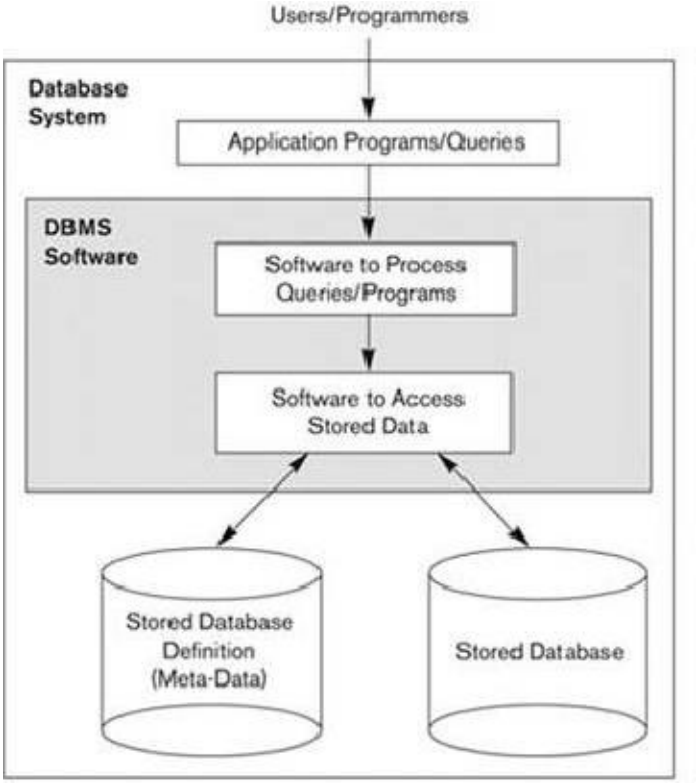
1. Introduction to Database System

Database: A database is **an organized collection of structured information, or data, typically stored electronically in a computer system**. A database is usually controlled by a database management system (DBMS).

Some examples of popular database software or DBMSs include **MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database, and dBASE**.

A database is **an organized collection of data**. More specifically, a database is an electronic system that allows data to be easily accessed, manipulated and updated. In other words, a database is used by an organization as an electronic way to store, manage and retrieve information.

A database environment is **a collective system of components that comprise and regulates the group of data, management, and use of data**, which consist of software, hardware, people, techniques of handling database, and the data also.



A simplified database system environment

Example Database

Consider the following university database, it consists of following tables:

Tables:

STUDENT(Student#, LastName, FirstName, Address, City, State, Zip, Enroll_Date, Undergrad?)

COURSE(Course#, Title, CrHour, InstName)

INSTRUCTOR(InstName, InstOffice, Rank)

Take(Student#, Course#, Grade)

STUDENT

Student#	Last name	First name	Address	City	State	Zip	Enroll_Date	UnderGrad?
Text 10	Text 50	Text 50	Text 50	Text 50	Text 2	Text 10	Date/Time	Yes/No
1y Key	2y Index				> Format	Input Mask	Short Date Format	
							System Date Default	

COURSE

Course#	Title	CrHour	InstName
Text 10	Text 60	Byte	Lookup Wizard (INSTRUCTOR)
1y Key		Valid values: 1-4 hours	

