UNIT 1 BASIC CONCEPTS

Structure **Page Nos.**

1.0 INTRODUCTION

Databases and database systems have become an essential part of our everyday life. We encounter several activities that involve some interaction with a database almost daily. The examples include deposit and/or withdrawal from a bank, hotel, airline or railway reservation, accessing a computerised library, order a magazine subscription from a publisher, purchase items from supermarkets. In all the above cases a database is accessed. These may be called **Traditional Database Applications.** In these types of databases, the information stored and accessed is textual or numeric. However, with advances in technology in the past few years, different databases have been developed such as **Multimedia Databases** that store pictures, video clips and sound messages; **Geographical Information Systems (GIS)** that can store maps, weather data and satellite images, etc., and Real time databases that can control industrial and manufacturing processes. In this unit, we will be introducing the concepts involved in the Database Management System.

1.1 OBJECTIVES

After going through this unit you should be able to:

- describe the File Based system and its limitations;
- describe the structure of DBMS;
- define the functions of DBA;
- explain the three-tier architecture of DBMS, and
- identify the need for three-tier architecture.

1.2 NEED FOR A DATABASE MANAGEMENT SYSTEM

A Database is an organised, persistent collection of data of an organisation. The database management system manages the database of an enterprise. But why do we need the database management system? To describe it, let us first discuss the alternative to it, that is the file-based system.

1.2.1 The File Based System

File based systems are an early attempt to computerise the manual filing system. For example, a manual file can be set up to hold all the correspondence relating to a particular matter as a project, product, task, client or employee. In an organisation there could be many such files which may be labeled and stored. The same could be done at homes where file relating to bank statements, receipts, tax payments, etc., could be maintained.

What do we do to find information from these files? For retrieval of information, the entries could be searched sequentially. Alternatively, an indexing system could be used to locate the information.

The manual filing system works well when the number of items to be stored is small. It even works quite well when the number of items stored is quite large and they are only needed to be stored and retrieved. However, a manual file system crashes when cross-referencing and processing of information in the files is carried out. For example, in a university a number of students are enrolled who have the options of doing various courses. The university may have separate files for the personal details of students, fees paid by them, the number and details of the courses taught, the number and details of each faculty member in various departments. Consider the effort to answer the following queries.

- Annual fees paid by the students of Computer science department.
- Number of students requiring transport facility from a particular area.
- This year's turnover of students as compared to last year.
- Number of students opting for different courses from different departments.

Please refer to *Figure 1*. The answer to all the questions above would be cumbersome and time consuming in the file based system.

1.2.2 Limitations of File Based System

The file-based system has certain limitations. The limitations are listed as follows:

- **Separation and isolation of data**: When the data is stored in separate files it becomes difficult to access. It becomes extremely complex when the data has to be retrieved from more than two files as a large amount of data has to be searched.
- **Duplication of data:** Due to the decentralised approach, the file system leads to uncontrolled duplication of data. This is undesirable as the duplication leads to wastage of a lot of storage space. It also costs time and money to enter the data more than once. For example, the address information of student may have to be duplicated in bus list file data (*Figure 1*).
- **Inconsistent Data:** The data in a file system can become inconsistent if more than one person modifies the data concurrently, for example, if any student

changes the residence and the change is notified to only his/her file and not to bus list. Entering wrong data is also another reason for inconsistencies.

- **Data dependence:** The physical structure and storage of data files and records are defined in the application code. This means that it is extremely difficult to make changes to the existing structure. The programmer would have to identify all the affected programs, modify them and retest them. This characteristic of the File Based system is called **program data dependence.**

- **Incompatible File Formats:** Since the structure of the files is embedded in application programs, the structure is dependent on application programming languages. Hence the structure of a file generated by COBOL programming language may be quite different from a file generated by 'C' programming language. This incompatibility makes them difficult to process jointly. The application developer may have to develop software to convert the files to some common format for processing. However, this may be time consuming and expensive.

- **Fixed Queries:** File based systems are very much dependent on application programs. Any query or report needed by the organisation has to be developed by the application programmer. With time, the type and number of queries or reports increases. Producing different types of queries or reports is not possible in File Based Systems. As a result, in some organisations the type of queries or reports to be produced is fixed. No new query or report of the data could be generated.

Besides the above, the maintenance of the File Based System is difficult and there is no provision for security. Recovery is inadequate or non-existent.

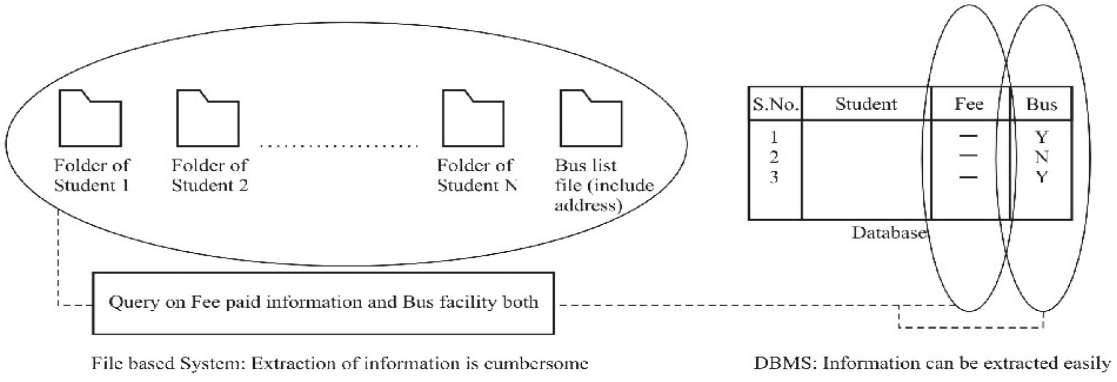


Fig: 1

Figure 1: File based system versus database system

1.2.3 The Database Approach

In order to overcome the limitations of a file system, a new approach was required. Hence a database approach emerged. *A database is a persistent collection of logically related data.* The initial attempts were to provide a centralised collection of data. A database has a self-describing nature. It contains not only the data but also the complete definition of the database structure and constraints, which are stored in a system catalog. A DBMS manages this data. It allows data sharing and integration of data of an organisation in a single database. *DBMS controls access to this data and thus needs to provide features for database creation, data manipulation such as data value modification, data retrieval, data integrity and security etc*. Let us describe some of the advantages of the database approach.

