Unit - I

Introduction to Database Management System and ER Model

Contents

Introduction

- A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner.
- It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

DBMS

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: 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
- Databases can be very large.
- Databases touch all aspects of our lives

Purpose of DBMS

- The purpose of DBMS is to transform the following –
  - Data into information.
  - Information into knowledge.
  - Knowledge to the action.

Drawbacks in File System

- Data redundancy and inconsistency: Different file formats, duplication of information in different files.
- Difficulty in accessing data: To carry out new task we need to write a new program.
- Data Isolation – Different files and formats.
- Integrity problems.
- Atomicity of updates – Failures leave the database in an inconsistent state. For example, the fund transfer from one account to another may be incomplete.
- Concurrent access by multiple users.
- Security problems.

Applications of DBMS

The different applications of DBMS are as follows −

- Railway Reservation System − It is used to keep record of booking of tickets, departure of the train and the status of arrival and give updates to the passengers with the help of a database.

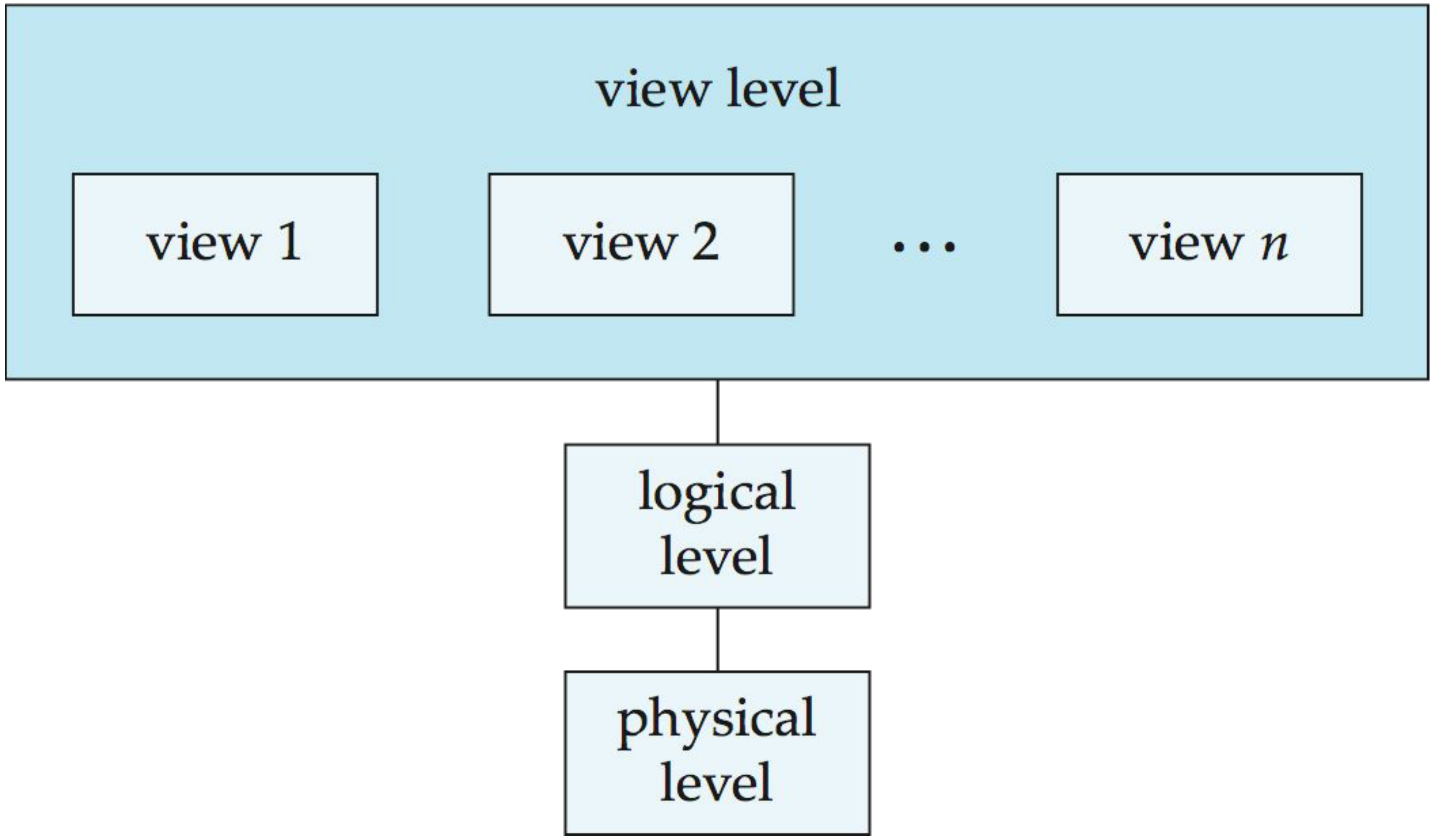
- Library Management System − There will be so many numbers of books in the library and it is very hard to keep a record of all the books in a register or a copy. So, DBMS is necessary to keep track of all the book records, issue dates, name of the books, author and maintain the records.

- Banking − We are doing a lot of transactions daily without directly going to the banks. The only reason is the usage of databases and it manages all the data of the customers over the database.

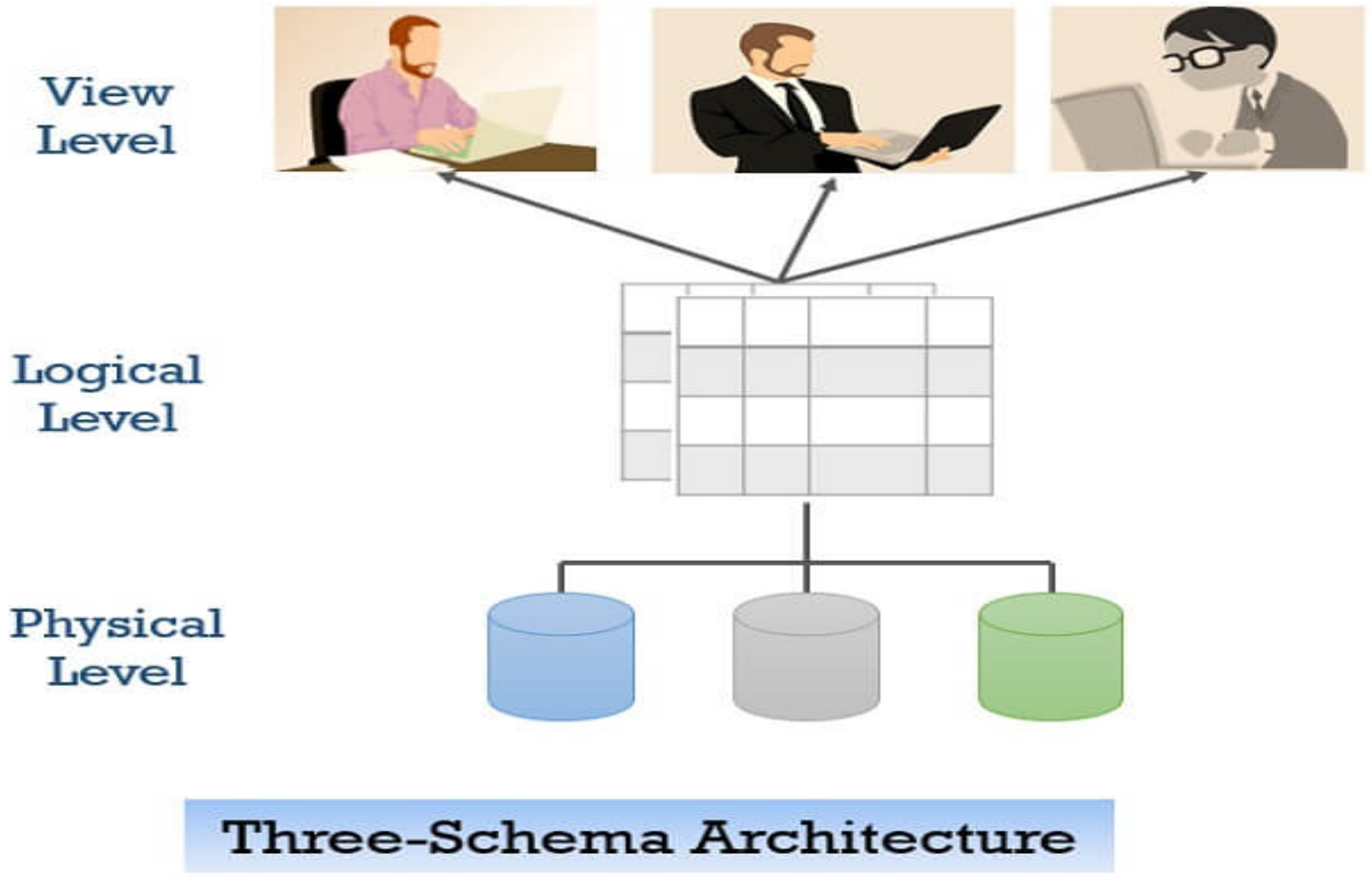
- Educational Institutions − All the examinations and the data related to the students maintained over the internet with the help of a database management system. It contains registration details of the student, results, grades and courses available. All these works can be done online without visiting an institution.

