CHAPTER-2

Database System Architecture

2.1 INTRODUCTION

One of the main aims of a database system is to provide users with an abstract view of data, hiding certain details of how data is stored and manipulated. To satisfy these needs, we need to develop architecture for the database systems.

The database architecture is a framework in which the structure of the DBMS is described.

2.2 Schemas, Sub-schemas, and Instances

2.2.1 Schema

- The overall design of the database is called the database schema OR
  The plan (or formulation of scheme) of a database is known as *schema.*
- The logical structure of database
- Schema gives the names of the entities and attributes. It specifies the *relationship* among them.
- Schema means an overall plan of all the data item (field) types and records type store in a database.
- Schema is stored in data dictionary.
- Schema does not change frequently.
- Example: the database consists of information about a set of customers and accounts and the relationship between them
- Database system has several schemas-
  - Physical schema at the lowest level(physical level)
  - Logical schema at the intermediate level(logical level)

PRODUCT		
PROD-ID	PROD-DESC	PROD-COST

CUSTOMER				
CUST-ID	CUST-NAME	CUST-STREET	CUST-CITY	CUST-BAL

SALES			
CUST-ID	PROD-ID	PROD-QTY	PROD-PRICE

Schema diagram for sales record database

```
Schema name is SALES-RECORD

        type CUSTOMER = record
              PROD-ID: string;               Attributes (or fields, or
              PROD-DESC: string;                       data item)
               UNIT-COST: integer;
             end

       type PRODUCT = record
              CUST-ID: integer;               Attributes (or fields, or
              CUST-NAME: string;                        data item)
               CUST-STREET: string;
              CUST-CITY: string;
              CUST-BAL: integer;
             end

      type PRODUCT = record
              CUST-ID: integer;               Attributes (or fields, or
              PROD-ID: string;                          data item)
               PROD-QTY: integer;
              PROD -PRICE: integer;
             end
```

Schema defined using database language

2.2.2 Sub-schema

- A *Subschema* is a subset of the schema and inherits same property that a schema has.
- Schemas may have many different subschemas.
- User can change his subschema without affecting its view of others.
- It is also known as view schema.
- Subschema refers to an application programmer(user's) view of the data item types and record types
- Subschema at the highest level
- Subschema improves security of database.

2.2.3 Instances

- The information stored in the database at a particular moment is called an instance of the database.
- It defines current state of database with existing contents and relationships.
- Instance may change frequently.
- Database change over time as information is inserted and deleted.
- For example value of mobile number in any record is an instance of that field.

OR

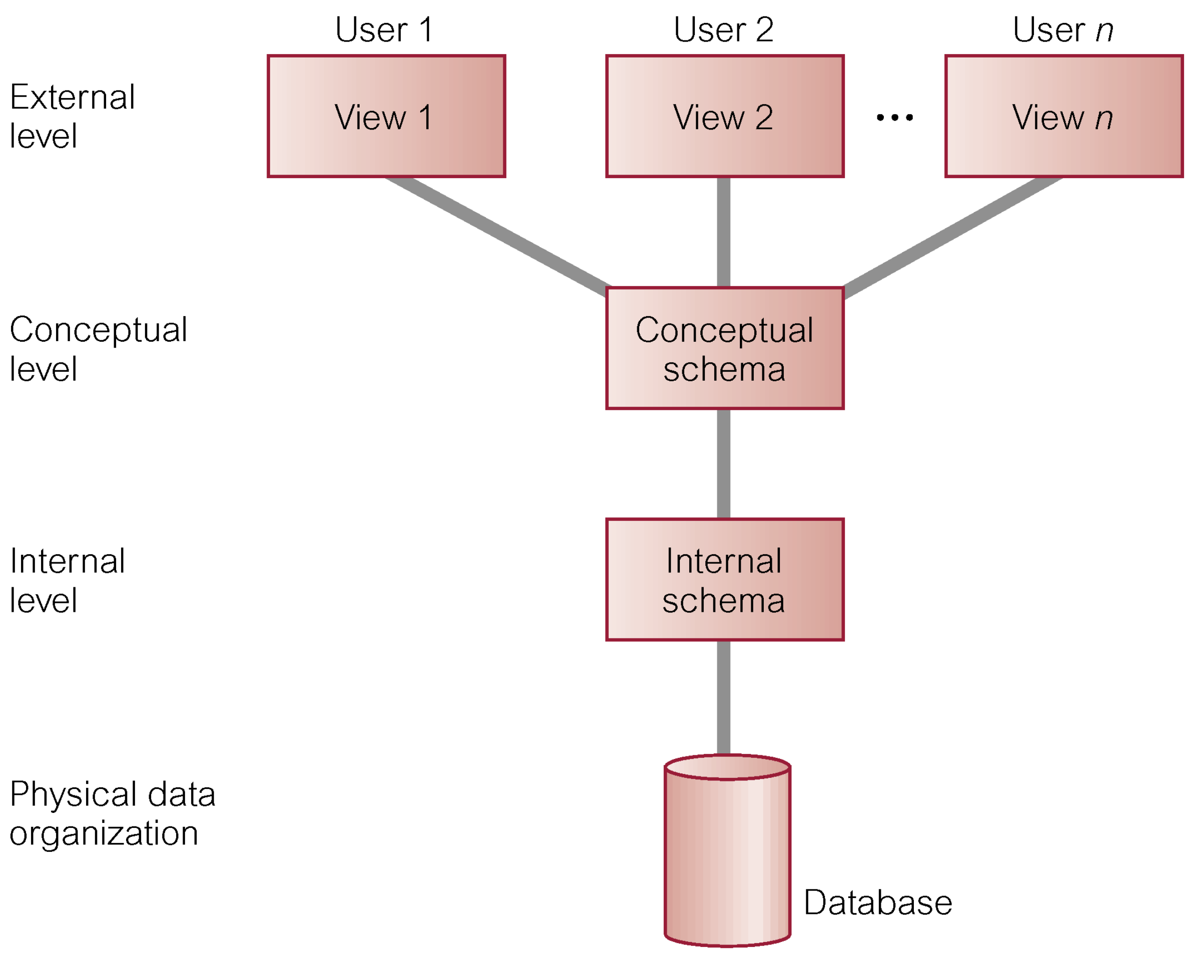
- ➢ The actual content of the database at a particular point in time
    - ▪ Analogous to the value of a variable

