UNIT - 3

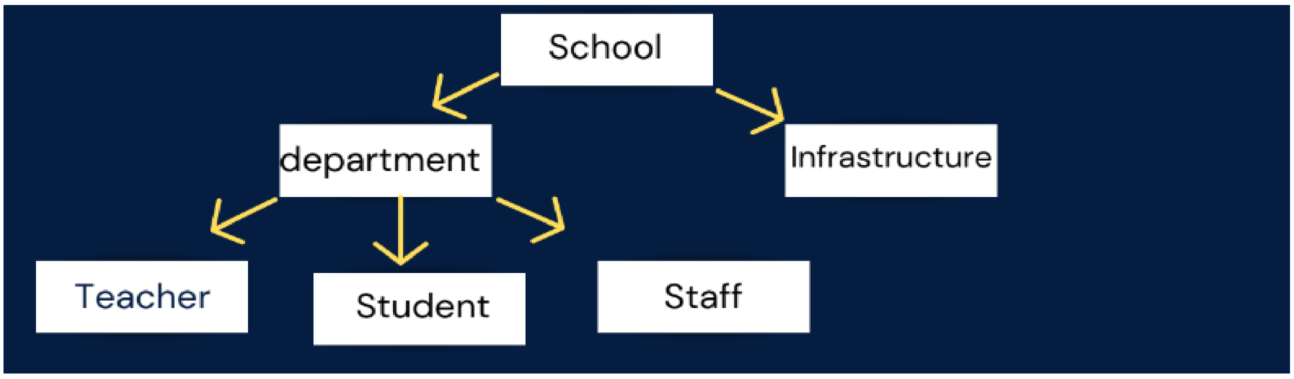
DATA MODELS

What is a Data Model?

- ➔ A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database.
- ➔ Data Models provide us with a transparent picture of data which helps us in creating an actual database.
- ➔ It determines how data can be stored, accessed and updated in a database management system.

Hierarchical Model

- ➔ This is one of the oldest models in a data model which was developed by IBM, in the 1950s.
- ➔ The data is organized into a tree-like structure where each record consists of one parent record and many children.
- ➔ The hierarchical model was popularized by early DBMS like IBM's IMS (Information Management System). It was effective for specific applications such as managing organizational structures, file systems, or assembly line processes.



Key Concepts:

1. **Nodes**: In a hierarchical model, data is stored in nodes. Each node represents a record or an entity, and it contains information fields (attributes) and pointers to its child nodes.
2. **Parent-Child Relationships**: Nodes are connected through parent-child relationships. Each parent can have multiple children, but each child has only

one parent. This creates a top-down structure where data flows from higher-level entities (parent nodes) to lower-level entities (child nodes).[1]

3. **Root Node**: At the top of the hierarchy is the root node, which does not have a parent but can have one or more child nodes.
4. **Leaf Nodes**: Nodes at the lowest level of the hierarchy that do not have children are called leaf nodes.
5. **Traversal**: Accessing data in a hierarchical model typically involves traversing from the root node down through its children to reach the desired data. This is known as a depth-first traversal.

Advantages:

- **Efficient for Certain Queries**: It's efficient for queries that follow a hierarchical path, like retrieving all employees in a department or all details of an employee.
- **Data Integrity**: Ensures data integrity through parent-child relationships.

Disadvantages:

- **Lack of Flexibility**: Not suitable for all types of data relationships (e.g., many-to-many relationships).
- **Complex Updates**: Updating the structure (e.g., inserting a new node) can be complex and may require reorganizing large parts of the hierarchy.
- **Scalability Issues**: Managing large hierarchical structures can become unwieldy.

Network Model

➔ This model is the generalization of the hierarchical model.
➔ The Network Model in a Database Management System (DBMS) is a data model that allows the representation of many-to-many relationships in a more flexible and complex structure compared to the Hierarchical Model.
➔ It uses a graph structure consisting of nodes (entities) and edges (relationships) to organize data, enabling more efficient and direct access paths.

