-

DATA BASE MANAGEMENT SYSTEMS

Unit - I

Introduction: Database and Database Users, Characteristics of the Database Approach, Different people behind DBMS, Implications of Database Approach, Advantages of using DBMS, When not to use a DBMS. Database System Concepts and architecture: Data Models, Schemas, and Instances. DBMS Architecture and Data Independence., Database languages and interfaces. The database system Environment,Classification of DBMS.

Introduction

Database is a collection of related data. Database management system is software designed to assist the maintenance and utilization of large scale collection of data. DBMS came into existence in 1960 by Charles. Integrated data store which is also called as the first general purpose DBMS. Again in 1960 IBM brought IMS-Information management system. In 1970 Edgor Codd at IBM came with new database called RDBMS. In 1980 then came SQL Architecture- Structure Query Language. In 1980 to 1990 there were advances in DBMS e.g. DB2, ORACLE.

Data: Data is raw fact or figures or entity.

Information: The processed data is called information.

Database: A database is a collection of related data.

For example, a university database might contain information about the following:

- Entities such as students, faculty and courses.
- Relationships between entities, such as students' enrollment in courses, faculty teaching courses.

Database Management System:

A Database Management System (DBMS) is a collection of program that enables user to create, maintain and manipulate a database.

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

Characteristics of DBMS

- To incorporate the requirements of the organization, system should be designed for easy maintenance.

- Information systems should allow interactive access to data to obtain new information without writing fresh programs.
- System should be designed to co-relate different data to meet new requirements.
- An independent central repository, which gives information and meaning of available data is required.
- Integrated database will help in understanding the inter-relationships between data stored in different applications.
- The stored data should be made available for access by different users simultaneously.
- Automatic recovery feature has to be provided to overcome the problems with processing system failure.

Different people behind DBMS

These apply to "large" databases, not "personal" databases that are defined, constructed, and used by a single person via, say, Microsoft Access.

There are two categories of people behind DBMS

a) Those who actually use and control the database content, and those who design, develop and maintain database applications (called *—Actors on the Scene*)

b) Those who design and develop the DBMS software and related tools, and the computer systems operators (called *—Workers Behind the Scene*).

a) Actors on the Scene

1. **Database Administrator** (DBA): DBA is a person who is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.
2. **Database Designers**: They are responsible for identifying the data to be

stored and for choosing an appropriate way to organize it. They also define **views** for different categories of users. The final design must be able to support the requirements of all the user sub-groups.

3.End Users: These are persons who access the database for **querying**, **updating**, and **report generation**. They are main reason for database's existence!

Casual end users: use database occasionally, needing different information each time; use query language to specify their requests; typically middle- or high-level managers.

Naive/Parametric end users: Typically the biggest group of users; frequently query/update the database using standard **canned transactions** that have been carefully programmed and tested in advance. Examples:

bank tellers check account balances, post withdrawals/deposits,reservation clerks for airlines, hotels, etc., check availability of seats/rooms and make reservations.

Sophisticated end users: engineers, scientists, business analysts who implement their own applications to meet their complex needs.

Stand-alone users: Use "personal" databases, possibly employing a special-purpose (e.g., financial) software package. Mostly maintain personal databases using ready-to-use packaged applications.

An example is a tax program user that creates its own internal database. Another example is maintaining an address book

4. System Analysts, Application Programmers, Software Engineers:

System Analysts: determine needs of end users, especially naive and parametric users, and develop specifications for canned transactions that meet these needs.

Application Programmers: Implement, test, document, and maintain programs that satisfy the specifications mentioned above.

c) Workers Behind the Scene

1)DBMS system designers/implementors: provide the DBMS software that is at the foundation of all this!

2) **Tool developers**: design and implement software tools facilitating database system design, performance monitoring, creation of graphical user interfaces, prototyping, etc.

3) Operators and maintenance personnel: responsible for the day-to-day operation of the system.

Advantages of a DBMS

Using a DBMS to manage data has many advantages:

Data independence: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.

Efficient data access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

Data integrity and security: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, the DBMS can enforce *access controls* that govern what data is visible to different classes of users.

Data administration: When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the

data to make retrieval efficient.

Concurrent access and crash recovery: A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.

Reduced application development time: Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick development of applications. Such applications are also likely to be more robust than applications developed from scratch because many important tasks are handled by the DBMS instead of being implemented by the application.

Functions of DBMS

• **Data Definition**: The DBMS provides functions to define the structure of the data in the application. These include defining and modifying the record structure, the type and size of fields and the various constraints to be satisfied by the data in each field.

• **Data Manipulation**: Once the data structure is defined, data needs to be inserted, modified or deleted. These functions which perform these operations are part of DBMS. These functions can handle plashud and unplashud data manipulation needs. Plashud queries are those which form part of the application. unplaced queries are ad-hoc queries which performed on a need basis.