2.3 <u>Three-level ANSI SPARC database Architecture</u>

ANSI-SPARC(American National Standards Institute- Standard Planning and Requirements Committee) produced a three – tier architecture with a system catalog.

ANSI-SPARC three – tier database architecture is shown in fig. It consists of following three levels:

- Internal Level
- Conceptual Level
- External Level



❖ **External Level**

- ➢ Users’ view of the database.
- ➢ The highest level of abstraction describes only part of the entire database.
- ➢ Describes that part of database that is relevant to a particular user.

- Mostly end users interact with this level most frequently.
- This level deals with mainly the end display of the database structure along with records stored here.

❖ Conceptual Level

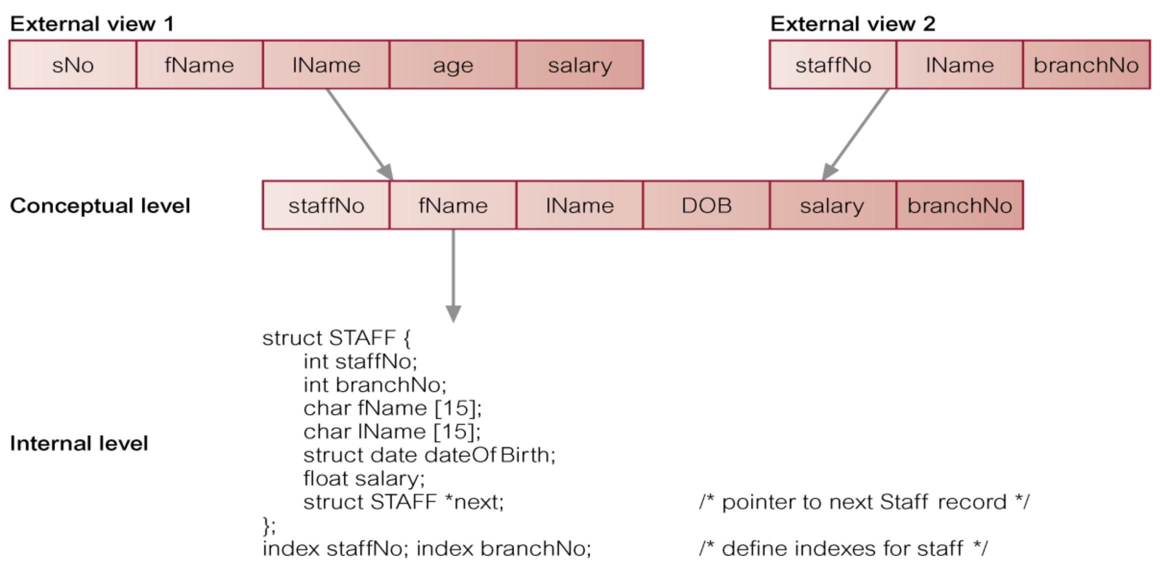
.

- The next higher level of abstraction.
- **Describes what data is stored in database and relationships among the data**.
- The logical level of abstraction is used by database administrator who must decide what information is to be kept in the database.
- Conceptual level is concerned with the following activities:
    - All entities, their attributes and their relationships.
    - Constraint on data.
    - Semantic information about data.
    - Security information.
    - Check data consistency and integrity.

❖ Internal Level

- Physical representation of the database on the computer.
- The lowest level of abstraction.
- **Describes how the data is stored in the database and describes the data structure.**
- Complex low-level data structures are described in detail.
- Internal level is concerned with following activities:
    - Storage space allocation for data and storage.
    - Record descriptions for storage with stored sizes for data items.'
    - Record placements.
    - Data compression and data encryption techniques.

Differences between Three Levels of ANSI-SPARC Architecture



```
struct STAFF {
      int staffNo;
      int branchNo;
      char fName [15];
      char lName [15];
      struct date dateOfBirth;
      float salary;
      struct STAFF *next;              /* pointer to next Staff record */
};
index staffNo; index branchNo;         /* define indexes for staff */
```

- Objectives of three-tier architecture/advantages of three tier architecture
    - All users should be able to access same data.
    - A user's view is immune to changes made in other views.
    - Users should not need to know physical database storage details.
    - DBA should be able to change database storage structures without affecting the users' views.
    - Internal structure of database should be unaffected by changes to physical aspects of storage.
    - DBA should be able to change conceptual structure of database without affecting all users.

2.4 <u>Data Independence</u>

- **Definition:**
  The ability to modify a schema definition in one level without affecting a schema definition in the next higher level is called data independence.

- There are two level of data independence.
    - Physical data independence
    - Logical data independence

    - **Physical data independence**
        - Refers to immunity of conceptual schema to changes in the internal schema.
        - Internal schema changes (e.g. using different file organizations, storage structures/devices).
        - When changes are made at internal schema ,Conceptual/internal mapping is changed
        - Should not require change to conceptual or external schemas.

