PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIT-1

Subject Name: **Database Management Systems**
Subject Code: **20CS3402**

SYLLABUS:

Introduction: Database system, Characteristics (Database Vs File System), Database Users, Advantages of Database systems, Database applications. Brief introduction of different Data Models; Concepts of Schema, Instance and data independence; Three tier schema architecture for data independence; Database system environment, Centralized and Client Server architecture for the databases.

Database system:

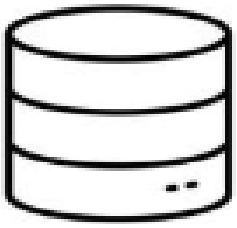
Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science.

DATABASE:

A database is a collection of data, typically describing the activities of one or more related organizations.

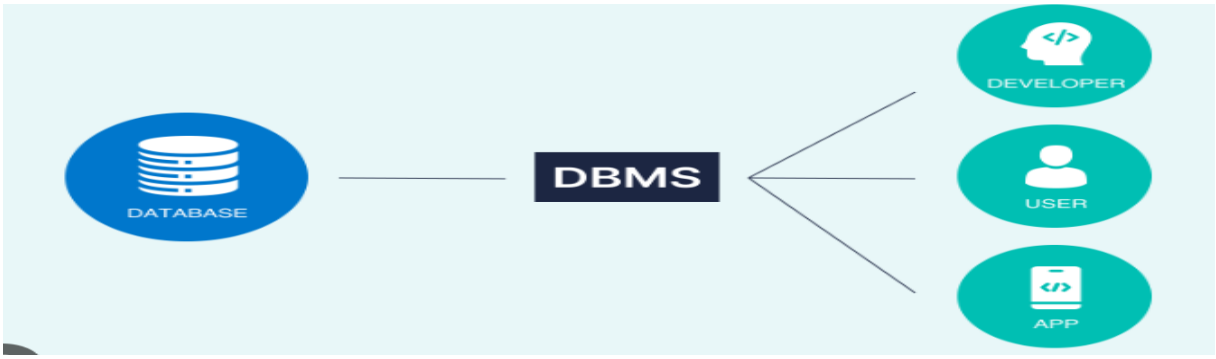
For example, a university database might contain information about the following:

- Entities such as students, faculty, courses, and classrooms.
- Relationships between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.



DBMS:

A database management system (DBMS) or DBMS, is software designed to assist in maintaining and utilizing large collections of data, and the need for such systems, as well as their use, is growing rapidly. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.



- Defining a database involves specifying the data types, 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**.**

Characteristics (Database Vs File System):

To understand the need for a DBMS, **let us consider a motivating scenario:** A company has a large collection (say, 500 GB) of data on employees, departments, products, sales, and so on. This data is accessed concurrently by several employees. Questions about the data must be answered quickly, changes made to the data by different users must be applied consistently, and access to certain parts of the data (e.g., salaries) must be restricted.

- We probably do not have 500 GB of main memory to hold all the data. We must therefore store data in a storage device such as a disk or tape and bring relevant parts into main memory for processing as needed.
- Even if we have 500 GB of main memory, on computer systems with 32-bit addressing, we cannot refer directly to more than about 4 GB of data! We have to program some method of identifying all data items.
- We have to write special programs to answer each question that users may want to ask about the data. These programs are likely to be complex because of the large volume of data to be searched.
- We must protect the data from inconsistent changes made by different users accessing the data concurrently. If programs that access the data are written with such concurrent access in mind, this adds greatly to their complexity.
- We must ensure that data is restored to a consistent state if the system crashes while changes are being made.
- Operating systems provide only a password mechanism for security. This is not sufficiently flexible to enforce security policies in which different users have permission to access different subsets of the data.