The database approach has many advantages. Let us discuss these in more detail.

Reduction of Redundancies

In a file processing system, each user group maintains its own files resulting in a considerable amount of redundancy of the stored data. This results in wastage of storage space but more importantly may result in data inconsistencies. Also, the same data has to be updated more than once resulting in duplication of effort. The files that represent the same data may become inconsistent as some may be updated whereas others may not be.

In database approach data can be stored at a single place or with controlled redundancy under DBMS, which saves space and does not permit inconsistency.

Shared Data

A DBMS allows the sharing of database under its control by any number of application programs or users. A database belongs to the entire organisation and is shared by all authorised users (may not be the complete data, why?). This scheme can be best explained with the help of a logical diagram (*Figure 2*). New applications can be built and added to the current system and data not currently stored can be stored.

Data Independence

In the file-based system, the descriptions of data and logic for accessing the data are built into each application program making the program more dependent on data. A change in the structure of data may require alterations to programs. Database Management systems separates data descriptions from data. Hence it is not affected by changes. This is called Data Independence, where details of data are not exposed. DBMS provides an abstract view and hides details. For example, logically we can say that the interface or window to data provided by DBMS to a user may still be the same although the internal structure of the data may be changed. (Refer to *Figure 2*).

Improved Integrity

Data Integrity refers to validity and consistency of data. Data Integrity means that the data should be accurate and consistent. This is done by providing some checks or constraints. These are consistency rules that the database is not permitted to violate. Constraints may apply to data items within a record or relationships between records. For example, the age of an employee can be between 18 and 70 years only. While entering the data for the age of an employee, the database should check this. However, if Grades of any student are entered, the data can be erroneously entered as Grade C for Grade A. In this case DBMS will not be able to provide any check as both A and C are of the same data type and are valid values.

Efficient Data Access

DBMS utilises techniques to store and retrieve the data efficiently at least for unforeseen queries. A complex DBMS should be able to provide services to end users, where they can efficiently retrieve the data almost immediately.

Multiple User Interfaces

Since many users having varying levels of technical knowledge use a database, a DBMS should be able to provide a variety of interfaces. This includes ─

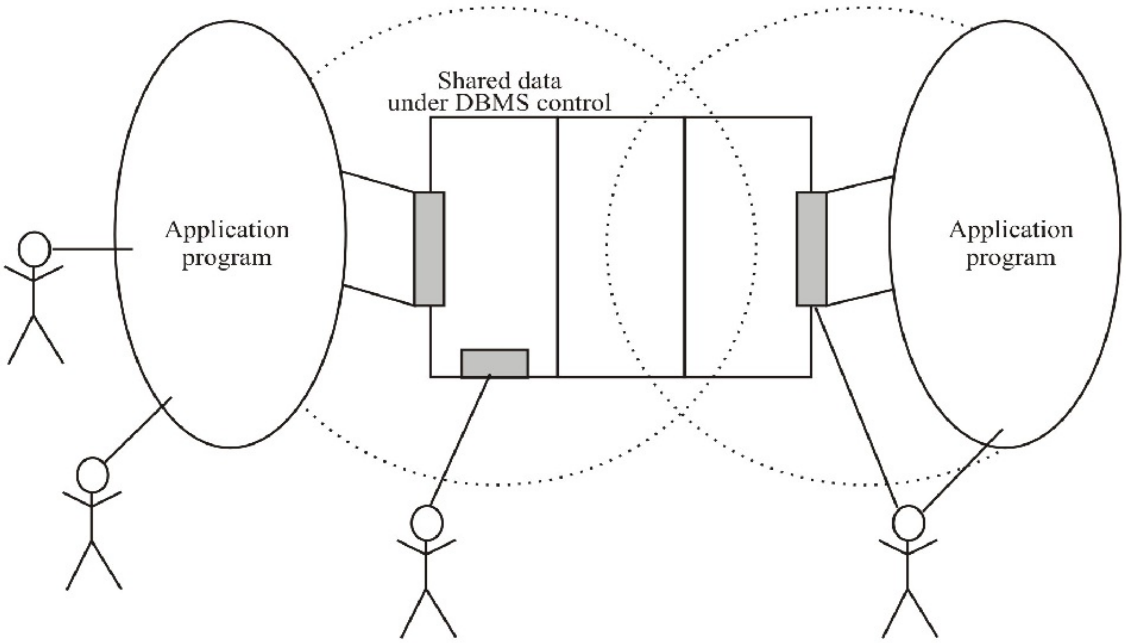
a. query language for casual users,

b. programming language interfaces for application programmers,

c. forms and codes for parametric users,

d. menu driven interfaces, and

e. natural language interfaces for standalone users, these interfaces are still not available in standard form with commercial database.



A user can either access data window through DBMS or use an application program.

Figure 2: User interaction to DBMS

Representing complex relationship among data

A database may include varieties of data interrelated to each other in many ways. A DBMS must have the capability to represent a variety of relationships among the data as well as to retrieve and update related data easily and efficiently.

Improved Security

Data is vital to any organisation and also confidential. In a shared system where multiple users share the data, all information should not be shared by all users. For example, the salary of the employees should not be visible to anyone other than the department dealing in this. Hence, database should be protected from unauthorised users. This is done by Database Administrator (DBA) by providing the usernames and passwords only to authorised users as well as granting privileges or the type of operation allowed. This is done by using security and authorisation subsystem. Only authorised users may use the database and their access types can be restricted to only retrieval, insert, update or delete or any of these. For example, the Branch Manager of any company may have access to all data whereas the Sales Assistant may not have access to salary details.

Improved Backup and Recovery

A file-based system may fail to provide measures to protect data from system failures. This lies solely on the user by taking backups periodically. DBMS provides facilities for recovering the hardware and software failures. A backup and recovery subsystem is responsible for this. In case a program fails, it restores the database to a state in which it was before the execution of the program.

Support for concurrent transactions

A transaction is defined as the unit of work. For example, a bank may be involved in a transaction where an amount of Rs.5000/- is transferred from account X to account Y. A DBMS also allows multiple transactions to occur simultaneously.

Check Your Progress 1

1) What is a DBMS?

……

……

……

……

……

2) What are the advantages of a DBMS?

