U-I: Database Management Systems 1-1

Introduction to Databases

A database is a collection of related data

A database has the following implicit properties:

- A database represents some aspects of the real world, sometimes called the miniworld or the universe of discourse (UOD). Changes to the miniworld are reflected in the database
- A database is a logically coherent collection of data with some inherent meaning — not random assortment
- A database is designed, built and populated with data for a specific purpose. — has an intended group

A database can be of any size and complexity.

An example of a large commercial database is Amazon.com. It contains data for over 20M million books, CDs, videos, DVDs, games, electronics, apparel and other items. The database occupies over 2TB and is stored on 200 different computers. About 10 million visitors access Amazon.com every day and use the database to make purchases. About 100 people will manage the database up-to-date

A DBMS is a collection of programs that enables users to create and maintain a database. The DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating and sharing databases among various users and applications.

Defining a database involves specifying the datatypes, structures and constraints of the data to be stored in the database

Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS.

Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.

Sharing a database allows multiple users and programs to access the database simultaneously.

An application program accesses the database by sending queries or requests for data to the DBMS

important functions provided by the DBMS include protecting the database and maintaining it over a long period of time.

Characteristics of the Database Approach :-

In traditional file processing, each user defines and implements the files needed for a specific s/w application as part of programming the application

In the database approach, a single repository maintains data that is defined once and then accessed by various users.

The main characteristics of the database approach versus the file processing approach are
- Self describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

Self describing nature :- A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition description of the database structure and constraints. This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The info stored in the catalog is called meta-data which describes the structure of the primary database.

The DBMS software must work equally well with any number of database applications

The definitions are specified by the database designer prior to creating the actual database and are stored in the catalog.

Relation between programs and data and Data abstraction

In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file.

But, DBMS access programs do not require such changes in most cases due to program data independence.

In object-oriented and object-relational database systems, users can define operations on data as part of the database definitions. An operation is specified in two parts: the interface and the implementation. User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented through program operation independence.

Data abstraction is the characteristic that allows program data independence and program operation independence.

A data model is a type of data abstraction that is used to provide the conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented.

Support of multiple views of the data

~~Two~~ Different users of the database may require a different perspective or view of the database. A view may contain virtual data that is derived from the database files but is not explicitly stored. A ~~multiple~~ multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

Sharing of Data and Multiuser Transaction processing -

A multiuser DBMS must allow multiple users to access the database at the same time. The DBMS must include concurrency control s/w to ensure that several users trying to update the same data do so in a controlled manner so that the result of the update is correct.

A fundamental role of multiuser DBMS s/w is to ensure that concurrent transactions operate correctly and efficiently.

A transaction is an executing program or process that includes one or more database accesses. The DBMS must enforce several transaction properties: Isolation, atomicity

In large organizations, many people are involved in the design, use and maintenance of a large database with hundreds of users. They are,

- Database Administrators (DBA), Database designers, End users, System Analysts and Application programmers

End users: Casual end users, Naive or parametric end users, Sophisticated end users, Standalone users.

Advantages of using the DBMS approach: —

1. Controlling Redundancy —
2. Restricting unauthorized access —
3. providing persistent storage for program objects —
4. providing storage structures and search techniques for efficient query processing —
5. providing Backup and Recovery
6. providing multiple user interfaces
7. Representing complex Relationships among Data
8. Enforcing integrity constraints —
9. permitting inferencing and Actions using Rules.
10. Additional implications: Potential for enforcing standards
    Reduced Application Development time
    Flexibility
    Availability of up-to-date information
    Economics of scale

History of Database Applications: —

Early Database Applications using Hierarchical and Network Systems: —

Many early database applications maitained records in large organizations such as cooporations, universities and banks. In many of these applications, there were large number of records of similar structure.

Early database Systems did not provide sufficient data abstraction and program data independence capabilities

Most of the early database systems were implemented on large and expensive to-use mainframe computers in the mid 1960s to 1970s & 80s. The shortcoming of these systems were that they provided only programming language interfaces.

The main paradigms used in the early systems are hierarchical systems, network model based systems and inverted file systems.

Providing Data abstraction and Application flexibility :— with Relational Databases

Relational databases were proposed to seperate the physical storage of data from its conceptual representation and to provide a mathematical foundation for data representation and querying. With relational database systems data abstraction and program-data independence were much improved when compared to early systems. The commercial RDBMS were available from early 1980s. Performance of these systems improved with the development of new storage and indexing techniques and better query processing and optimization.