- Social Media Websites − By filling the required details we are able to access social media platforms. Many users daily sign up for social websites such as Facebook, Pinterest and Instagram. All the information related to the users are stored and maintained with the help of DBMS.

View of Data



View of Data


View Level
Logical Level
Physical Level
Three-Schema Architecture

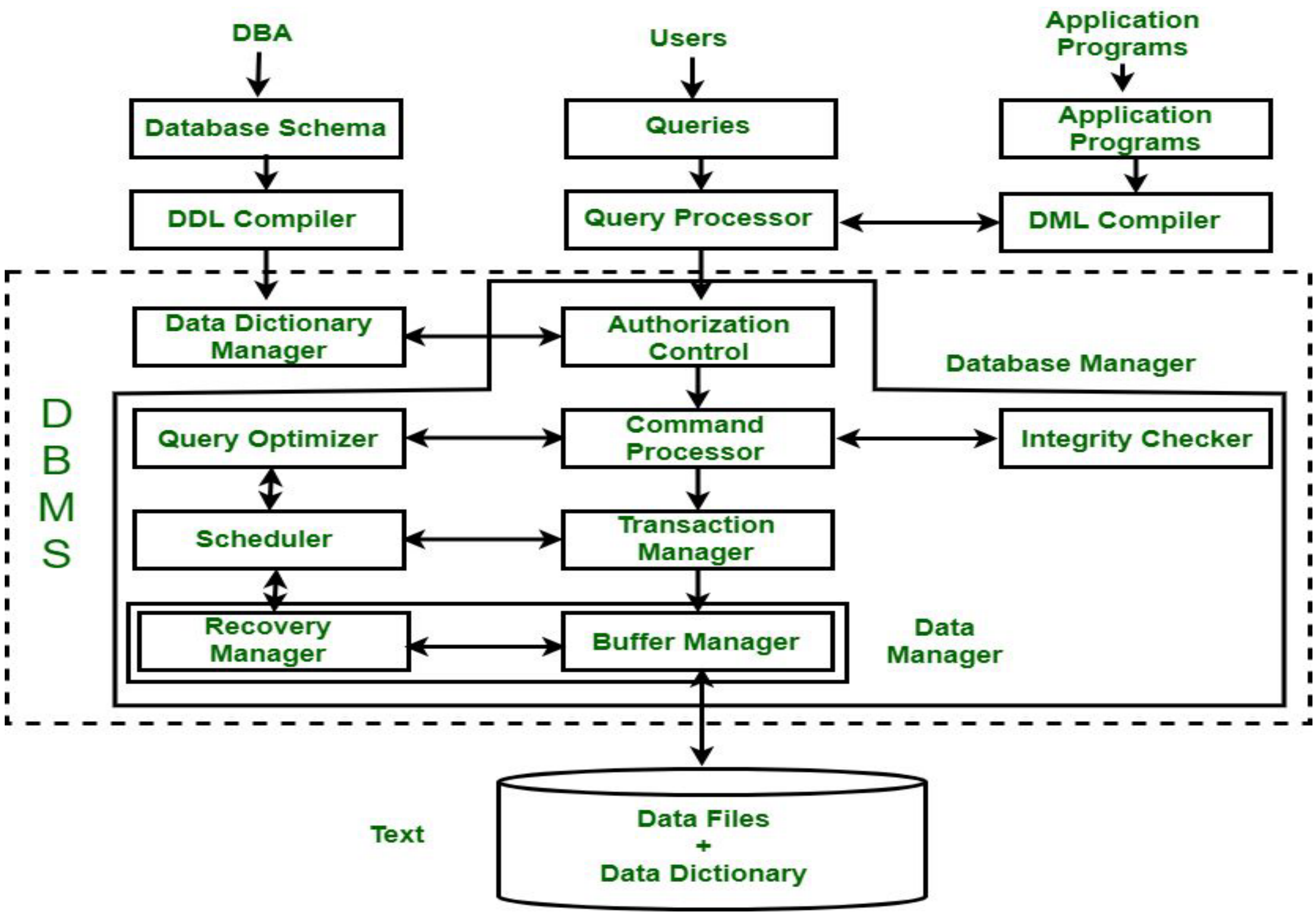
Database Languages

- **Database Languages** are the set of statements, that are used to define and manipulate a database.
- A Database language has

Data Definition Language (DDL), which is used to construct a database & it has

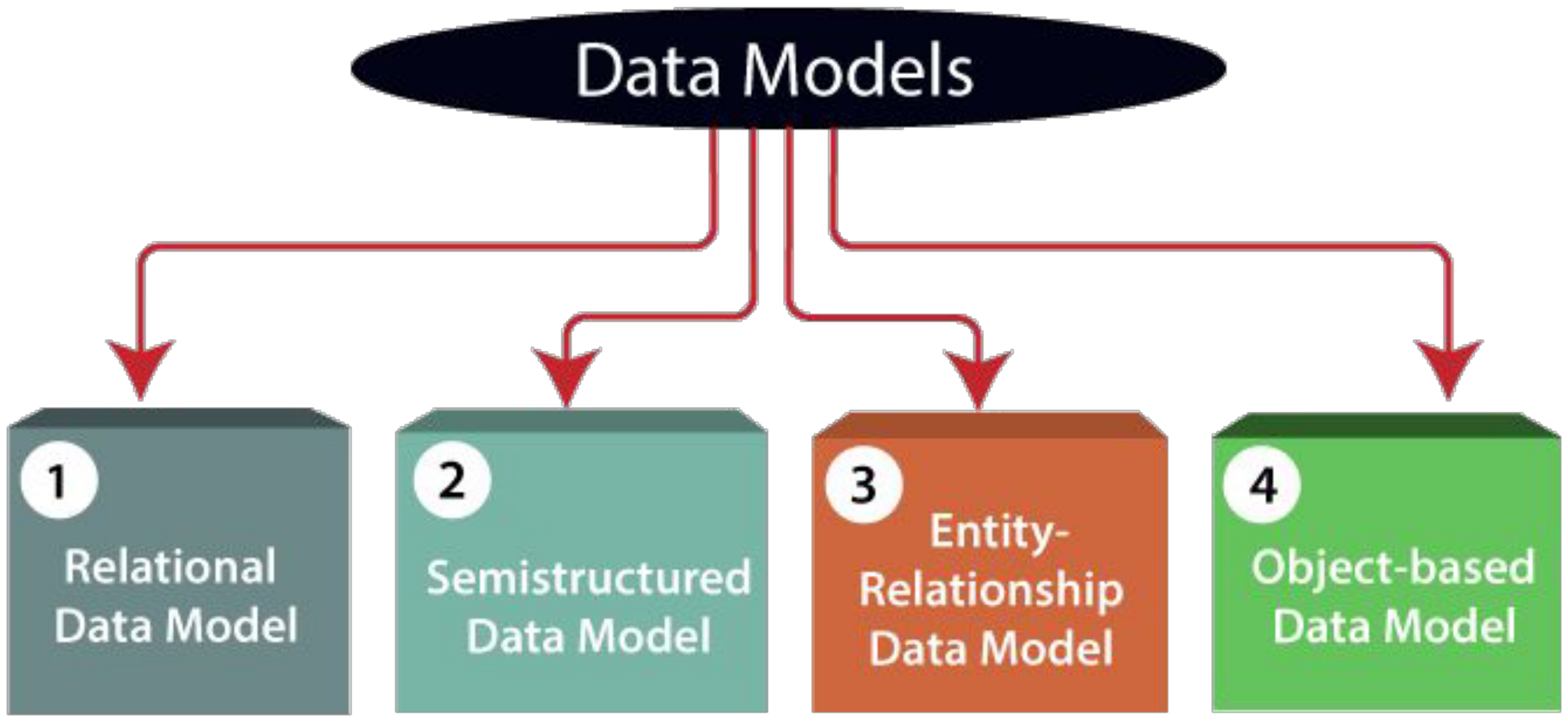
Data Manipulation Language (DML), which is used to access a database.

