TYPES OF VIEWS

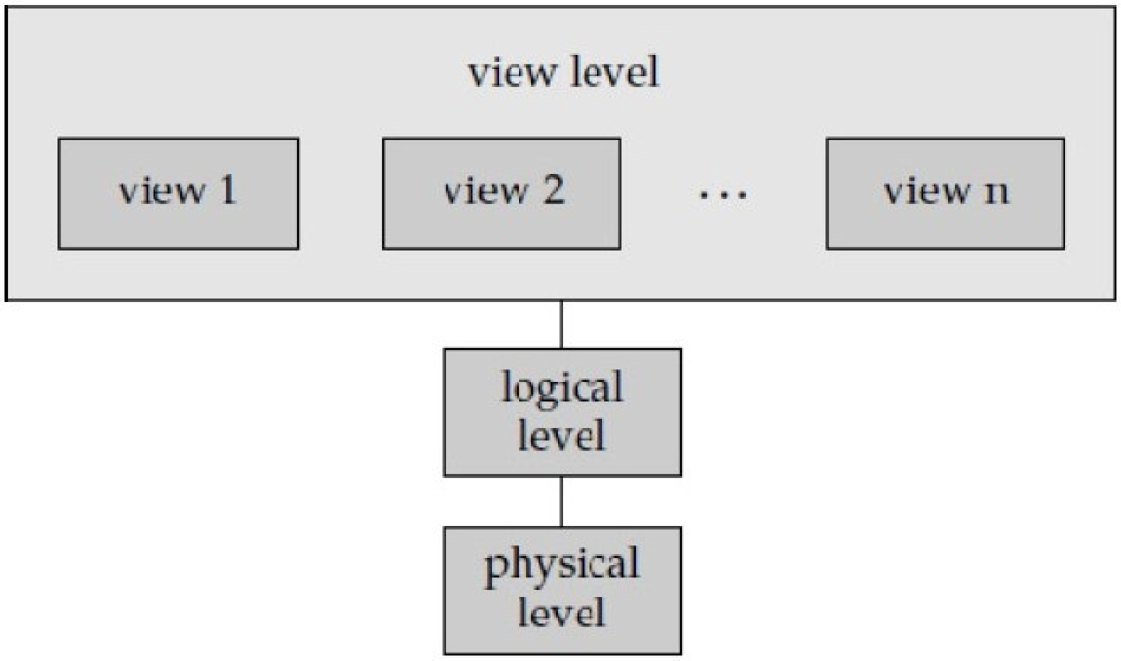
Views of Data

- Database is a collection of interrelated data and set of programs that allow users to access and modify data.
- The major purpose of a database system is to provide users with an **abstract view** of the system.
- The system hides certain details of how data is stored and created and maintained
- Complexity should be hidden from database users.

a) Data Abstraction

There are several levels of abstraction:

1. **Physical Level:**
   - Lowest level of abstraction.
   - How the data are stored.
   - E.g. index, B-tree, hashing.
   - Complex low-level structures described in detail.
2. Conceptual Level:
   - Next highest level of abstraction.
   - Describes *what* data are stored.
   - Describes the relationships among data.
   - Database administrator level.
3. View Level:
   - Highest level.
   - Describes *part* of the database for a particular group of users.
   - Can be many different views of a database.
   - E.g. tellers in a bank get a view of customer accounts, but not of payroll data.



The three levels of data abstraction

Example

- **Customer,** with field cust_id, cust_name, cust_street, cust_city
- **account,** with field acc_num, balance
- **employee,** with field emp_name, salary

At Physical Level,

- Customer, account, employee record can be described as block of consecutive storage locations (word, byte etc.,)

At Logical level,

- Each record will be described by the type definition.

```
type customer = record
                    cust_id : string
                    cust_name : string
                    cust_street : string
                    cust_city : string
                end;
```

At View level

- Applications that hide details of data types.