Advantages of Network Model

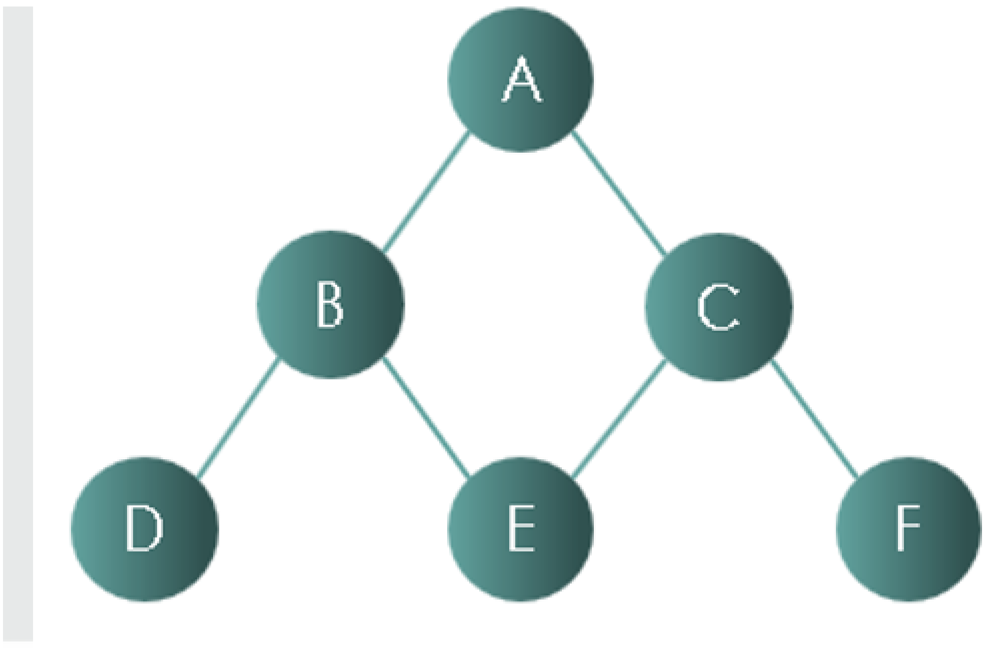
- This model is very simple and easy to design like the hierarchical data model.

[1]**Faculty: Shubham Upadhyay**

- This model is capable of handling multiple types of relationships which can help in modeling real-life applications, for example, 1: 1, 1: M, M: N relationships.
- In this model, we can access the data easily, and also there is a chance that the application can access the owner's and the member's records within a set.

Disadvantages of Network Model

- The design or the structure of this model is not user-friendly.
- This model does not have any scope of automated query optimization.
- This model fails in achieving structural independence even though the network database model is capable of achieving data independence.



Relational Model:

- ➔ The relational model uses a collection of tables to represent both data and the relationships among those data.
- ➔ Relational model can represent as a table with columns and rows.
- ➔ Each row is known as a tuple. Each table of the column has a name or attribute.

Terminologies:

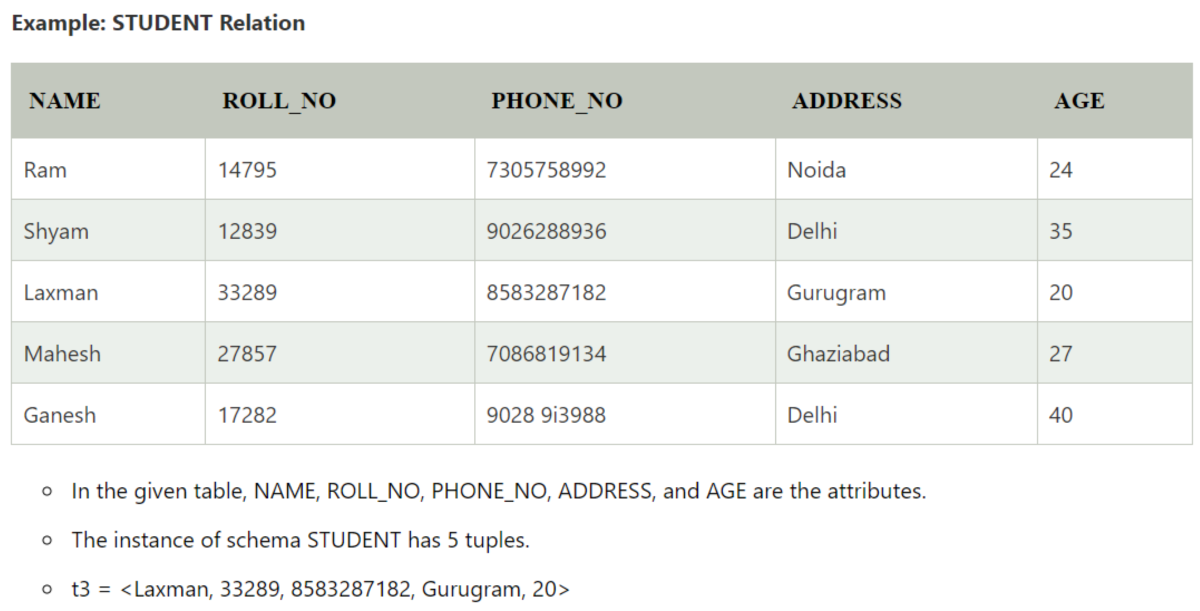
Domain: It contains a set of atomic values that an attribute can take.