• **Data Security & Integrity**: The DBMS contains modules which handle the security and integrity of data in the application.

• **Data Recovery and Concurrency**: Recovery of the data after system failure and concurrent access of records by multiple users is also handled by DBMS.

• **Data Dictionary Maintenance**: Maintaining the data dictionary which contains the data definition of the application is also one of the functions of DBMS.

• **Performance**: Optimizing the performance of the queries is one of the important functions of DBMS.

When not to use a DBMS

a) Main inhibitors (costs) of using a DBMS:

i) High initial investment and possible need for additional hardware.

ii) Overhead for providing generality, security, concurrency control, recovery, and integrity functions.

b) When a DBMS may be unnecessary:

i) If the database and applications are simple, well defined and not expected to change.

ii) If there are stringent real-time requirements that may not be met because of DBMS overhead.

iii)If access to data by multiple users is not required.

c) When no DBMS may be sufficient:

i) If the database system is not able to handle the complexity of data because of modeling limitations

ii) If the database users need special operations not supported by the DBMS.

Role of Database Administrator.

Typically there are three types of users for a DBMS:

1. The END User who uses the application. Ultimately he is the one who actually puts the data into the system into use in business. This user need not know anything about the organization of data in the physical level.

2. The Application Programmer who develops the application programs. He/She has more knowledge about the data and its structure. He/she can manipulate the data using his/her programs. He/she also need not have access and knowledge of the complete data in the system.

3. The Data base Administrator (DBA) who is like the super-user of the system.

The role of DBA is very important and is defined by the following functions.

• *Defining the schema*: The DBA defines the schema which contains the structure of the data in the application. The DBA determines what data needs to be present in the system and how this data has to be presented and organized.

• *Liaising with users*: The DBA needs to interact continuously with the users to understand the data in the system and its use.

• *Defining Security & Integrity checks*: The DBA finds about the access restrictions to be defined and defines security checks accordingly. Data Integrity checks are defined by the DBA.

• *Defining Backup/Recovery Procedures*: The DBA also defines procedures for backup and recovery. Defining backup procedure includes specifying what data is to be backed up, the periodicity of taking backups and also the medium and storage place to backup data.

• *Monitoring performance*: The DBA has to continuously monitor the performance of the queries and take the measures to optimize all the queries in the application.

Database Manager

Database manager is a program module which provides the interface between the low level data stored in the database and the application programs and queries submitted to the system:

– The database manager would translate DML statement into low level file system commands for storing, retrieving, and updating data in the database.

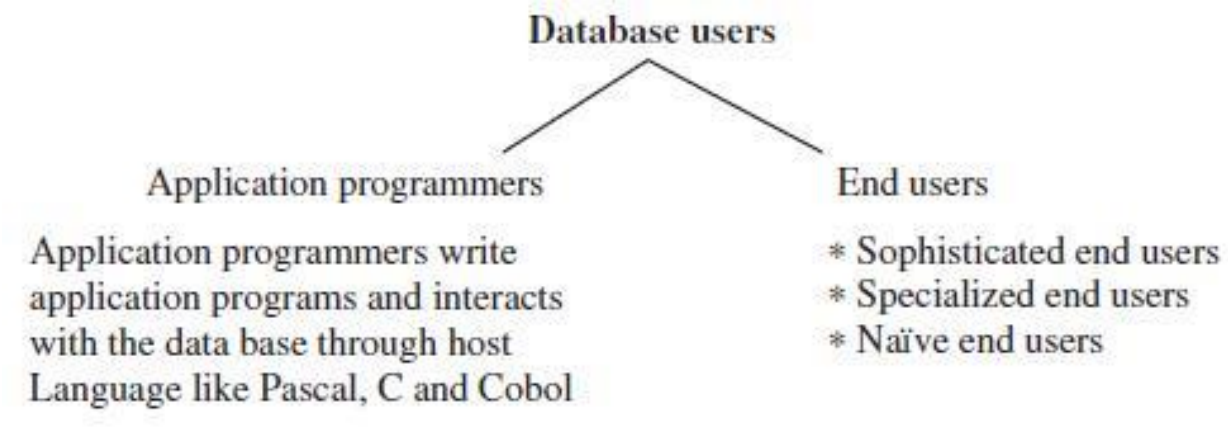
– *Integrity enforcement.* Database manager enforces integrity by checking consistency constraints like the bank balance of customer must be maintained to a minimum of Rs. 1000, etc.

– *Security enforcement.* Unauthorized users are prohibited to view the information stored in the data base.

– *Backup and recovery*. Backup and recovery of database is necessary to ensure that the database must remain consistent despite the fact of failures.

Database Users

Database users are the people who need information from the database to carry out their business responsibility. The database users can be broadly classified into two categories like application programmers and end users.



Sophisticated End Users

Sophisticated end users interact with the system without writing programs. They form requests by writing queries in a database query language. These are submitted to query processor. Analysts who submit queries to explore data in the database fall in this category.