Object-oriented Applications and the need for more complex databases

Object Oriented Databases (OODBs) were developed in the 1980s with the emergence of object oriented programming languages. These databases need to store and share complex and structured objects and incorporated many of the useful object oriented paradigms. They are mainly used in specialized applications such as engineering design, multimedia publishing and manufacturing systems. Later ORDBMS were developed

Interchanging Data on the web for E-commerce using XML

The WWW provides a large network of interconnected computers. In the 1990s E-commerce emerged as a major application on the web. A variety of techniques were developed to allow the interchange of data on the web. XML is considered to be

The primary standard for interchanging data among various types of databases and web pages

Extending Database capabilities for new applications :

Database systems now offer extensions to better support the specialized requirements for some of the applications :

- Scientific applications
- Storage and retrieval of images
- Storage and retrieval of videos
- Data mining applications
- Spatial applications
- Time series applications

DBMS developers add functionality to the systems. Many large organizations use a variety of also application packages that work closely with database back-ends.

ERP and CRM are examples

Data Models, Schemas and Instances :-

A data model is a collection of concepts that can be used to describe the structure of a database (data types, relationships and constraints)

Data models also include a set of basic operations for specifying retrievals and updates on the database, and concepts to specify the dynamic or aspect or behavior of a database application

Categories of data models :-

1. High level or Conceptual data models.
2. Low level or physical data models
3. Representation or implementation data models.

Conceptual data models provide concepts that are close to the way many users perceive data. They use concepts such as entities, attributes and relationships.

An entity represents a real world object or concept.

An attribute represents some property of interest that further describes an entity.

Relationship among two or more entities represents an association among the entities

Representational data models provide concepts that describe the details of how data is stored on the computer storage media may be easily understood by end users but that are not too far removed from the way data is organized in computer storage.

Some of the representational models are

relational data model, network data model and hierarchical data model

Relational data model is used most frequently and widely in traditional commercial DBMSs

- These data models represent data by using record structures and hence sometimes are called record based data models

Object data model is an example of a new family of higher level implementation data models that are closer to conceptual data models.

physical data models provide concepts that describe the details of how data is stored on the computer storage media. These models describe how data is stored as files in the computer
- by representing information such as record formats, record orderings and access paths.

Schemas, Instances and Database state :-

The description of a database is called the database schema which is specified during database design and is not expected to change frequently.

A display schema is called a schema diagram

The data in the database at a particular moment in time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database

The DBMS is partly responsible for ensuring that every state of the database is a valid state - a state that satisfies the structure and constraints specified in the schema

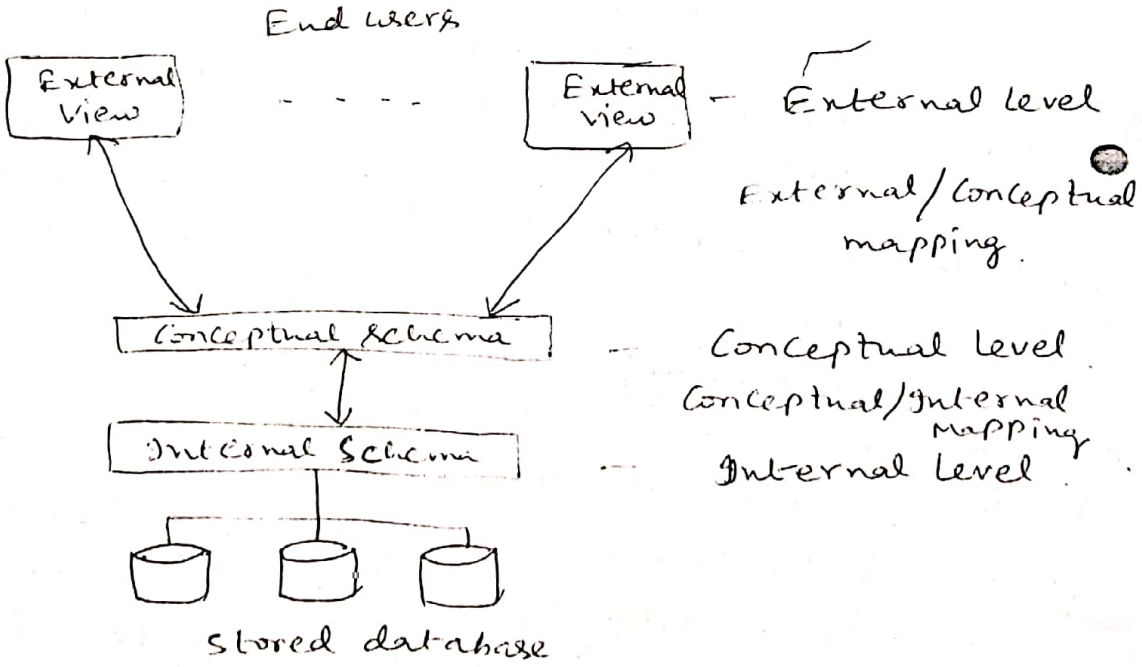
Three Schema Architecture :-

On the architecture, Schemas can be classified as 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 representational data model is used to describe the conceptual schema.