Attribute: It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai)

Relational instance: In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.

Relational schema: A relational schema contains the name of the relation and name of all columns or attributes.

Relational key: In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.



Example: STUDENT Relation

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.
- The instance of schema STUDENT has 5 tuples.
- t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

E-R MODEL(Entity-Relational Model):

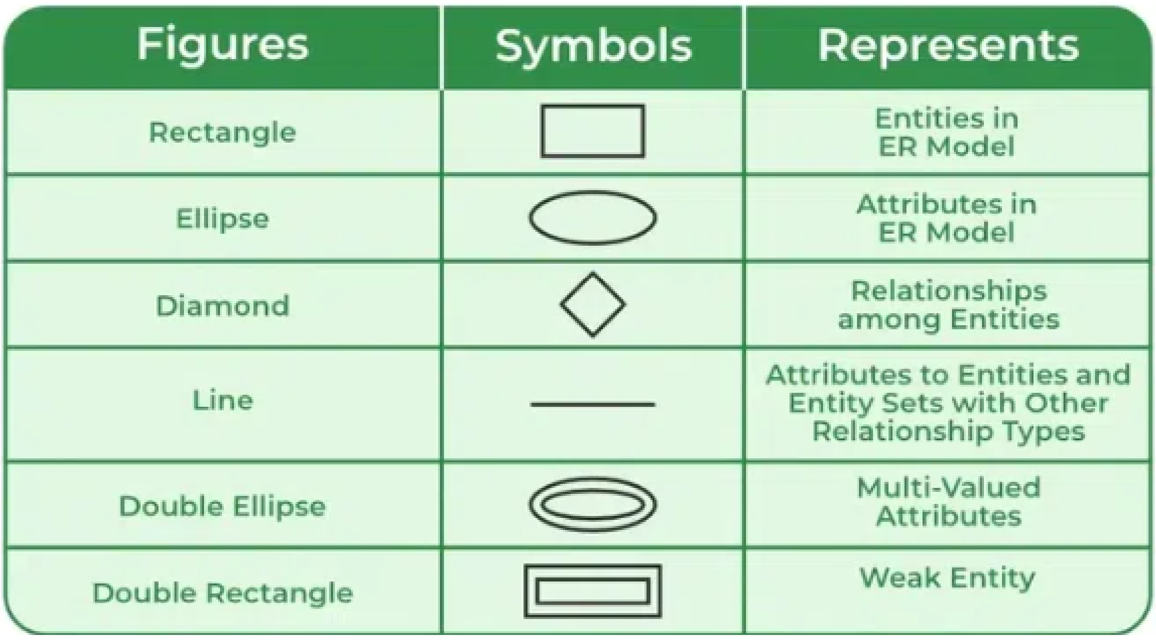
➔ Peter Chen developed the ER diagram in 1976 .

➔ The ER model was created to provide a simple and understandable model for representing the structure and logic of databases.
➔ The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related.
➔ ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects.