    - **Logical data independence**
        - Refers to immunity of external schemas to changes in conceptual schema.
        - Conceptual schema changes (e.g. addition/removal of entities).
        - Should not require changes to external schema or rewrites of application programs.
        - When changes are made at Conceptual schema , external /Conceptual mapping is changed
        - It is done whenever logical structure of the database is altered.

2.5 <u>Mappings</u>

The process of transforming request and reply at different levels is known as mapping.

- There are the two types of mappings
    - Conceptual/Internal Mapping
    - External/Conceptual Mapping

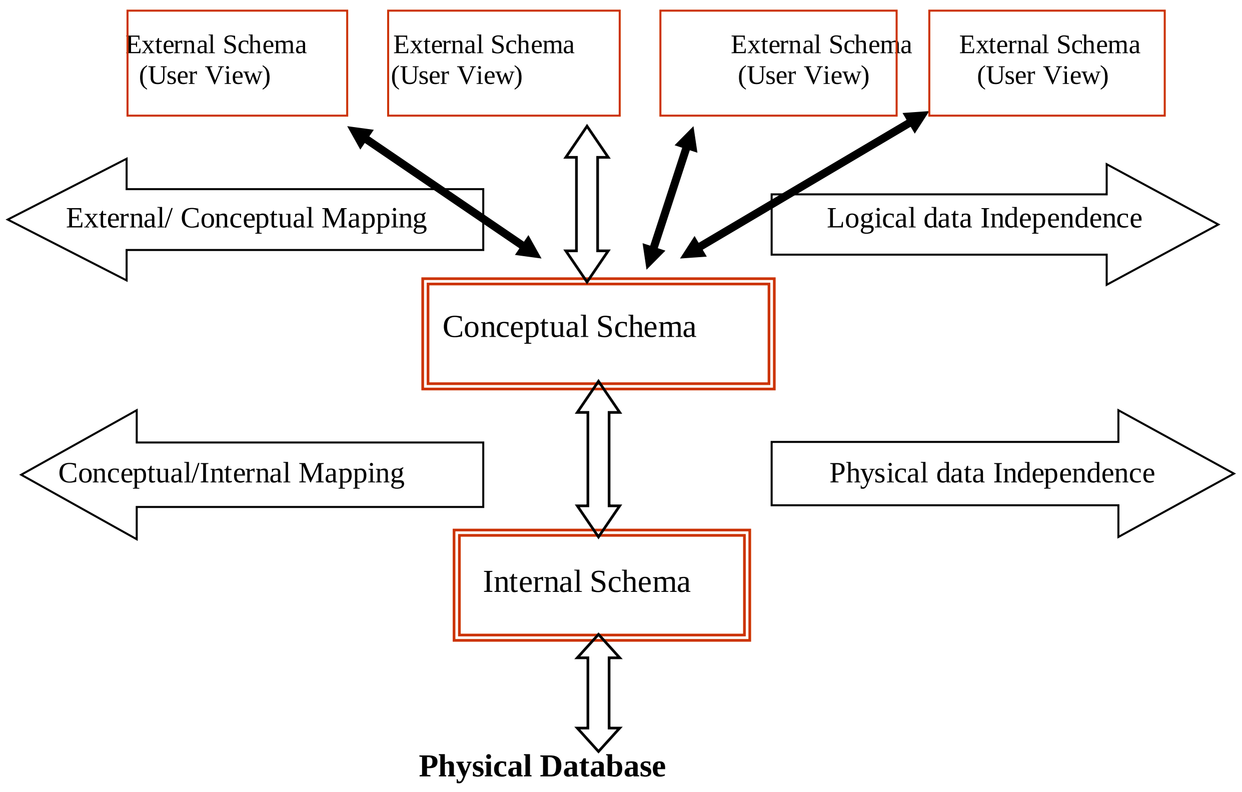
    - Conceptual/Internal Mapping
        - The conceptual schema is related to the internal schema through conceptual/internal mapping.

- The conceptual/internal mapping defines the correspondence between the conceptual view and the store database; it specifies how conceptual record and fields are represented at the internal level.
- If structure of the store database is changed.
- If changed is made to the storage structure definition-then the conceptual/internal mapping must be changed accordingly, so that the conceptual schema can remain invariant.

➢ External/Conceptual Mapping

- Each external schema is related to the conceptual schema through external/conceptual mapping.
- The external/conceptual mapping defines the correspondence between a particular external view and conceptual view.
- The differences that can exist between these two levels are analogous to those that can exist between the conceptual view and the stored database.
- Example: fields can have different data types; fields and record name can be changed; several conceptual fields can be combined into a single external field.
- Any number of external views can exist at the same time; any number of users can share a given external view: different external views can overlap.

Mapping of three – tier Architecture.