Database System Structure



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)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model


Data Models
1
Relational Data Model
2
Semistructured Data Model
3
Entity-Relationship Data Model
4
Object-based Data Model

1) **Relational Data Model:** 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. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

2) **Entity-Relationship Data Model:** An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student_name, student_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

3) **Object-based Data Model:** An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

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.

Database Design and ER Model

- Entity
- Attributes
- Relationships
- Constraints
- Keys

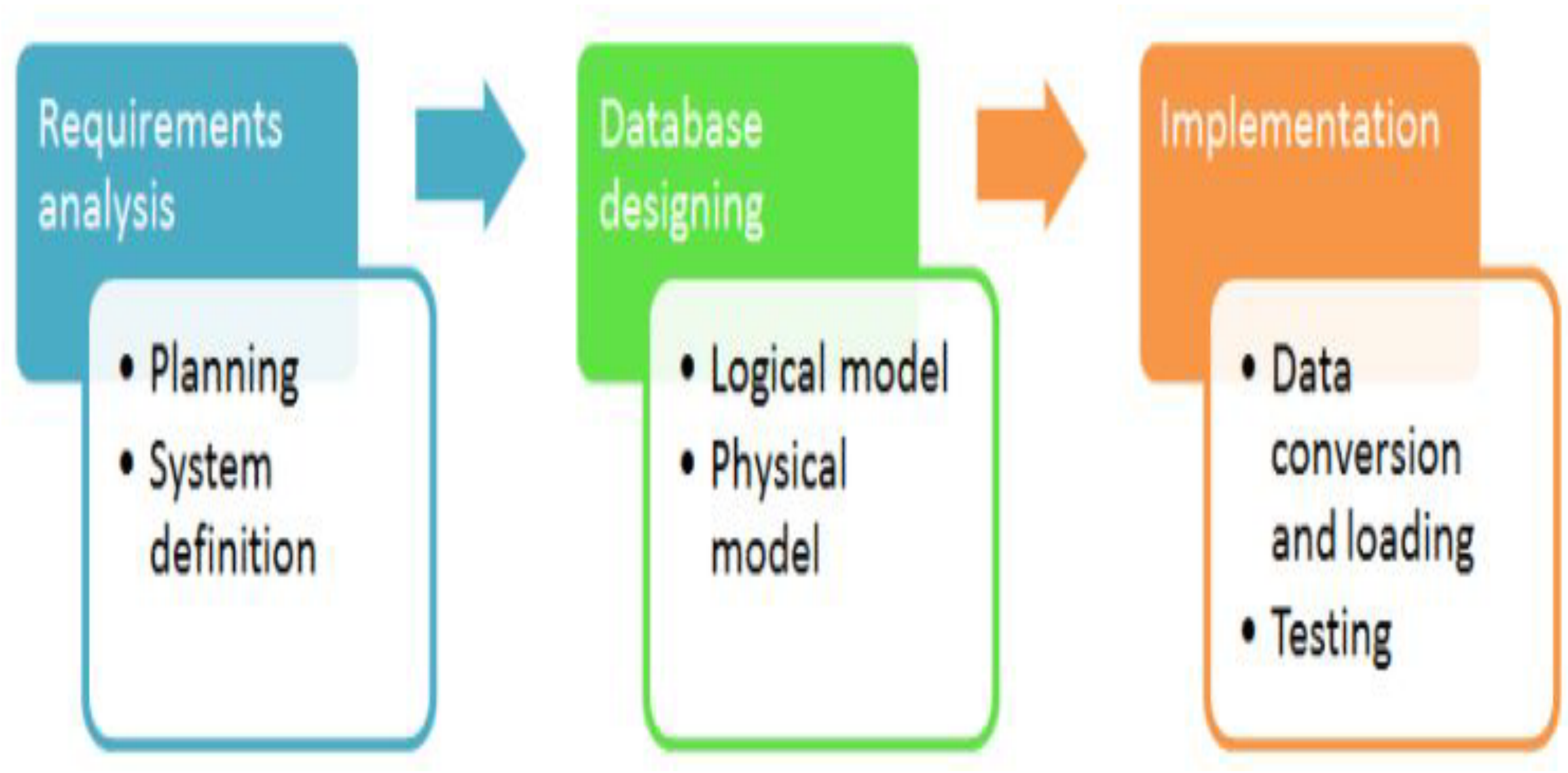
Database Design

- Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system.
- Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space.
- The main objectives behind database designing are to produce physical and logical design models of the proposed database system.

Why is Database Design important?

- Database designs provide the blueprints of how the data is going to be stored in a system. A proper design of a database highly affects the overall performance of any application.
- The designing principles defined for a database give a clear idea of the behavior of any application and how the requests are processed.
- Another instance to emphasize the database design is that a proper database design meets all the requirements of users.
- Lastly, the processing time of an application is greatly reduced if the constraints of designing a highly efficient database are properly implemented.

Design Process



Design Process

The process of designing a database carries various conceptual approaches that are needed to be kept in mind. An ideal and well-structured database design must be able to:

- Save disk space by eliminating redundant data.
- Maintains data integrity and accuracy.
- Provides data access in useful ways.
- Comparing Logical and Physical data models.

Important Terms

- **Entity** - An entity in the database can be defined as abstract data that we save in our database. For example, a customer, products.
- **Attributes** - An attribute is a detailed form of data consisting of entities like length, name, price, etc.
- **Relationship** - A relationship can be defined as the connection between two entities or figures. For example, a person can relate to multiple persons in a family.
- **Foreign key** - It acts as a referral to the Primary Key of another table. A foreign key contains columns with values that exist only in the primary key column they refer to.
- **Primary key** - A primary key is the pointer of records that is unique and not null and is used to uniquely identify attributes of a table.
- **Normalization** - A flexible data model needs to follow certain rules. Applying these rules is called normalizing.

Logical DB design Process

A logical data model generally describes the data in as many details as possible, without having to be concerned about the physical implementations in the database.

Features of logical data model might include:

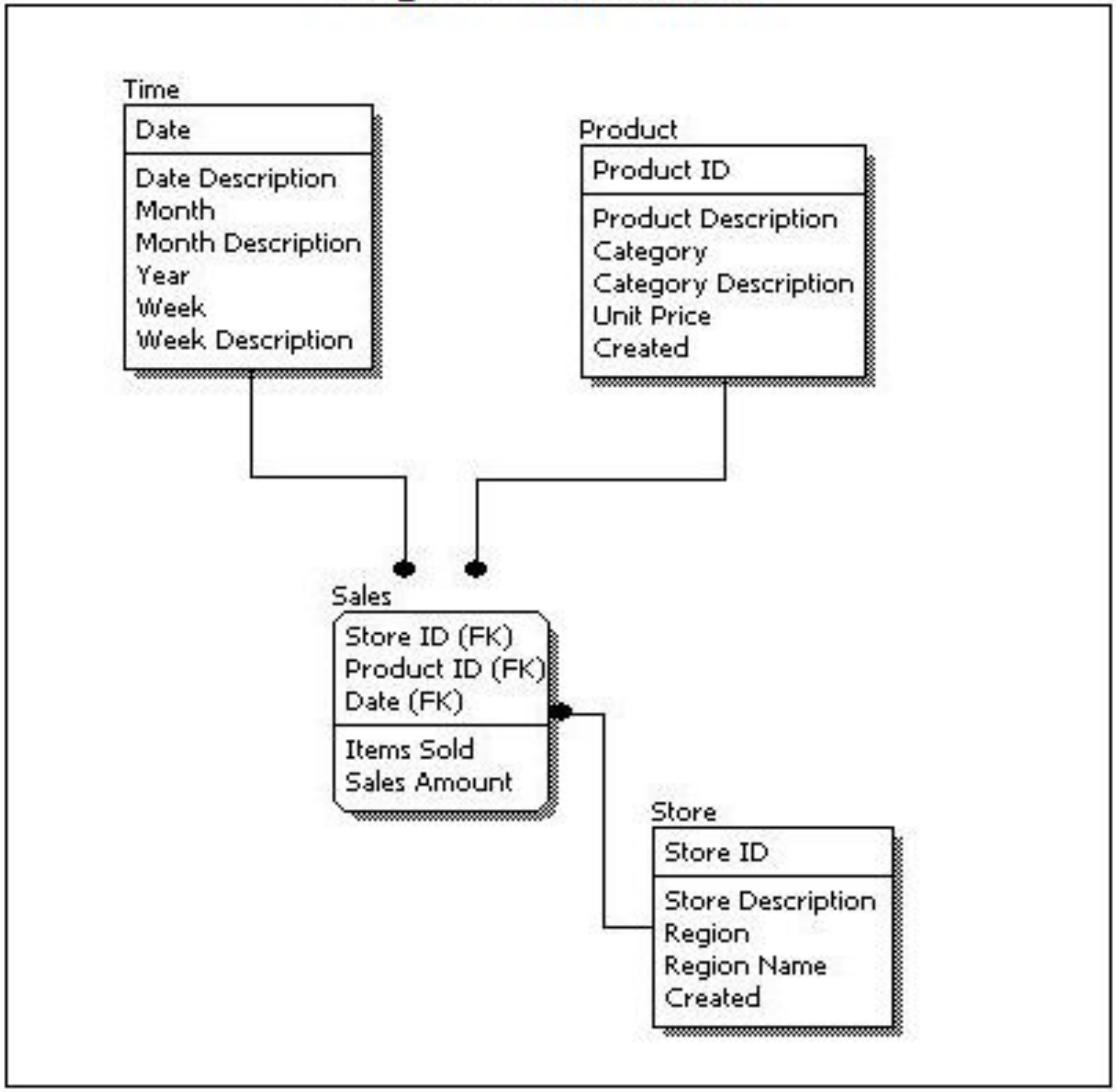
- All the entities and relationships amongst them.
- Each entity has well-specified attributes.
- The primary key for each entity is specified.
- Foreign keys which are used to identify a relationship between different entities are specified.
- Normalization occurs at this level.