Basics	File System	DBMS
Structure	The file system is a way of arranging the files in a storage medium within a computer.	DBMS is software for managing the database.
Data Redundancy	Redundant data can be present in a file system.	In DBMS there is no redundant data.
Backup and Recovery	It doesn't provide Inbuilt mechanism for backup and recovery of data if it is lost.	It provides in house tools for backup and recovery of data even if it is lost.
Query processing	There is no efficient query processing in the file system.	Efficient query processing is there in DBMS.
Consistency	There is less data consistency in the file system.	There is more data consistency because of the process of normalization
Complexity	It is less complex as compared to DBMS.	It has more complexity in handling as compared to the file system.
Security Constraints	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file systems.
Cost	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.
Data Independence	There is no data independence.	In DBMS Data Independence exists, mainly of two types: 1).Conceptual Data Independence. 2)Physical Data Independence.
User Access	Only one user can access data at a time.	Multiple users can access data at a time.
Meaning	The users are not required to write procedures.	The user has to write procedures for managing databases
Sharing	Data is distributed in many files. So, it is not easy to share data.	Due to centralized nature data sharing is easy
Data Abstraction	It give details of storage and representation of data	It hides the internal details of Database.

Basics	File System	DBMS
Integrity Constraints	Integrity Constraints are difficult to implement	Integrity constraints are easy to implement
Attributes	To access data in a file , user requires attributes such as file name, file location.	No such attributes are required.
Example	Cobol, C++.	Oracle, SQL Server

Database Users :

Users may be divided into

- Those who actually use and control the database content, and those who design, develop and maintain database applications called **"Actors on the scene".**
- Those who design and develop the DBMS software and related tools, and the computer systems operators called **"Workers behind the scene".**

Actors on the Scene:

1) Database Administrator (DBA): chief administrator, who oversees and manages the database system (including the data and software). Duties include authorizing users to access the database, coordinating/monitoring its use, acquiring hardware/software for upgrades, etc. The DBA is accountable for problems such as security breaches and poor system response time. In large organizations, the DBA might have a support staff.

2) Database Designers: responsible for identifying the data to be stored and for choosing an appropriate way to organize it. 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. The final database design must be capable of supporting the requirements of all user groups.

3) End Users: These are persons who access the database for querying, updating, and report generation. There are several categories of end users:

- ✓ **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:** 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:** Include engineer scientist's 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.
- ✓ **Stand-alone users:** maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces.

Example: user of a tax package that stores a variety of personal financial data for tax purposes

4) System Analysts and 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.

Workers behind the Scene

1. 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.

2. Tool developers: design and implement tools that facilitate database modelling and design, database system design, and improved performance.

3. Operators and maintenance personnel (system administration personnel): responsible for the actual running and maintenance of the hardware and software environment for the database system.

Advantages of Database systems:

1)**Controlling Redundancy:** Data redundancy such as tends to occur in the "file processing" approach leads to wasted storage space, duplication of effort and a higher likelihood of the introduction of inconsistency.

In the database approach, the views of different user groups are integrated during database design. This is known as data normalization, and it ensures consistency and saves storage space. However, it is sometimes necessary to use controlled redundancy to improve the performance of queries. For example, we may store Student_name and Course_number redundantly in a GRADE_REPORT file because whenever we retrieve a GRADE_REPORT record, we want to retrieve the student name and course number along with the grade, student number, and section identifier.

A DBMS should provide the capability to automatically enforce the rule that no inconsistencies are introduced when data is updated.

2) Restricting Unauthorized Access: When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. For example, financial data is often considered confidential and only authorized persons are allowed to access such data. In addition, some users may only be permitted to retrieve data, whereas others are allowed to retrieve and update. Hence, the type of access operation retrieval or update must also be controlled. A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts, to specify account restrictions and enforce these restrictions automatically.

3) Providing Persistent Storage for Program Objects: The values of program variables or objects are discarded once a program terminates, unless the programmer explicitly stores them in permanent files, which often involves converting these complex structures into a format suitable for file storage. Object-oriented database systems make it easier for complex runtime objects to be saved in secondary storage so as to survive beyond program termination and to be retrievable at a later time.

Object-oriented database systems are compatible with programming languages such as C++ and Java, and the DBMS software automatically performs any necessary conversions.

4) Providing Storage Structures and Search Techniques for Efficient Query Processing: DBMS maintains indexes that are utilized to improve the execution time of queries and updates. DBMS has a buffering or caching module that maintains parts of the database in main memory buffers. The query processing and optimization module is responsible for choosing an efficient query execution plan for each query submitted to the system.

5) Providing Backup and Recovery: The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing. Disk backup is also necessary in case of a catastrophic disk failure.

