18CSC303J- DATABASE MANAGEMENT SYSTEMS

UNIT-I

What is Database Management System- Advantage of DBMS over File Processing System - Introduction and applications of DBMS- Purpose of database system- Views of data- Database system Architecture- Data Independence- The evolution of Data Models- Degrees of Data Abstraction- Database Users and DBA- Database Languages

1.1 WHAT IS DATABASE MANAGEMENT SYSTEMS?

- Collection of interrelated data
- Set of programs to access the data
- DMBS contains information about a particular enterprise
- DBMS provides an environment that it both convenient and efficient to use

1.2 Disadvantage of File-oriented system:

1. Data Redundancy:

It is possible that the same information may be duplicated in different files. this leads to data redundancy results in memory wastage.

2. Data Inconsistency:

Because of data redundancy, it is possible that data may not be in consistent state.

3. Difficulty in Accessing Data:

Accessing data is not convenient and efficient in file processing system.

4. Limited Data Sharing:

Data are scattered in various files.also different files may have different formats and these files may be stored in different folders may be of different departments.

So, due to this data isolation, it is difficult to share data among different applications.

5. Integrity Problems:

Data integrity means that the data contained in the database in both correct and consistent.for this purpose the data stored in database must satisfy correct and constraints.

6. Atomicity Problems:

Any operation on database must be atomic.
this means, it must happen in its **entirely** or not at all.

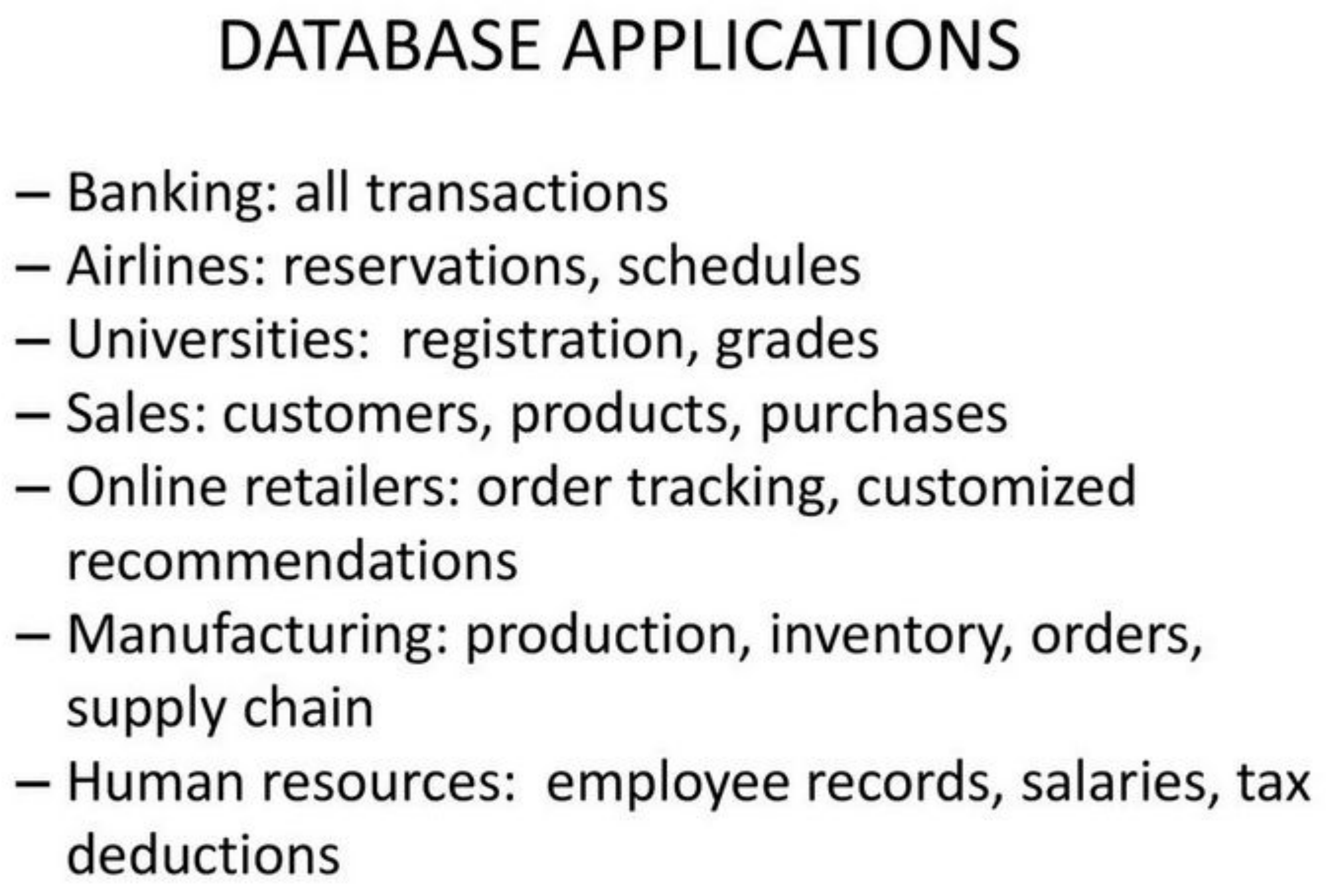
7. Concurrent Access Anomalies:

Multiple users are allowed to access data simultaneously. this is for the sake of better performance and faster response.

8. Security Problems:

Database should be accessible to users in limited way.
Each user should be allowed to access data concerning his requirements only.