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 program. Each external schema is implemented using a representational data model.



Most DBMSs do not separate the three levels completely and explicitly, but support the three schema architecture to some extent. The three schemas are only descriptions of data; the stored data that actually exists is at the physical level only.

The DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called mappings.

Data Independence :— It is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

Two types of data independence — Logical
physical

- Logical data independence :— It is the capacity to change the conceptual schema without having to change external schemas or application programs. The conceptual schema may be changed to expand the database, to change constraints or to reduce the database. Only the view definition and the mappings need to be changed in a DBMS that supports logical data independence

- Physical data independence :— It is the capacity to change the internal schema without having to change the conceptual schema. Changes to the internal schema may be needed because some physical files were reorganized.

Generally physical data independence exists in most data bases and file environments. Logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs.

Database languages and Interfaces :— The DBMS must provide appropriate languages and interfaces for each category of users.

DBMS Languages :— In many DBMSs, where no strict separation of levels is maintained, the data definition language (DDL), is used by the DBA and by database designers to define both schemas.

In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only. Another language, the storage definition language (SDL) is used to specify the internal schema. In many DBMSs where no clear separation is maintained between the two levels, the DDL is used to specify both conceptual and internal schemas. In most relational DBMSs the internal schema is specified by a combination of functions, parameters and specifications related to storage. The view definition language (VDL) is used to specify user views and their mappings to the conceptual schema. In most DBMSs the DDL is used to define both conceptual and external schemas. In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries.

The Data manipulation language (DML) is used to manipulate the database (retrieval, insertion, deletion & modification of the data).

Two main types of DMLs - A high level or nonprocedural DML
A low level or procedural DML

A non procedural DML can be used on its own to specify complex database operations concisely.

A low procedural DML must be embedded in a general purpose programming language. This type of DML retrieves individual records or objects from the database and processes each separately. These DMLs are called record-at-a-time DMLs. High level DMLs, such as SQL, can specify and retrieve many records in a single DML statement; therefore, they are called set-at-a-time or set-oriented DMLs.

Whenever DML commands, are embedded in a general-purpose programming language, that language is called the host language and the DML is called the data sublanguage.
A high level DML used in a standalone interactive manner is called a query language.

DBMS Interfaces :-

Menu-based Interfaces for web clients or Browsing :- These interfaces present the user with lists of options that lead the user through the formulation of a request. The query is composed step-by-step by picking options from a menu that is displayed by the system

Forms-Based Interfaces :- A form-based interface displays a form to each user. Users can fill out all of the form entries to insert new data, or fill only certain entries. Many DBMSs have forms specification languages which are special languages that help programmers specify such forms

- Oracle forms is a component of the Oracle product suite that provides an extensive set of features to design and build applications using forms.

Graphical User Interfaces :- A GUI displays a schema to the user in diagrammatic form. The user can then specify a query by manipulating the diagram. Most GUIs use a pointing device such as a mouse, to select parts of the displayed schema diagram.