……

……

……

……

……..

3) Compare and contrast the traditional File based system with Database approach.

……

……

……

……

……..

1.3 THE LOGICAL DBMS ARCHITECTURE

Database Management Systems are very complex, sophisticated software applications that provide reliable management of large amounts of data. To describe general database concepts and the structure and capabilities of a DBMS better, the architecture of a typical database management system should be studied.

There are two different ways to look at the architecture of a DBMS: the *logical DBMS architecture* and the *physical DBMS architecture*. The logical architecture deals with the way data is stored and presented to users, while the physical architecture is concerned with the software components that make up a DBMS.

1.3.1 Three Level Architecture of DBMS or Logical DBMS Architecture

The logical architecture describes how data in the database is perceived by users. It is not concerned with how the data is handled and processed by the DBMS, but only with how it looks. The method of data storage on the underlying file system is not revealed, and the users can manipulate the data without worrying about where it is located or how it is actually stored. This results in the database having different levels of abstraction.

The majority of commercial Database Management S ystems available today are based on the ANSI/SPARC generalised DBMS architecture, as proposed by the ANSI/SPARC Study Group on Data Base Management Systems. Hence this is also called as the ANSI/SPARC model. It divides the system into three levels of abstraction: the *internal* or *physical level*, the *conceptual level*, and the *external* or *view level*. The diagram below shows the logical architecture for a typical DBMS.

The External or View Level

The external or view level is the highest level of abstraction of database. It provides a window on the conceptual view, which allows the user to see only the data of interest to them. The user can be either an application program or an end user. There can be many external views as any number of external schema can be defined and they can overlap each other. It consists of the definition of logical records and relationships in the external view. It also contains the methods for deriving the objects such as entities, attributes and relationships in the external view from the Conceptual view.

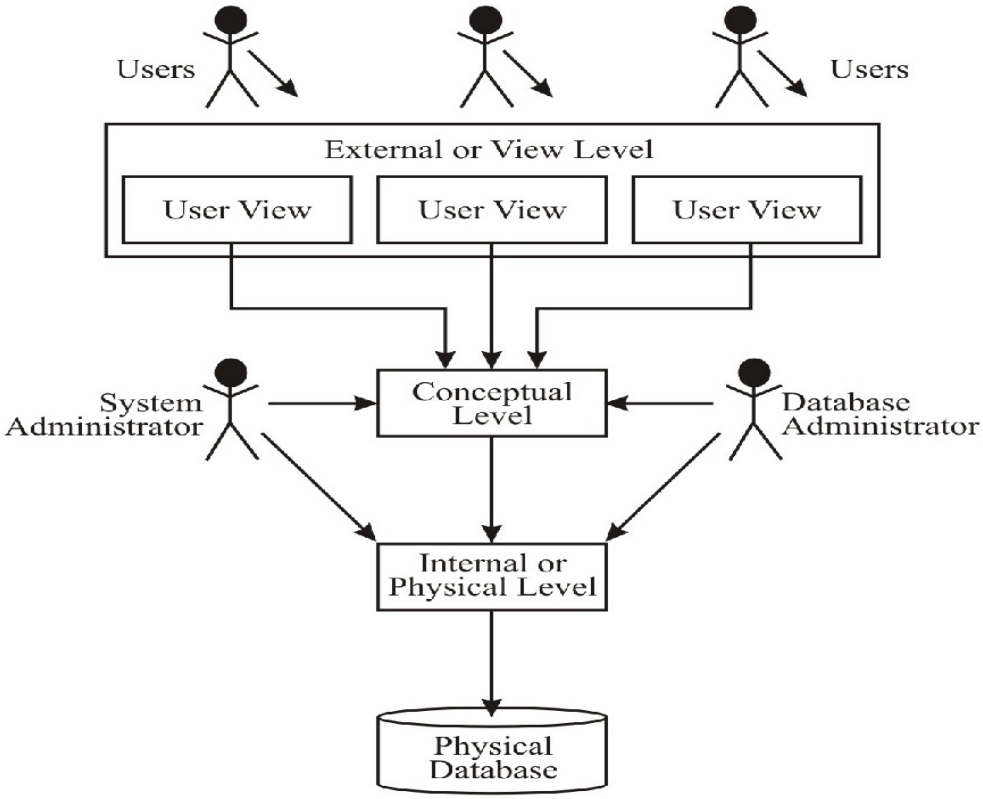


Figure 3: Logical DBMS Architecture

The Conceptual Level or Global level

The conceptual level presents a logical view of the entire database as a unified whole. It allows the user to bring all the data in the database together and see it in a consistent manner. Hence, there is only one conceptual schema per database. The first stage in the design of a database is to define the conceptual view, and a DBMS provides a data definition language for this purpose. It describes all the records and relationships included in the database.

The data definition language used to create the conceptual level must not specify any physical storage considerations that should be handled by the physical level. It does not provide any storage or access details, but defines the information content *only*.

The Internal or Physical Level

The collection of files permanently stored on secondary storage devices is known as the physical database. The physical or internal level is the one closest to physical storage, and it provides a low-level description of the physical database, and an interface between the operating systems file system and the record structures used in higher levels of abstraction. It is at this level that record types and methods of storage are defined, as well as how stored fields are represented, what physical sequence the stored records are in, and what other physical structures exist.

1.3.2 Mappings between Levels and Data Independence

The three levels of abstraction in the database do not exist independently of each other. There must be some correspondence, or mapping, between the levels. There are two types of mappings: the conceptual/internal mapping and the external/conceptual mapping.

The conceptual/internal mapping lies between the conceptual and internal levels, and defines the correspondence between the records and the fields of the conceptual view and the files and data structures of the internal view. *If the structure of the stored database is changed, then the conceptual/ internal mapping must also be changed accordingly so that the view from the conceptual level remains constant.* It is this mapping that provides physical data independence for the database. For example, we may change the internal view of student relation by breaking the student file into two files, one containing enrolment, name and address and other containing enrolment, programme. However, the mapping will make sure that the conceptual view is restored as original. The storage decision is primarily taken for optimisation purposes.