Logical DB design process

A logical model can be designed using the following approach:

- Specify all the entities with primary keys.
- Specify concurrent relationships between different entities.
- Figure out each entity attributes
- Resolve many-to-many relationships.
- Carry out the process of normalization.

Logical Data Model



Physical DB design Process

A Physical data mode generally represents how the approach or concept of designing the database. The main purpose of the physical data model is to show all the **structures** of the table including the **column name, column data type, constraints, keys(primary and foreign)**, and the relationship among tables.

The following are the features of a physical data model:

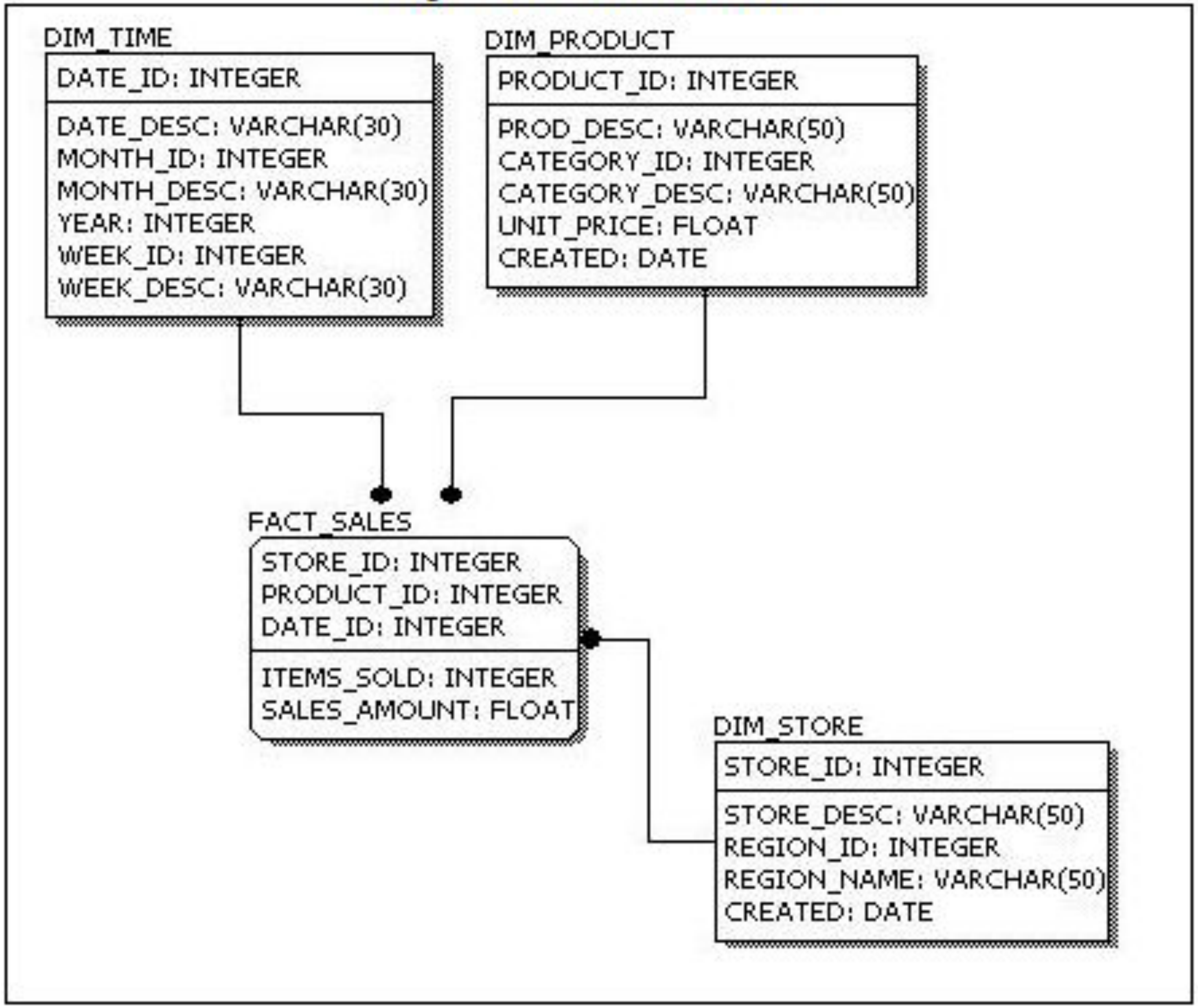
- Specifies all the columns and tables.
- Specifies foreign keys that usually define the relationship between tables.
- Based on user requirements, de-normalization might occur.
- Since the physical consideration is taken into account so there will straightforward reasons for difference than a logical model.
- Physical models might be different for different RDBMS. For example,

Physical DB design Process

While designing a physical data model, the following points should be taken into consideration:

- Convert the entities into tables.
- Convert the defined relationships into foreign keys.
- Convert the data attributes into columns.
- Modify the data model constraints based on physical requirements.