External Schema (User View)
External Schema (User View)
External Schema (User View)
External Schema (User View)
External/ Conceptual Mapping
Logical data Independence
Conceptual Schema
Conceptual/Internal Mapping
Physical data Independence
Internal Schema
Physical Database

2.6 Structure, Components and Functions of DBMS

2.6.1 Structure of DBMS

DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users. The various components of DBMS are shown below: -

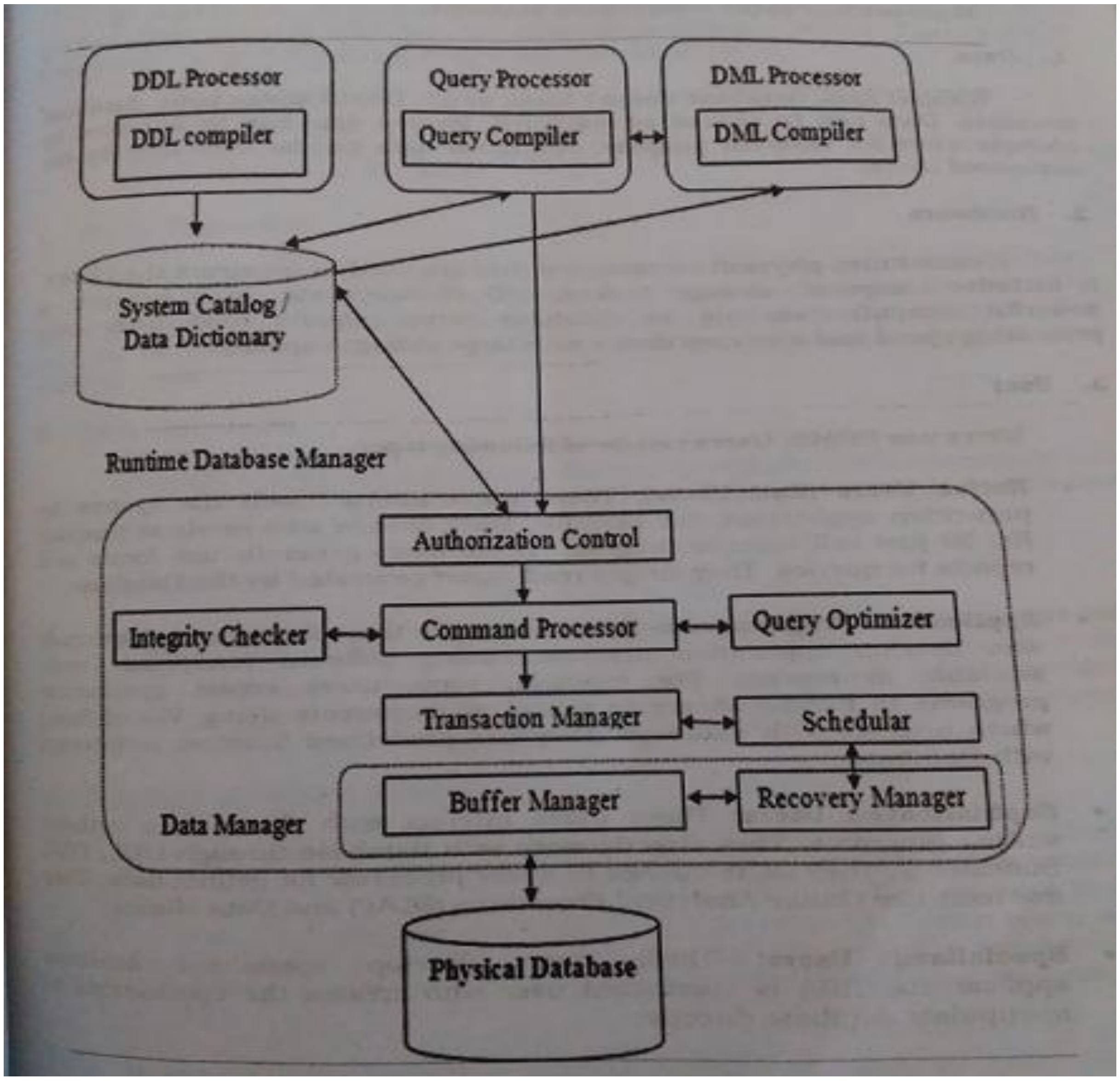


Fig. 2.1 Structure of DBMS

2.6.2 Components of database system.

- **Query Processor:** The query processor transforms users queries into a series of low-level instructions directed to the run time database manager.
  It is used to interpret the online user's query and convert it into an efficient series of operations in a form capable of being sent to runtime data manager for execution.

- **Run time Database Manager:**
  Run time database manager is central software components of the DBMS, which interfaces with user-submitted application programs and queries.
  It handles database access at run time.
  Run time database manager is sometimes referred to as the *database control system* and has the following components:

  - **Authorization Control:-** checks the authority of users to access data.
  - **Command Processor:-** Processes Queries passed by Authorization control module.
  - **Integrity checker:-** Checks for necessary integrity constraints
  - **Query optimizer: -** Produce an efficient execution plan for evaluating query.
  - **Transaction Manager:-**Preserves Atomicity and control concurrency.
  - **Scheduler: -** Controls the relevant order in which transaction operations are executed.
  - **Data Manager:** - actual handling of data in the database.
    -*Recovery Manager:* Auto Commit (Save) our database. we can recover(restore) our database in case of failures.
    -*Buffer Manager:* transfer of data between the main memory and secondary storage (such as disk or tape).

- **DML Processor:-** using DML compiler, DML processor converts DML statements written in to host programming language in to object code for the database access.

- **DDL Processor:-** using a DDL compiler, the DDL processor converts the DDL statements into a set of tables containing metadata.
  - This tables are then stored in the system catalog while control information is stored in data files headers.
  - the system catalog includes information such as the names of data files, data items, storage details of each data files, mapping information amongst schema, and constraints.