Specialized End Users

Specialized end users write specialized database application that does not fit into data-processing frame work. Application involves knowledge base and expert system, environment modeling system, etc.

Naive End Users

Naive end user interact with the system by using permanent application program Example: Query made by the student, namely number of books borrowed in library database.

System Analysts

System analysts determine the requirements of end user, and develop specification for canned transaction that meets this requirement.

Canned Transaction

Readymade programs through which naive end users interact with the database is called canned transaction.

Database System Concepts and Architecture

Data Models, Schemas, and Instances

One fundamental characteristic of the database approach is that it provides some level of *data abstraction* by hiding details of data storage that are irrelevant to database users.

A **data model** is a collection of concepts that can be used to describe the conceptual/logical structure of a database.

The structure of a database means that holds the data's data types, relationships, and constraints.

According to C.J. Date (one of the leading database experts), a **data model** is an abstract, self-contained, logical definition of the objects, operators, and so forth, that together constitute the *abstract machine* with which users interact. The objects allow us to model the *structure* of data; the operators allow us to model its *behavior*.

Types of Data Models

1. High Level- Conceptual data model.
2. Low Level – Physical data model.
3. Relational or Representational
4. Object-oriented Data Models:
5. Object-Relational Models:

1. High Level-conceptual data model: User level data model is the high level

or conceptual model. This provides concepts that are close to the way that many users perceive data.

2 .Low level-Physical data model: provides concepts that describe the details of how data is stored in the computer model. Low level data model is only for Computer specialists not for end-user.

3. Representation data model: It is between High level & Low level data model Which provides concepts that may be understood by end-user but that are not too far removed from the way data is organized by within the computer.

The most common data models are

1. Relational Model

The Relational Model uses a collection of tables both data and the relationship among those data. Each table has multiple columns and each column has a unique name.

Relational database comprising of two tables.

Student_Table

Regno	Name	Gender	DOB	Course
101	Varsh G Kalyan	F	20-Sep-1985	BCA
102	Mohith G Kalyan	M	20-Aug-1980	BBM
103	Nisarga	F	15-Jul-1983	BCom
104	Eenchara	F	04-Dec-1985	BCA

Result_Table

Regno	Sem	Result
101	II	First Class
102	II	First Class
103	II	Passes
104	II	Second Class

Advantages

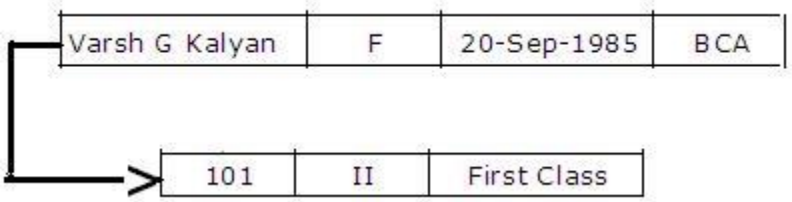
1. The main advantage of this model is its ability to represent data in a simplified format.

2. The process of manipulating record is simplified with the use of certain key attributes used to retrieve data.

3. Representation of different types of relationship is possible with this model.

2. Network Model

The data in the network model are represented by collection of records and relationships among data are represented by links, which can be viewed as pointers.



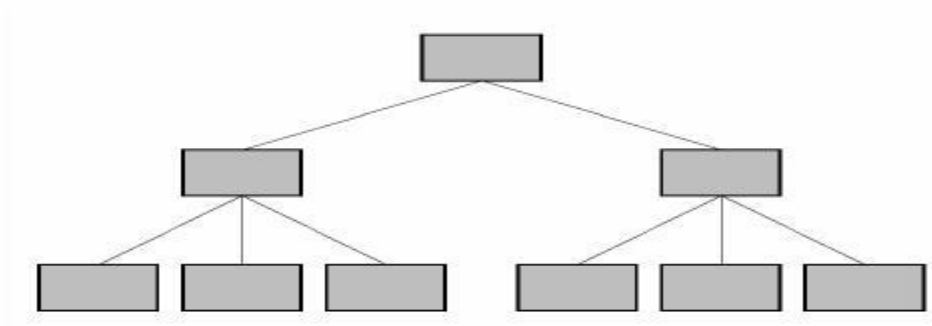
The records in the database are organized as collection of arbitrary groups.

Advantages:

1. Representation of relationship between entities is implemented using pointers which allows the representation of arbitrary relationship
2. Unlike the hierarchical model it is easy.
3. Data manipulation can be done easily with this model.

3. Hierarchical Model

A hierarchical data model is a data model which the data is organized into a tree like structure. The structure allows repeating information using parent/child relationships: each parent can have many children but each child has one parent. All attributes of a specific record are listed under an entity type.



Advantages:

1. The representation of records is done using an ordered tree, which is natural method of implementation of one–to-many relationships.
2. Proper ordering of the tree results in easier and faster retrieval of records.
3. Allows the use of virtual records. This result in a stable database especially when modification of the data base is made.