b) Instance and Schema

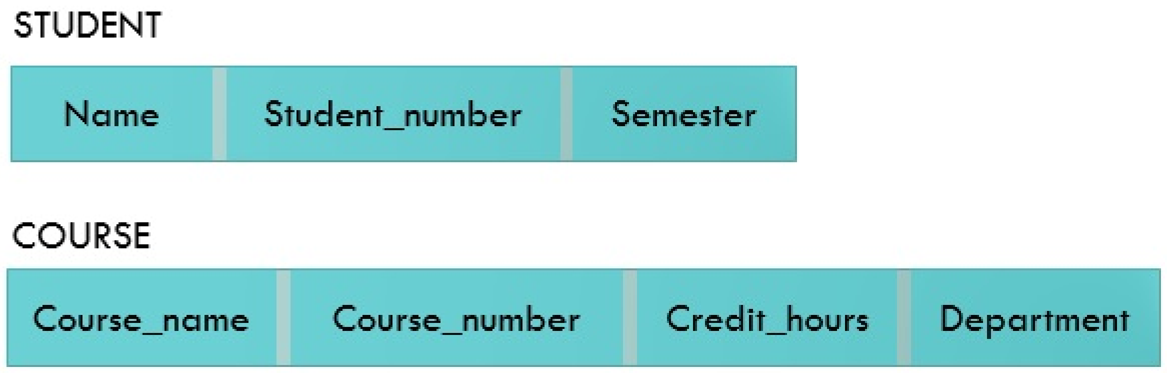
Instance

- Collection of information stored in the database at a particular moment of time is called as instance
- Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.
- For example,
    - Let’s say we have a single table student in the database, today the table has 100 records,
    - so today the instance of the database has 100 records.
    - Let’s say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table.
    - In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

Schema

- Overall design of the database is called as Schema.
- Description of a database is called as the database schema, which is specified during database design and is not expected to change frequently.
- Database systems have several schema, partitioned according to the levels of data abstraction.
    - Physical Schema
    - Logical Schema
- A **schema** is the complete design of database it is also known as **intension**.
- It is the collection of named objects.
- The names of tables, columns of each table, datatype, triggers, functions views packages and other objects are included in the schema.
- The changes in a schema are not applied so frequently, but occasionally changes need to be applied as the requirements of application changes.
- The schema modification or alteration is known as **schema revolution**.

Example of Schema



The database system has various schemata separated according to the levels of abstraction such as physical, logical and external/subschema. Generally, DBMS assist one physical one logical and several sub-schemata.

- The **physical schema** is the lowest level of a schema which describes how the data stored on the disk or the physical storage.
- The **logical schema** is the intermediate level of a schema which describes the structure of the database to the database designers. It also specifies what relationship exists between the data.
- The **external schema** or **subschema** is the highest level of a schema which defines the views for the end users.

Basic Comparison of schema and instance

BASIS FOR COMPARISON	SCHEMA	INSTANCE
Basic	Description of the database.	Snapshot of a database at a specific moment.
Change occurrence	Rare	Frequent
Initial state	Empty	Always have some data.

Data Model

- **Data models** are a collection of conceptual tools for
  - describing data,
  - data relationships,
  - data semantics and
  - data constraints.
- Data model provides a way to describe the design of a database at physical, logical and view level.

Types of database models

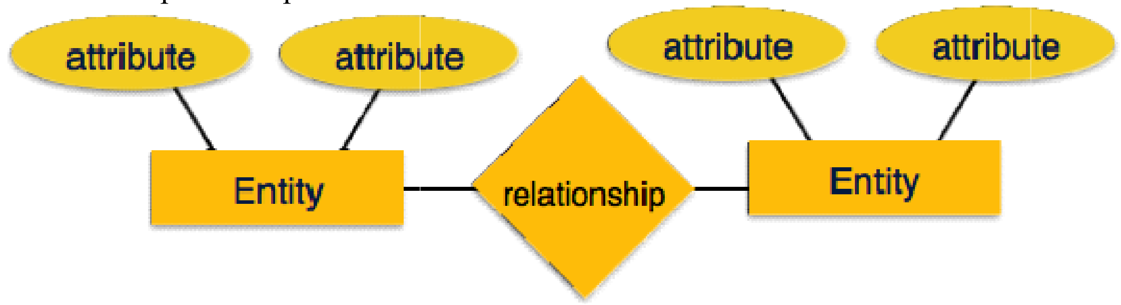
There are many kinds of data models.

a) Entity relationship model
b) Relational model
c) Hierarchical database model
d) Network model
e) Object-oriented database model

a) Entity relationship model

- Entity relationship ER model consists of collection of basic object called entities and relationship among entities.
- ENTITY
    - An entity is a thing or object in the real world that is distinguishable from other objects
    - Example: person, bank_account is an entity.
- RELATIONSHIP
    - A relationship is an association among several entities.
    - Example: depositor → associates customer with account.

These concepts are explained below.