Physical Data Model

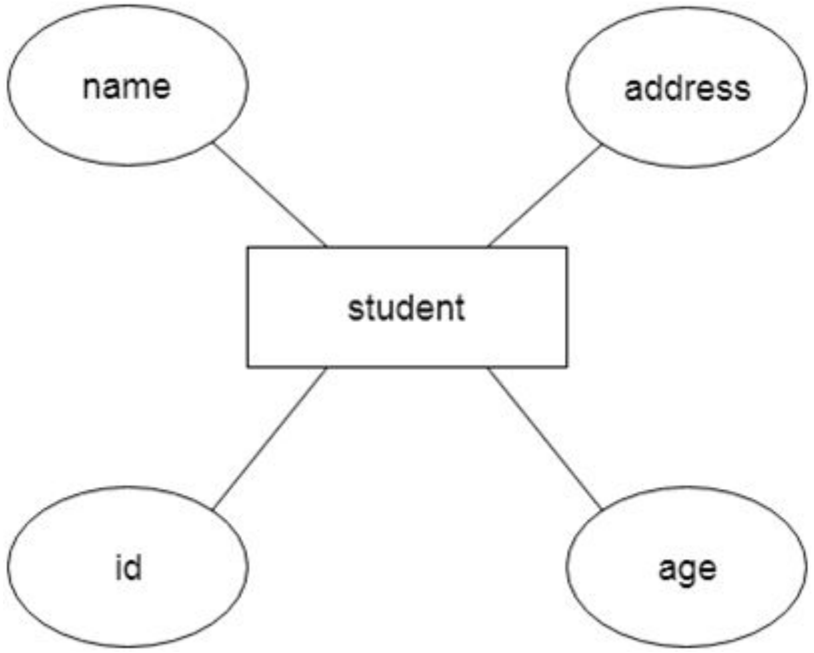


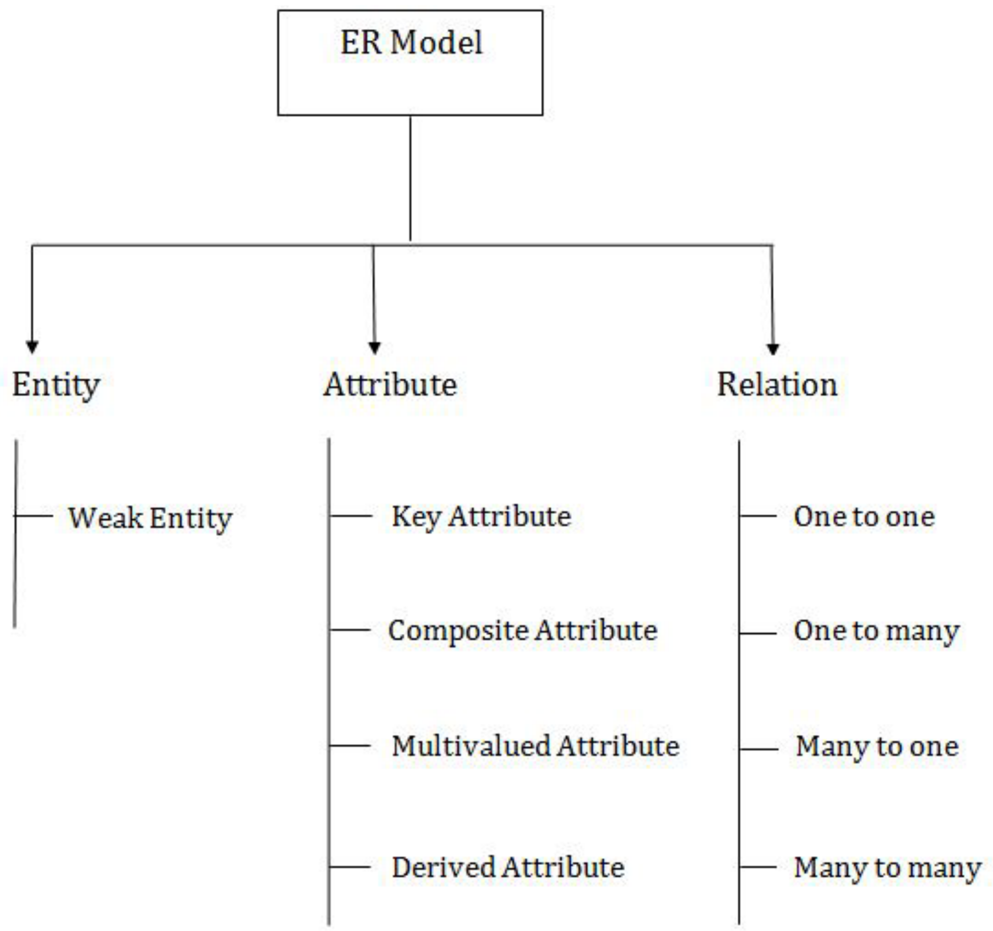
Entity-Relationship Model

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

ER Diagram

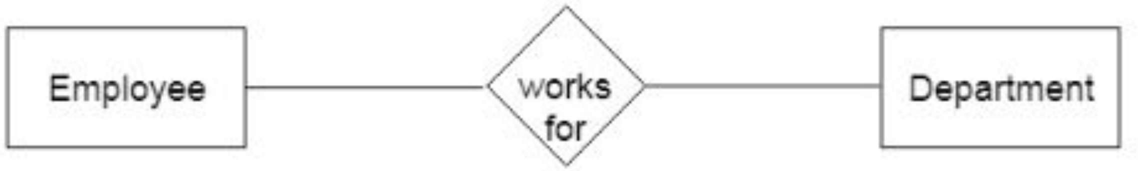
- **For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.




ER Model
Entity
Weak Entity
Attribute
Key Attribute
Composite Attribute
Multivalued Attribute
Derived Attribute
Relation
One to one
One to many
Many to one
Many to many

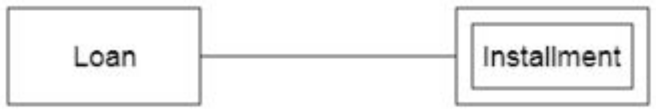
1. Entity

- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



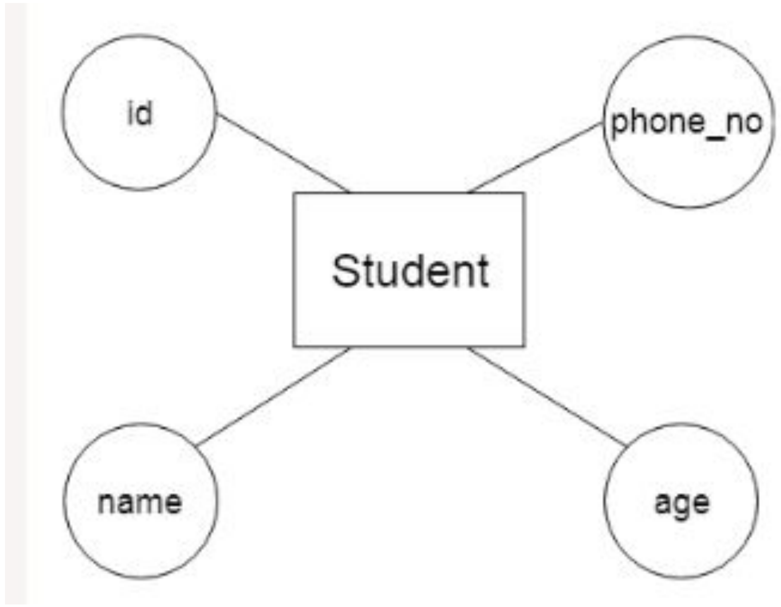
a. Weak Entity

- An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

- The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
- **For example,** id, age, contact number, name, etc. can be attributes of a student.