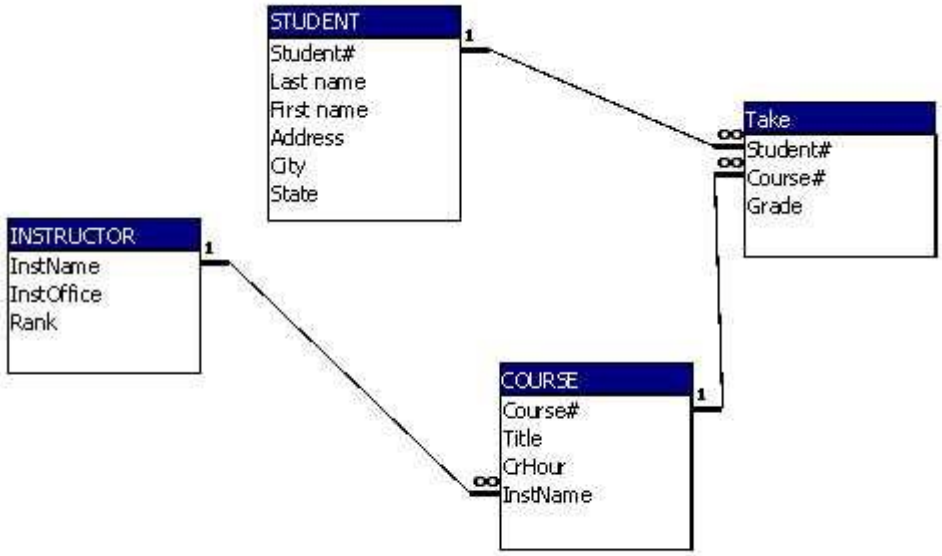
INSTRUCTOR

InstName	InstOffice	Rank
Text 50	Text 10	Text 20
1y Key		

Take

Student#	Course#	Grade
Text 10	Text 10	Text 2
1y Key	1y Key	

2. Create the following relationships



1.1 Characteristics of the Database Approach

1) Self-Describing Nature of a Database System

- One of the most fundamental characteristics of the database approach is that the database system contains not only the database itself but also an entire definition or description of the database structure and constraints also known as metadata of the database.
- This definition is stored within the DBMS catalog, which contains information like the structure of every file, the sort and storage format of every data item, and various constraints/rules on the information.
- The knowledge stored within the catalog is named meta-data, and it describes the structure of the first database the catalog is employed by the DBMS software and also by database users such as database administrators who required to know the information about the database structure.
- A general-purpose DBMS software package is not written for a selected database application. Therefore, it must ask the catalog to understand the structure of the files during a specific database, like the sort and format of knowledge it will access.
- The DBMS software must work equally well with any number of database applications, For example, a university database, a banking database, or a corporation database as long as

because the database definition is stored within the catalog In traditional file processing, data definition is usually a part of the files. File processing software can access only specific databases, Database Management software can access various databases by extracting the database definitions or schemas from the catalog and using these definitions.

2) Isolation between Programs and Data, and Data Abstraction :

- In a traditional file processing system, the structure of database knowledge files is embedded within the application programs, so any changes to the structure of a file may require changing all programs that access that file.
- Against this, DBMS access programs don't require such changes in most cases, so independence is achieved between them.
- The structure of knowledge files is stored within the DBMS catalog separately from the programs that access them. We call this property program-data independence.
- The characteristic that allows program-data independence and program-operation independence is known as data abstraction.
- A DBMS provides users with a conceptual representation of knowledge that doesn't include much of the small print of how the information is stored or how the operations are implemented internally. Informally, a knowledge model may be a sort of data abstraction that won't provide this conceptual representation.
- The information model uses logical concepts, like objects, their properties, and their relationships between them, which will be easier for many users to know than memory concepts or storage concepts. Hence, the information model hides storage and implementation details that are not of interest to most database users, so unnecessary complications are hidden from them.

3) **Support for Multiple Views of the Data :**

- A database sometimes has many users, each of whom may require a special perspective or view of the database.
- A view could also be a subset of the database, or it's going to contain virtual data that is derived from the database files but isn't explicitly stored.
- Some users might not get to remember whether the information they ask for is stored or derived.
- A multi-user DBMS whose users have a spread of distinct applications must provide facilities for outlining multiple views. This provides many benefits for large databases such as the Aadhaar database.

4) **Sharing of knowledge and Multi-user Transaction Processing :**

- A multi-user DBMS, as its name implies, must allow multiple users to access the database at an equivalent time or concurrently.
- This is often essential if data for multiple applications is to be integrated and maintained during a single database such as the latest feature of WhatsApp integration with Facebook.
- The DBMS must implement concurrency control in the software to make sure that several users trying to update equivalent data do so in a controlled manner in order that the results of the updates are correct.
- For instance, when several reservation agents attempt to assign a seat on an airline flight, the DBMS should make sure that each seat is often accessed by just one user agent at a single time for an assignment to a passenger.
- These sorts of applications are generally called online transaction processing (OLTP) applications. A fundamental role of multi-user DBMS software is to make sure that concurrent transactions operate correctly and efficiently with no inconsistency.
- The concept of a transaction has become central to several database applications. A transaction is an executing program or process that has one or more database accesses, like reading or updating of database records or inserting new records.
- The isolation property ensures that every transaction appears to execute in isolation from other transactions, many transactions could also be executed concurrently without affecting each other.

1.2 Actors on Scene

For a small personal database, such as the list of addresses discussed in Previous Section , one person typically defines, constructs, and manipulates the database, and there is no sharing. However, in large organizations, many people are involved in the design, use, and maintenance of a large database with hundreds of users. In this section we identify the people whose jobs involve the day-to-day use of a large database; we call them the *actors on the scene*. In Section 1.5 we consider people who may be called *workers behind the scene*—those who work to maintain the database system environment but who are not actively interested in the database contents as part of their daily job.