Data Instances and Schemas

Database Instances

Database change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of the database.

Database Schema

The overall design of the database is called the database schema. A schema is a collection of named objects. Schemas provide a logical classification of objects in the database. A schema can contain tables, views, triggers, functions, packages, and other objects.

DBMS Architecture

A commonly used view of data approach is the three-level architecture suggested by the ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee). ANSI/SPARC proposed an architectural framework for databases.

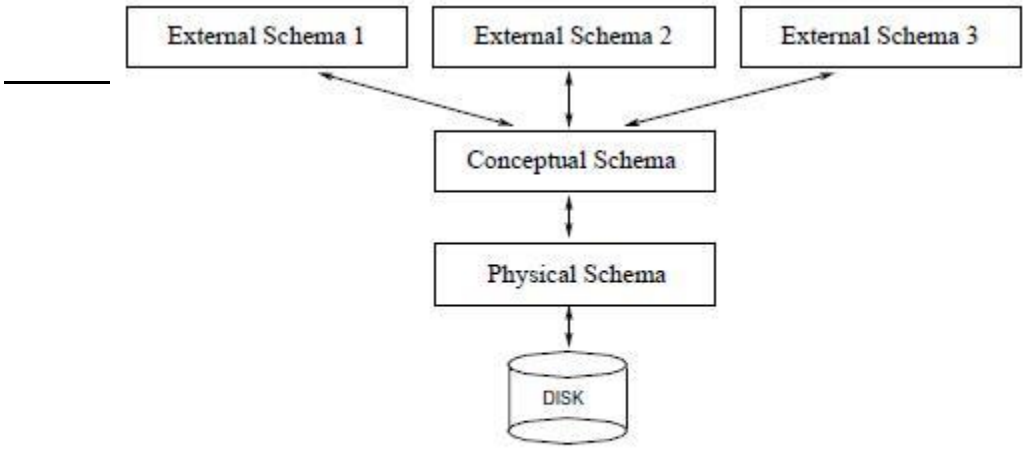


Fig: Three level DBMS architecture / Three schema architecture

The three levels of the architecture are three different views of the data:

External Schema - individual user view

Conceptual Schema- Logical or community user view

Physical Schema -Internal or storage view

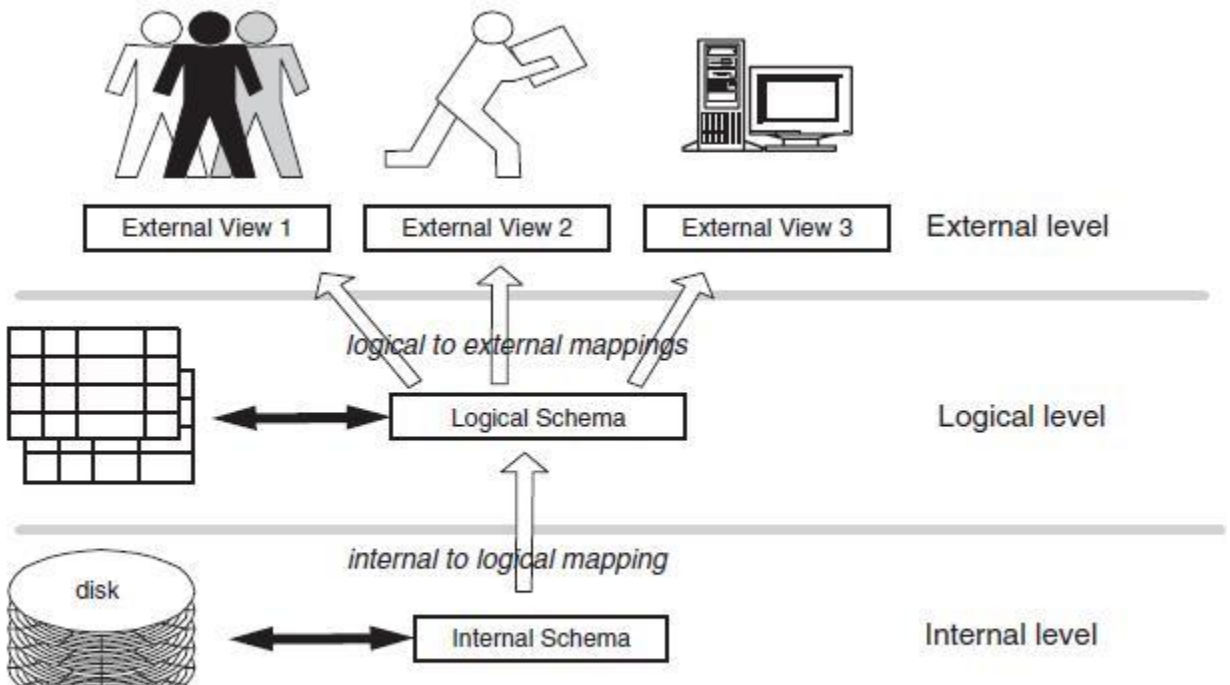
The three level database architecture allows a clear separation of the information meaning (conceptual view) from the external data representation and from the physical data structure layout. A database system that is able to separate the three different views of data is likely to be flexible and adaptable.