Advantage

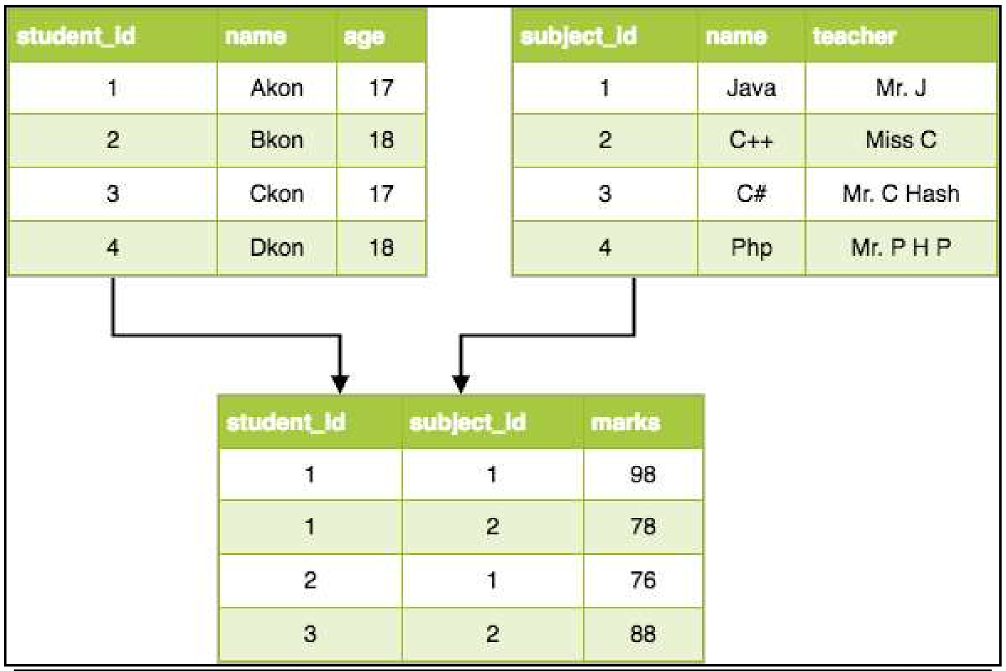
- It is easy to develop relational model using ER model.
- ER model specifies mapping cordialities
- Can specify generalization and specialization using ER diagram

Disadvantage

- It is just used for design
- Not for implementation.

b) Relational model

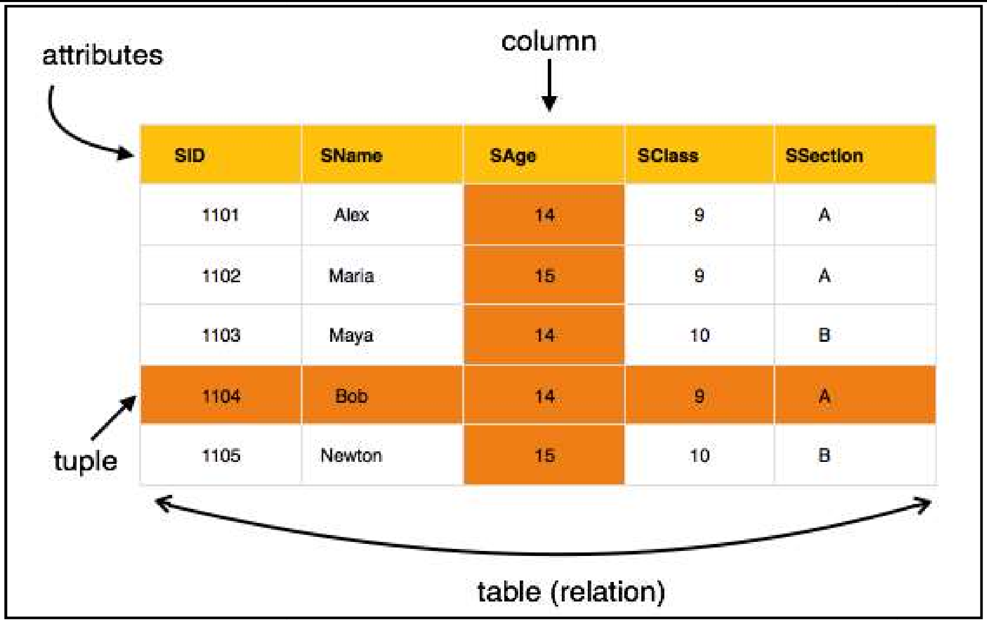
- Relational model uses a **collection of tables to represent both data and relationship among those data.**
- Each table has **multiple columns** and each column has a **unique name**.
- Relational data model is the **most widely used model** and vast majority of the current database are based on the relational database model.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88