1. Database Administrators

In any organization where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources. In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The DBA is responsible for authorizing access

to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed. The DBA is accountable for problems such as secu-rity breaches and poor system response time. In large organizations, the DBA is assisted by a staff that carries out these functions.

2. Database Designers

Database designers are responsible for identifying the data to be stored in the data-base and for choosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and popu-lated with data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to cre-ate a design that meets these requirements. In many cases, the designers are on the staff of the DBA and may be assigned other staff responsibilities after the database design is completed. Database designers typically interact with each potential group of users and develop views of the database that meet the data and processing requirements of these groups. Each view is then analyzed and *integrated* with the views of other user groups. The final database design must be capable of supporting the requirements of all user groups.

3. End Users

End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users

Casual end users occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.

- **Naive** or **parametric end users** make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called **canned transactions—**that have been carefully programmed and tested. The tasks that such users perform are varied:

Bank tellers check account balances and post withdrawals and deposits.

Reservation agents for airlines, hotels, and car rental companies check availability for a given request and make reservations.

Employees at receiving stations for shipping companies enter package identifications via bar codes and descriptive information through buttons to update a central database of received and in-transit packages.

- **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.
- **Standalone users** maintain personal databases by using ready-made pro-gram packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes.

A typical DBMS provides multiple facilities to access a database. Naive end users need to learn very little about the facilities provided by the DBMS; they simply have to understand the user interfaces of the standard transactions designed and implemented for their use. Casual users learn only a few facilities that they may use repeatedly. Sophisticated users try to learn most of the DBMS facilities in order to achieve their complex requirements. Standalone users typically become very proficient in using a specific software package.

4. System Analysts and Application Programmers (Software Engineers)

System analysts determine the requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions that meet these requirements. **Application programmers** implement these specifi-cations as programs; then they test, debug, document, and maintain these canned transactions. Such analysts and programmers—commonly referred to as **software developers** or **software engineers**—should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.

1.3Workers behind the scene

In addition to those who design, use, and administer a database, others are associated with the design, development, and operation of the DBMS *software and system environment.* These persons are typically not interested in the database content itself. We call them the *workers behind the scene*, and they include the following categories:

- **DBMS system designers and implementers** design and implement the DBMS modules and interfaces as a software package. A DBMS is a very complex software system that consists of many components or **modules**, including modules for implementing the catalog, query language processing, interface processing, accessing and buffering data, controlling concurrency, and handling data recovery and security. The DBMS must interface with other system software such as the operating system and compilers for various programming languages.

- **Tool developers** design and implement **tools**—the software packages that facilitate database modeling and design, database system design, and improved performance. Tools are optional packages that are often purchased separately. They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation. In many cases, independent software vendors develop and market these tools.
- **Operators and maintenance personnel** (system administration personnel) are responsible for the actual running and maintenance of the hardware and software environment for the database system.

Although these categories of workers behind the scene are instrumental in making the database system available to end users, they typically do not use the database contents for their own purposes.

1.4 Advantages of Database approach

- **Reducing Data Redundancy**

The file based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. Because of this, there were sometimes multiple copies of the same file which lead to data redundancy.

This is prevented in a database as there is a single database and any change in it is reflected immediately. Because of this, there is no chance of encountering duplicate data.

- **Sharing of Data**

In a database, the users of the database can share the data among themselves. There are various levels of authorization to access the data, and consequently the data can only be shared based on the correct authorization protocols being followed.

Many remote users can also access the database simultaneously and share the data between themselves.

- **Data Integrity**

Data integrity means that the data is accurate and consistent in the database. Data Integrity is very important as there are multiple databases in a DBMS. All of these databases contain data that is visible to multiple users. So it is necessary to ensure that the data is correct and consistent in all the databases and for all the users.

- **Data Security**

Data Security is vital concept in a database. Only authorised users should be allowed to access the database and their identity should be authenticated using a username and password. Unauthorised users should not be allowed to access the database under any circumstances as it violates the integrity constraints.

- **Privacy**

The privacy rule in a database means only the authorized users can access a database according to its privacy constraints. There are levels of database access and a user can only view the data he is allowed to. For example - In social networking sites, access constraints are different for different accounts a user may want to access.