The external/conceptual view lies between the external and conceptual levels, and defines the correspondence between a particular external view and the conceptual view. Although these two levels are similar, some elements found in a particular external view may be different from the conceptual view. For example, several fields can be combined into a single (virtual) field, which can also have different names from the original fields. If the structure of the database at the conceptual level is changed, then the external/conceptual mapping must change accordingly so that the view from the external level remains constant. It is this mapping that provides logical data independence for the database. For example, we may change the student relation to have more fields at conceptual level, yet this will not change the two user views at all.

It is also possible to have another mapping, where one external view is expressed in terms of other external views (this could be called an external/external mapping). This is useful if several external views are closely related to one another, as it allows you to avoid mapping each of the similar external views directly to the conceptual level.

1.3.3 The need for three level architecture

The objective of the three level architecture is to separate each user's view of the database from the way the database is physically represented.

- **Support of multiple user views:** Each user is able to access the same data, but have a different customized view of the data. Each user should be able to change the way he or she views the data and this change should not affect other users.
- **Insulation between user programs and data that does not concern them:** Users should not directly deal with physical storage details, such as indexing or hashing. The user's interactions with the database should be independent of storage considerations.

Insulation between conceptual and physical structures

It can be defined as:

1. The Database Administrator should be able to change the storage structures without affecting users' views.
2. The internal structure of the database should be unaffected by the changes to the physical aspects of the storage, such as changing to a new storage device.
3. The DBA should be able to change the conceptual structure of the database without affecting all users.

1.4 PHYSICAL DBMS ARCHITECTURE

The physical architecture describes the software components used to enter and process data, and how these software components are related and interconnected. Although it is not possible to generalise the component structure of a DBMS, it is possible to identify a number of key functions which are common to most database management systems. The components that normally implement these functions are shown in *Figure 4*, which depicts the physical architecture of a typical DBMS.

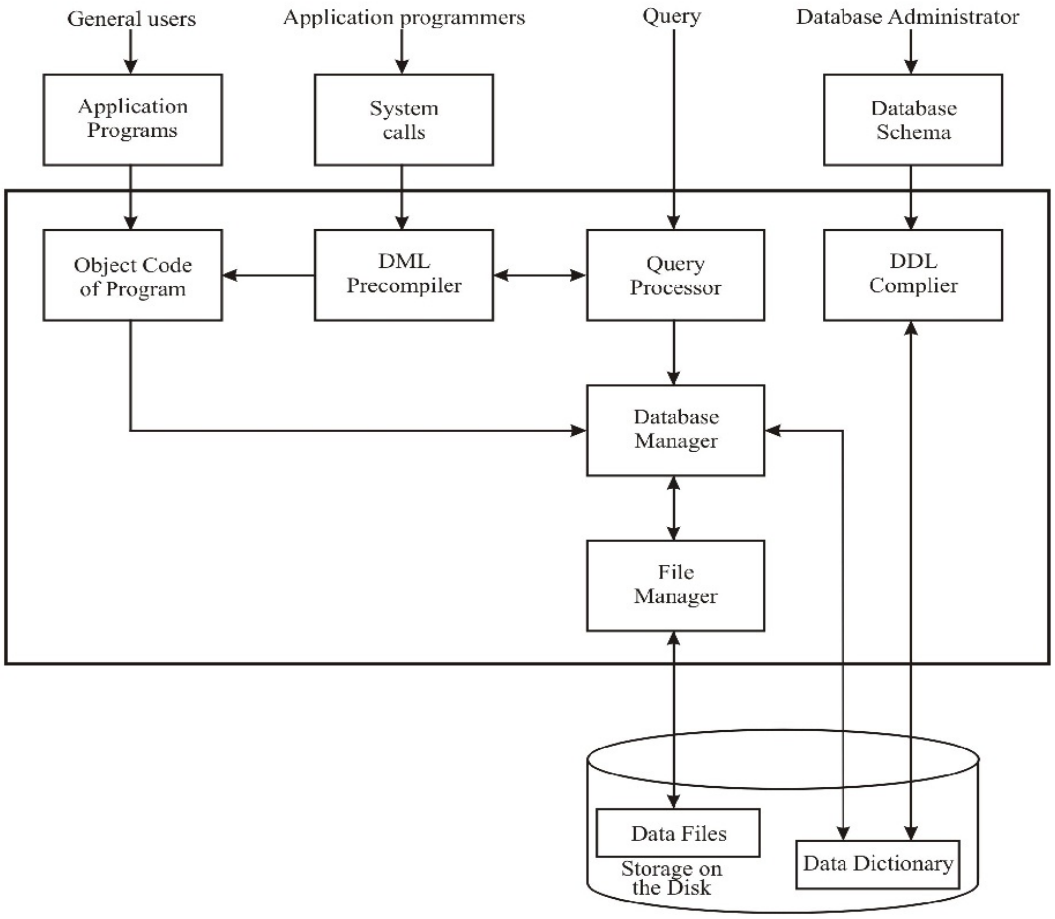


Figure 4: DBMS Structure

Based on various functions, the database system may be partitioned into the following modules. Some functions (for example, file systems) may be provided by the operating system.

1.4.1 DML Precompiler

All the Database Management systems have two basic sets of Languages – Data Definition Language (DDL) that contains the set of commands required to define the format of the data that is being stored and Data Manipulation Language (DML) which defines the set of commands that modify, process data to create user definable output. The DML statements can also be written in an application program. The DML precompiler converts DML statements (such as SELECT…FROM in Structured Query Language (SQL) covered in Block 2) embedded in an application program to normal procedural calls in the host language. The precompiler interacts with the query processor in order to generate the appropriate code.

1.4.2 DDL Compiler

The DDL compiler converts the data definition statements (such as CREATE TABLE …. in SQL) into a set of tables containing metadata tables. These tables contain information concerning the database and are in a form that can be used by other components of the DBMS. These tables are then stored in a system catalog or data dictionary.

1.4.3 File Manager

File manager manages the allocation of space on disk storage. It establishes and maintains the list of structures and indices defined in the internal schema that is used to represent information stored on disk. However, the file manager does not directly manage the physical output and input of data. It passes the requests on to the appropriate access methods, which either read data from or write data into the system buffer or cache. The file manager can be implemented using an interface to the existing file subsystem provided by the operating system of the host computer or it can include a file subsystem written especially for the DBMS.