The **External Schema** is the view that the individual user of the database has. This view is often a restricted view of the database and the same database may provide a number of different views for different classes of users.

The **Conceptual schema** (sometimes called the **logical schema**) describes the stored data in terms of the data model of the DBMS. In a relational DBMS, the conceptual schema describes all relations that are stored in the database.

It hides physical storage details, concentrating upon describing entities, data types, relationships, user operations, and constraints.

The **physical schema** specifies additional storage details. Essentially, the physical schema summarizes how the relations described in the conceptual schema are actually stored on secondary storage devices such as disks and tapes. It tells us what data is stored in the database and how.



Data Independence

Data independence can be defined as the capacity to change the schema at one level without changing the schema at next higher level.

It also means the internal structure of database should be unaffected by changes to physical aspects of storage. Because of data independence, the Database administrator can change the database storage structures without affecting the users view.

The different levels of data abstraction are:

1. Physical data independence
2. Logical data independence

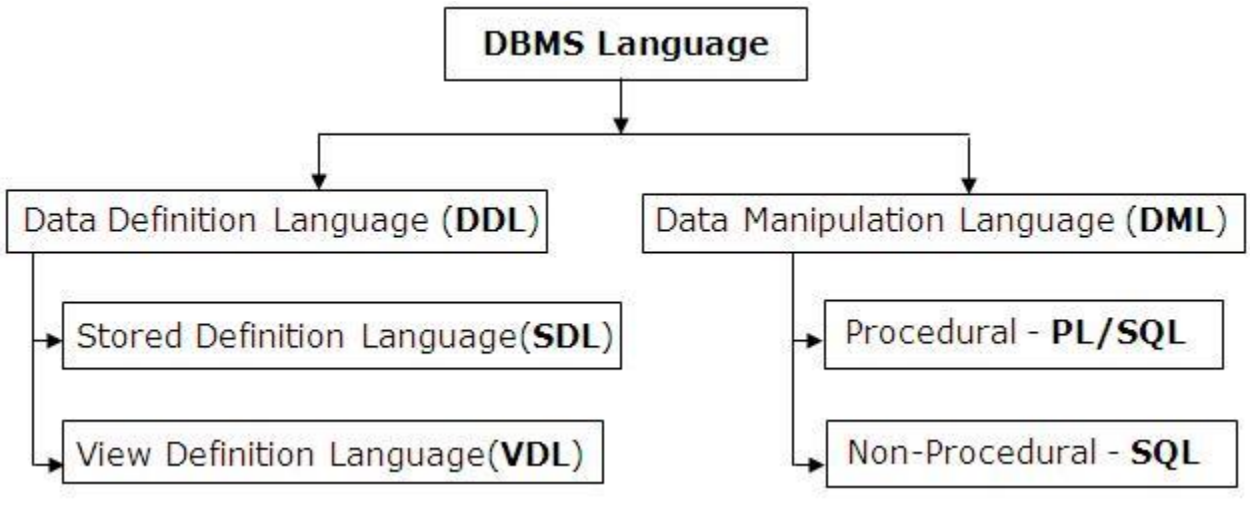
1. Physical data independence is the capacity to change the internal schema without changing the conceptual schema(logical).

2. Logical data independence is the capacity to change the conceptual schema without having to change the external schema(physical).

Database languages and interfaces

A database system provides a **data definition language** to specify the database schema and a **data manipulation language** to express database queries and updates.

In practice, the data definition and data manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.



Data Definition Language

Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed

For instance, the following statement in the SQL language is used to create the *account* table:

create table *account*

(*accountnumber number*(10),
balance number(8));

The storage definition language (**SDL**), is used to specify the internal schema. The mappings between the two schemas may be specified in either one of these languages.

The view definition language (**VDL**), to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.

In addition, it updates a special set of tables called the **data dictionary** or **data directory**.

A data dictionary contains **metadata**—that is, data about data. The schema of a table is an example of metadata. A database system consults the data dictionary before reading or modifying actual data.

Data Manipulation Language

Data manipulation is

- The retrieval of information stored in the database

- The insertion of new information into the database
- The deletion of information from the database
- The modification of information stored in the database

Data Manipulation Language (DML) statements are used for managing data within schema objects. Some examples:

- SELECT - retrieve data from the a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - deletes all records from a table

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model.

There are basically two types:

• **Procedural DMLs** require a user to specify *what* data are needed and *how* to get those data, The DML component of the PL/SQL language is procedural.

•**Nonprocedural DMLs** require a user to specify *what* data are needed *without* specifying how to get those data. The DML component of the SQL language is nonprocedural.