- **Backup and Recovery**

Database Management System automatically takes care of backup and recovery. The users don't need to backup data periodically because this is taken care of by the DBMS. Moreover, it also restores the database after a crash or system failure to its previous condition.

- **Data Consistency**

Data consistency is ensured in a database because there is no data redundancy. All data appears consistently across the database and the data is same for all the users viewing the database. Moreover, any changes made to the database are immediately reflected to all the users and there is no data inconsistency.

1.5 A Brief History of Database

1. **Early Database Applications**:

- The Hierarchical and Network Models were introduced in mid 1960s and dominated during theseventies.

- A bulk of the worldwide database processing still occurs using these models.

2. **Relational Model based Systems**:

- Relational model was originally introduced in 1970, was heavily researched and experimentedwith in IBM Research and several universities.

3. Object-oriented and emerging applications:

- Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980sand early 1990s to cater to the need of complex data processing in CAD and other applications.
- Their use has not taken off much.
- Many relational DBMSs have incorporated object database concepts, leading to a new categorycalled object-relational DBMSs (ORDBMSs)

4. Extended relational systems add further capabilities (e.g. for multimedia data, XML, and otherdata types)

- Relational DBMS Products emerged in the 1980s
- Data on the Web and E-commerce Applications:
- Web contains data in HTML (Hypertext markup language) with links among pages.
- This has given rise to a new set of applications and E-commerce is using new standards likeXML (eXtended Markup Language).
- Script programming languages such as PHP and JavaScript allow generation of dynamic Webpages that are partially generated from a database
- New functionality is being added to DBMSs in the following areas:
- Scientific Applications
- XML (eXtensible Markup Language)
- Image Storage and Management
- Audio and Video data management
- Data Warehousing and Data Mining
- Spatial data management
- Time Series and Historical Data Management

The above gives rise to new research and development in incorporating new data types,

- Complex data structures, new operations and storage and indexing schemes in database systems.
- Also allow database updates through Web pages

When not to use a DBMS

In spite of the advantages of using a DBMS, there are a few situations in which a DBMS may involve unnecessary overhead costs that would not be incurred in traditional file processing. The overhead costs of using a DBMS are due to the following:

• High initial investment in hardware, software, and training

• The generality that a DBMS provides for defining and processing data

• Overhead for providing security, concurrency control, recovery, and integrity functions

Therefore, it may be more desirable to use regular files under the following circumstances:

• Simple, well-defined database applications that are not expected to change at all

• Stringent, real-time requirements for some application programs that may not be met becauseof DBMS overhead

• Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit

• No multiple-user access to data

1.6 Data Models, schemas and Instances

- The data which is stored in the database at a particular moment of time is called an instance of the database.
- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
- A database schema is designed by the database designers to help programmers whose software will interact with the database. The process of database creation is called data modeling.

A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram. For example, the given figure neither shows the data type of each data item nor the relationship among various files.

In the database, actual data changes quite frequently. For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database.

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

1.7 Data Independence

- Data independence can be explained using the three-schema architecture.
- Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

There are two types of data independence:

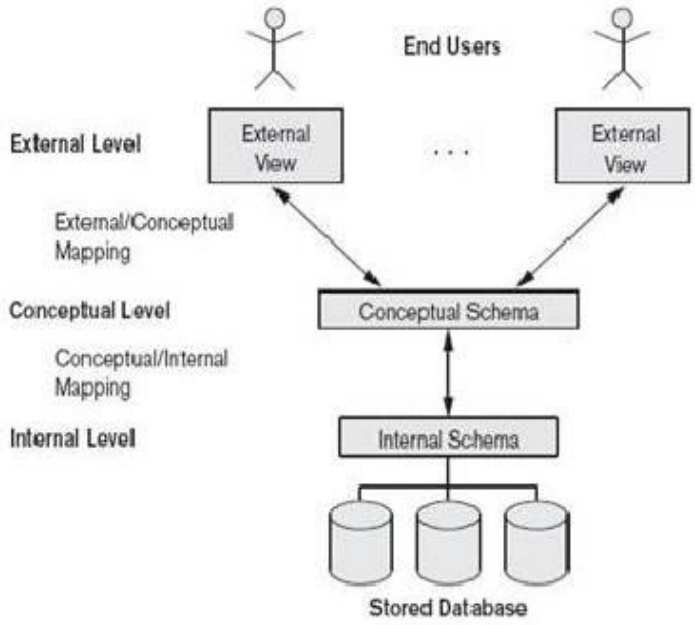
1. Logical Data Independence

- Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual view.
- If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
- Logical data independence occurs at the user interface level.