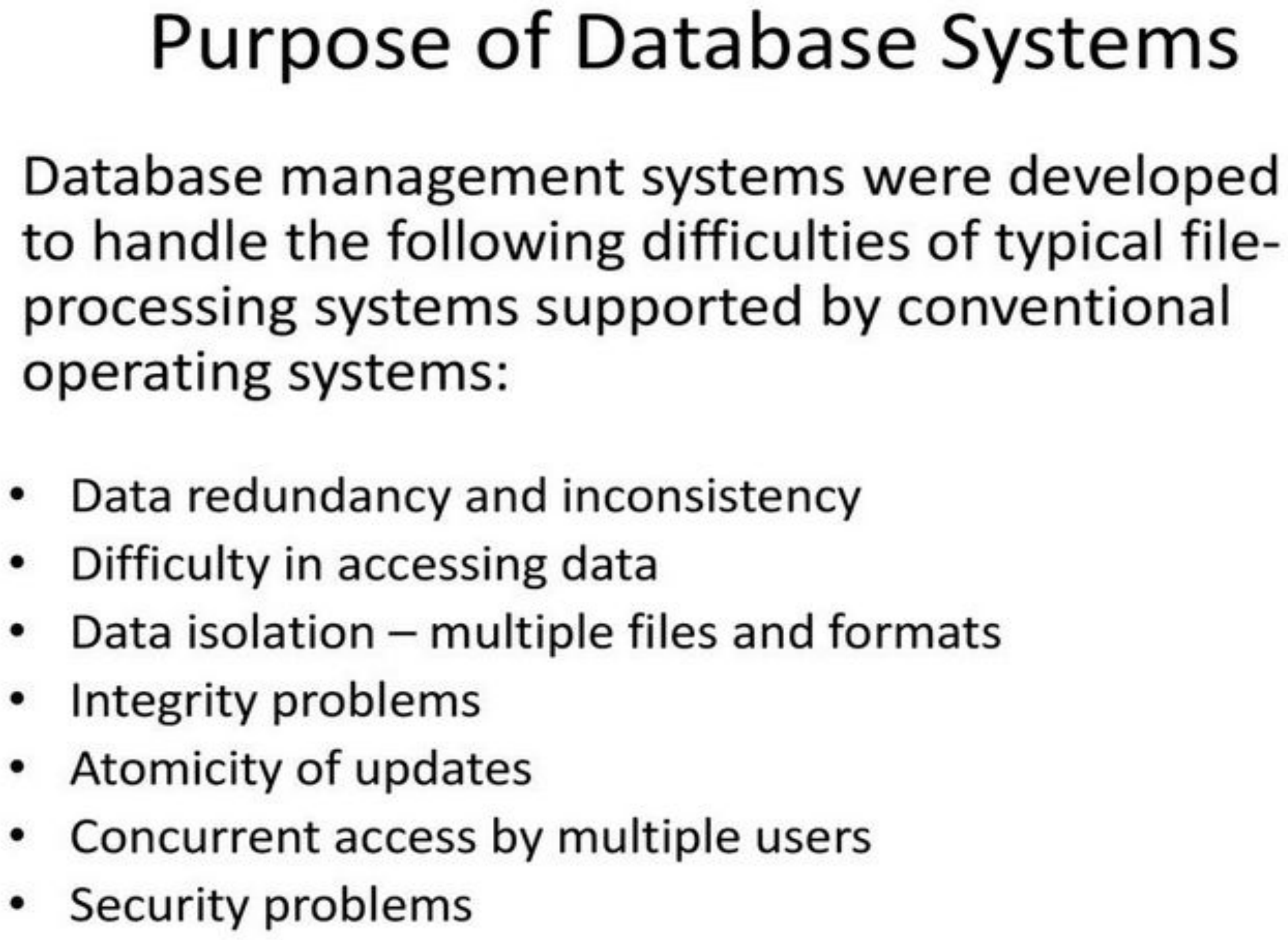
1.3 Database Applications:



DATABASE APPLICATIONS

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

1.4 Purpose of Database Systems



Purpose of Database Systems

Database management systems were developed to handle the following difficulties of typical file-processing systems supported by conventional operating systems:

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation – multiple files and formats
- Integrity problems
- Atomicity of updates
- Concurrent access by multiple users
- Security problems

Purpose of Database Systems

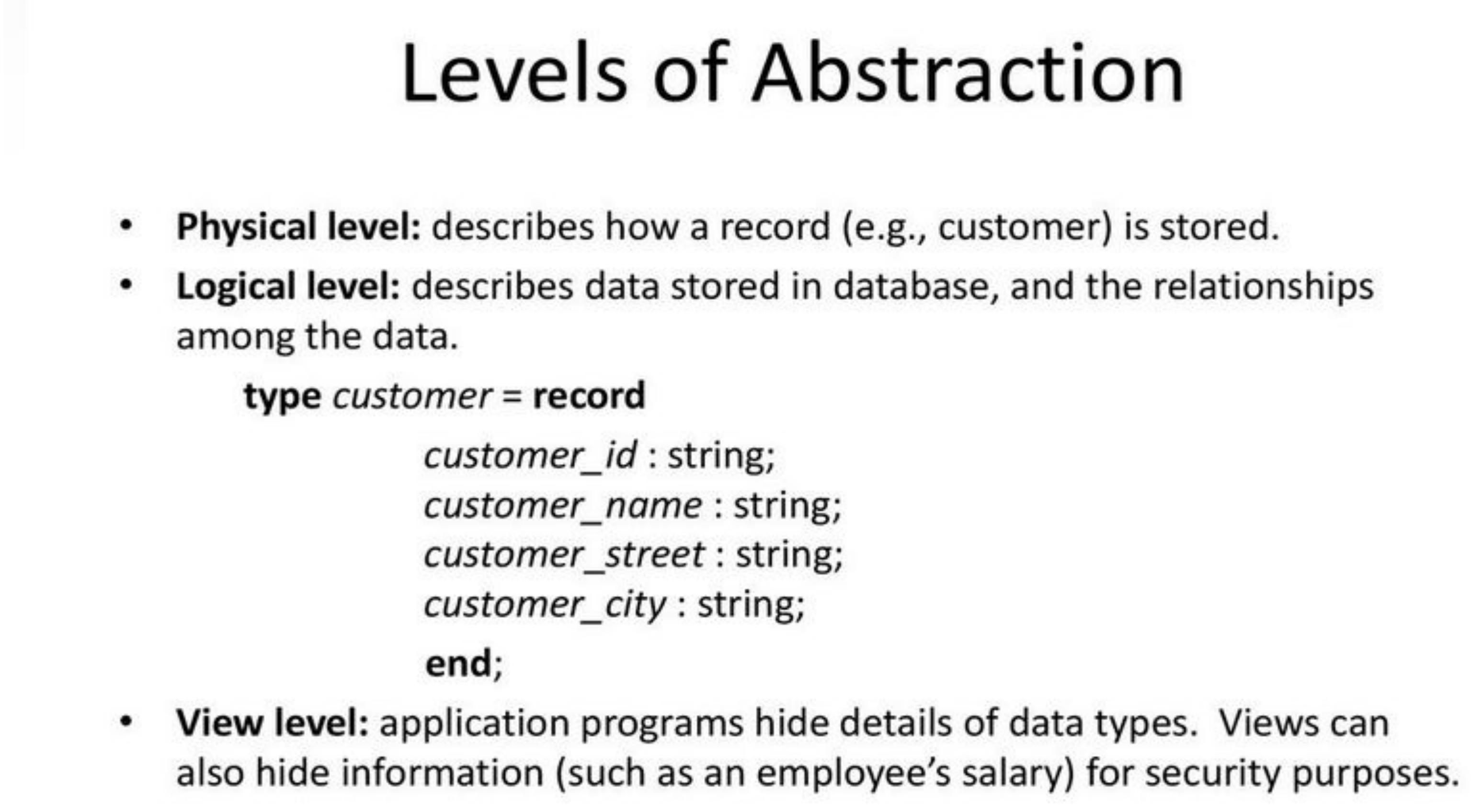
- In the early days, database applications were built directly on top of file systems
- Drawbacks of using file systems to store data:
  - Data redundancy and inconsistency
    - Multiple file formats, duplication of information in different files
  - Difficulty in accessing data
    - Need to write a new program to carry out each new task
  - Data isolation — multiple files and formats
  - Integrity problems
    - Integrity constraints (e.g. account balance > 0) become "buried" in program code rather than being stated explicitly
    - Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