Types of Attribute

a. **Key Attribute**

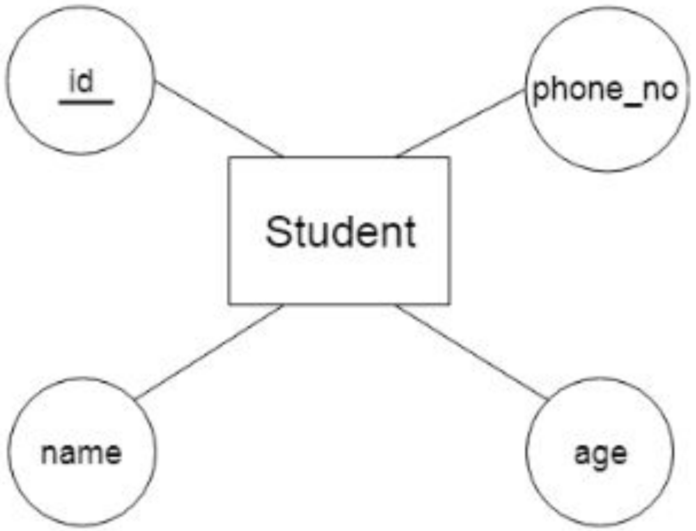
b. **Composite Attribute**

c. **Multivalued Attribute**

d. **Derived Attribute**

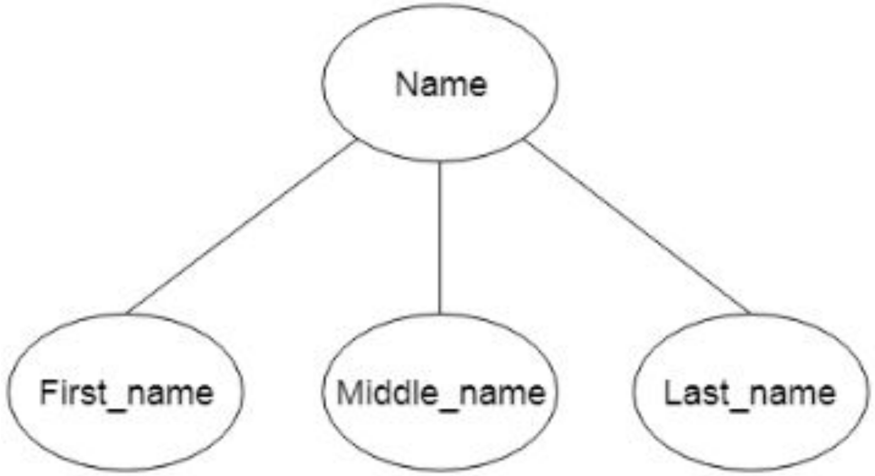
Key Attribute

- The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlinec



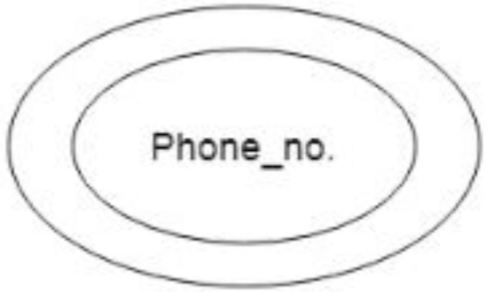
Composite Attribute

- An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



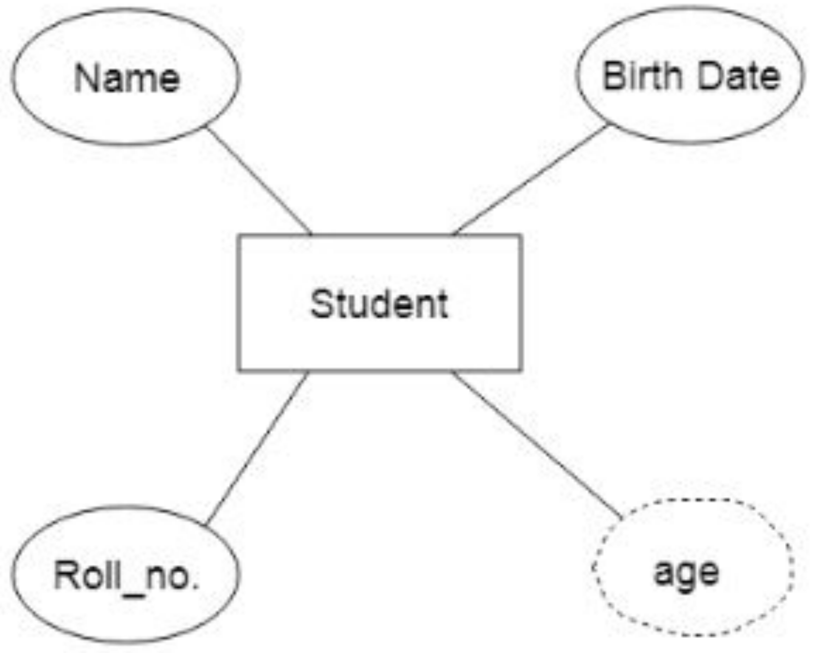
Multivalued Attribute

- An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.
- **For example,** a student can have more than one phone number.



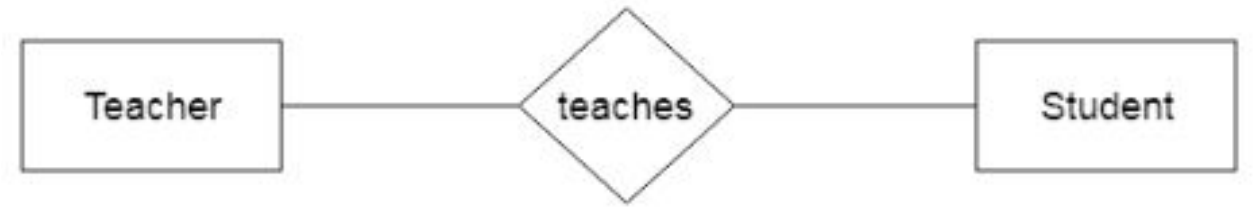
Derived Attribute

- An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.
- **For example,** A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

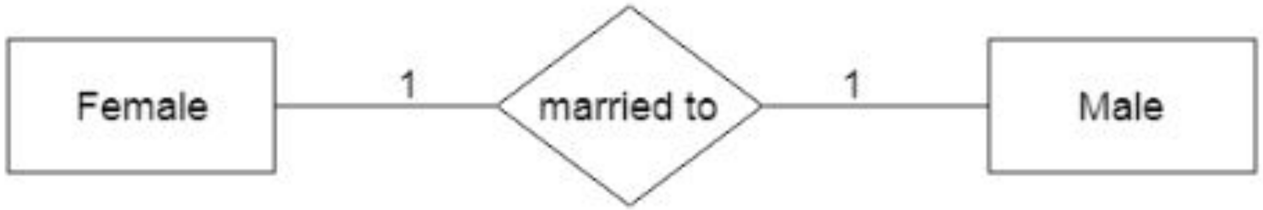
- A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



- Types of relationship are as follows:
  - **a. One-to-One Relationship**
  - **b. One-to-many relationship**
  - **c. Many-to-one relationship**
  - **d. Many-to-many relationship**

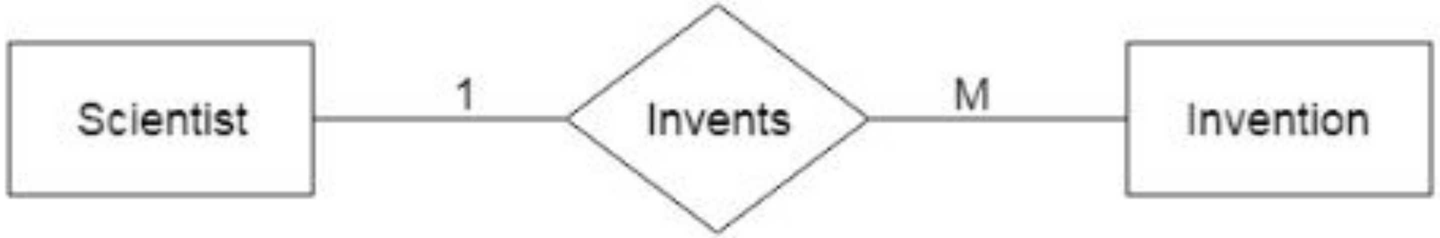
One-to-One Relationship

- When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.
- **For example,** A female can marry to one male, and a male can marry to one female.



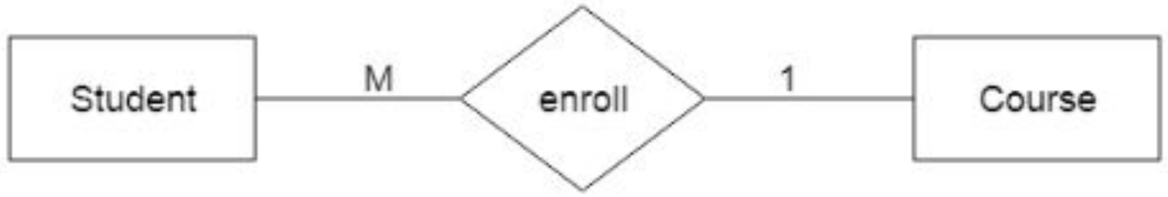
One-to-many relationship

- When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.
- **For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



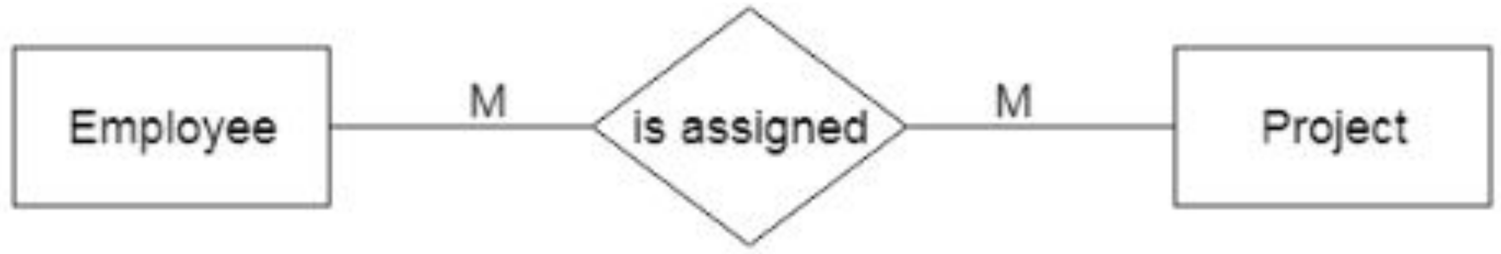
Many-to-one relationship

- When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
- **For example,** Student enrolls for only one course, but a course can have many students.



Many-to-many relationship

- When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
- **For example,** Employee can assign by many projects and project can have many employees.



Design Issues

1. Use of Entity Set vs Attributes
2. Use of Entity Set vs. Relationship Sets
3. Use of Binary vs n-ary Relationship Sets
4. Placing Relationship Attributes

1. Use of Entity Set vs Attributes

- The use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled and the semantics associated with its attributes.
- It leads to a mistake when the user use the primary key of an entity set as an attribute of another entity set.
- Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets.