2. Physical Data Independence

- Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
- If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
- Physical data independence is used to separate conceptual levels from the internal levels.
- Physical data independence occurs at the logical interface level.

Three- Schema Architecture



The Three-Schema Architecture

The goal of the three-schema architecture, illustrated in Figure 1.3 is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

1. The **internal level** has an **internal schema**, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The **conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. A representational data model is used to describe the conceptual schema when a database system is implemented.

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 andhides the rest of the database from that user group. An external schema is described in a high- level data model.

The stored data *actually* exists is at the physical level only. In a DBMS based on the three- schema architecture, each user group refers to its own external schema. Hence, 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. Ifthe request is 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**

1.8 Database Language

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.

Types of Database Language

1. Data Definition Language

- **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

2. Data Manipulation Language

DML stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

1.9 DBMS INTERFACES

A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself.

User-friendly interfaces provide by DBMS may include the following:

1. **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 formation of a request. Basic advantage of using menus is that they removes the tension of remembering specific commands and syntax of any query language, rather than query is basically composed step by step by collecting or picking options from a menu that is basically shown by the system. Pull-down menus are a very popular technique in *Web based interfaces*. They are also often used in *browsing interface* which allow a user to look through the contents of a database in an exploratory and unstructured manner.
2. **Forms-Based Interfaces –**
   A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert a new data, or they can fill out only certain entries, in which case the DBMS will redeem same type of data for other remaining entries. This type of forms are usually designed or created and programmed for the users that have no expertise in operating system. Many DBMSs have *forms specification languages* which are special languages that help specify such forms.
   Example: SQL* Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.b>
3. **Graphical User Interface –**
   A GUI typically displays a schema to the user in diagrammatic form.The user then can specify a query by manipulating the diagram. In many cases, GUI's utilize both menus and forms. Most GUIs use a pointing device such as mouse, to pick certain part of the displayed schema diagram.

4. **Natural language Interfaces –**

These interfaces accept request written in English or some other language and attempt to understand them. A Natural language interface 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 a dictionary to interpret the request. If the interpretation is successful, the interface generates a high-level query corresponding to the natural language and submits it to the DBMS for processing; otherwise a dialogue is started with the user to clarify any provided condition or request. The main disadvantage with this is that the capabilities of this type of interfaces are not that much advance.

5. Speech Input and Output –

There is an limited use of speech say it for a query or an answer to a question or being a result of a request it is becoming commonplace Applications with limited vocabularies such as inquiries for telephone directory, flight arrival/departure, and bank account information are allowed speech for input and output to enable ordinary folks to access this information.

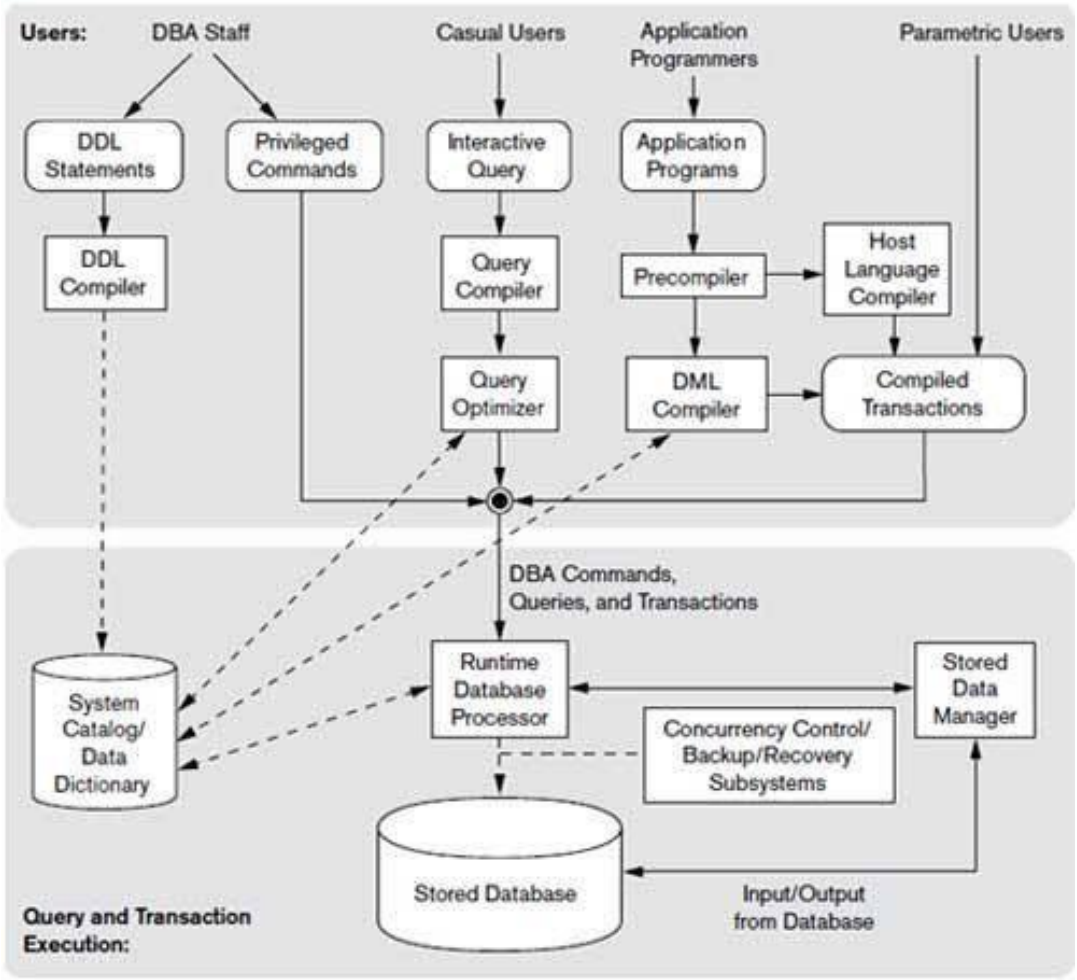
The Speech input is detected using a predefined word and used to set up the parameters that are supplied to the queries. For output, a similar conversion from text or numbers into speech takes place.