- Drawbacks of using file systems (cont.)
  - Atomicity of updates
    - Failures may leave database in an inconsistent state with partial updates carried out
    - Example: Transfer of funds from one account to another should either complete or not happen at all
  - Concurrent access by multiple users
    - Concurrent accessed needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
      - Example: Two people reading a balance and updating it at the same time
  - Security problems
    - Hard to provide user access to some, but not all, data
- Database systems offer solutions to all the above problems

1.5 Views of Data:

1.5.1 Levels of Abstraction:



Levels of Abstraction

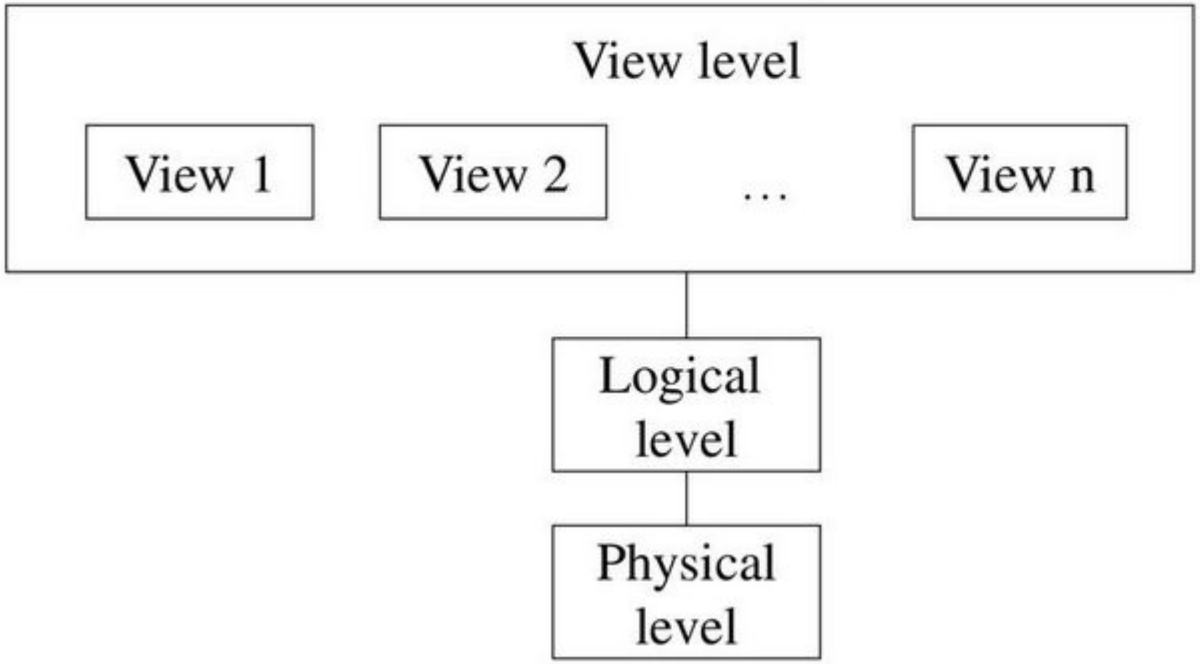
- **Physical level:** describes how a record (e.g., customer) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type customer = record
        customer_id : string;
        customer_name : string;
        customer_street : string;
        customer_city : string;
        end;
```

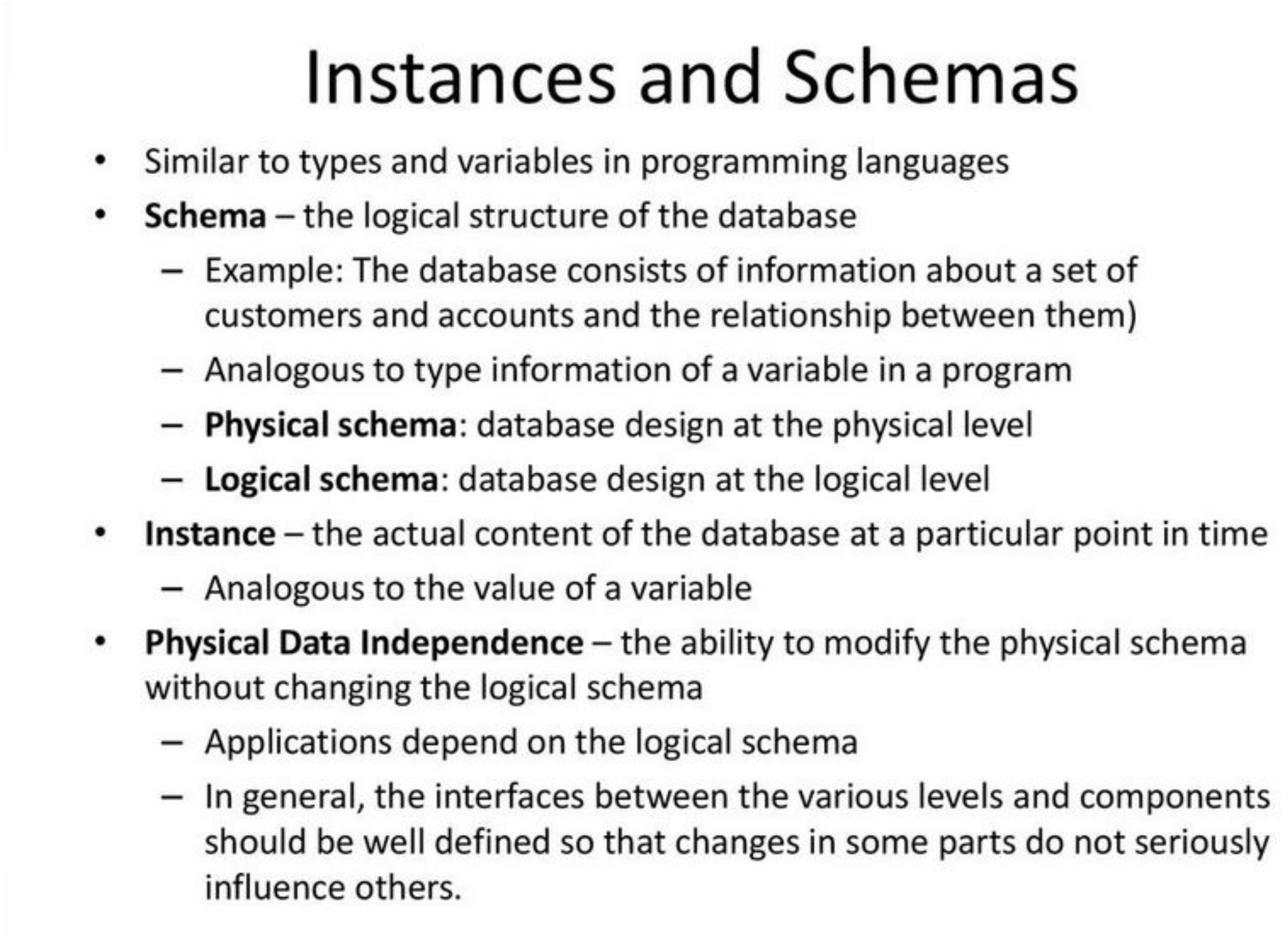
- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

Views of Data

An architecture for a database system



1.5.2 Instances and Schema:



Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them)
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

1.6 DATABASE ARCHITECTURE

- A Database Architecture is **a representation of DBMS design**. It helps to design, develop, implement, and maintain the database management system.
- A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered.
- The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs.
- The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- DBMS architecture depends upon how users are connected to the database to get their request done.

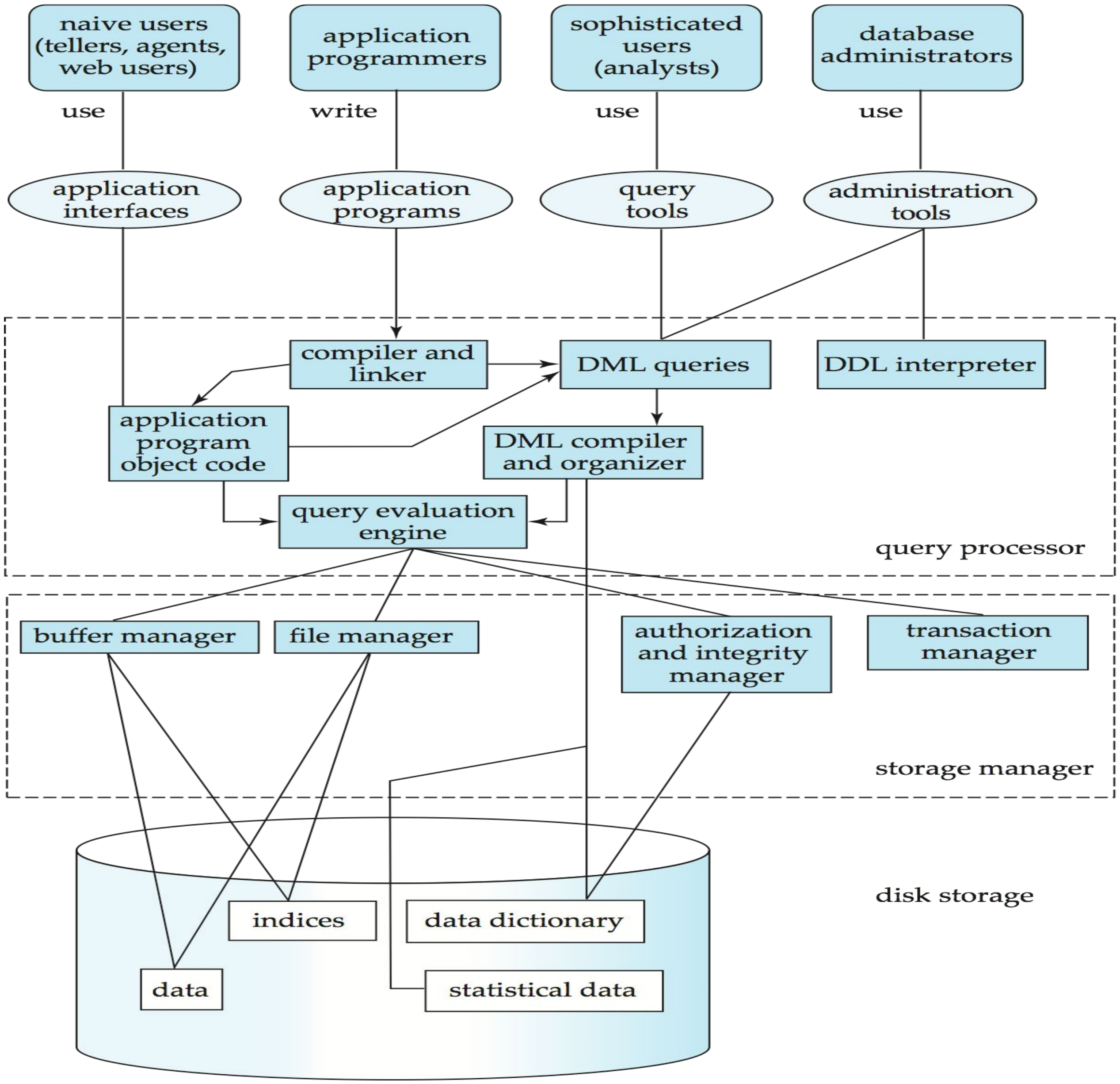


Fig: Database system Architecture

1.6.1 Storage Manager

•Storage Manager is a program module that provides an interface between the data stored in the database and the application programs and queries submitted to the system. It is also known as Database Control System.