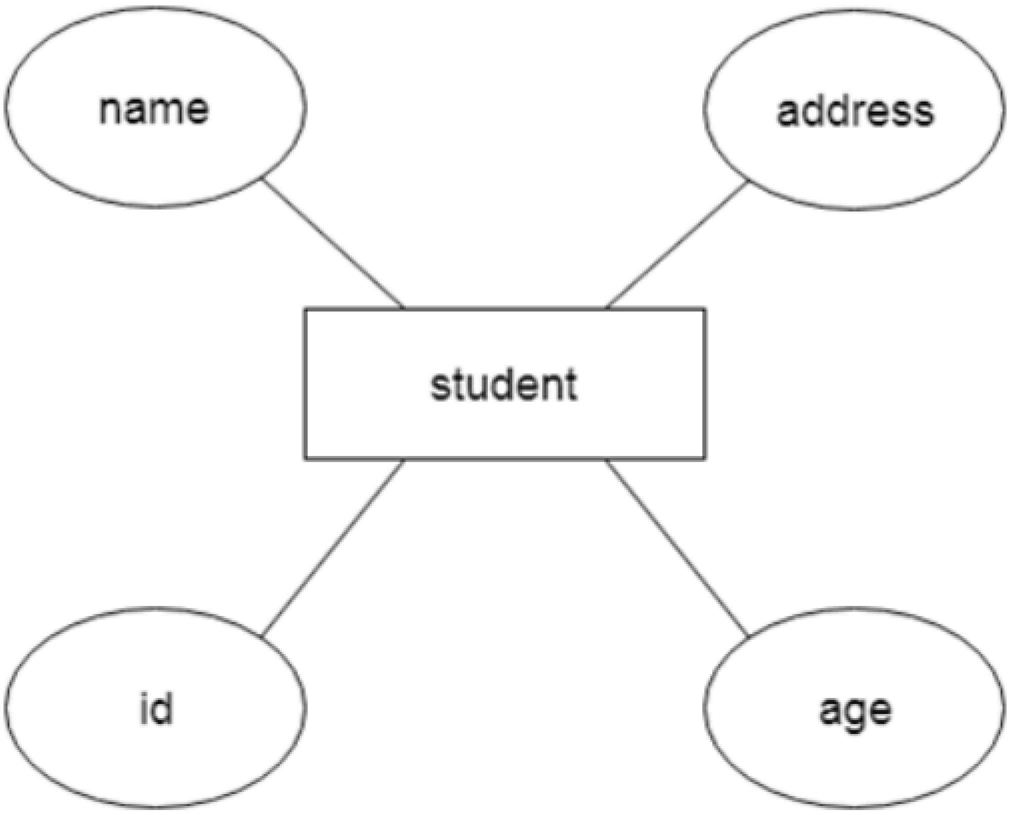
Why Use ER Diagrams In DBMS?

- ER diagrams are used to represent the E-R model in a database, which makes them easy to convert into relations (tables).
- ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- ER diagrams require no technical knowledge and no hardware support.
- These diagrams are very easy to understand and easy to create even for a naive user.
- It gives a standard solution for visualizing the data logically.

Symbols Used in ER Model



Figures	Symbols	Represents
Rectangle	▭	Entities in ER Model
Ellipse	⬭	Attributes in ER Model
Diamond	◇	Relationships among Entities
Line	—	Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse	◎	Multi-Valued Attributes
Double Rectangle	▣	Weak Entity



Object Oriented Model:

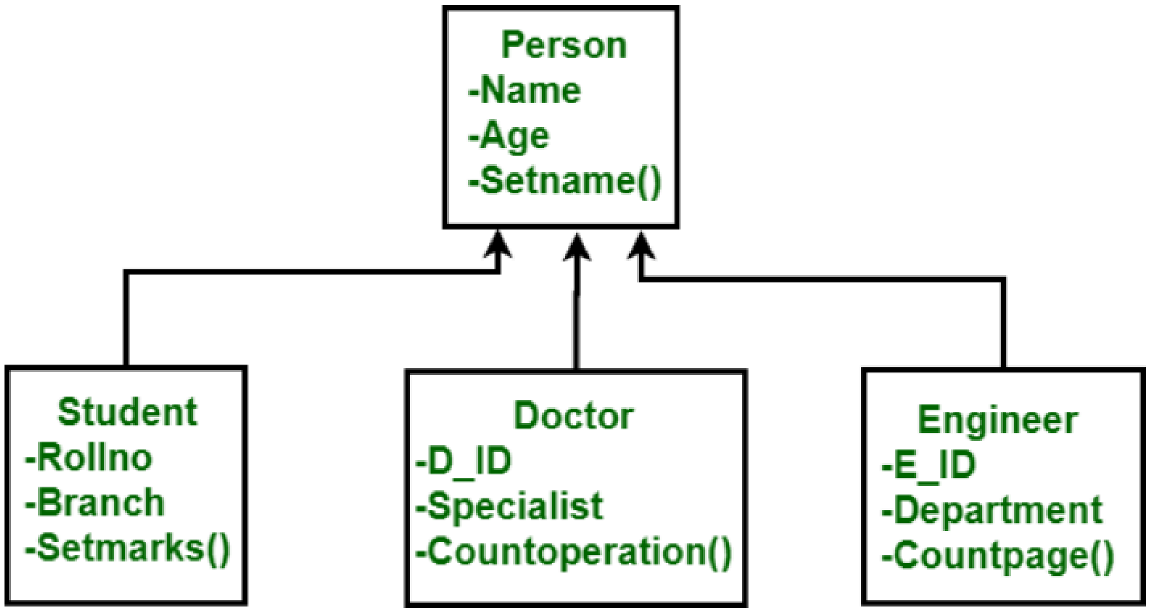
- ➔ In Object Oriented Data Model, data and their relationships are contained in a single structure which is referred to as an object in this data model.
- ➔ In this, real world problems are represented as objects with different attributes.
- ➔ Basically, it is combination of Object Oriented programming and Relational Database Model .

Advantages of Object Oriented Data Model :

- Codes can be reused due to inheritance.
- Easily understandable.
- Cost of maintenance can be reduced due to reusability of attributes and functions because of inheritance.

Disadvantages of Object Oriented Data Model :

- It is not properly developed so not accepted by users easily.



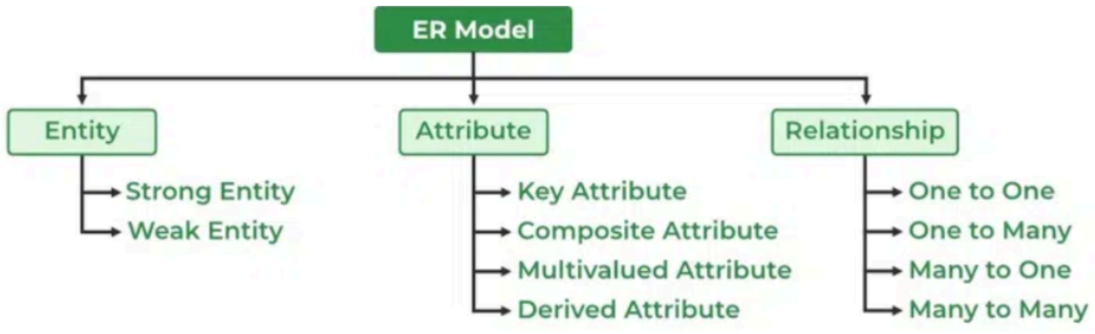
Basic Object Oriented Data Model

E- R Diagram:

➔ An Entity-Relationship (E-R) diagram is a visual representation used in database design to describe the relationships between entities.

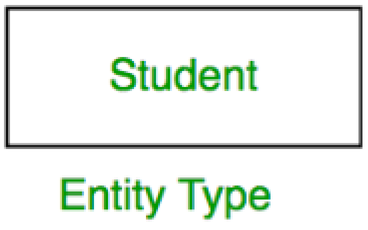
Components of ER Diagram

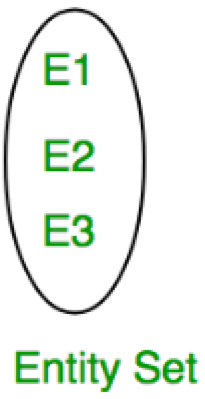
ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.