6. Interfaces for DBA –

Most database system 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, reorganizing the storage structures of databases.

1.10 The Database System Environment

1.10.1DBMS Components Modules



DBMS Components and their interactions

Figure illustrates the typical DBMS components in a simplified form. The figure is divided into two parts.

- 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.

DBMS Components modules are:

1. Stored data manager
2. DDL compiler
3. Interactive query interface
   - Query compiler

- Query optimizer
- Precompiler

4. Runtime database processor

- System catalog
- Concurrency control system
- Backup and recovery system

Stored data manager:

- Storage manager is a program module that provides the interface between the low-level datastored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for efficient storing, retrieving and updating of data.
- The dotted lines and circle shows access that are under the control of this stored data
- manager.

DDL Compiler:

- The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.

Interactive Query Interface:

- This interface may be used to generate the interactive query automatically. These queries are parsed and validated for correctness of the query syntax by a **query compiler** that compiles theminto 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.

Pre-complier:

- The **pre-compiler** 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 acanned transaction whose executable code includes calls to the runtime database processor

Runtime database processor:

- Runtime Database Processor executes
- The privileged commands,
- The executable query plans, and
- The canned transactions with runtime parameters.

Concurrency Control System:

Concurrency control (CC) is a process to ensure that data is updated correctly and appropriately when multiple transactions are concurrently executed in **DBMS Backup and Recovery System.**

A database backup operation is performed by backup system, when a problem that damages the database , all committed data is recovered by recovery subsystem

1.10.2 Database System Utilities

The DBMS have **database utilities** that help the DBA to manage the database system. Commonutilities have the following types of functions:

- Loading
- Backup
- Database storage Reorganization
- Performance Monitoring

■ Loading

A loading utility is used to load existing data files into the database.

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. Such tools are also called **conversion tools**

■ **Backup**

A backup utility creates a backup copy of the database, 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 catastrophic 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**

- Monitors database usage and provide statistics to the DBA.
- The DBA uses the statistics in making decision such as whether or not to reorganize files or whether to add or drop indexes to improve performance.

Tools, Application Environments, and Communications Facilities Tools

CASE tools are used in the design phase of database systems.

- Another tool that can be useful in large organizations is an expanded **data dictionary system**.
- The data dictionary stores other information, such as design decisions, usage standards, application program descriptions, and user information. Such a system is also called an **information repository**.

  This information can be accessed *directly* by users or the DBA when needed.

  Application development environments
- The systems provide an environment for developing database applications and include facilities that help in many facets of database systems, including database design, GUI development, querying and updating, and application program development.

Communications software

The DBMS also needs to interface with **communications software**, whose function is to allow users at locations remote from the database system site to access the database through computer

terminals, workstations, or personal computers.

These are connected to the database site through data communications hardware such as Internet Router.