•It is responsible for the interaction with the file manager.
•Translates DML statements into low-level file-system commands.
•It is responsible for updating, storing, deleting, and retrieving data in the database.

It contains the following components –

- **Authorization Manager**
- **Integrity Manager**
- **Transaction Manager**
- **File Manager**
- **Buffer Manager**

- **Authorization Manager –**
  It ensures role-based access control, i.e,. checks whether the particular person is privileged to perform the requested operation or not.

- **Integrity Manager –**
  It checks the integrity constraints when the database is modified.

- **Transaction Manager –**
  It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.

- **File Manager –**
  It manages the file space and the data structure used to represent information in the database.

- **Buffer Manager –**
  It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

1.6.2 Disk Storage :

❖ It contains the following components –

❖Data Files

❖Data Dictionary

❖Indices

•**Data Files –**
which store the database itself.

•**Data Dictionary –**
It conatins meta data that is data about data. The schema of a table is an example of meta data. A database system consults the data dictionary before reading and modifying actual data.

•**Indices –**
which provides fast access to data limits that hold particular values.

1.6.3 Query Processor :

It interprets the requests (queries) received from end user via an application program into instructions. It also executes the user request which is received from the DML compiler.

Query Processor contains the following components –

DML Compiler –

It processes the DML statements into low level instruction (machine language), so that they can be executed.

Select * from student;(address,aadhar no)

DDL Interpreter –

It processes the DDL statements into a set of table containing meta data (data about data).

Embedded DML Pre-compiler –

It processes DML statements embedded in an application program into procedural calls.