Entity: An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

Entity Set: An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:





1. Strong Entity

A Strong Entity is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined. These are called Weak Entity types.

Attributes

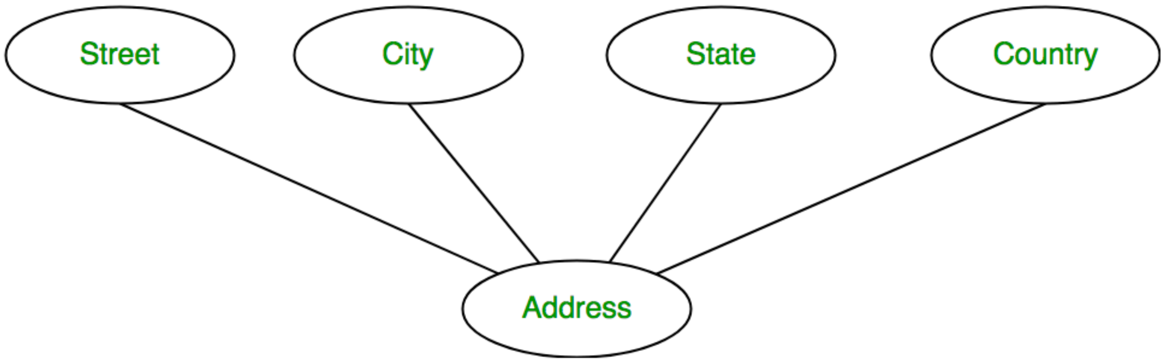
Attributes are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.

1. Key Attribute

The attribute which **uniquely identifies each entity** in the entity set is called the key attribute. For example, Roll_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.

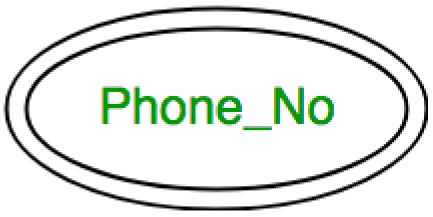
2. Composite Attribute

An attribute **composed of many other attributes** is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In the ER diagram, the composite attribute is represented by an oval comprising of ovals.



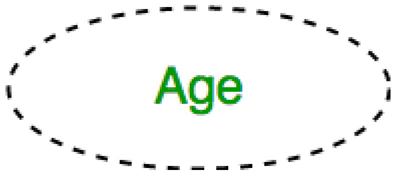
3. Multivalued Attribute

An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.