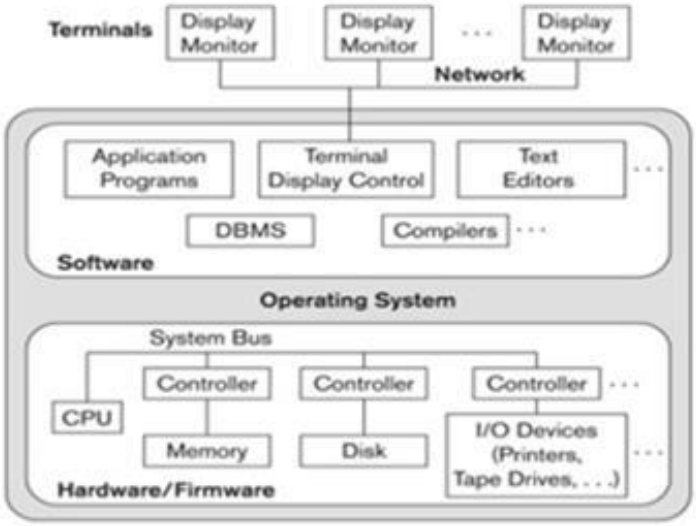
The integrated DBMS and data communications system is called a **DB/DC** system.

1.11 Centralized and Client/Server Architectures for DBMSs

Centralized DBMSs Architecture

Centralized DBMS combines everything into single system including- DBMS software, hardware, application programs and user interface processing software.

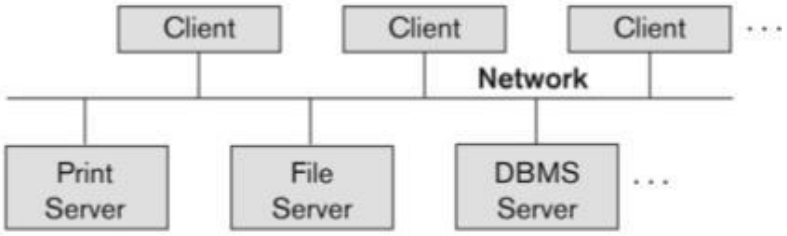
User can connect through a remote terminal – however, all processing is done at centralized site. Figure 1.5 illustrates the physical components in a centralized architecture. Gradually, DBMSsystems started to exploit the available processing power at the user side, which led toclient/server DBMS architectures.



Physical components in a centralized architecture

Basic Client/Server Architectures

The **client/server architecture** was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, data base servers, Web servers, e-mail servers, and other software and equipment are connected via a network. Illustrates client/server architecture at the logical level:



Logical two-tier architecture

- 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.
- A **server** is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access.
- The **file server** that maintains the files of the client machines, another machine can be designatedas a **printer server** by being connected to various printers; all print requests by the clients are forwarded to this machine
- **Web servers** or **e-mail servers** also fall into the specialized server category. The resources Provided by specialized servers can be accessed by many client machines.

Two-Tier Client/Server Architectures for DBMSs.

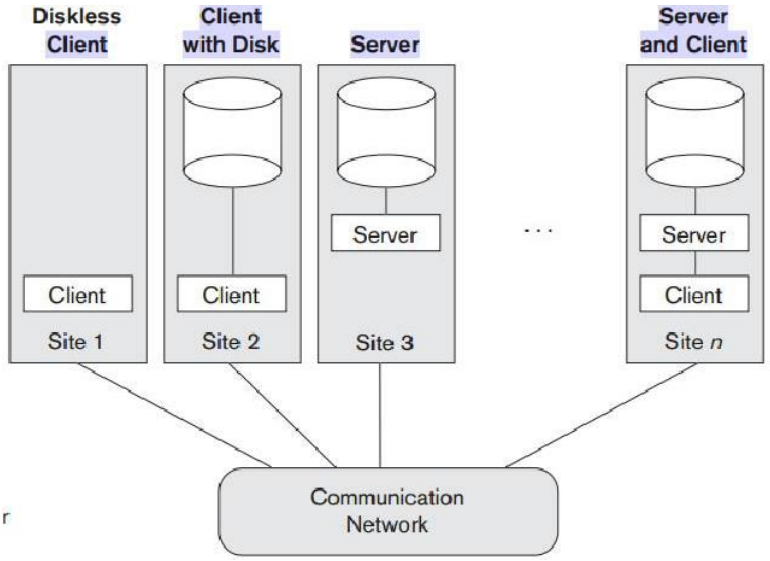
In two tier architecture user interface programs and application programs can run on the client side. When DBMS access is required, the program establishes a connection to the DBMS server.