1.4.4 Database Manager

It is the interface between low-level data, application programs and queries. Databases typically require a large amount of storage space. It is stored on disks, as the main memory of computers cannot store this information. Data is moved between disk storage and main memory as needed. Since the movement of data to and from disk is slow relative to the speed of the control processing unit of computers, it is imperative that database system structure data so as to minimise the need to move data between disk and main memory.

A database manager is a program module responsible for interfacing with the database file system to the user queries. In addition, the tasks of enforcing constraints to maintain the consistency and integrity of the data as well as its security are also performed by database manager. Synchronising the simultaneous operations performed by concurrent users is under the control of the data manager. It also performs backup and recovery operations. Let us summarise now the important responsibilities of Database manager.

- **Interaction with file manager:** The raw data is stored on the disk using the file system which is usually provided by a conventional operating system. The database manager translates the various DML statements into low-level file system commands. Thus, the database manager is responsible for the actual storing, retrieving and updating of data in the database.
- **Integrity enforcement:** 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 (for example, Rs. 1000/-). Similarly the number of holidays per year an employee may be having should not exceed 8 days. These constraints must be specified explicitly by the DBA. If such constraints are specified, then the database manager can check whether updates to the database result in the violation of any of these constraints and if so appropriate action may be imposed.
- **Security enforcement:** As discussed above, not every user of the database needs to have access to the entire content of the database. It is the job of the database manager to enforce these security requirements.
- **Backup and recovery:** A computer system like any other mechanical or electrical device, is subject to failure. There are a variety of causes of such failure, including disk crash, power failure and software errors. In each of these cases, information concerning the database is lost. It is the responsibility of database manager to detect such failures and restore the database to a state that existed prior to the occurrence of the failure. This is usually accomplished through the backup and recovery procedures.
- **Concurrency control:** When several users update the database concurrently, the consistency of data may no longer be preserved. It is necessary for the system to control the interaction among the concurrent users, and achieving such a control is one of the responsibilities of database manager.

The above functions are achieved through the database manager. The major components of a database manager are:

- **Authorisation control:** This module checks that the user has necessary authorisation to carry out the required function.
- **Command Processor:** Once the system has checked that the user has authority to carry out the operation, control is passed to the command processor, which converts commands to a logical sequence of steps.
- **Integrity checker:** For an operation that changes the database, the integrity checker checks that the requested operation satisfies all necessary integrity constraints such as key constraints.
- **Query Optimiser:** This module determines an optimal strategy for the query execution.
- **Transaction Manager:** This module performs the required processing of operations of various transactions. The transaction manager maintains tables of authorisation Concurrency. The DBMS may use authorisation tables to allow the transaction manager to ensure that the user has permission to execute the desired operation on the database. The authorisation tables can only be modified by properly authorised user commands, which are themselves checked against the authorisation tables.

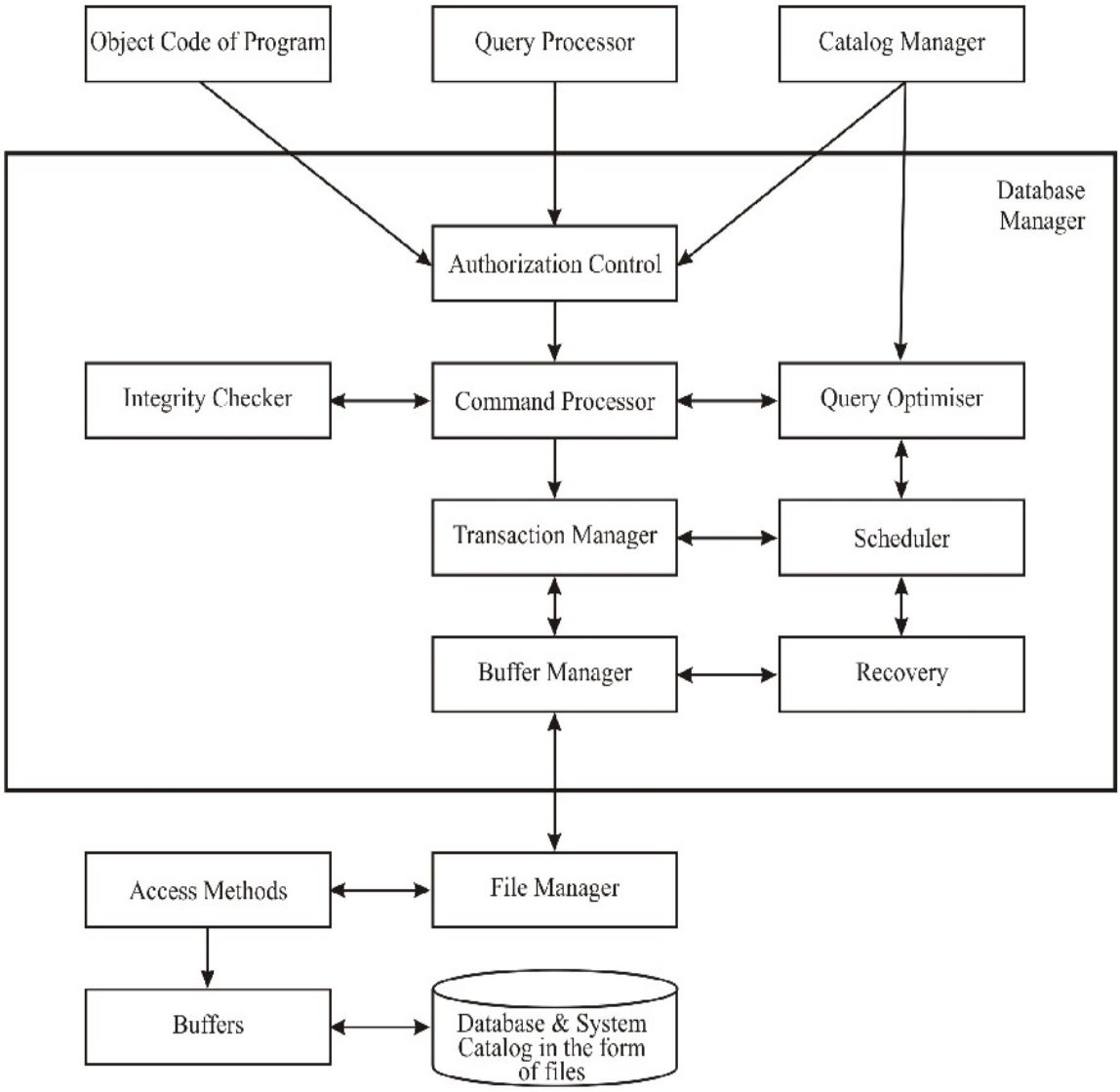


Figure 5: Components of Database Manager

- **Scheduler:** This module is responsible for ensuring that concurrent operations or transactions on the database proceed without conflicting with one another. It controls the relative order in which transaction operations are executed. A database may also support concurrency control tables to prevent conflicts when simultaneous, conflicting commands are executed. The DBMS checks the concurrency control tables before executing an operation to ensure that the data used by it is not locked by another statement.