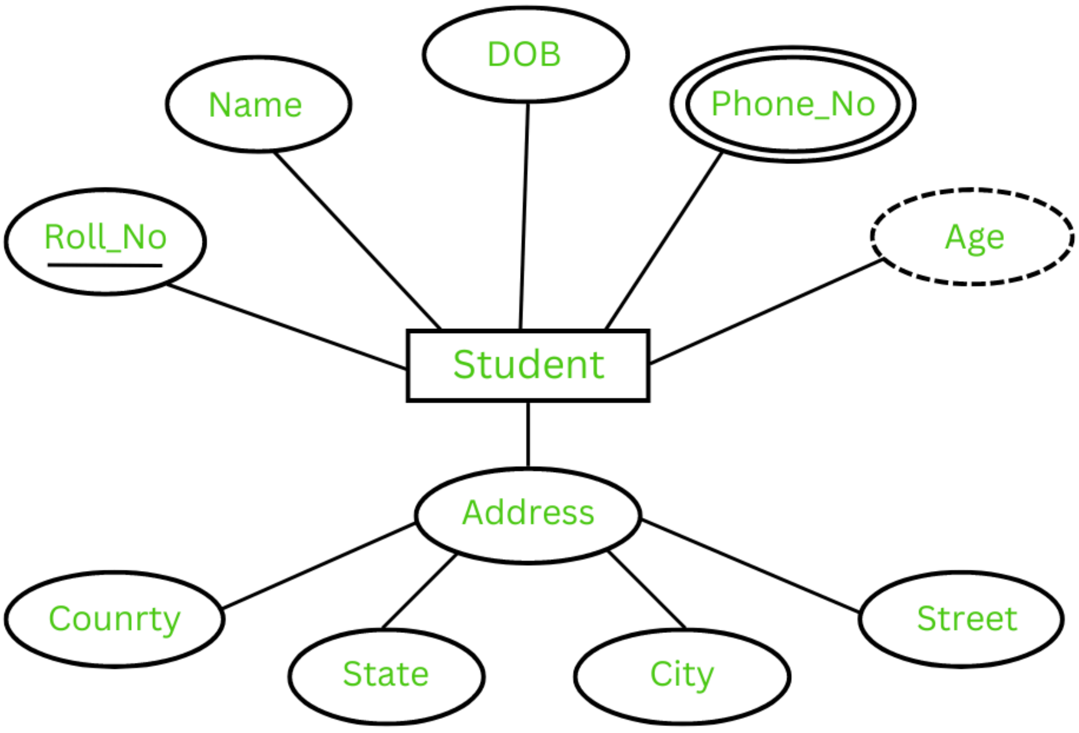


4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In the ER diagram, the derived attribute is represented by a dashed oval.

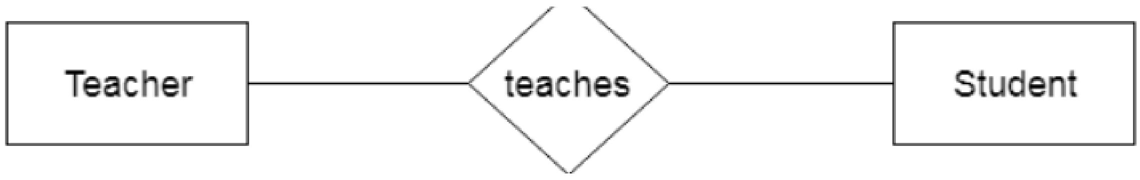


The Complete Entity Type Student with its Attributes can be represented as:



Relationship

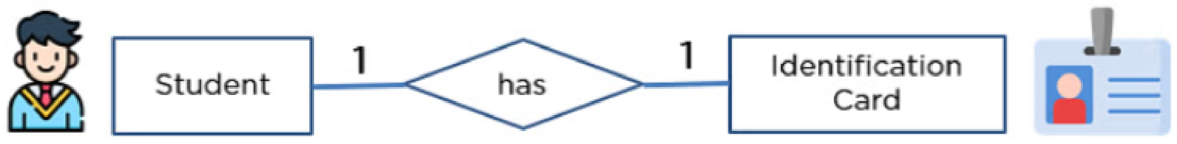
A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



Types of relationship are as follows:

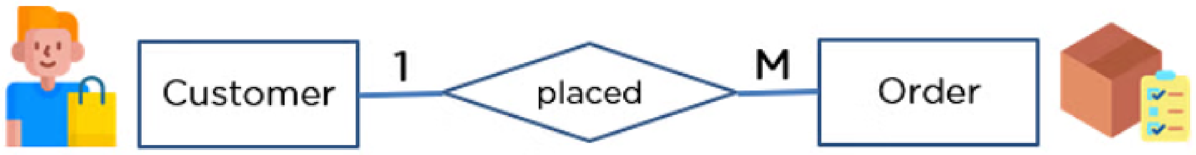
a) One-to-One Relationship: When a single element of an entity is associated with a single element of another entity, it is called a one-to-one relationship.

For example, a student has only one identification card and an identification card is given to one person.



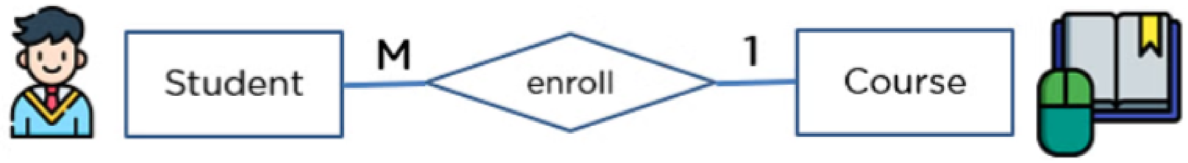
b) One-to-Many relationship: When more than one element of an entity is related to a single element of another entity, then it is called a many-to-one relationship.