6) Providing Multiple User Interfaces: Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include

Query languages for casual users

Programming language interfaces for application

programmers Forms and command codes for parametric users

Menu-driven interfaces and natural language interfaces for standalone users

7) Representing Complex Relationships among Data: A database may include numerous varieties of data that are interrelated in many ways.

For example each section record is related to one course record and to a number of GRADE_REPORT records one for each student who completed that section. A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

8) Enforcing Integrity Constraints: Most database applications are such that the semantics of the data require that it satisfy certain restrictions in order to make sense.

The simplest type of integrity constraint involves specifying a data type for each data item. For example, in student table we specified that the value of Name must be a string of no more than 30 alphabetic characters.

More complex type of constraint is referential integrity involves specifying that a record in one file must be related to records in other files. For example, in university database, we can specify that every section record must be related to a course record.

Another type of constraint specifies uniqueness on data item values, such as every course record must have a unique value for Course_number. This is known as a key or uniqueness constraint.

9) Permitting Inferencing and Actions Using Rules: In a deductive database system, one may specify declarative rules that allow the database to infer new data. For example, figure out which students are on academic probation. Such capabilities would take the place of application programs that would be used to ascertain such information otherwise.

Active database systems go one step further by allowing "active rules" that can be used to initiate actions automatically.

In today's relational database system it is possible to associate triggers with tables.

10) Additional Implications of Using the Database Approach:

Potential for Enforcing Standards: database approach permits the DBA to define and enforce standards among database users in a large organization which facilitates communication and cooperation among various departments, projects, and users within the organization. Standards can be defined for names and formats of data elements, display formats, report structures and so on.

Reduced Application Development Time: once a database is up and running, substantially less time is generally required to create new applications using DBMS facilities. Development time using a DBMS is estimated to be one-sixth to one-fourth of that for a traditional file systems.

Flexibility: It may be necessary to change the structure of a database as requirements change. DBMSs allow changes to the structure of the database without affecting the stored data and the existing application programs.

Availability of Up-to-Date Information: DBMS makes the database available to all users. Availability of up-to-date information is essential for many transaction processing applications, such as reservation systems or banking databases.

Economies of Scale: DBMS approach permits consolidation of data and applications, to overlap between activities of data-processing in different projects or departments. This enables the whole organization to invest in more powerful processors, storage devices, or communication gear, rather than having each department purchase its equipment thus reducing overall costs of operation and management.

Database applications:

Early Database Applications Using Hierarchical and Network Systems

Early database applications maintained records in large organizations such as corporations, universities, hospitals, and banks. In many of these applications, there were large numbers of records of similar structure. There were also many types of records and many interrelationships among them.

Problems with the early database systems

o Lack of data abstraction and program-data independence capabilities
o Provided only programming language interfaces. This made it time-consuming and expensive to implement new queries and transactions, since new programs had to be written, tested, and debugged.

Providing Data Abstraction and Application Flexibility with Relational Databases

Relational databases were originally proposed to separate the physical storage of data from its conceptual representation and to provide a mathematical foundation for data representation and querying. The relational data model also introduced high-level query languages that provided an alternative to programming language interfaces, making it much faster to write new queries. Hence, data abstraction and program-data independence were much improved when compared to earlier systems.

Object-Oriented Applications and the Need for More Complex Databases

Object-oriented databases (OODBs) mainly used in specialized applications, such as engineering design, multimedia publishing, and manufacturing systems. In addition, many object-oriented concepts were incorporated into the newer versions of relational DBMSs, leading to object-relational database management systems, known as ORDBMSs.

Interchanging Data on the Web for E-Commerce Using XML

The World Wide Web provides a large network of interconnected computers. Users can create documents using a Web publishing language, such as HyperText Markup Language (HTML), and store these documents on Web servers where other users (clients) can access them. Documents can be linked through hyperlinks, which are pointers to other documents.

Currently, eXtended Markup Language (XML) is considered to be the primary standard for interchanging data among various types of databases and Web pages. XML combines concepts from the models used in document systems with database modeling concepts.