A **query** is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a **query language**.

The query in the SQL language finds the name of the customer whose customer-id is 192:

```
select
customername from
customer
where customerid = 192;
```

The query specifies that those rows *from* the table *customer where* the *customerid* is 192 must be retrieved.

DBMS Interfaces

User-friendly interfaces provided by a DBMS may include the following.

Menu Based Interfaces for Web Clients or Browsing. These interfaces present the user with lists of options, called menus, that lead the user through the formulation of a request. Menus do away with the need to memorize the specific commands and syntax of a query language; rather, the query is composed step by step by picking options from a menu that is displayed by the system. Pull-down menus are a very popular technique in Web-based user interfaces. They are also often used in browsing interfaces, which allow a user to look through the contents of a database in an exploratory and unstructured manner.

Forms Based Interfaces. A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert new data, or they fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries. Forms are usually designed and programmed for naive users as interfaces to canned transactions.

Graphical User Interfaces. A graphical interface (GUI) typically displays a schema to the user in diagrammatic form. The user can then specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms. Most GUIs use a pointing device, such as a mouse, to pick certain parts of the displayed schema diagram.

Natural Language Interfaces. These interfaces accept requests written in English or some other language and attempt to "understand" them. A natural language interface usually has its own "schema," which is similar to the database conceptual

.

schema, as well as a dictionary of important words. 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.

Interfaces for Parametric Users. Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly. Systems analysts and programmers design implement a special interface for naive users. Usually, a small set of abbreviated commands is included, with the goal of

minimizing the number of keystrokes required for each request.

Interfaces for the DBA. Most database systems contain privileged commands that can be used only by the DBA's staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

The Database System Environment

A DBMS is a complex software system. The database and the DBMS catalog are usually stored on disk. Access to the disk is controlled primarily by the **operating system** (**OS**), which schedules disk read/write. Many DBMSs have their own **buffer management** module to schedule disk read/write, because this has a considerable effect on performance. Reducing disk read/write improves performance considerably. A higher-level **stored data manager** module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

the top part of Figure, It shows interfaces for the DBA staff, casual users who work with interactive interfaces to formulate queries, application programmers who create programs using some host programming languages, and parametric users who do data entry work by supplying parameters to predefined transactions. The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.

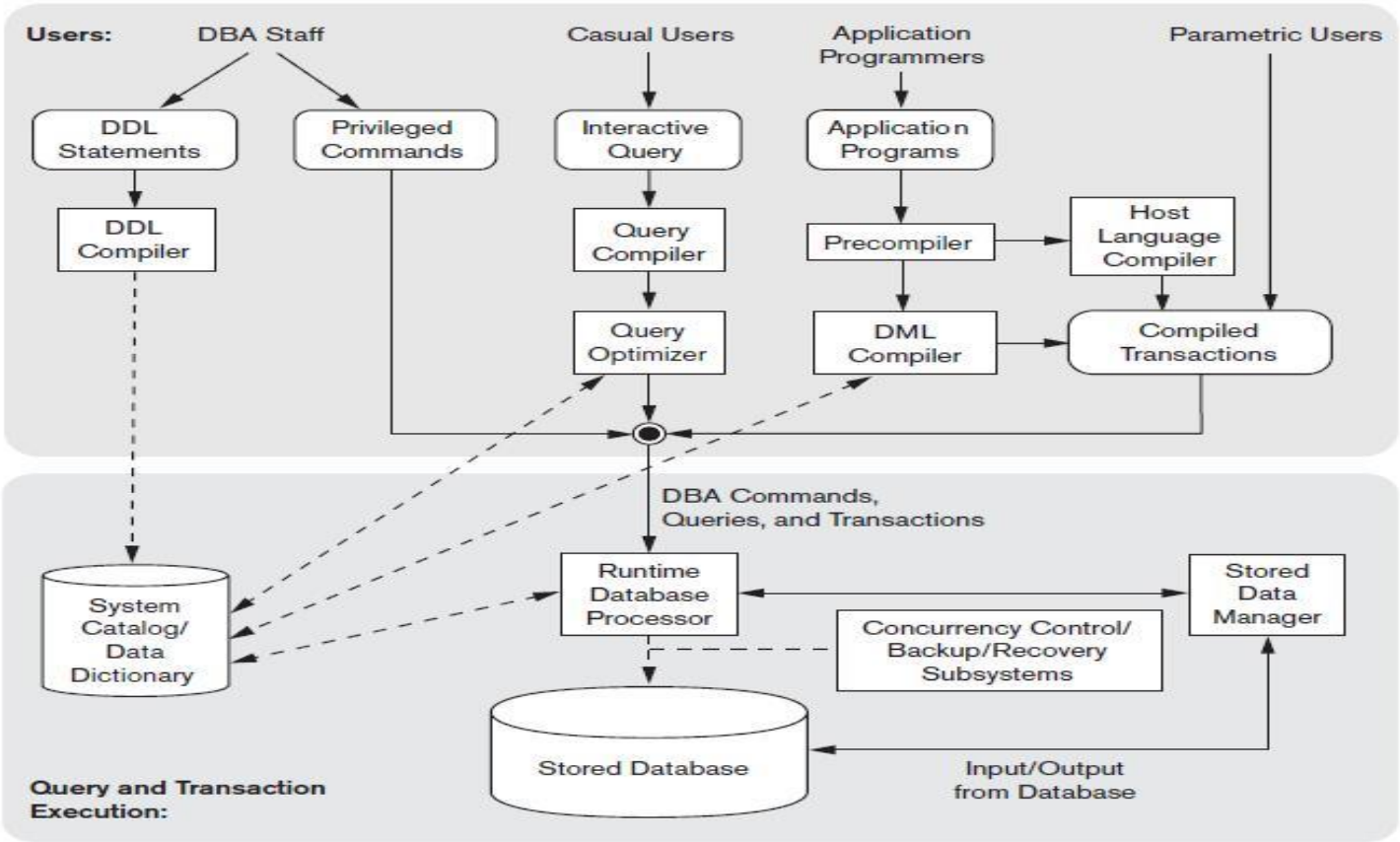
The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog. The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints.