Query Optimizer –

It executes the instruction generated by DML Compiler.

1.6.4 Database Users and Administrators

The primary goal of DBMS - to **retrieve the information from the database & store new information** into the database.

People who **work with a database can be categorized into:**

- **Database Users and**
- **Data base administrators**

1.6.4.1 Database Users and User Interfaces:

4 types of database-system users, depending upon the way they interact with system

(1) Naıve Users:
unsophisticated users
interact with system by invoke one of the permanent application programs that have been written previously
For example, a **clerk** in university needs to **add new instructor** to department A **invokes a program** called **new _hire** and enter the required data.

(2)Application programmers:

- **Computer professionals** who **write** application programs.
- Application programmers can choose from many tools to develop user interfaces.
- Rapid application development (RAD) tools are the tools that enable an application programmer to construct forms and reports with minimal programming effort.

(3)Sophisticated Users:

- Interact with the system without writing programs.
- Instead they form their request either using a database query language or using tools such as data analysis software.
- Analysts who submit queries to explore the data in the database falls in this category.

(4)Specialized users:

- Write specialized database applications that do not fit into the traditional data processing framework.
- Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store with complex data types (for example, graphics and audio data) and environment modeling systems.

1.6.4.2 Database Administrator(DBA):

The person who has central control over the system is called **Database Administrator(DBA):**

The functions of DBA includes:

- **Schema definition :** The DBA creates a original database schema by executing a set of data definition statements in **DDL.**

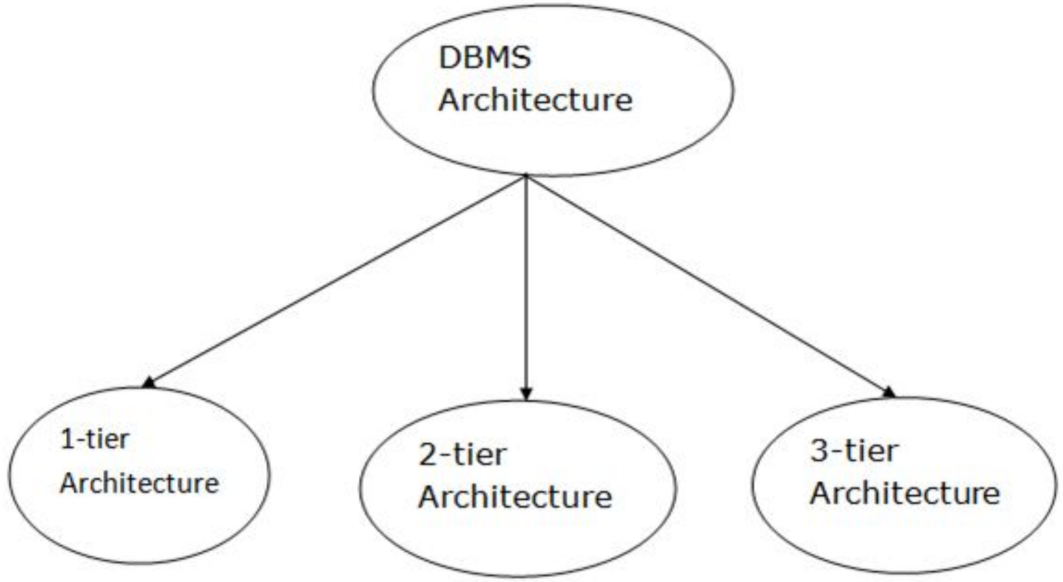
- **Storage structure and access-method definition**