Extending Database Capabilities for New Applications

The success of database systems in traditional applications encouraged developers of other types of applications to attempt to use them. The following are some examples of these applications: Scientific applications that store large amounts of data resulting from scientific experiments in areas such as high-energy physics, the mapping of the human genome, and the discovery of protein structures.
o Storage and retrieval of images, including scanned news or personal photographs, satellite photographic images, and images from medical procedures such as x-rays and MRIs (magnetic resonance imaging).
o Storage and retrieval of videos, such as movies, and video clips from news or personal digital cameras.
o Data mining applications that analyze large amounts of data searching for the occurrences of specific patterns or relationships, and for identifying unusual patterns in areas such as credit card usage.
o Spatial applications that store spatial locations of data, such as weather information, maps used in geographical information systems, and in automobile navigational systems. o Time series applications that store information such as economic data at regular points in time, such as daily sales and monthly gross national product figures.

Databases versus Information Retrieval

Database technology is heavily used in manufacturing, retail, banking, insurance, finance, and health care industries, where structured data is collected through forms, such as invoices or patient registration documents. An area related to database technology is Information Retrieval (IR), which deals with books, manuscripts, and various forms of library based articles. Data is indexed, cataloged, and annotated using keywords. IR is concerned with searching for material based on these keywords, and with the many problems dealing with document processing and free-form text processing.

Data Model, Schemas and Instances:

Data Model

A data model is a collection of concepts that can be used to describe the structure of a database. By structure of a database we mean the data types, relationships and constraints that apply to the data. Most data models also include a set of basic operations for specifying retrievals and updates on the database. Data model provides the necessary means to achieve abstraction.

Categories of Data Models

Data models can be categorized according to the types of concepts they use to describe the database structure.

1) High-level or conceptual data models: provide concepts that are close to the way many users perceive data. Conceptual data models use concepts such as entities, attributes, and relationships.

2) Representational or implementation data models: provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly. Representational or implementation data models are the models used most frequently in traditional commercial DBMSs. These include the widely used relational data model, as well as the so-called legacy data models the network and hierarchical models. Representational data models represent data by using record structures and hence are sometimes called record-based data models.

3) Low-level or physical data models: provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks. Physical data models describe how data is stored as files in the computer by representing information such as record formats, record orderings, and access paths.

Database schema

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

Schema diagram

A displayed schema is called a ***schema diagram.*** A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints.

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

Database state or snapshot

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. In a given database state, each schema construct has its own current set of instances; for example, the STUDENT construct will contain the set of individual student entities (records) as its instances.

The distinction between database schema and database state is very important. When we define a new database, we specify its database schema only to the DBMS. At this point, the corresponding database state is the empty state with no data. We get the initial state of the database when the database is first populated or loaded with the initial data. From then on, every time an update operation is applied to the database, we get another database state. At any point in time, the database has a current state.

The DBMS is partly responsible for ensuring that every state of the database is a valid state that is, a state that satisfies the structure and constraints specified in the schema. The DBMS stores the descriptions of the schema constructs and constraints also called the meta data in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to. The schema is sometimes called the intension, and a database state is called an extension of the schema.