2.Use of Entity Set vs. Relationship Sets

- It is difficult to examine if an object can be best expressed by an entity set or relationship set.
- To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities.
- If there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

3. Use of Binary vs n-ary Relationship Sets

- Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships.
- For example, we can create and represent a ternary relationship 'parent' that may relate to a child, his father, as well as his mother.
- Such relationship can also be represented by two binary relationships i.e, mother and father, that may relate to their child.
- Thus, it is possible to represent a non-binary relationship by a set of distinct binary relationships.

4. Placing Relationship Attributes

- The cardinality ratios can become an affective measure in the placement of the relationship attributes.
- So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set.
- The decision of placing the specified attribute as a relationship or entity attribute should possess the charactestics of the real world enterprise that is being modelled.
- **For example**, if there is an entity which can be determined by the combination of participating entity sets, instead of determing it as a separate entity. Such type of attribute must be associated with the many-to-many relationship sets.

Extended E-R Features

- Extended ER is a high-level data model that incorporates the extensions to the original ER model. Enhanced ER models are high level models that represent the requirements and complexities of complex databases.

Extended E-R Features

The extended Entity Relationship (ER) models are three types as given below –

- Aggregation
- Specialization
- Generalization

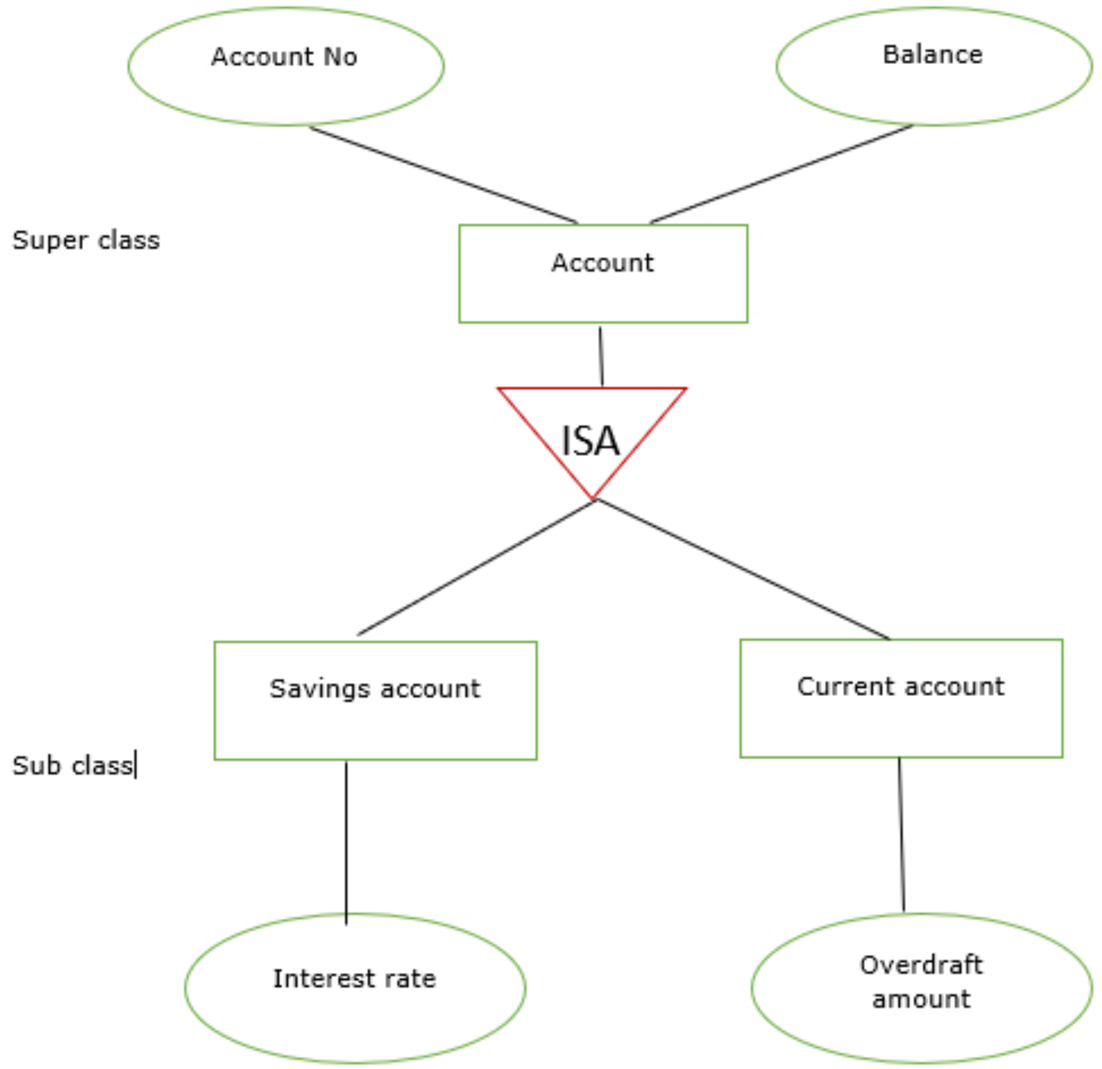
Specialization

- The process of designing sub groupings within an entity set is called specialization.
- It is a top-down process.
- If an entity set is given with all the attributes in which the instances of the entity set are differentiated according to the given attribute value, then that sub-classes or the sub-entity sets can be formed from the given attribute.

Example

- Specialization of a person allows us to distinguish a person according to whether they are employees or customers. Specialization of account creates two entity sets: savings account and current account.
- In the E-R diagram specialization is represented by triangle components labeled ISA. The ISA relationship is referred as superclass- subclass relationship.