2.6.3 Function and services of DBMS

i. **Data storage management**: provides a permanent storage for data on physical devices.
ii. **Data manipulation management**: provides data manipulation functionality like Insert, update, delete
iii. **Data definition services**
iv. **Data dictionary /system catalog management:** manages Data dictionary /system Catalog to track metadata
v. **Database communication interface:**
vi. **Authorization management:** like log on to the DBMS start the database stop the Database etc.
vii. **Security management:** authorized users can access data
viii. **Backup and recovery management:** provides way to back-up data periodically and recovery from different failures.
ix. **Concurrency control services:** control concurrent access
x. **Transaction management:** ensures atomicity of transaction
xi. **Integrity services:** follows integrity constraints**.**
xii. **Data independence services:** provides data independency
xiii. **Utility services**
xiv. **Import and Export of Data.**
xv. **Maintaining data dictionary**
xvi. **User's Monitoring**

2.6.3 Execution steps of DBMS

i. users issue a query using particular database language, for example SQL commands
ii. The passed query is presented to query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query.
iii. The DBMS accepts the users SQL commands and analyses them.
iv. DBMS produces query evaluation plans that are the external schema for the user, the correspondence External/Conceptual Mapping, the conceptual schema, the conceptual/internal mapping and the storage structure definition.
v. The DBMS executes these plans against physical database and returns the answers to the users.

2.7 Data Models

- Data models are the conceptual method to represent the structure of the database. In other words, a data model describes the structure of a database.
- It consists entities, attributes, relationship, constrains, operations etc.
- There are three different groups.

  - Object-based logical models
    - The entity-relationship model
    - The object-oriented model
    - The semantic data model
    - The functional data model
  - Record-based logical models
    - Relational model
    - Network model
    - Hierarchical model
  - Physical models
    - Unifying model
    - Frame-memory model

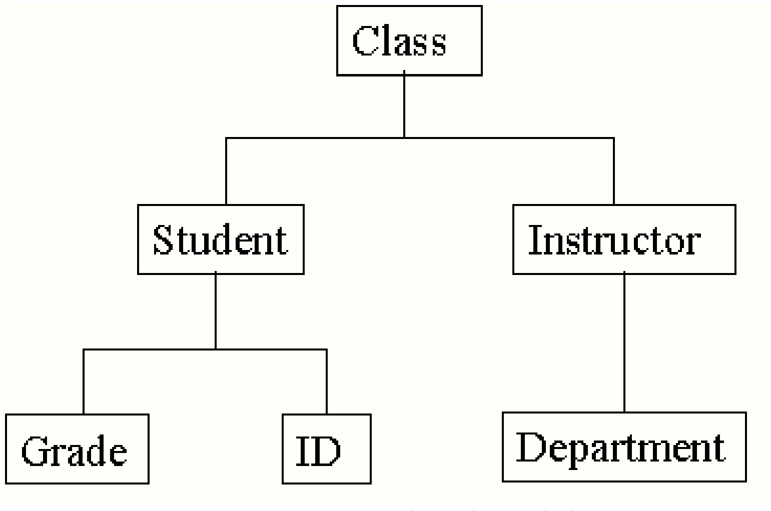
2.7.1 Record-based logical models

- Record based logical models are used in describing data at the logical and view levels.
- They are used to specify overall logical structure of the database.
- Record-base models are named as database structure has fixed format records of several types.
- Each record type defines a fixed number of fields or attributes.
- Each attributes and each fields is a usually of a fixed length.

- There are three most widely used record-based models are:

  i. Hierarchical model
  ii. Network model
  iii. Relational model

i. **Hierarchical model:**
   - In hierarchical model the data and relationships among the data are represented by records and links.
   - In hierarchical model, data is organized in the form of tree.
   - Model starts with a special node known as root node or parent node.
   - One parent can have many children but children are allowed only one parent.
   - It is same as network model but differs in terms of organization of records as collections of trees rather than graphs.



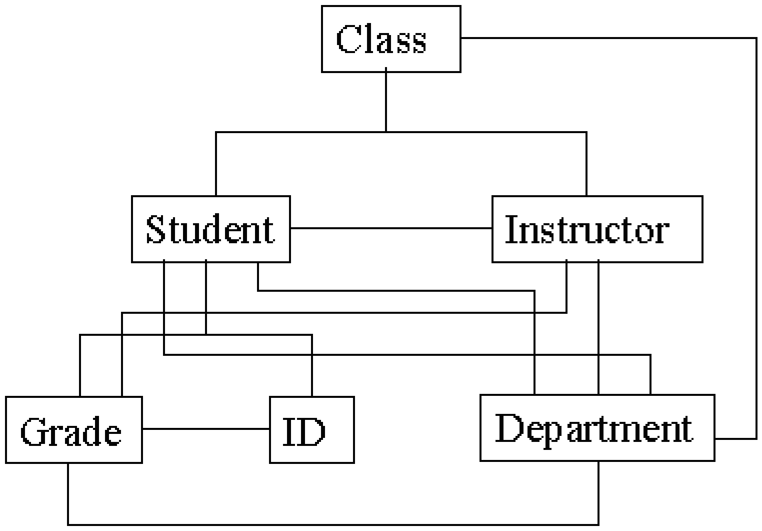
Hierarchical model

- Advantages Hierarchical model
  - ♦ Simplicity
  - ♦ Design of database in this type is simple
  - ♦ Data sharing is good.
  - ♦ Data Security and Data Integrity
  - ♦ Efficiency

- Disadvantages Hierarchical model
  - ♦ Implementation Complexity
  - ♦ Lack of structural independence
  - ♦ Programming complexity
  - ♦ There is no any fixed standard to implement.

ii. **Network model:**

- ♦ Data in the network model are represented by collections of record
- ♦ Relationships among data are represented by links, which can be viewed as pointers.
- ♦ Network data model is almost same as Hierarchical model except that record can have multiple parents.
- ♦ The records in the database are organized as collection of arbitrary graphs



Network model

- Advantages Network model.
    - ♦ Simple and easy to design.
    - ♦ Ease of data access
    - ♦ Implements one to many and many to many relationships.
    - ♦ Data Integrity and capability to handle more relationship types
    - ♦ Data independence
    - ♦ Database standards

- Disadvantages Network model.
    - ♦ System complexity
    - ♦ Absence of structural independence
    - ♦ It is not user friendly system.

iii. **Relational model:**

- ♦ The relational model uses a collection of tables to represent both data and the relationships among those data.
- ♦ Each table has multiple columns, and each column has unique name.
- ♦ Each row must have unique key based on data.
- ♦ It does not use pointers or links.
- ♦ This model relates record by values that they contain.