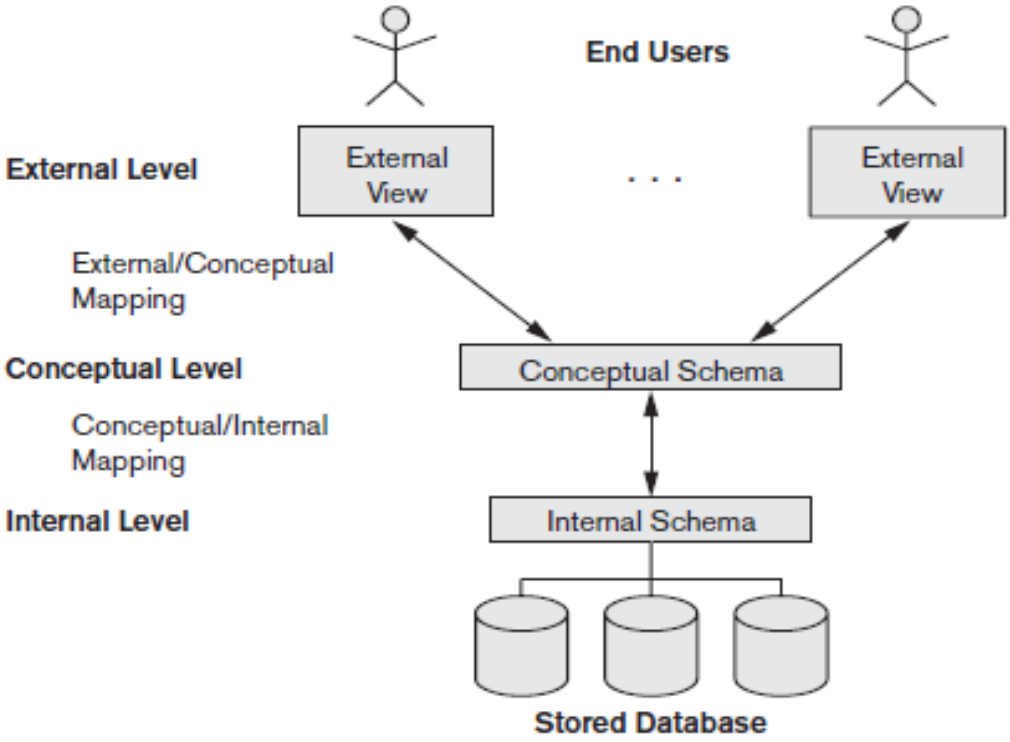
The Three-Schema Architecture

The goal of the three-schema architecture 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. Usually, 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 and hides the rest of the database from that user group. Each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.



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. 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

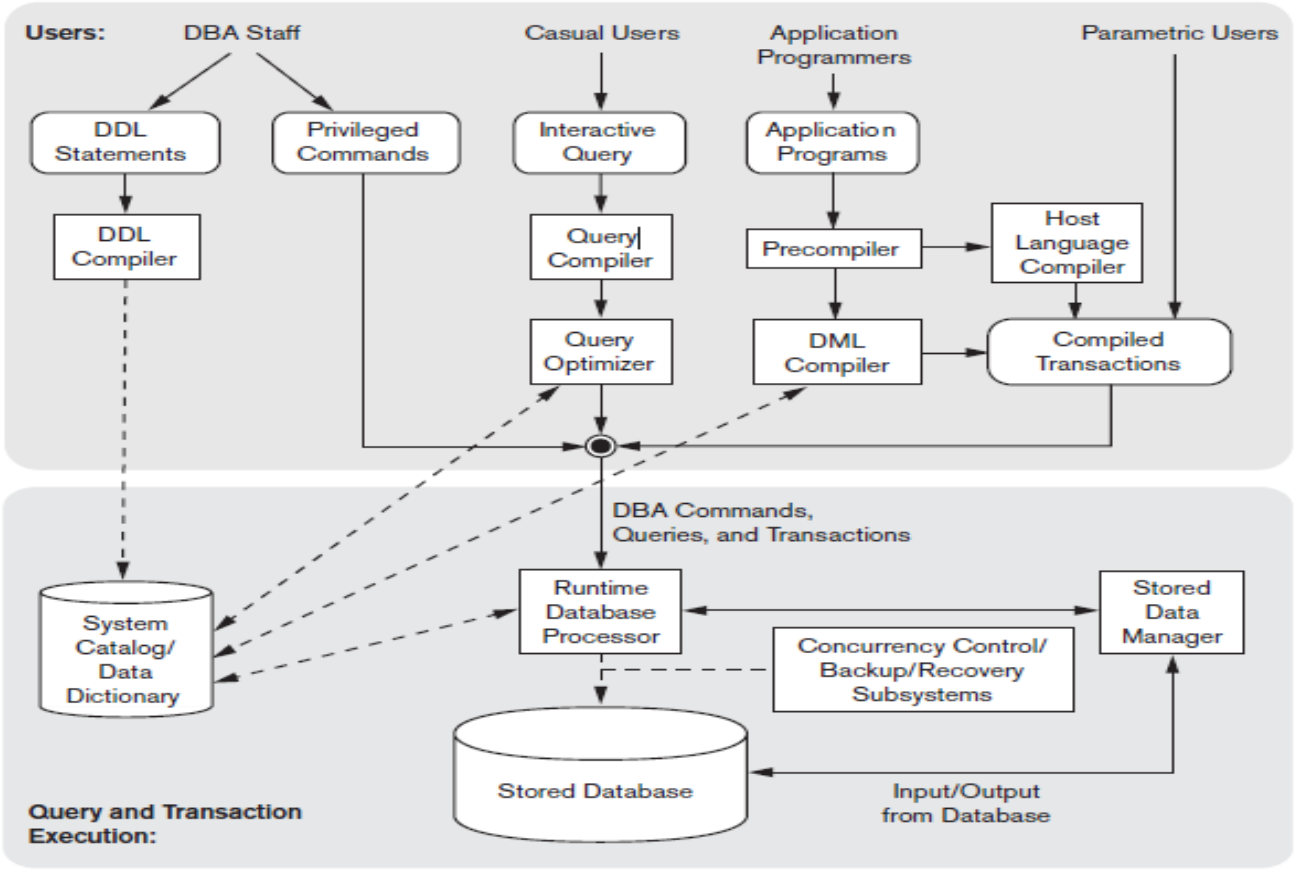
Data independence can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

