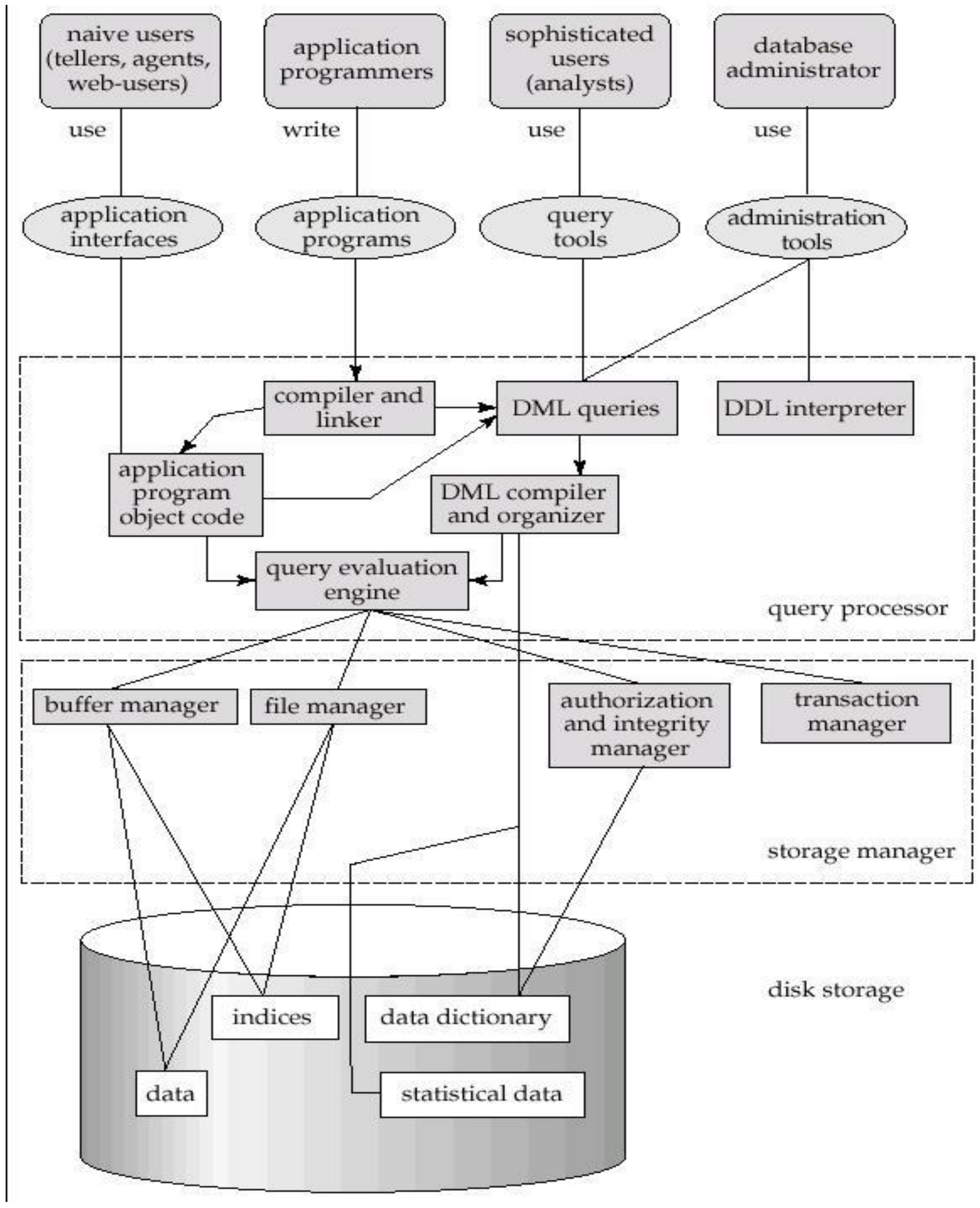
- Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data.

DATABASE SYSTEM STRUCTURE:

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because databases typically require a large amount of storage space. Some Big organizations Database ranges from Giga bytes to Tera bytes. So the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed.