- **Schema and physical-organization modification**: The DBA carries out the changes to the schema.
- **Granting of authorization for data access:** By granting different types of authorization, the DBA can regulate different users accessing different parts of database.

- **Routine maintenance:** Examples of DBA routine maintenance activities are:
  - ***Periodically backing up the database***
  - ***Ensuring enough free disk space***
  - ***Monitoring jobs running on the database***

Types of DBMS Architecture

Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.



1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

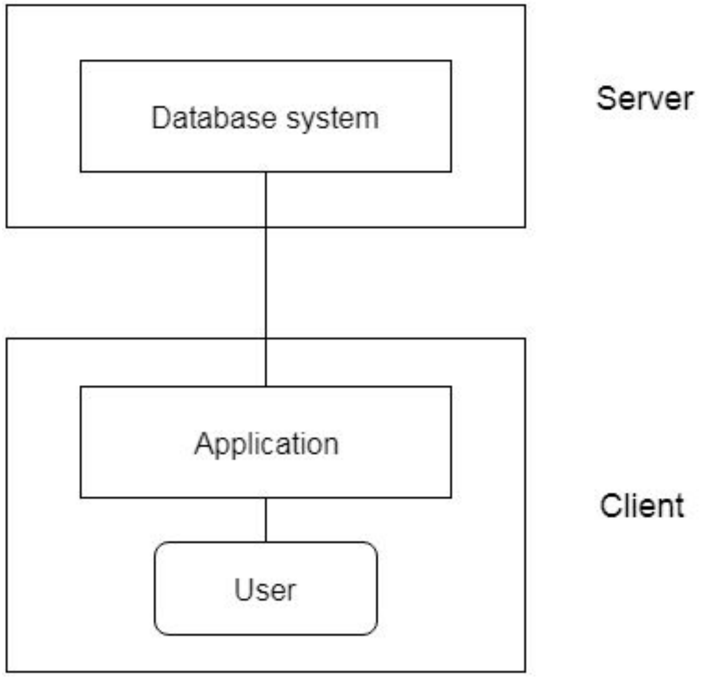


Fig: 2-tier Architecture

3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

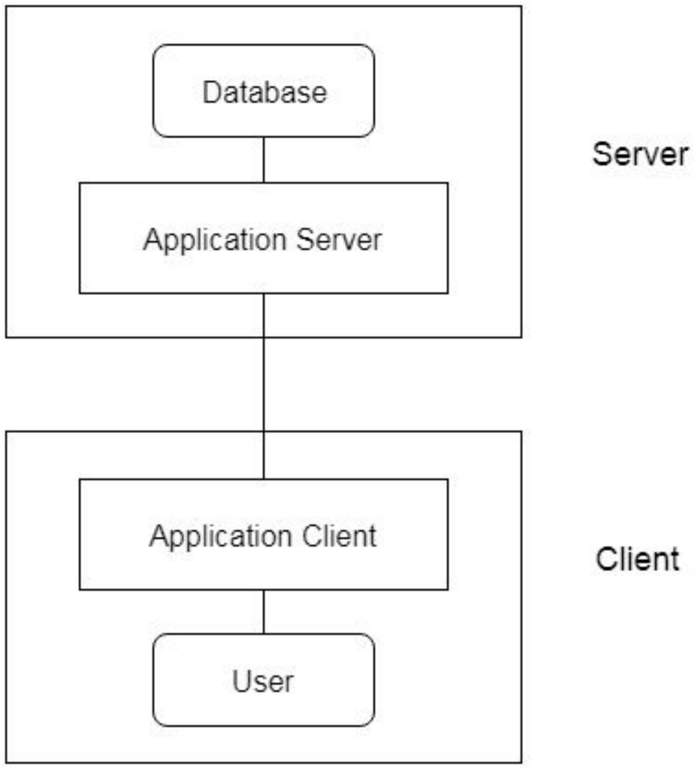


Fig: 3-tier Architecture

Data Independence

- Ability to modify a schema definition in one level without affecting a schema definition in the other levels.
- The interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
- Two levels of data independence
  - Physical data independence
  - Logical data independence

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:

1.7.1 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.

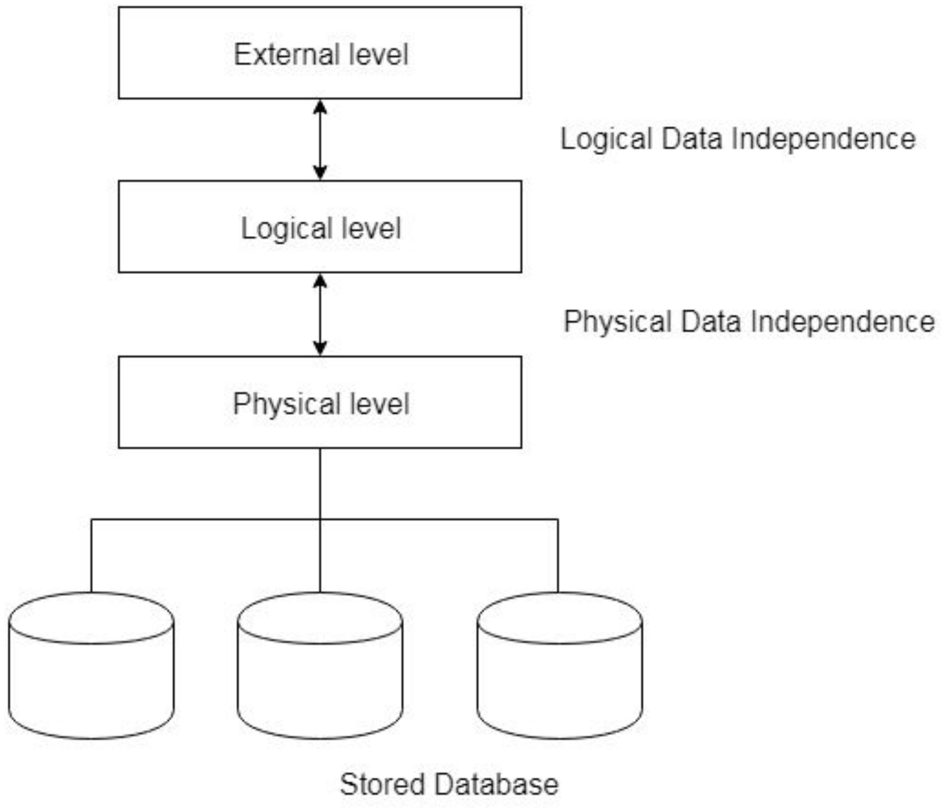
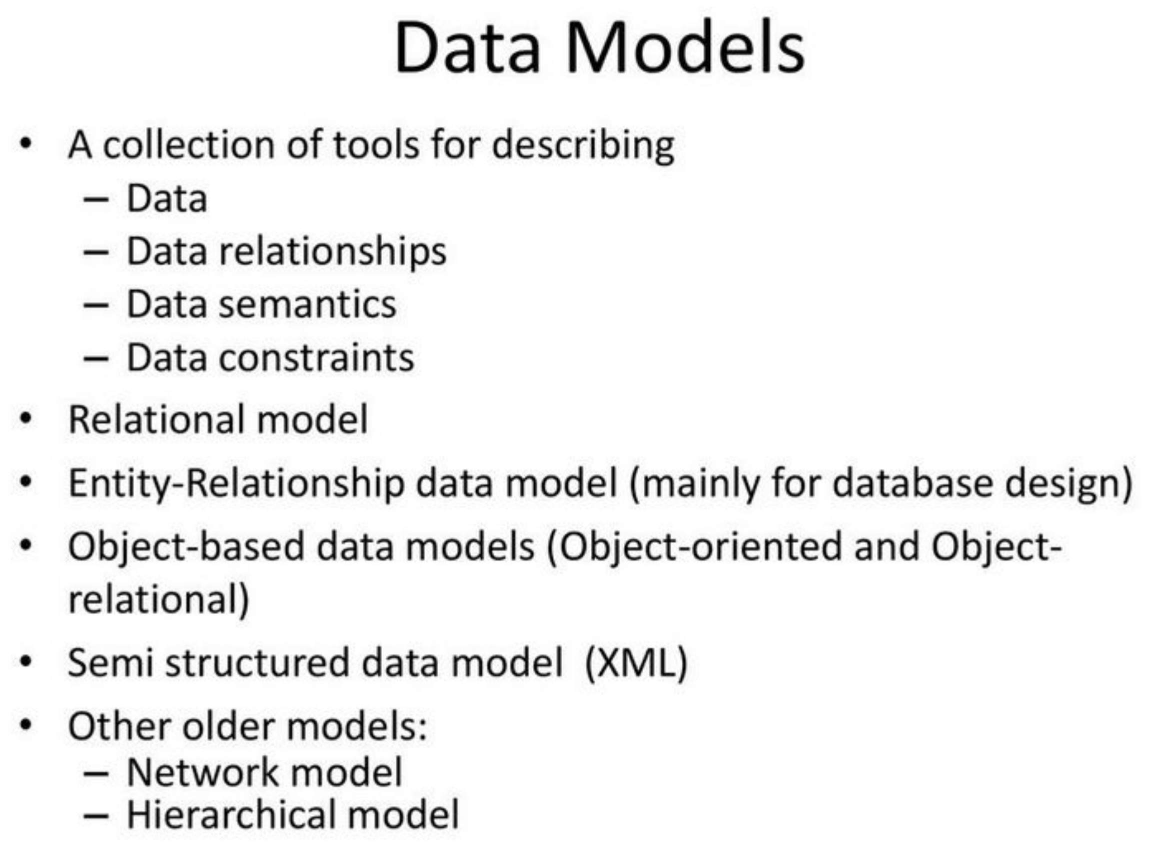


Fig: Data Independence

1.8 Data Models

- Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

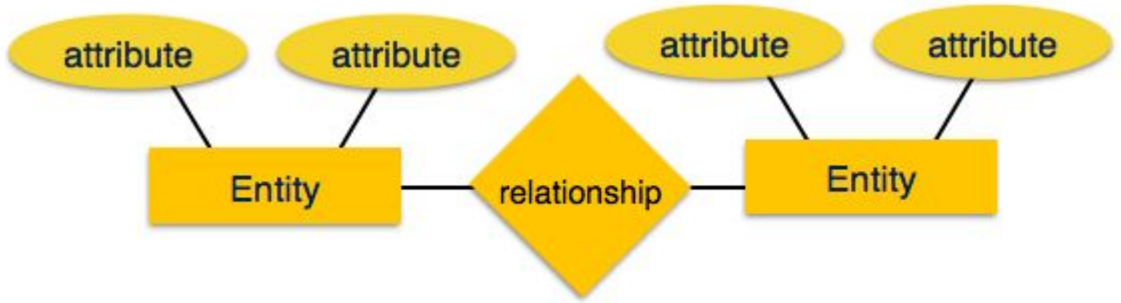


Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

1.8.1 Entity-Relationship Model

- Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.
- ER Model is best used for the conceptual design of a database.
- ER Model is based on –
    - **Entities** and their *attributes.*
    - **Relationships** among entities.
- These concepts are explained below.



Entity − An entity in an ER Model is a “real-world objects” or “thing” having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**.

For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