1) **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema 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.

2) **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized for example, by creating additional access structures to improve the performance of retrieval or update.

Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed.

Database system environment:



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.

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

Interactive query interface: interface for Casual users and persons with occasional need for information from the database.

Query compiler- validates for correctness of the query syntax, the names of files and data elements & compiles them into an internal form.

Query optimizer concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution. It consults the system catalog for statistical and other physical information about the stored data and generates executable code that performs the necessary operations for the query and makes calls on the runtime processor.

Pre compiler - extracts DML commands from an application program and sends to the DML compiler for compilation into object code for database access.

Host language compiler - 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.

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 may update it with statistics. It also works with the stored data manager, which in turn uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory. The runtime database processor handles other

aspects of data transfer, such as management of buffers in the main memory.

Stored data manager uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory. Concurrency control and backup and recovery systems integrated into the working of the runtime database processor for purposes of transaction management.

Database System Utilities

Database utilities help the DBA to manage the database system.

Common utilities have the following types of functions:

Loading: used to load existing data files such as text files or sequential files into the database.

Backup: 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 catastrophic disk failure. Incremental backups are also often used, where only changes since the previous backup are recorded. Incremental backup is more complex, but saves storage space.

Database storage reorganization: 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 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.

Tools, Application Environments, and Communications Facilities

Tools

CASE : used in the design phase of database systems

Data dictionary: In addition to storing catalog information about schemas and constraints, 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

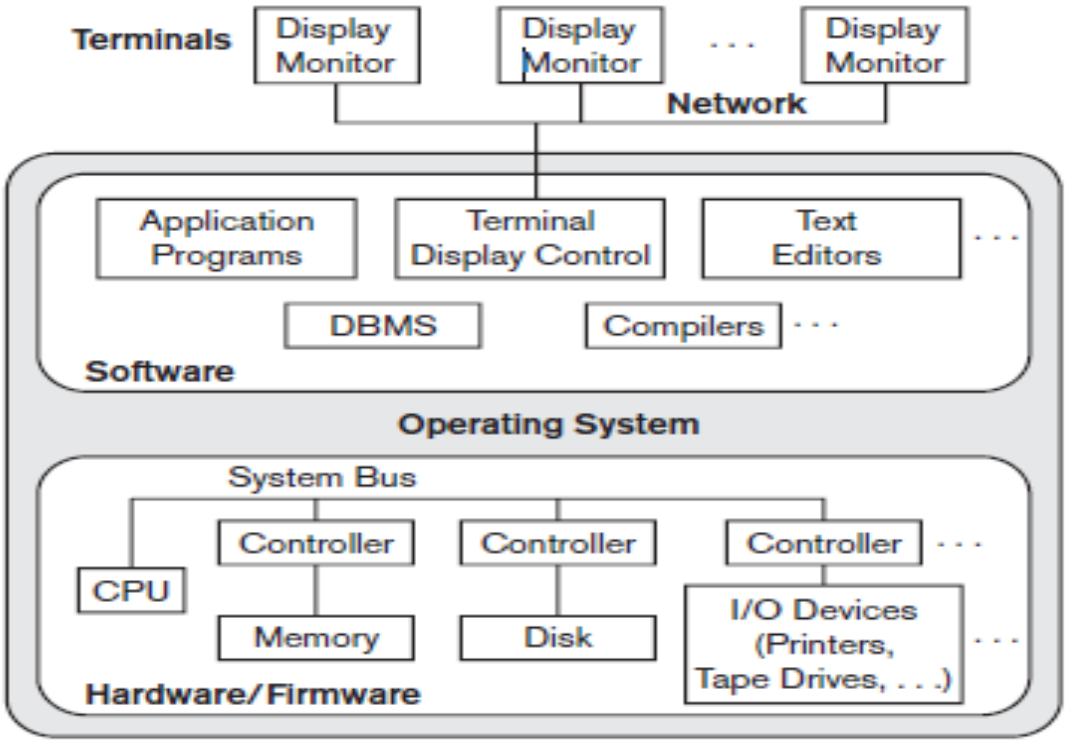
PowerBuilder (Sybase) or JBuilder (Borland): provide an environment for developing database applications including database design, GUI development, querying and updating, and application program development.