Customer-ID	Customer-name	Customer-street	Customer-city
C00001	Smith	4 north St	Rye
C00002	Jones	3 main St	Harrison

(a) Customer table

Account-number	Balance
A-101	500
A-215	700

(b) Account table

Customer-ID	Account-number
C00001	A-101
C00002	A-215

(c) Depositor table

- Advantages Relational model.
  - ♦ Simple and easy to design
  - ♦ Structural independence
  - ♦ Supports data independence
  - ♦ Data access through queries is very fast and efficient.

- Disadvantages Relational model
  - ♦ Needs more powerful hardware and data storage

2.7.2 Object-based logical models

- ▪ Object-based logical models express data and its relationships.
- ▪ They provide fairly flexible structure.
- ▪ There are many different models more widely know ones are;

i. Entity-relationship model
ii. Object-oriented model
iii. Semantic model
iv. Functional data model

i. Entity-Relational model

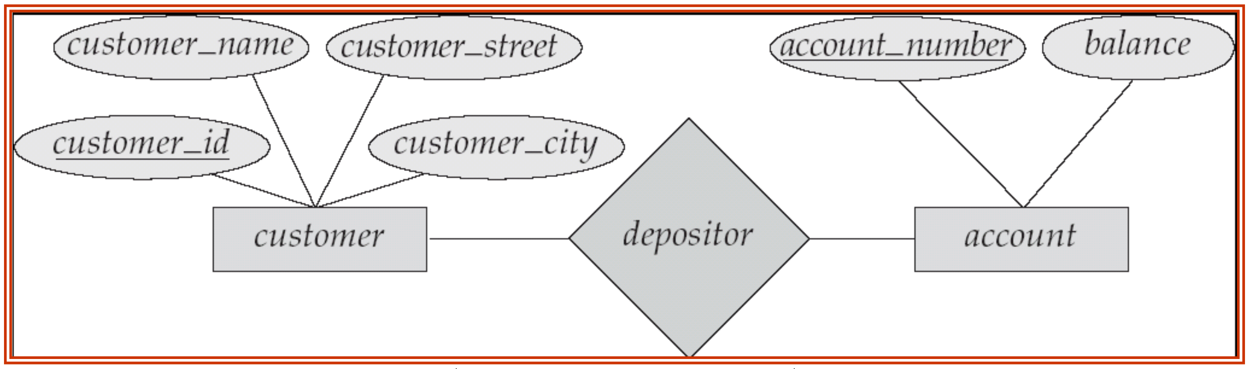
- ▪ This Models is based on real world entities and relationships
- ▪ Entity: a “thing” or “object” in real world

  Ex. each person is an entity, bank account is an entity.
- ▪ Attribute: properties of entities.

  Ex. the attributes account-number and balance describes one particular account
- ▪ Relationship: an association among several entities

  E.g. Depositor relationship associates a customer with each account
- ▪ Entity set and Relationship: The set of all entities of the same type are called entity set and the set of relationship of the same type are called the relationship set.

- Represented diagrammatically by an entity-relationship diagram:



(A sample E-R diagram)

- **Rectangles,** which represent entity sets

- **Ellipses,** which represent attributes

- **Diamonds,** which represent relationships among entity sets

- **Lines,** which link attributes to entity sets and entity sets to relationships

Each component is labelled with the entity or the relationship that it represents.

ii. Object- Oriented Data Model

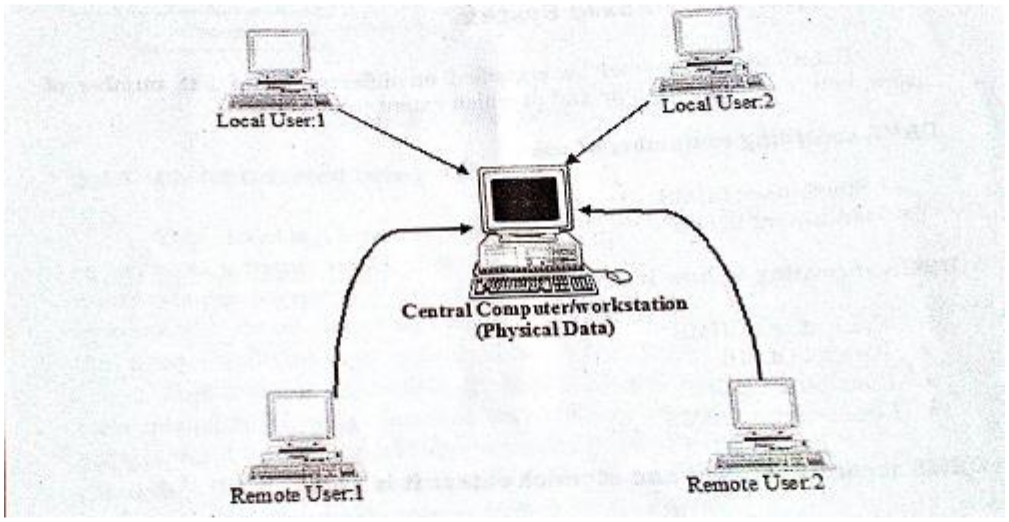
- This model is based on collection of objects.
- Object corresponds to an entity in the E-R model
- This model represents objects also known as entities, constraints on them and relationship among objects.
- Entity is represented by class with its attribute and behaviour.
- Object of class is instance of that class.
- Attributes of individual object has specific value.
- This model provides data encapsulation and abstraction
- Also hide data from outside world.