Relationship − The logical association among entities is called ***relationship***. The overall logical structure of a database can be expressed graphically by an E-R diagram, which consists of following components:

Rectangles: represents entity set

Ellipses : represents attributes

Diamonds: represents relationships among entity sets

Lines: links attributes to entity sets and entity sets to relationships

Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Mapping cardinalities −

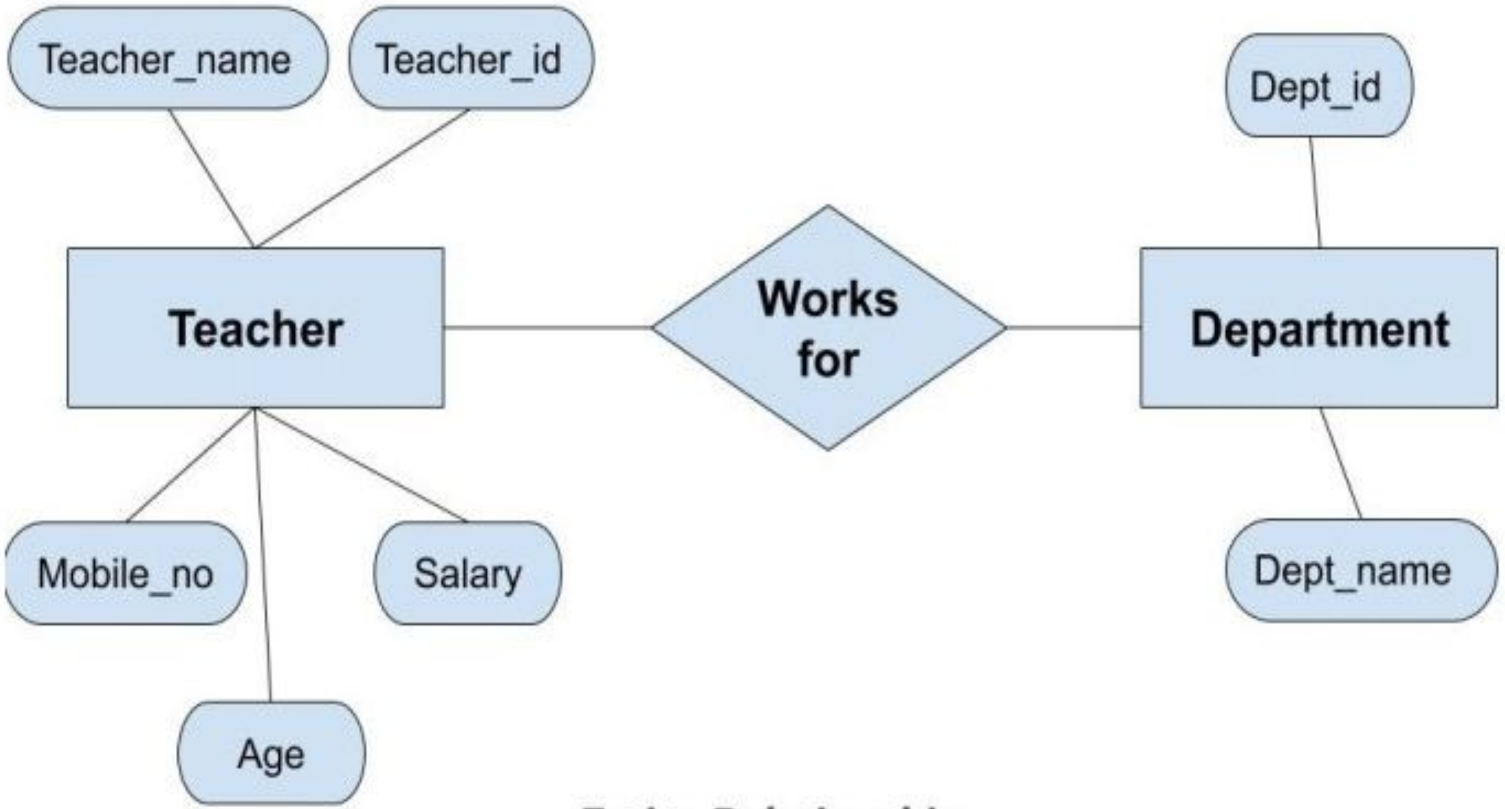
one to one

one to many

many to one

many to many

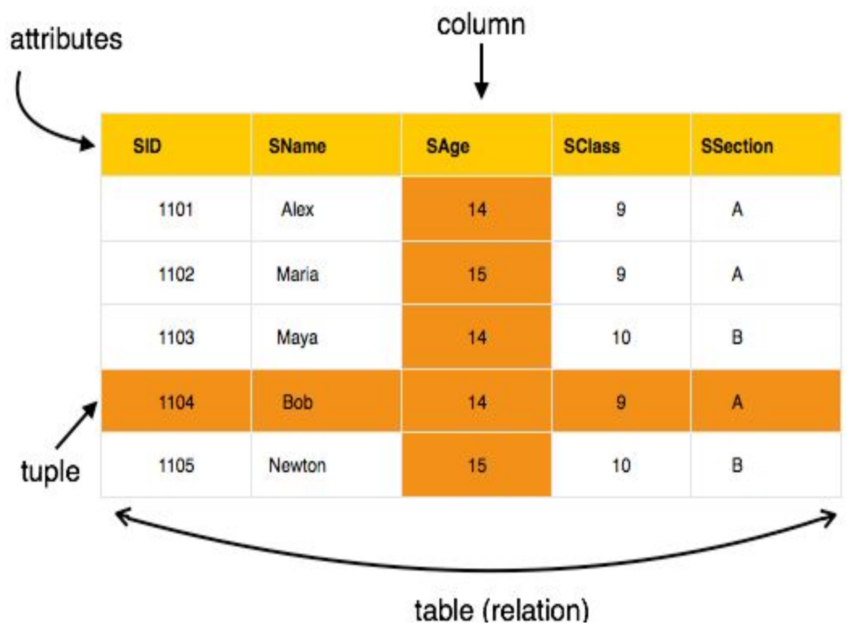
Example:



Entity-Relationship Model

1.8.2 Relational Model

- The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an **n-ary relation**.
- This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations.



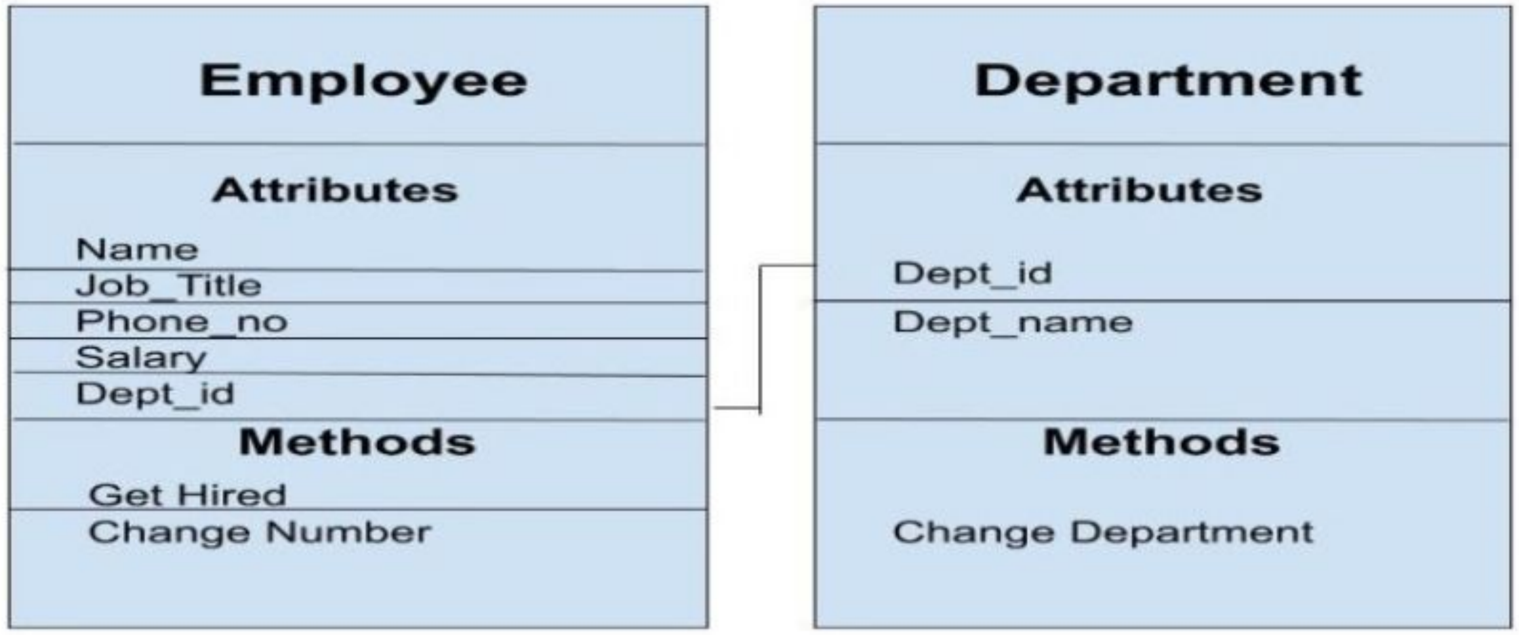
SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

The main highlights of this model are −

- Data is stored in tables called **relations**.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

1.8.3 Object-oriented model

- The real-world problems are more closely represented through the object-oriented data model.
- In this model, both the data and relationship are present in a single structure known as an object.
- In this model, two are more objects are connected through links. We use this link to relate one object to other objects.
- This can be understood by the example given below.



Object_Oriented_Model

1.8.4. Semistructured Data Model:

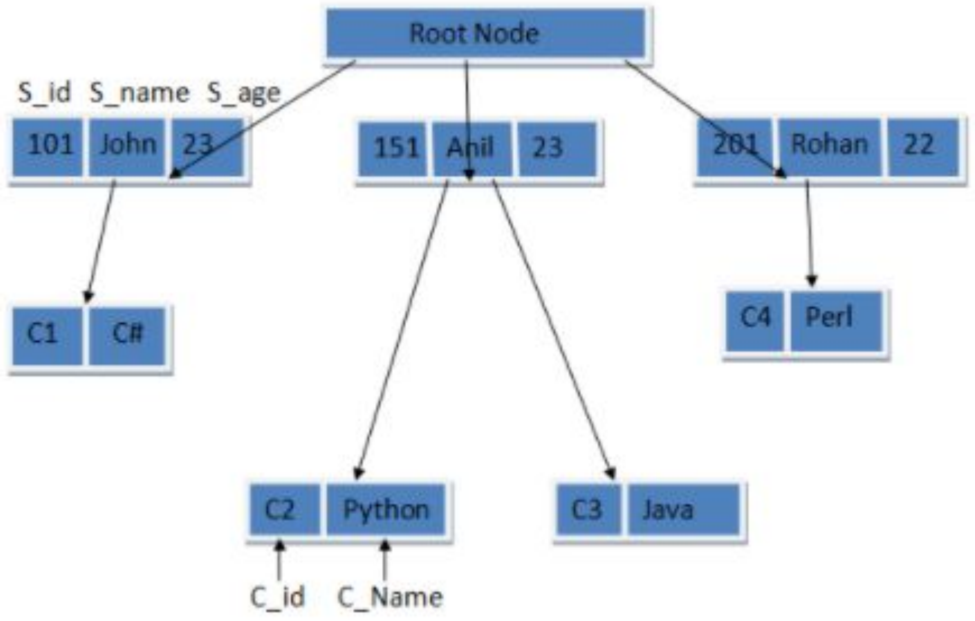
- This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets.

- The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data.

- Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

1.8.5. Hierarchical database model

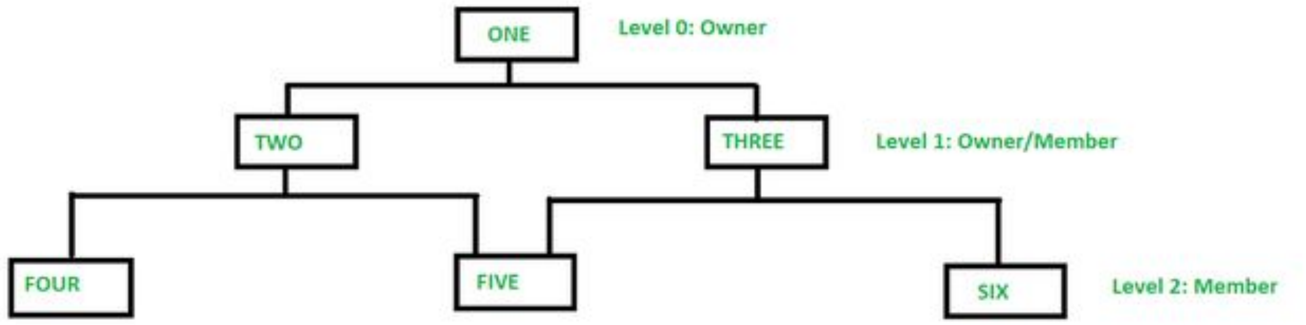
A **hierarchical database model** is a data model in which the data are organized into a tree-like structure. The data are stored as **records** which are connected to one another through **links**.
A record is a collection of fields, with each field containing only one value.