- **Recovery Manager:** This module ensures that the database remains in a consistent state in the presence of failures. It is responsible for transaction commit and abort, that is success or failure of transaction.

- **Buffer Manager:** This module is responsible for the transfer of data between main memory and secondary storage, such as disk and tape. The recovery manager and the buffer manager are sometimes collectively referred to as *data manager*. The buffer manager is sometimes known as *cache manager*.

1.4.5 Query Processor

The query language processor is responsible for receiving query language statements and changing them from the English-like syntax of the query language to a form the DBMS can understand. The query language processor usually consists of two separate parts: the parser and the query optimizer.

The parser receives query language statements from application programs or command-line utilities and examines the syntax of the statements to ensure they are correct. To do this, the parser breaks a statement down into basic units of syntax and examines them to make sure each statement consists of the proper component parts. If the statements follow the syntax rules, the tokens are passed to the query optimizer.

The query optimiser examines the query language statement, and tries to choose the best and most efficient way of executing the query. To do this, the query optimizer will generate several query plans in which operations are performed in different orders, and then try to estimate which plan will execute most efficiently. When making this estimate, the query optimizer may examine factors such as: CPU time, disk time, network time, sorting methods, and scanning methods.

1.4.6 Database Administrator

One of the main reasons for having the database management system is to have control of both data and programs accessing that data. The person having such control over the system is called the database administrator (DBA). The DBA administers the three levels of the database and defines the global view or conceptual level of the database. The DBA also specifies the external view of the various users and applications and is responsible for the definition and implementation of the internal level, including the storage structure and access methods to be used for the optimum performance of the DBMS. Changes to any of the three levels due to changes in the organisation and/or emerging technology are under the control of the DBA. Mappings between the internal and the conceptual levels, as well as between the conceptual and external levels, are also defined by the DBA. The DBA is responsible for granting permission to the users of the database and stores the profile of each user in the database. This profile describes the permissible activities of a user on that portion of the database accessible to the user via one or more user views. The user profile can be used by the database system to verify that a particular user can perform a given operation on the database.

The DBA is also responsible for defining procedures to recover the database from failures due to human, natural, or hardware causes with minimal loss of data. This

recovery procedure should enable the organisation to continue to function and the intact portion of the database should continue to be available.

Thus, the functions of DBA are:

- **Schema definition:** Creation of the original database schema is accomplished by writing a set of definitions which are translated by the DDL compiler to a set of tables that are permanently stored in the data dictionary.
- **Storage Structure and access method definition:** The creation of appropriate storage structure and access method is accomplished by writing a set of definitions which are translated by the data storage and definition language compiler.
- **Schema and Physical organisation modification:** DBA involves either the modification of the database schema or the description of the physical storage organisation. These changes, although relatively rare, are accomplished by writing a set of definitions which are used by either the DDL compiler or the data storage and definition language compiler to generate modification to the appropriate internal system tables (for example the data dictionary).
- **Granting of authorisation for data access:** DBA allows the granting of different types of authorisation for data access to the various users of the database.
- **Integrity constraint specification:** The DBA specifies the constraints. These constraints are kept in a special system structure, the data dictionary that is consulted by the database manager prior to any data manipulation. Data Dictionary is one of the valuable tools that the DBA uses to carry out data administration.

1.4.7 Data files Indices and Data Dictionary

The data is stored in the data files. The indices are stored in the index files. Indices provide fast access to data items. For example, a book database may be organised in the order of Accession number, yet may be indexed on Author name and Book titles.

Data Dictionary: A Data Dictionary stores information about the structure of the database. It is used heavily. Hence a good data dictionary should have a good design and efficient implementation. It is seen that when a program becomes somewhat large in size, keeping track of all the available names that are used and the purpose for which they were used becomes more and more difficult. After a significant time if the same or another programmer has to modify the program, it becomes extremely difficult.

The problem becomes even more difficult when the number of data types that an organisation has in its database increases. The data of an organisation is a valuable corporate resource and therefore some kind of inventory and catalog of it must be maintained so as to assist in both the utilisation and management of the resource. It is for this purpose that a data dictionary or dictionary/directory is emerging as a major tool. A dictionary provides definitions of things. A directory tells you where to find them. A data dictionary/directory contains information (or data) about the data.

A comprehensive data dictionary would provide the definition of data items, how they fit into the data structure and how they relate to other entities in the database. In DBMS, the data dictionary stores the information concerning the external, conceptual and internal levels of the databases. It would combine the source of each data field value, that is from where the authenticate value is obtained. The frequency of its use and audit trail regarding the updates including user identification with the time of each update is also recorded in Data dictionary.

The Database administrator (DBA) uses the data dictionary in every phase of a database life cycle, starting from the data gathering phase to the design,

implementation and maintenance phases. Documentation provided by a data dictionary is as valuable to end users and managers, as it is essential to the programmers. Users can plan their applications with the database only if they know exactly what is stored in it. For example, the description of a data item in a data dictionary may include its origin and other text description in plain English, in addition to its data format. Thus, users and managers will be able to see exactly what is available in the database. A data dictionary is a road map which guides users to access information within a large database.

An ideal data dictionary should include everything a DBA wants to know about the database.

1. External, conceptual and internal database descriptions.
2. Descriptions of entities (record types), attributes (fields), as well as cross-references, origin and meaning of data elements.
3. Synonyms, authorisation and security codes.
4. Which external schemas are used by which programs, who the users are, and what their authorisations are.
5. Statistics about database and its usage including number of records, etc.

A data dictionary is implemented as a database so that users can query its contents.