2.7.3 Physical data model:

- This model is used to describe the data at the lowest level.
- This model concerns about how data is stored at physical level.
- It describes storage structure and access methods of database.
- Two widely know ones are the *unifying model* and the *frame-memory* model.

2.8 Types of Database System

2.8.1 Centralized Database Systems



A Centralized Computer System

- Centralized Database Systems is one in which database is located , stored and maintained at single location.
- There is a powerful computer with advance processor and high storage capacity to store complete database.
- Different users can access this database from local machines or from remote machines.
- For example small company has database stored at one personal computer.

Advantages:

- Operations such as delete, update, insert, backup, query, recovery are easier.
- Data access is simple.
- Maintenance costs are less.
- Data integrity is good
- Data duplication is less.
- Data security is good.

Disadvantages

- When central computer on which whole database is stored crashes or goes down, everything is blocked.
- Central computer must be highly protected from unauthorized access.
- All terminals must have connection with the central computer and this incurs cost in networking.

2.8.2 Parallel Database Systems

- Parallel database systems database is kept in multiple disks.
- Also, multiple processors are used.
- Different database operations are performed in parallel.
- This improves processing and I/O speed.
- To implement parallel DBMS different architectures are used
    - **Shared memory**: multiple CPUs share main memory space
    - **Shared disk storage space**: Each node has its own memory, but all nodes share storage disk.
    - **Hybrid**: combination of all types.
    - **Independent resources:** each node has its own mass storage and main memory.