The **type** of a record defines which fields the record contains.



1.8.6. Network model

- This model is the generalization of the hierarchical model.
- This model can consist of multiple parent segments and these segments are grouped as levels but there exists a logical association between the segments belonging to any level.
- Mostly, there exists a many-to-many logical association between any of the two segments.
- We called **graphs** the logical associations between the segments.
- Therefore, this model replaces the hierarchical tree with a graph-like structure, and with that, there can more general connections among different nodes.

Structure of a Network Model :



Database Languages

Data Definition Language

- Specification notation for defining the database schema
- DDL compiler generates a set of tables stored in a data dictionary
- Data dictionary contains ***metadata*** (data about data)
- Data storage and definition language – special type of DDL in which the storage structure and access methods used by the database system are specified

Data Definition Language-DDL

- **Data Definition Language (DDL)** statements are used to define the database structure or schema.
  Some examples:
- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
  TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data

Data Manipulation Language[DML]:

- used for **accessing and manipulating data** in database.
- Handles **user requests.**

Commands:

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

Data Control Language[DCL]:

- is used to access the stored data.
- DCL execution is transactional.
- It also has rollback parameters.

Commands:

Grant: used to give user access privileges to database.

Revoke: used to take back permissions from user.

operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

Transaction Control (TCL)

- **Transaction Control** (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

  Some examples:

  COMMIT – save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use