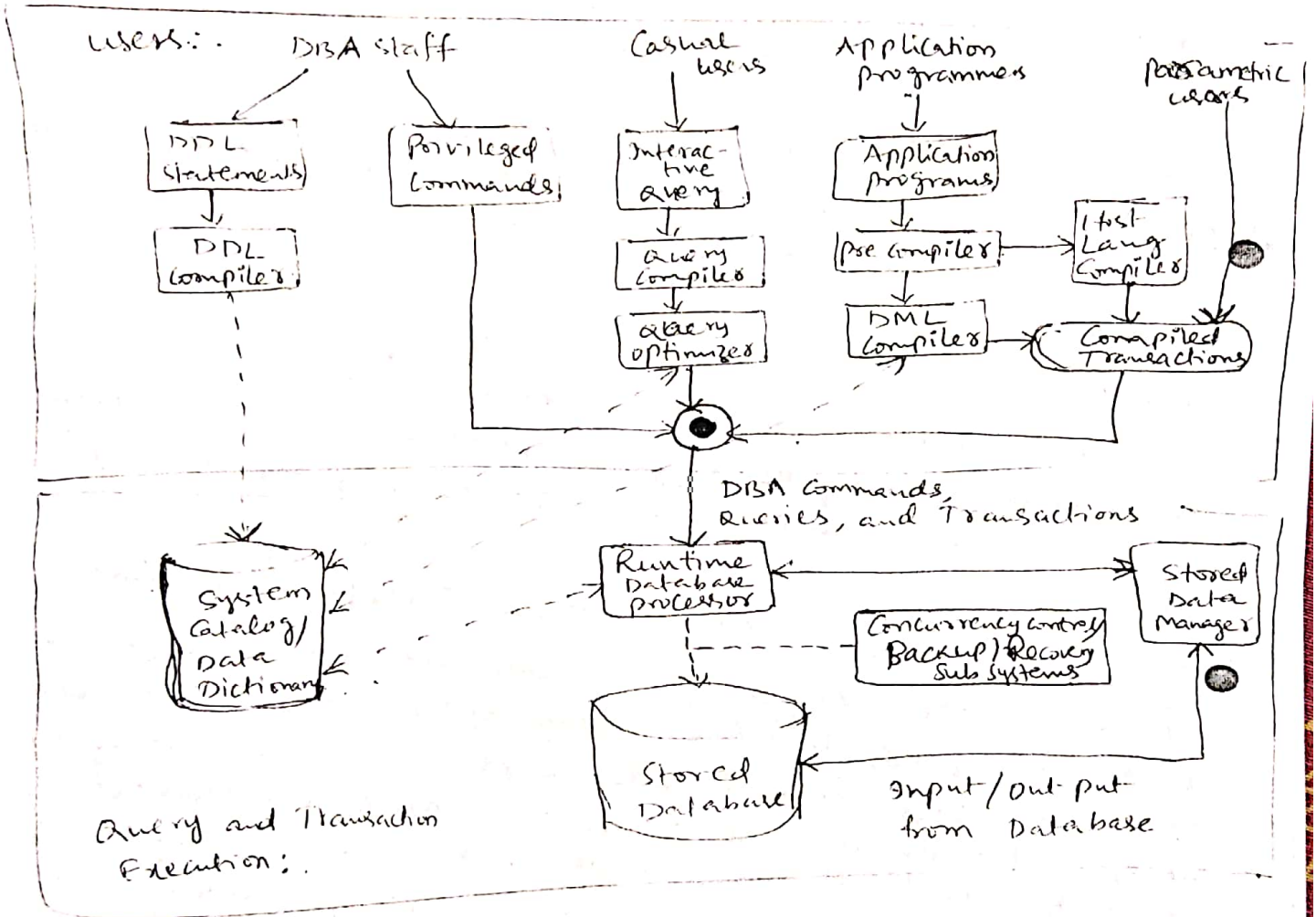
- **Natural Language Interfaces :-** These interfaces have their own schemas similar to the database conceptual schemas. The natural language interface refers to the words in its schema as well as to the set of standard words in its dictionary, to interpret the request. If the interpretation is successful, the interface generates a high level query corresponding to the natural language request and submits it to the DBMS for processing. They use predefined indexes on words and use ranking functions to retrive and present resulting documents in a decreasing degree of match.

Speech I/P and O/P :- Applications with limited vocabularies such as Enquiries for telephone directory, flight arrival/dep. and credit card account information are allowing speech for input and output to enable customers to access this information

Interfaces for parametric users :- These users have a small set of operations that they must perform repeatedly. System analysts and programmers design and implement a special interface for each known class of naive users. Usually a small set of abbreviated commands is included, with the goal of minimizing the number of keystrokes required for each request.

The database System Environment :-

DBMS Component Modules :-



The top part of the figure refers to the various users of the database environment and their interfaces. The lower part shows the internals of the DBMS responsible for storage of data and processing of transactions.

The top part of the figure shows interfaces for the DBA staff, Casual users who work with interactive interfaces to formulate queries, application programmers who create programs [illegible] some host programming languages and parametric users

who do data entry work by supplying parameters to predefined transactions. The DBA staff works on defining and tuning the database

The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas in the DBMS Catalog.

The interactive query interface is used by casual users with occasional need for information. These queries are parsed and validated for correctness of the query syntax, the names of files and data elements by a query compiler that compiles them into an internal form.

This internal query is subjected to query optimizer which is concerned with the rearrangement and possible reordering of operations, elimination of redundancies and use of correct algorithms and indexes during execution.

Application programmers write programs in host languages such as Java, C, or C++ that are submitted to a precompiler. The precompiler extracts DML commands from an application program written in a host programming language. These commands are sent to the DML compiler for compilation into object code for database access. The rest of the program is sent to the host language compiler.