The cost effectiveness of a data dictionary increases as the complexity of an information system increases. A data dictionary can be a great asset not only to the DBA for database design, implementation and maintenance, but also to managers or end users in their project planning.

Check Your Progress 2

1) What are the major components of Database Manager?

……

……

……

2) Explain the functions of the person who has the control of both data and programs accessing that data.

……

……

1.5 COMMERCIAL DATABASE ARCHITECTURE

At its most basic level the DBMS architecture can be broken down into two parts: the *back end* and the *front end*.

The back end is responsible for managing the physical database and providing the necessary support and mappings for the internal, conceptual, and external levels described in a later section. Other benefits of a DBMS, such as security, integrity, and access control, are also the responsibility of the back end.

The front end is really just any application that runs on top of the DBMS. These may be applications provided by the DBMS vendor, the user, or a third party. The user interacts with the front end, and may not even be aware that the back end exists. This interaction is done through Applications and Utilities which are the main interface to the DBMS for most users.

There are three main sources of applications and utilities for a DBMS:

a. *Vendor applications and utilities* are provided for working with or maintaining the database, and usually allow users to create and manipulate a database without the need to write custom applications.

b. *User applications* are generally custom-made application programs written for a specific purpose using a conventional programming language. This programming language is coupled to the DBMS query language through the *application program interface (API)*. This allows the user to utilise the power of the DBMS query language with the flexibility of a custom application.

c. *Third party applications* may be similar to those provided by the vendor, but with enhancements, or they may fill a perceived need that the vendor hasn't created an application for. They can also be similar to user applications, being written for a specific purpose they think a large majority of users will need.

The most common applications and utilities used with a database can be divided into several well-defined categories. These are:

- **Command Line Interfaces:** These are character-based, interactive interfaces that let you use the full power and functionality of the DBMS query language directly. They allow you to manipulate the database and perform ad-hoc queries and see the results immediately. They are often the only method of exploiting the full power of the database without creating programs using a conventional programming language.
- **Graphical User Interface (GUI) tools:** These are graphical, interactive interfaces that hide the complexity of the DBMS and query language behind an intuitive, easy to understand, and convenient interface. This allows casual users the ability to access the database without having to learn the query language, and it allows advanced users to quickly manage and manipulate the database without the trouble of entering formal commands using the query language. However, graphical interfaces usually do not provide the same level of functionality as a command line interface because it is not always possible to implement all commands or options using a graphical interface.
- **Backup/Restore Utilities:** These are designed to minimise the effects of a database failure and ensure a database is restored to a consistent state if a failure does occur. Manual backup/restore utilities require the user to initiate the backup, while automatic utilities will back up the database at regular intervals without any intervention from the user. Proper use of a backup/restore utility allows a DBMS to recover from a system failure correctly and reliably.
- **Load/Unload Utilities:** These allow the user to unload a database or parts of a database and reload the data on the same machine, or on another machine in a different location. This can be useful in several situations, such as for creating backup copies of a database at a specific point in time, or for loading data into a new version of the database or into a completely different database. These load/unload utilities may also be used for rearranging the data in the database to improve performance, such as clustering data together in a particular way or reclaiming space occupied by data that has become obsolete.
- **Reporting/Analysis Utilities:** These are used to analyse and report on the data contained in the database. This may include analysing trends in data, computing values from data, or displaying data that meets some specified criteria, and then displaying or printing a report containing this information.

1.6 DATA MODELS

After going through the database architecture, let us now dwell on an important question: how is the data organised in a database? There are many basic structures

that exist in a database system. They are called the database models. A database model defines

- The logical data structure
- Data relationships
- Data consistency constraints.

The following Table defines various types of Data Models

Model Type	Examples
Object-based Models: Use objects as key data representation components.	• Entity-Relationship Model: It is a collection of real world objects called entities and their relationships. It is mainly represented in graphical form using E-R diagrams. This is very useful in Database design. These have been explained in Unit 2 of this Block 1. • Object-Oriented Model: Defines the database as a collection of objects that contains both data members/values and operations that are allowed on the data. The interrelationships and constraints are implemented through objects, links and message passing mechanisms. Object-Models are useful for databases where data interrelationship are complex, for example, Computer Assisted Design based components.
Record based Logical Models: Use records as the key data representation components	Relational Model: It represents data as well as relationship among data in the form of tables. Constraints are stored in a meta-data table. This is a very simple model and is based on a proven mathematical theory. This is the most widely used data base model and will be discussed in more detail in the subsequent units. Network Model: In this model data is represented as records and relationship as links. A simple network model example is explained in *Figure 6*. It shows a sample diagram for such a system. This model is a very good model as far as conceptual framework is concerned but is nowadays not used in database management systems.
Hierarchical Data Representation Model	Hierarchical Model: It defines data as and relationships through hierarchy of data values. *Figure 7* shows an example of hierarchical model. These models are now not used in commercial DBMS products.