The query processor also very important because it helps the database system simplify and facilitate access to data. So quick processing of updates and queries is important. It is the job of the database system to translate updates and queries written in a nonprocedural language,


naive users (tellers, agents, web-users)
application programmers
sophisticated users (analysts)
database administrator
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
indices
data dictionary
data
statistical data

StorageManager:

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 queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

Storage Manager Components:

Authorization and integrity manager which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

Transaction manager which ensures that the database itself remains in a consistent state despite system failures, and that concurrent transaction executions proceed without conflicting.

File manager: which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

Buffer manager which is responsible for fetching data from disk storage into main memory. Storage manager implements several data structures as part of the physical system implementation. Data files are used to store the database itself. Data dictionary is used to stores metadata about the structure of the database, in particular the schema of the database.

Query Processor Components:

DDL interpreter: It interprets DDL statements and records the definitions in the data dictionary.

DML compiler: It translates 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 the DML compiler.

Application Architectures:

Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines, on which remote database users' work, and server machines, on which the database system runs. Database applications are usually partitioned into two or three parts. They are:

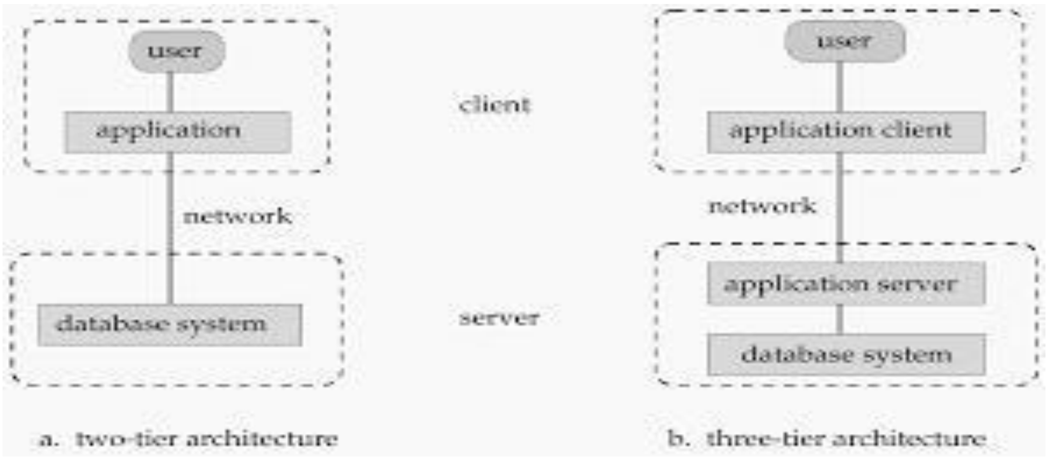
1. Two – Tier Architecture
2. Three – Tier Architecture.

Two-Tier Architecture:

The application is partitioned into a component that resides at the client machine, which invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.

Three-Tier Architecture:

The client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through forms interface. The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.



DATABASE DESIGN:

The database design process can be divided into six steps. The ER Model is most relevant to the first three steps. Next three steps are beyond the ER Model.

1. Requirements Analysis:

The very first step in designing a database application is to understand what data is to be stored in the database, what applications must be built on top of it, and what operations are most frequent and subject to performance requirements. The database designers collect information of the organization and analyzer, the information to identify the user's requirements. The database designers must find out what the users want from the database.

2. Conceptual Database Design:

Once the information is gathered in the requirements analysis step a conceptual database design is developed and is used to develop a high level description of the data to be stored in the database, along with the constraints that are known to hold over this data. This step is often carried out using the ER model, or a similar high-level data model.

3. Logical Database Design:

In this step convert the conceptual database design into a database schema (Logical Database Design) in the data model of the chosen DBMS. We will only consider **relational DBMSs**, and therefore, the task in the

logical design step is to convert an ER schema into a relational database schema. The result is a conceptual schema, sometimes called the **logical schema**, in the relational data model.

Beyond the ER Design:

The first three steps are more relevant to the ER Model. Once the logical scheme is defined designer consider the physical level implementation and finally provide certain security measures. The remaining three steps of database design are briefly described below:

4. Schema Refinement:

The fourth step in database design is to analyze the collection of relations in our relational database schema to identify potential problems, and to refine it. In contrast to the requirements analysis and conceptual design steps, which are essentially subjective, schema refinement can be guided by some elegant and powerful theory.