Communications software: allow users at locations remote from the database system site to access the database through computer terminals, workstations, or personal computers. Integrated DBMS and data communications system is called a DB/DC system.

Centralized and Client/Server Architectures for DBMSs:

Centralized DBMSs Architecture

All DBMS functionality, application program execution, and user interface processing carried out on one machine.



Disadvantages:

When the central site computer or database system goes down, then everyone is blocked from using the system.

Communication costs from the terminals to the central site can expensive.

Basic Client/Server Architectures

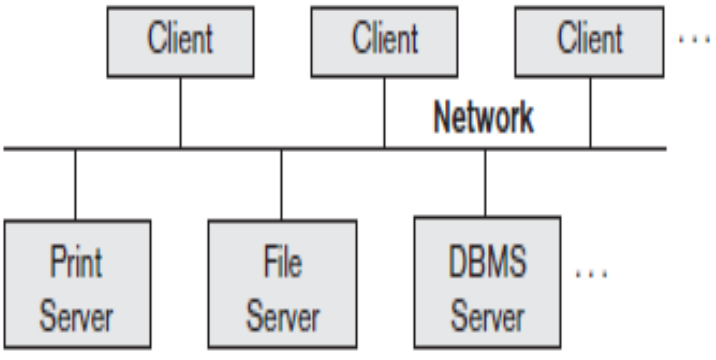
The client/server architecture was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, database servers, Web servers, e-mail servers, and other software and equipment are connected via a network.

Define specialized servers with specific functionalities.

For example file server that maintains the files of the client machines.

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 and local processing power to run local applications



The concept of client/server architecture assumes an underlying framework that consists of many PCs and workstations as well as a smaller number of mainframe machines, connected via LANs and other types of computer networks.

A client is a user machine that provides user interface capabilities and local processing.

When a client requires access to additional functionality such as database access that does not exist at that machine, it connects to a server that provides the needed functionality.

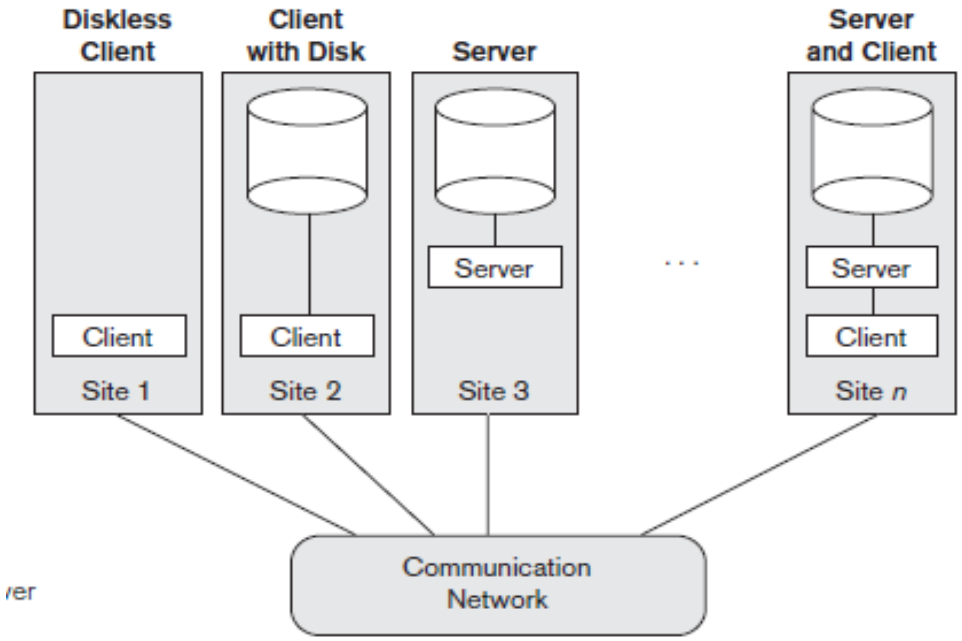
A server is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access. The software components are distributed over two systems: client and server

✓ **Server handles**

- Query and transaction functionality related to SQL

✓ **Processing Client handles**

- User interface programs and application programs



The 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 (which is on the server side) once the connection is created, the client program can communicate with the DBMS.