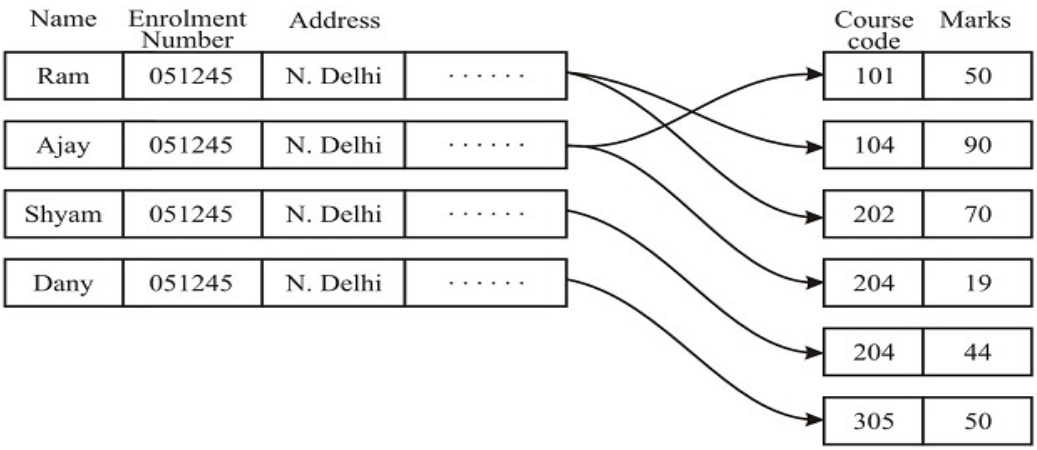


Figure 6: An example of Network Model

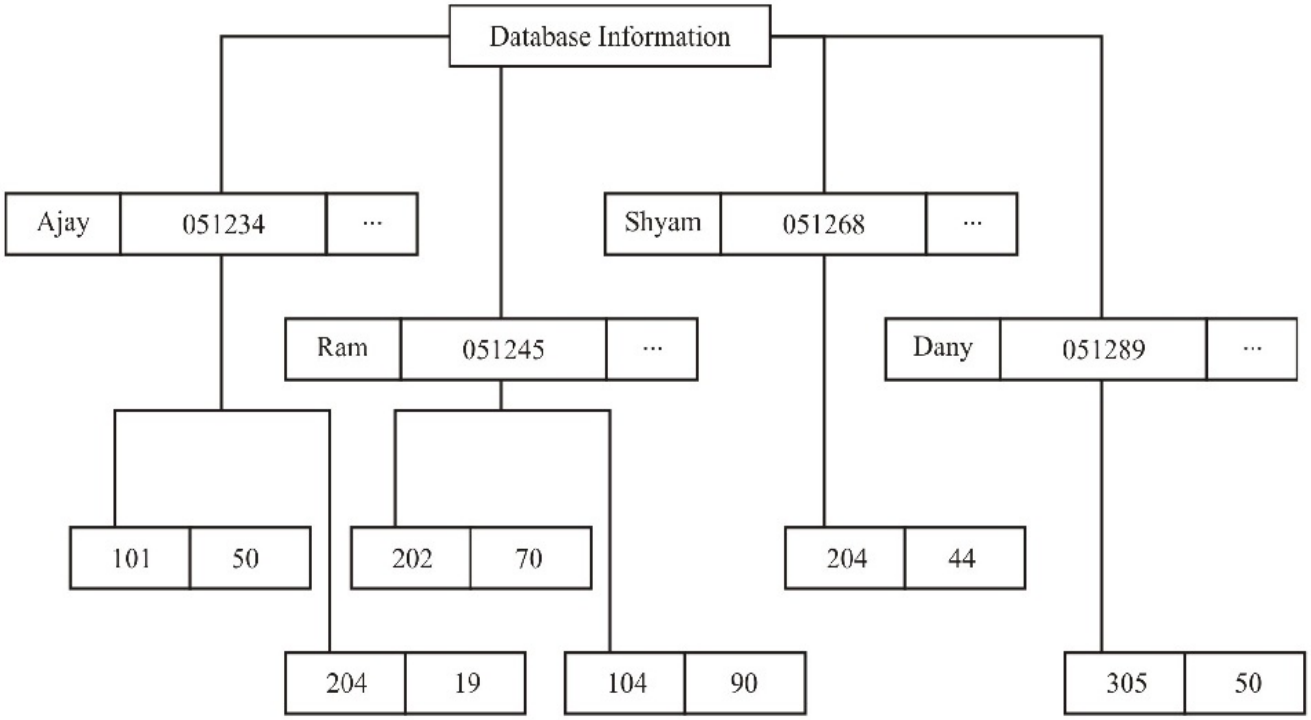


Figure 7: An example of Hierarchical Model

Check your Progress 3

State whether the following are **True** or **False.**

T	F

1) The external schema defines how and where data are organised in physical data storage. ☐
2) A schema separates the physical aspects of data storage from the logical aspects of data representation. ☐
3) The conceptual schema defines a view or views of the database for particular users. ☐
4) A collection of data designed to be used by different people is called a database. ☐
5) In a database, the data are stored in such a fashion that they are independent of the programs of people using the data. ☐
6) Using a database redundancy can be reduced. ☐
7) The data in a database cannot be shared. ☐
8) Security restrictions are impossible to apply in a database. ☐
9) In a database data integrity can be maintained. ☐
10) Independence means that the three levels in the schema (internal, conceptual and external) should be independent of each other so that the changes in the schema at one level should not affect the other levels. ☐

1.7 SUMMARY

Databases and database systems have become an essential part of our everyday life. We encounter several activities that involve some interaction with a database almost daily. File based systems were an early attempt to computerise the manual filing system. This system has certain limitations. In order to overcome the limitations of

file-based system, a new approach, a database approach, emerged. A database is a collection of logically related data. This has a large number of advantages over the file-based approach. These systems are very complex, sophisticated software applications that provide reliable management of large amounts of data.
There are two different ways to look at the architecture of a DBMS: the *logical DBMS architecture* and the *physical DBMS architecture*. The logical architecture deals with the way data is stored and presented to users, while the physical architecture is concerned with the software components that make up a DBMS.

The physical architecture describes the software components used to enter and process data, and how these software components are related and interconnected. At its most basic level the physical DBMS architecture can be broken down into two parts: the *back end* and the *front end*.

The logical architecture describes how data in the database is perceived by users. It is not concerned with how the data is handled and processed by the DBMS, but only with how it looks. The method of data storage on the underlying file system is not revealed, and the users can manipulate the data without worrying about where it is located or how it is actually stored. This results in the database having different levels of abstraction such as the *internal* or *physical level*, the *conceptual level*, and the *external* or *view level*. The objective of the three level architecture is to separate each user's view of the database from the way the database is physically represented.

Finally, we have a brief introduction to the concepts of database Models.

1.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1) DBMS manages the data of an organisation. It allows facilities for defining, updating and retrieving data of an organisation in a sharable, secure and reliable way.

2)
- Reduces redundancies
- Provides environment for data independence
- Enforces integrity
- Security
- Answers unforeseen queries
- Provides support for transactions, recovery etc.

3)

File Based System	Database Approach
Cheaper	Costly
Data dependent	Data independent
Data redundancy	Controlled data redundancy
Inconsistent Data	Consistent Data
Fixed Queries	Unforeseen queries can be answered

Check Your Progress 2

1) Integrity enforcement, control of file manager, security, backup, recovery, concurrency control, etc.

2) A database administrator is normally given such controls. His/her functions are: defining database, defining and optimising storage structures, and control of security, integrity and recovery.

Check Your Progress 3

1. False 2. True 3. False 4. False 5. True 6. True 7. False 8. False 9. True 10. True