Casual users and persons with occasional need for information from the database interact using some form of interface, which we call the **interactive query**

interface. These queries are parsed and validated for correctness of the query syntax, the names of files and a **query compiler** that compiles them into an internal form. This internal query is subjected to query optimization, the **query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution.

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 for the DML commands and the rest of the program are linked, forming a canned transaction whose executable code includes calls to the runtime database processor.

It is now common to have the **client program** that accesses the DBMS running on a separate computer from the computer on which the database resides. The former is called the **client computer** running a DBMS client software and the latter is called the **database server**. In some cases, the client accesses a middle computer, called the **application server**, which in turn accesses the database server.



Database System Utilities

In addition to possessing the software modules just described, most DBMSs have **database utilities** that help the DBA manage the database system. Common utilities have the following types of functions:

Loading. A loading utility is used to load existing data files—such as text files or sequential files—into the database. Usually, the current (source) format of the data file and the desired (target) database file structure are specified to the utility, which then automatically reformats the data and stores it in the database.

Backup. A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium. The backup copy can be used to restore the database in case of disk failure.

Database storage reorganization. This utility can be used to reorganize a set of database files into different file organizations, and create new access paths to improve performance.

Performance monitoring. Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files or whether to add or drop indexes to improve performance.

Other utilities may be available for sorting files, handling data compression, monitoring access by users, interfacing with the network, and performing other functions.

Centralized and Client/Server Architectures for DBMSs

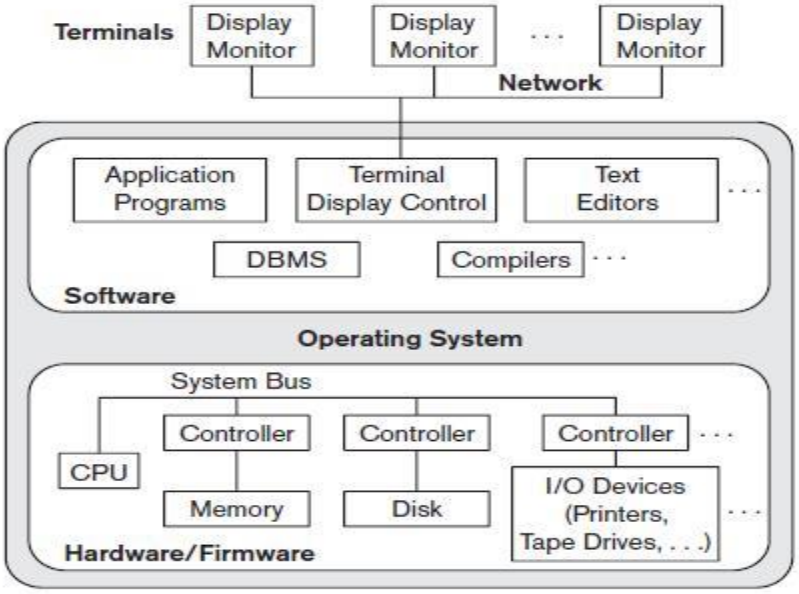
Centralized DBMSs Architecture

Architectures for DBMSs have followed trends similar to those for general computer system architectures. Earlier 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. The reason was that most users accessed such systems via computer terminals that

did not have processing power and only provided display capabilities. Therefore, all processing was performed remotely on the computer system, and only display information and controls were sent from the computer to the display terminals, which were connected to the central computer via various types of communications networks.