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

Advantage

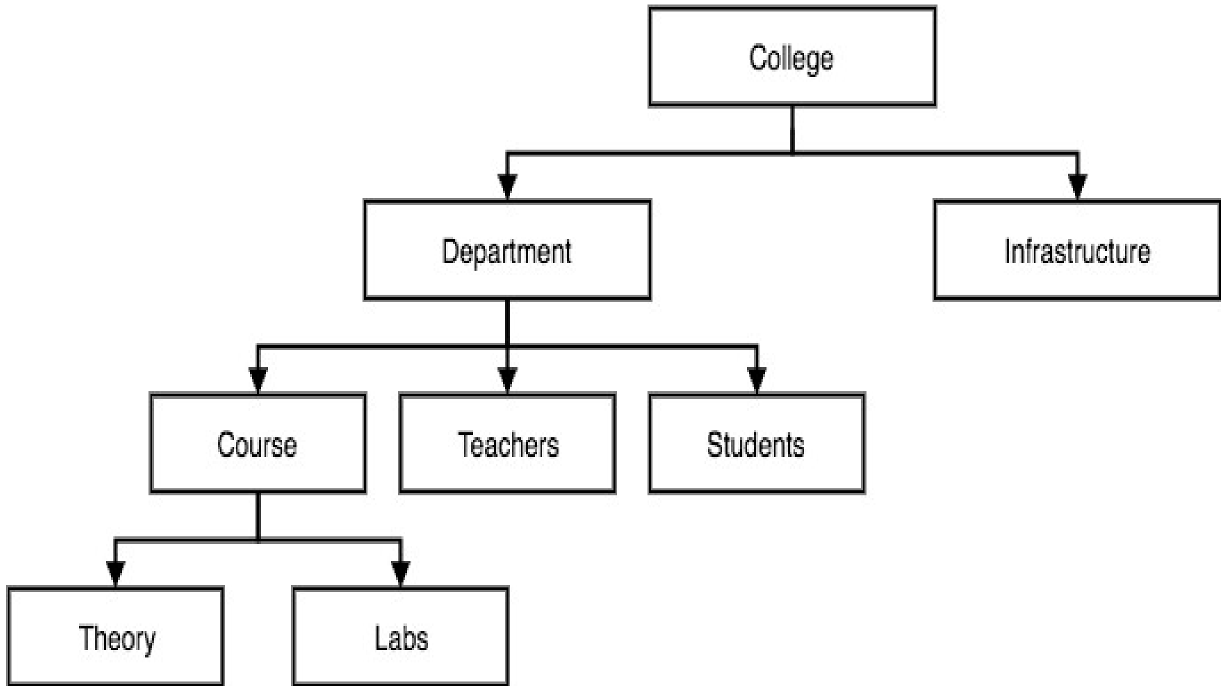
- Structural independencies → change in database structure does not affect data & data access.
- Conceptual simplicity → this model is simple at conceptual level, make designer to concentrate on the logical view of data.
- Greater flexibility & simple to navigate

Disadvantage

- Overhead
- Have slower processing time
- Not as good for transaction process as hierarchical and network model.

c) Hierarchical model

- This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked.
- The heirarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
- In hierarchical model, data is organised into tree-like structure with one
  - one-to-many relationship between two different types of data,
  - for example, one department can have many courses, many professors and of-course many students.



Advantage

- Easy to access
- High speed access to large datasets
- Simplicity due to hierarchical design
- Data security → this is the first model that offer data security.
- Efficiency → This model is very efficient which contain 1: n relationship.

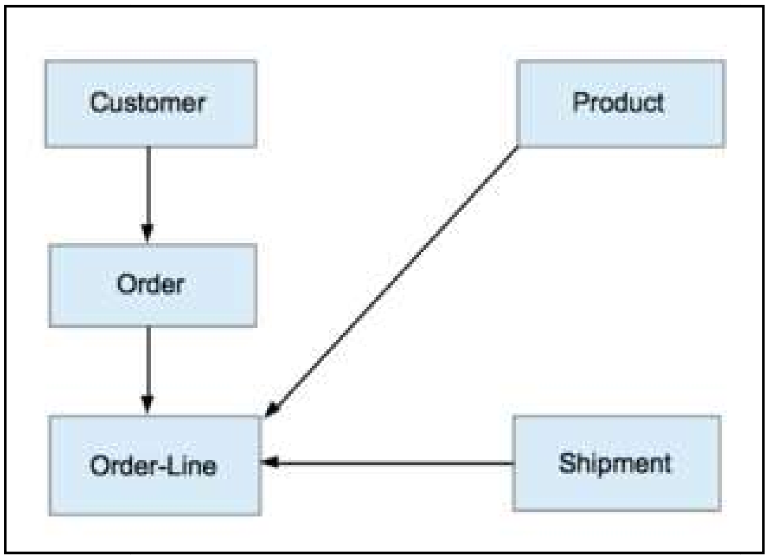
Disadvantage

- Implementation complexity → simple to design but complex to implement
- Lack of structural independence