The object codes and the rest of the program are linked, forming a canned transaction. These transactions are executed ~~are executed~~ repeatedly by parameteric users.

In the lower part of the figure, the runtime database processor executes 1) the privileged commands, 2) the executable query plans, and 3) the canned transactions with runtime parameters. It works with the system catalog and updates it with statistics. It also works with the stored data manager, which in turn uses basic operating system services for carrying out low level I/O operations between the disk and main memory

The Concurrency Control and backup and recovery systems are integrated into the working of the runtime database processor for purposes of transaction management.

The DBMS interacts with the OS when disk accesses to the database or to the catalog are needed. If the computer system is shared by many users, the OS will schedule DBMS disk access requests and DBMS processing along with other processes.

Database System Utilities :- Most DBMSs have database utilities that help the DBA to manage the database system.

Functions of common utilities are :

Loading, Backup, Database storage reorganization, Performance monitoring

Centralized and Client/server Architectures for DBMSs :-

Centralized DBMSs architecture :- These architectures used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality. Fig Pg no: 45

users accessed these systems via computer terminals that did not have processing power and only provided display capabilities. These terminals were connected to the central computer via various types of communication networks.

Basic client/server architecture :- This architecture was developed to deal with computing environments in which large number of PCs, workstations, file servers, printers, database servers, web servers, email servers and other s/w and equipment are connected via a n/w.

The idea is to define specialized servers with specific functionalities Eg: file servers, printer servers, web or email server etc

The resources provided by specialized servers can be accessed by many client machines.

The client machines provide the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications.

The concept of this architecture assumes an underlying framework that consists of many PCs and workstations as well as a smaller number of mainframe machines connected via different types of computer networks.

Client - Provides user interface capabilities & local processing

Server - Contains h/w & s/w and provide services to the client such as file access, printing or database access.

Two tier Client/Server architecture for DBMS:

In RDBMS SQL acts as a logical dividing point between client and server

Client - User interface & application programs

Server - Query & transaction functionality

Server is called as a query server or transaction server or SQL server.

When DBMS access is required, the program establishes a connection to the DBMS on the client machine. Once the connection is created, the client can communicate with the DBMS.

A standard, ODBC provides an API that allows client side programs to call the DBMS. ODBC exists as an interface that is installed on the client machine. A client program can be compiled to several RDBMSs and send query

and transaction requests using the ODBC API, which are then processed at server sites.

JDBC is another standard used by Java client programs to access one or more DBMSs through a standard interface. fig pg no: 46

Object oriented ~~data~~ DBMSs — s/w modules of the DBMS were divided s/w client & server in a more integrated way.

For eg, Server level — handles s/w for data storage on disk pages, local concurrency control, buffering & caching of disk pages etc

Client level — ~~user~~ handles user interface, data dictionary functions, interaction with PL compilers, global query optimization etc.

Client/server interaction is more tightly coupled and is done internally by DBMS modules.

In this architecture the server has been called a data server because it provides data in disk pages to the client.

Adv :- Simplicity & seamless compatibility with existing systems.

Three tier and n-tier architectures for web applications :-

In this type of architecture an intermediate layer is added b/w the client and the database server. This intermediate layer or middle tier is called the application server or the web server depending on the application

It is goal is to run application programs and storing of business rules to

It also improves database security

Clients contain GUI interfaces & some additional application specific business rules.

Intermediate servers accepts requests from client, processes the request and send database queries and commands to the database server and then takes back the result from the server and presents it to users on clients in GUI format.

fig - Pg 48 (a)

Three tiers — user interface, application rules, data access.

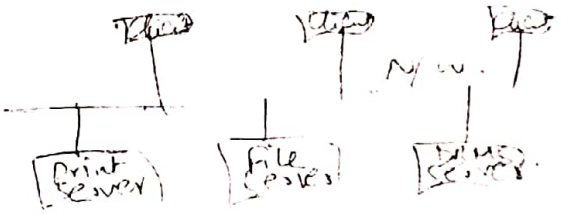
Another three tier architecture :-

fig Pg 48 (b)

The presentation layer displays info to the users and allows data entry.

The business logic layer handles intermediate rules and constraints before data ~~management services~~ is passed up to the user or down to the DBMS.

The bottom layer includes all data management services.



Responsibilities of DBA :-

- Schema definition
- Storage structure and access method definition
- Schema and physical organization modification
- Granting user authority to access the database
- Specifying integrity constraints
- Monitoring performance and responding to changes in requirements.