A standard called **Open Database Connectivity** (**ODBC**) provides an **application programming interface** (**API**). is a simplified diagram that shows the physical Architecture. Some machines wouldbe client sites only.

A client program can actually connect to several RDBMSs and send query and transaction requests using the ODBC API, which are then processed at the server sites. Any query results aresent back to the client program, which can process and display the results as needed.
A related standard for the Java programming language, called **JDBC**, has also been defined. Thisallows Java client programs to DBMS server through a standard interface.

The architectures described here are called **two-tier architectures** because the software Components are distributed over two systems: client and server.

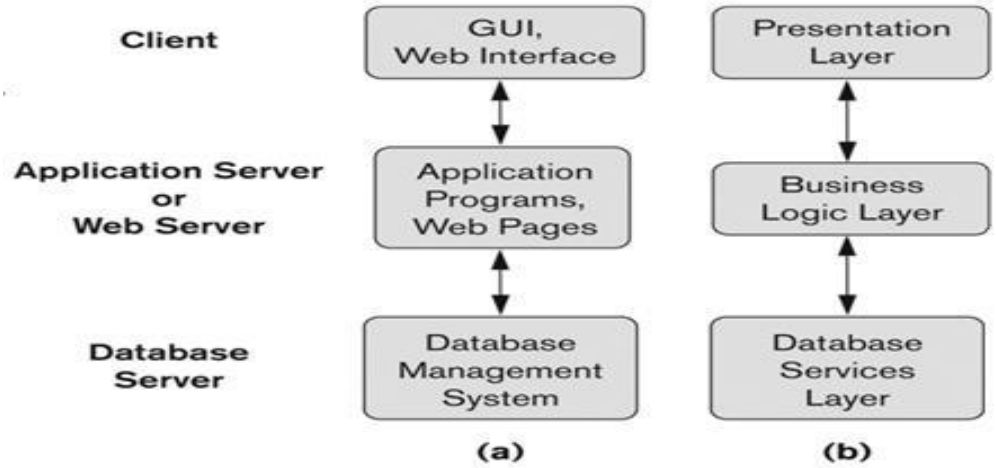
The advantages of this architecture are its simplicity and seamless compatibility with existingsystems.



Physical two-tier client/server architecture.

Three-Tier and n-Tier Architectures for Web Applications

Many Web applications use an architecture called the **three-tier architecture**, which adds an intermediate layer between the client and the database server, as illustrated in Figure



Logical three-tier architecture

This intermediate layer or **middle tier** is called the **application server** or the **Web server**, depending on the application. This server plays an intermediary role by running application programs and storing business rules that are used to access data from the database server. It can also improve database security by checking a client‘s credentials before forwarding a request to the database server.

Clients contain GUI interfaces and some additional application-specific business rules. The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing (partially) processed data from the database server to the clients, where it may be presented to users in GUI format. Thus, the *user interface, application rules,* and *data access* act as the three tiers

1.12 Classification of DBMS

The following criteria are normally used to classify DBMSs.

- **Data model**
- **Number of users**
- **Number of sites**
- **Cost**

Based on the *Data model* being used the DBMS is classified as follows:

Hierarchical data model

- Hierarchical data model represents data as hierarchical tree structure.
- There is no standard language for the hierarchical model; most hierarchical DBMS have
- record-at-time languages
- Legacy applications still run on database systems based on the **hierarchical** and **networkdata models**.

Relational data model

Stores data in the form of a table.

- A relation database is a data driven not design driv
- Maintaining consistency among all applications is easier
- It supports power query language SQL.

Object-Oriented mode

- It is based on the collection of objects
- This database manages objects for multimedia applications and manages data with complex relationships that are difficult to model and process in a RDBMS.

Object-relational DBMS

- Relational DBMS have been extending their model to incorporate object database conceptsand other capabilities

Based on number of *users*, DBMS can be classified as follows:

- S**ingle-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 multipleusers.

Based on *number of Sites* DBMS can be classified as follows:

Centralized DBMS

- 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 residetotally at a single computer site.

Distributed DBMS

- A **distributed** DBMS (DDBMS) can have the actual database and DBMS softwaredistributed over many sites, connected by a computer network.

Based on *the cost* DBMS can be classified as follows:

- The DBMS package cost between $10000 to 100000
- Single user work with micro computers cost between $100 and $3000
- Other few elaborate packages cost more than $10000.

****