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 are sent 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 to allow Java client programs to access one or more DBMSs through a standard interface.

Object-oriented DBMSs

The different approach to two-tier client/server architecture was taken by some object oriented DBMSs, where the software modules of the DBMS were divided between client and server in a more integrated way.

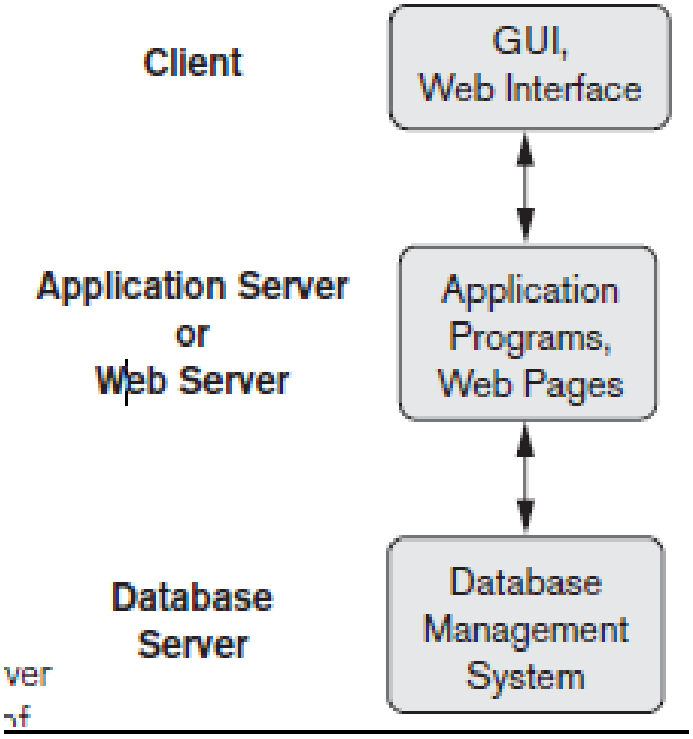
Server level: May include the part of the DBMS software responsible for handling data storage on disk pages, local concurrency control and recovery, buffering and caching of disk pages.

Client level: May handle the user interface, data dictionary functions, DBMS interactions with programming language compilers, global query optimization, concurrency control, and recovery across multiple servers, structuring of complex objects from the data in the buffers.

In this approach, the client/server interaction is more tightly coupled and is done internally by the DBMS modules some of which reside on the client and some on the server rather than by the users/programmers.

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.



Client

- Contain GUI interfaces and some additional application-specific business rules.

Application Server or the Web Server

- Accepts requests from the client, processes the request and sends database queries and commands to the database server, and then passes processed data from the database server to the clients, where it may be processed further and filtered to be presented to users in GUI format.
- It can also improve database security by checking client's credentials before forwarding a request from the database server.

The figure shows another architecture used by database and other application package vendors.

Presentation Layer: Displays information to the user and allows data entry.

The business Logic Layer:

- Handles intermediate rules and constraints before data is passed up to the user or down to the DBMS.
- Can also act as a Web server, which retrieves query results from the database server and formats them into dynamic Web pages that are viewed by the Web browser at the client side

The bottom Layer: Includes all the data management services.

N-tier Architecture:

It is possible to divide the layers between the user and the stored data further into finer components, thereby

giving rise to n-tier architectures; where n may be four or five tiers. The business logic layer is divided into multiple layers.

Advantage:

- Any one tier can run on an appropriate processor or operating system platform and can be handled independently.

Vendors of ERP (enterprise resource planning) and CRM (customer relationship management) packages often use a middleware layer, which accounts for the front-end modules (clients) communicating with a number of back-end databases (servers).