d) Network model

- Network model is based on **directed graph theory**
- This is an **extension of the Hierarchical model.**
- In this model data is organised more like a graph, and are allowed to have more than one parent node.
- In this database model data is more related as more relationships are established in this database model.
- Also, as the data is more related, hence accessing the data is also easier and fast.
- This database model was used to map many-to-many data relationships.

- **Hierarchical model → Tree structure → 1: n relationship**
- **Network model → Graph structure → n:n relationship**



Advantage

- Conceptual simplicity → easy to design and understand
- Capability to handle many relation type
  - One-to-many (1:n)
  - Many-to-many (n:n)
  - Many-to-one (n:1)
- Data independence
  - Changes in data characteristics do not require change to application program

Disadvantage

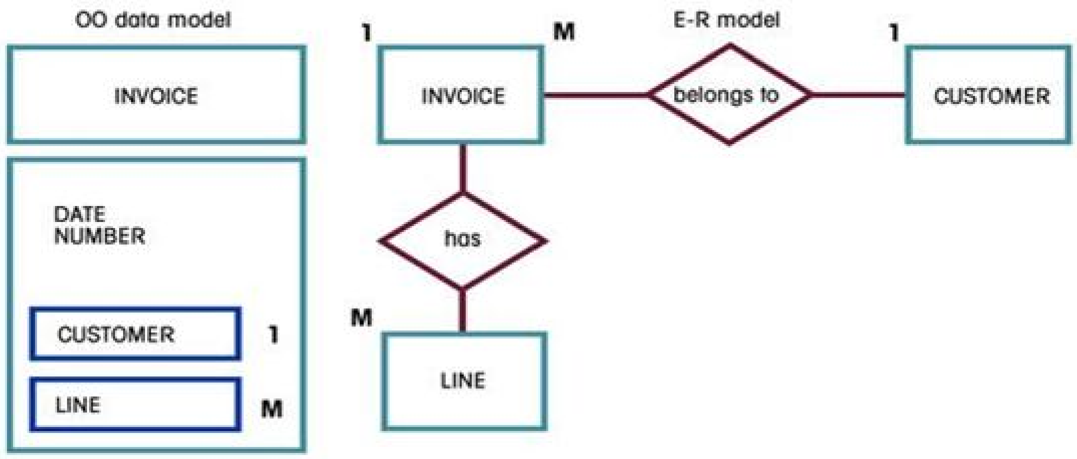
- Detailed structured knowledge is required
- Lack of structural dependencies

e) Object oriented model

- Based on **collection of objects**
- AN object contain values stored in instance variable within the object
- It considers each object in the world as objects and isolates it from each other.
- It groups its related functionalities together and allows inheriting its functionality to other related sub-groups.

The Components of the Object Oriented Data Model

• **An object** is an abstraction of a real-world entity. In general terms, an object may be considered equivalent to an ER model's entity.
• **Attributes** describe the properties of an object. For example, a PERSON object includes the attributes Name, Social Security Number, and Date of Birth.
• Objects that share similar characteristics are grouped in classes. A **class** is a collection of similar objects with shared structure (attributes) and behavior (methods).
• Classes are organized in a **class hierarchy**. The class hierarchy resembles an upside-down tree in which each class has only one parent. For example, the CUSTOMER class and the EMPLOYEE class share a parent PERSON class.
• **Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of the classes** above it. For example, two classes, CUSTOMER and EMPLOYEE, can be created as subclasses from the class PERSON. In this case, CUSTOMER and EMPLOYEE will inherit all attributes and methods from PERSON.



A comparison of OO data model and ER model

Advantage

- Application require less code
- Code is easier to maintain
- Database integrity
- Both structural and data independence
- Codes are re-used because of inheritance.

Disadvantage

- It is not widely developed and complete to use it in the database systems. Hence it is not accepted by the users.
- It is an approach for solving the requirement. It is not a technology. Hence it fails to put it in the database management systems.