As prices of hardware declined, most users replaced their terminals with PCs and workstations. At first, database systems used these computers similarly to how they had used display terminals, so that the DBMS itself was still a **centralized** DBMS in which all the DBMS functionality,application program execution, and user interface processing were carried out on one machine.

The Figure illustrates the physical components in a centralized architecture. Gradually, DBMS systems started to exploit the available processing power at the user side, which led to client/server DBMS architectures.



Client/Server Architectures for DBMSs

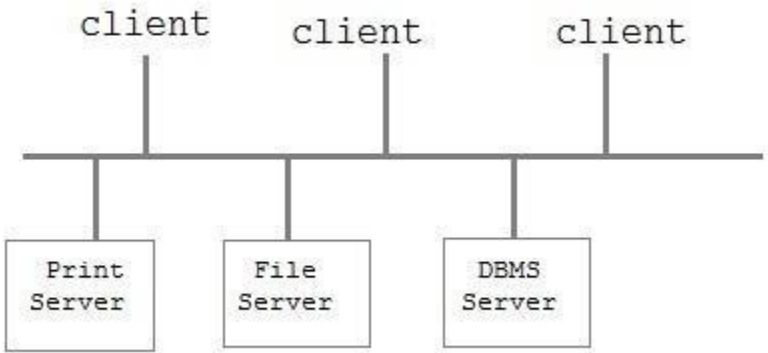
Database architecture essentially describes the location of all the pieces of information that make up the database application.

Database architecture is logically divided into two types.

a) Logical two-tier Client / Server architecture

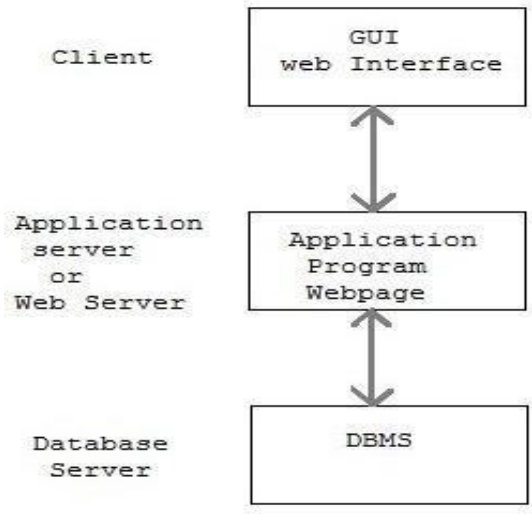
b) Logical three-tier Client / Server architecture

Two-tier Client / Server Architecture



Two-tier Client / Server architecture is used for User Interface program and Application Programs that runs on client side. An interface called ODBC(Open Database Connectivity) provides an API that allow client side program to call the dbms. Most DBMS vendors provide ODBC drivers. A client program may connect to several DBMS's. In this architecture some variation of client is also possible for example in some DBMS's more functionality is transferred to the client including data dictionary, optimization etc. Such clients are called Data server.

Three-tier Client / Server Architecture



Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called Application server or Web Server stores the web connectivity software and the business logic(constraints) part of application used to access the right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client.

Differentiate between centralized and distributed data base

Centralized	Distributed
Database is maintained at one site	Database is maintained at a number of different sites
If centralized system fails, entire system is halted.	If one system fails, system continues work with other sites
Less reliable	More reliable

Classification of Database Management Systems

Several criteria are normally used to classify DBMSs. The first is the **data model** on which the DBMS is based. The main data model used in many current commercial DBMSs is the **relational data model**. The **object data model** has been implemented in some commercial systems but has not had widespread use. Many legacy applications still run on database systems based on the **hierarchical** and **network data models**.

The second criterion used to classify DBMSs is the **number of users** supported by the system. **Single-user systems** support only one user at a time and are mostly used with PCs. **Multiuser systems**, which include the majority of DBMSs, support concurrent multiple users.

The third criterion is the **number of sites** over which the database is distributed. A DBMS is **centralized** if the data is stored at a single computer site. A centralized DBMS can support multiple users, but the DBMS and the database reside totally at a single computer site. A **distributed** DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network. **Homogeneous** DDBMSs use the same DBMS software at all

the sites, whereas **heterogeneous** DDBMSs can use different DBMS software at each site.

The fourth criterion is cost. It is difficult to propose a classification of DBMSs based on cost. Today we have open source (free) DBMS products like MySQL and PostgreSQL that are supported by third-party vendors with additional services. The main RDBMS products are available as free examination 30-day copy versions as well as personal versions,

We can also classify a DBMS on the basis of the **types of access path** options for storing files. One well-known family of DBMSs is based on inverted file structures. Finally, a DBMS can be **general purpose** or **special purpose**. When performance is a primary consideration, a special-purpose DBMS can be designed and built for a specific application; such a system cannot be used for other applications without major changes. Many airline reservations and telephone directory systems developed in the past are special-purpose DBMSs. These fall into the category of **online transaction processing** (**OLTP**) systems, which must support a large number of concurrent transactions without imposing excessive delays.