5. Physical Database Design:

In this step we must consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired performance criteria. This step may simply involve building indexes on some tables and clustering some tables, or it may involve a substantial redesign of parts of the database schema obtained from the earlier design steps.

6. Security Design:

The last step of database design is to include security features. This is required to avoid unauthorized access to database practice after all the six steps. We required Tuning step in which all the steps are interleaved and repeated until the design is satisfactory.

DBMS FUNCTIONS:

- DBMS performs several important functions that guarantee the integrity and consistency of the data in the database.
- Those functions transparent to end users and can be accessed only through the use of DBMS. They include:
  - Data Dictionary Management
  - Data Storage Management
  - Data transformation and Presentation
  - Security Management
  - Multiple Access Control
  - Backup and Recovery Management
  - Data Integrity Management
  - Database Access Languages
  - Databases Communication Interfaces

Data Dictionary Management:

- DBMS stores definitions of database elements and their relationship (Metadata) in the data dictionary.
- The DBMS uses the data dictionary to look up the required data component structures and relationships.
- Any change made in database structure is automatically recorded in the data dictionary.

Data Storage Management:

- Modern DBMS provides storage not only for data but also for related data entities.
- Data Storage Management is also important for database “performance tuning”.
- Performance tuning related to activities that make database more efficiently.

Data Transformation and Presentation:

- DBMS transforms entered data to confirm to required data structures.
- DBMS formats the physically retrieved data to make it confirms to user's logical expectations.
- DBMS also presents the data in the user's expected format.

Security Management:

- DBMS creates a security system that enforces the user security and data privacy.
- Security rules determines which users can access the database, which data items each user can access etc.

- DBA and authenticated user logged to DBMS through username and password or through Biometric authentication such as Finger print and face reorganization etc.

Multiuser Access Control:

- To provide data integrity and data consistency, DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of database.

Backup and Recovery Management:

- DBMS provides backup and recovery to ensure data safety and integrity.

- Recovery management deals with the recovery of database after failure such as bad sector in the disk or power failure. Such capability is critical to preserve database integrity.

Data Integrity Management:

- DBMS provides and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency.

- Ensuring data integrity is especially important in transaction- oriented database systems.

Database Access Languages:

- DBMS provides data access through a query language.

- A query language is a non-procedural language i.e. it lets the user specify what must be done without specifying how it is to be done.

- SQL is the default query language for data access.

<u>Databases Communication Interfaces:</u>

- Current DBMS's are accepting end-user requests via different network environments.

- For example, DBMS might provide access to database via Internet through the use of web browsers such as Mozilla Firefox or Microsoft Internet Explorer.

What is Schema?

A database schema is the skeleton structure that represents the logical view of the entire database. (or)

The logical structure of the database is called as Database Schema. (or)

The overall design of the database is the database schema.

- It defines how the data is organized and how the relations among them are associated.

- It formulates all the constraints that are to be applied on the data.

EG:

STUDENT

SID	SNAME	PHNO

What is Instance?

The actual content of the database at a particular point in time. (Or)

The data stored in the database at any given time is an instance of the database

Student

Sid	Name	phno
1201	Venkat	9014901442
1202	teja	9014774422

In the above table 1201, 1202, Venkat etc are said to be instance of student table.

Difference between File system & DBMS:

File system	DBMS
1. File system is a collection of data. Any management with the file system, user has to write the procedures	1. DBMS is a collection of data and user is not required to write the procedures for managing the database.
2. File system gives the details of the data representation and Storage of data.	2. DBMS provides an abstract view of data that hides the details.
3. In File system storing and retrieving of data cannot be done efficiently.	3. DBMS is efficient to use since there are wide varieties of sophisticated techniques to store and retrieve the data.
4. Concurrent access to the data in the file system has many problems like : Reading the file while other deleting some information, updating some information	4. DBMS takes care of Concurrent access using some form of locking.
5. File system doesn't provide crash recovery mechanism. Eg. While we are entering some data into the file if System crashes then content of the file is lost	5. DBMS has crash recovery mechanism, DBMS protects user from the effects of system failures.
6. Protecting a file under file system is very difficult.	6. DBMS has a good protection mechanism.