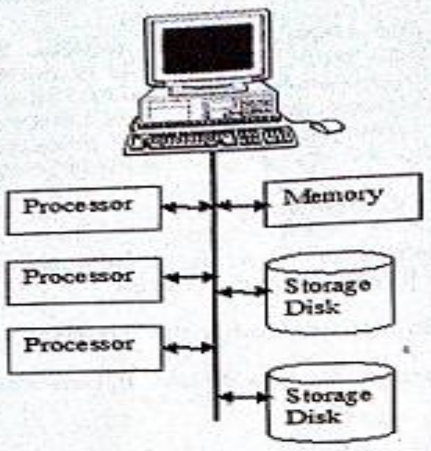


Figure 2.8 (a) Shared Memory

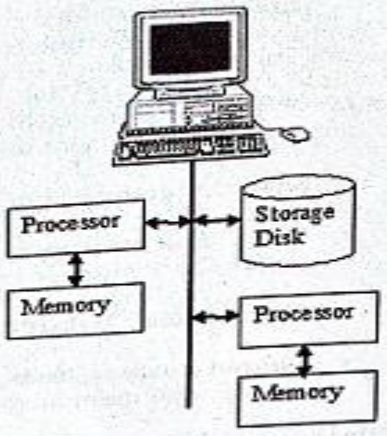


Figure 2.8 (b) Shared Storage Disk

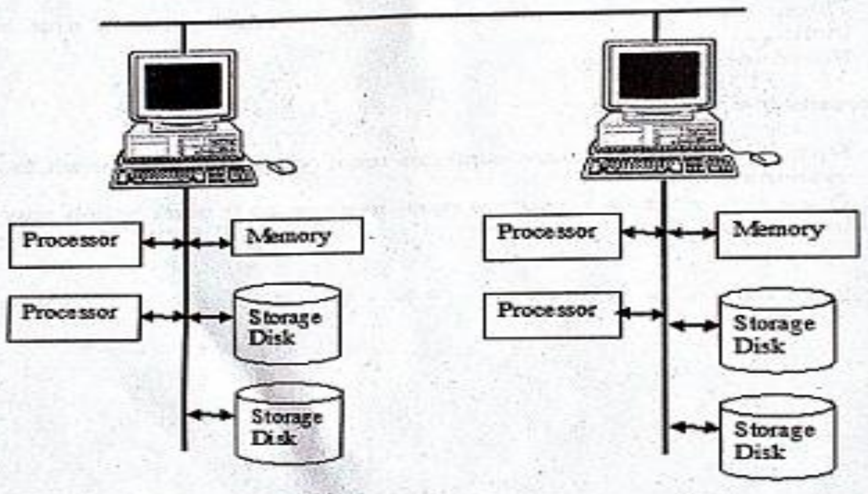


Figure 2.8 (c) Hierarchical

<u>Advantages</u>

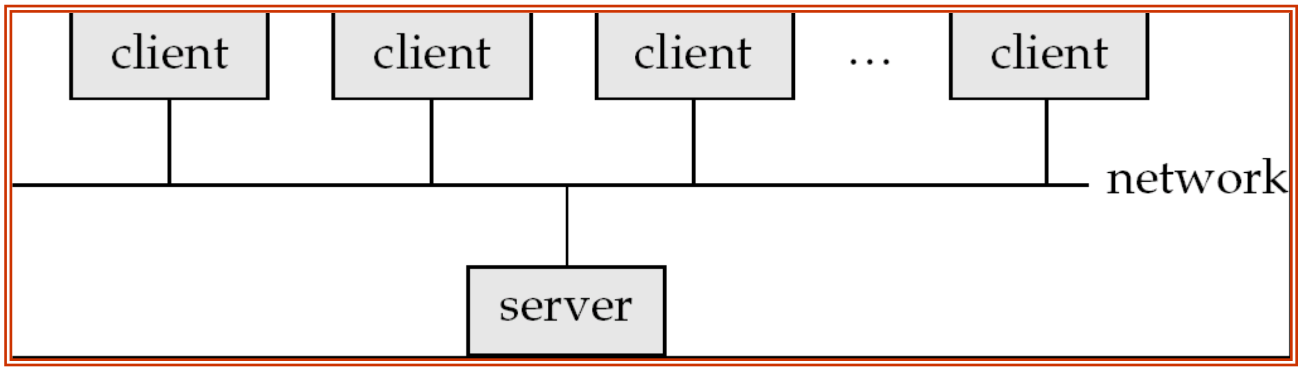
- Used when large number of queries to be processed in unit time
- Throughput, that is the number of tasks completed in unit time, is high.
- Response time is high.

Disadvantages

- Networking costs are very high
- Management of shared resources becomes very vomplex.

2.8.3 Client-Server Database Systems

- Client-server DBMS consists of *client, server* and *network.*
- Clients are personal computer or workstation.
- Server is powerful computer where DBMS software resides.
- Client and servers are connected via network.
- Clients send request to server using different SQL statements.
- Server performs operations requested by client and sends reply to client.
- Server systems satisfy requests generated at *m* client systems, whose general structure is shown below:



- Client is known as **Front-end**
- Server is known as **Back-end**

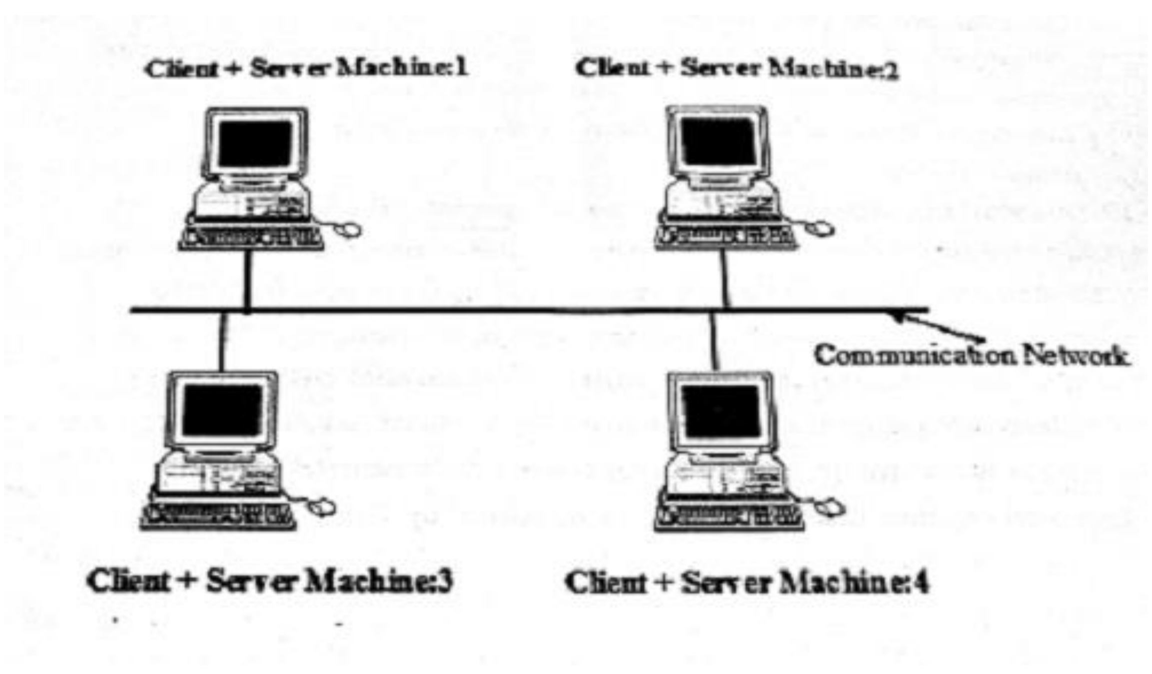
Advantages

- Provides more productivity and efficient use of data
- Less cost compared to other type of DBMS
- Applications are user friendly and attractive.
- Response time and throughput is high
- More flexible
- Database can be shared in efficient manner.

Disadvantages

- Client-server programming cost is high.
- Security needs to be implemented.
- Set up cost is high.

2.8.4 Distributed Database Systems


Client + Server Machine:1
Client + Server Machine:2
Communication Network
Client + Server Machine:3
Client + Server Machine:4

- Data spread over multiple machines (also referred to as **sites** or **nodes**).
- Network interconnects the machines
- Data shared by users on multiple machines
- Each machine can act as server as well as client.
- Each machine can have data and applications of its own.
- Homogeneous distributed databases
  - Same software/schema on all sites, data may be partitioned among sites
  - Goal: provide a view of a single database, hiding details of distribution
- Heterogeneous distributed databases
  - Different software/schema on different sites
  - Goal: integrate existing databases to provide useful functionality

Advantages

- Proved greater efficiency and performance
- Provide high throughput and better response time
- Single database can be shared across several distinct client machines.
- Provide local autonomy
- Distribution of data is transparent.
- When user add new machines to store apart of database ,it doesnot have any effect on performance of system.

Disadvantage
- Added complexity required to ensure proper coordination among sites.
- Software development cost.
- Recovery from failure is more complex.