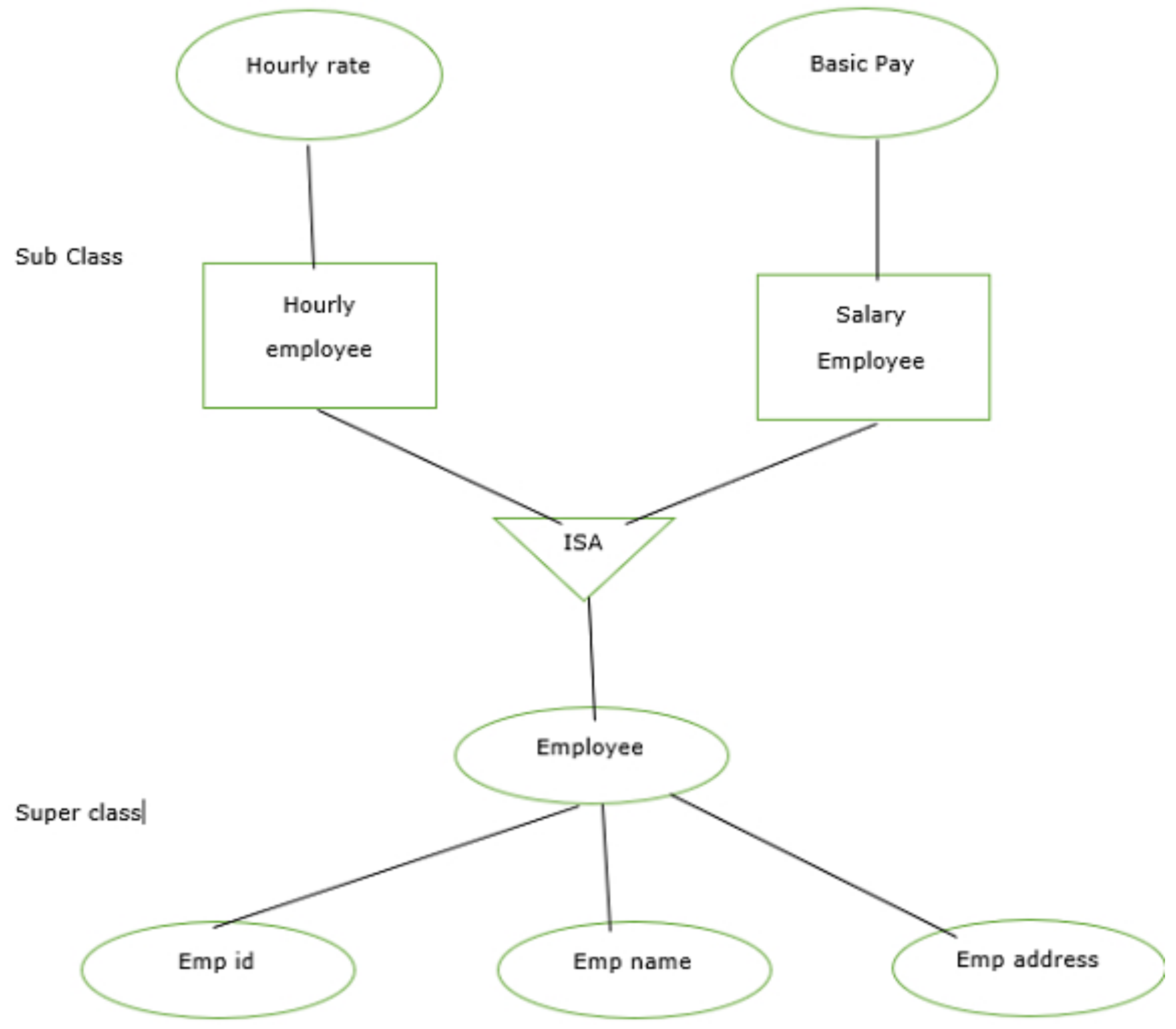
Specialization



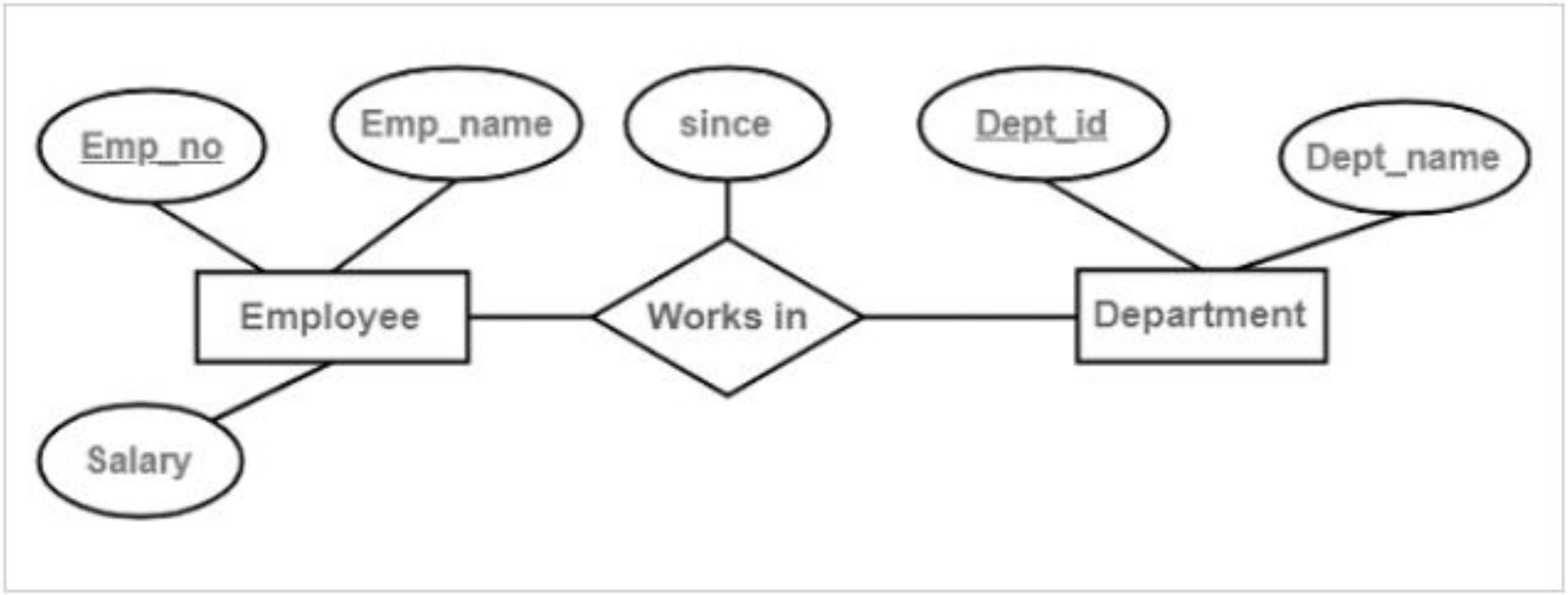
Generalization

- It is the reverse process of specialization.
- It is a bottom-up approach.
- It converts subclasses to superclasses.
- This process combines a number of entity sets that share the same features into higher-level entity sets.
- If the sub-class information is given for the given entity set then, ISA relationship type will be used to represent the connectivity between the subclass and superclass as shown below –
- **Example**

Generalization



Converting ER and EER diagram into tables



Conversion of ER diagrams to tables

Step 1 – Conversion of strong entities

- For each strong entity create a separate table with the same name.
- Includes all attributes, if there is any composite attribute divided into simple attributes and has to be included.
- Ignore multivalued attributes at this stage.
- Select the p key for the table.

Conversion of ER diagrams to tables

Step 2 – Conversion of weak entity

- For each weak entity create a separate table with the same name.
- Include all attributes.
- Include the P key of a strong entity as foreign key is the weak entity.
- Declare the combination of foreign key and decimator attribute as P key from the weak entity.

Conversion of ER diagrams to tables

Step 3 – Conversion of one-to-one relationship

- For each one to one relation, say A and B modify either A side or B side to include the P key of the other side as a foreign key.
- If A or B is having total participation, then that should be a modified table.
- If a relationship consists of attributes, include them also in the modified table.

Conversion of ER diagrams to tables

Step 4 – Conversion of one-to-many relationship

- For each one to many relationships, modify the M side to include the P key of one side as a foreign key.
- If relationships consist of attributes, include them as well.

Conversion of ER diagrams to tables

Step 5 – Conversion of many-many relationship

- For each many-many relationship, create a separate table including the P key of M side and N side as foreign keys in the new table.
- Declare the combination of foreign keys as P for the new table.
- If relationships consist of attributes, include them also in the new table.

Conversion of ER diagrams to tables

Step 6 – Conversion of multivalued attributes

- For each multivalued attribute create a separate table and include the P key of the present table as foreign key.
- Declare the combination of foreign key and multivalued attribute as P keys.

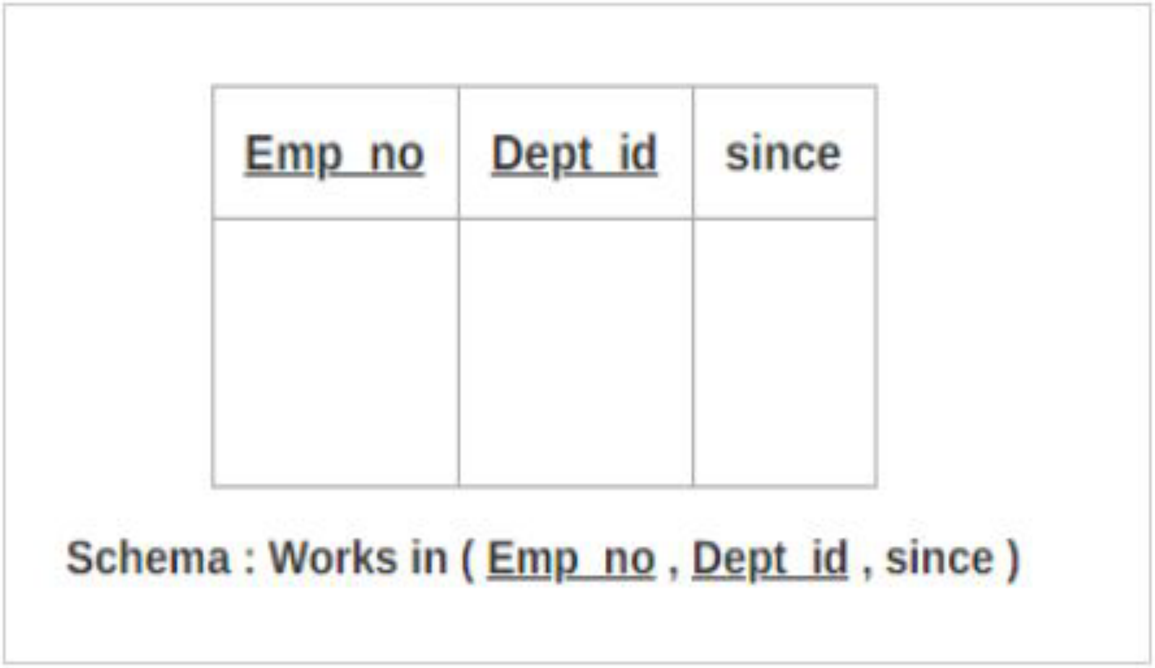
Conversion of ER diagrams to tables

Step 7 – Conversion of n-ary relationship

- For each n-ary relationship create a separate table and include the P key of all entities as foreign key.
- Declare the combination of foreign keys as P key.

Table

- After successful conversion, the result will be as follows –



Emp_no	Dept_id	since

Schema : Works in (Emp_no , Dept_id , since)