For example, students have to opt for a single course, but a course can have many students.



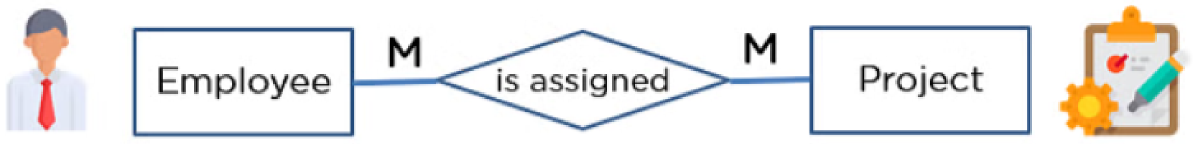
c) Many-to-One relationship: When more than one element of an entity is related to a single element of another entity, then it is called a many-to-one relationship.

For example, students have to opt for a single course, but a course can have many students.



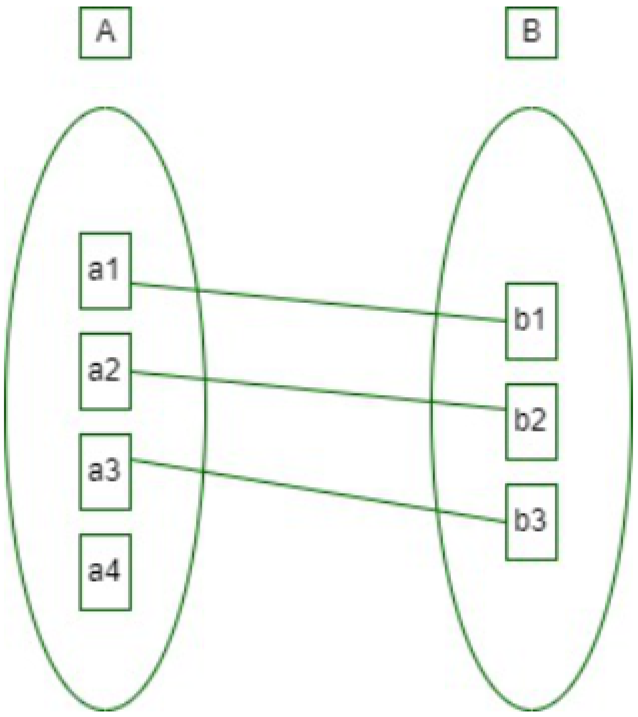
d) Many-to-Many relationship: When more than one element of an entity is associated with more than one element of another entity, this is called a many-to-many relationship.

For example, you can assign an employee to many projects and a project can have many employees.



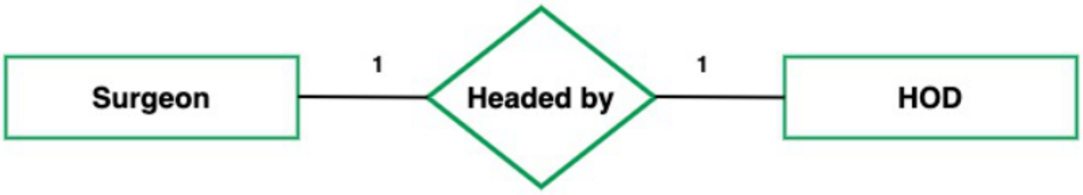
Mapping Cardinalities:

- ➔ The mapping cardinality or cardinality ratio means to denote the number of entities to which another entity can be linked through a certain relation set.
- ➔ Mapping cardinality is most useful in describing binary relation sets, although they can contribute to the description of relation sets containing more than two entity sets.
- ➔ Here, we will focus only on binary relation sets, which means we will find the relation between entity sets A and B for the set R.
- ➔ There are four types of Cardinality Mapping.
    1. **One-to-one(1-1):** In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say that a unit or item in B is connected to at most one unit or item in A.

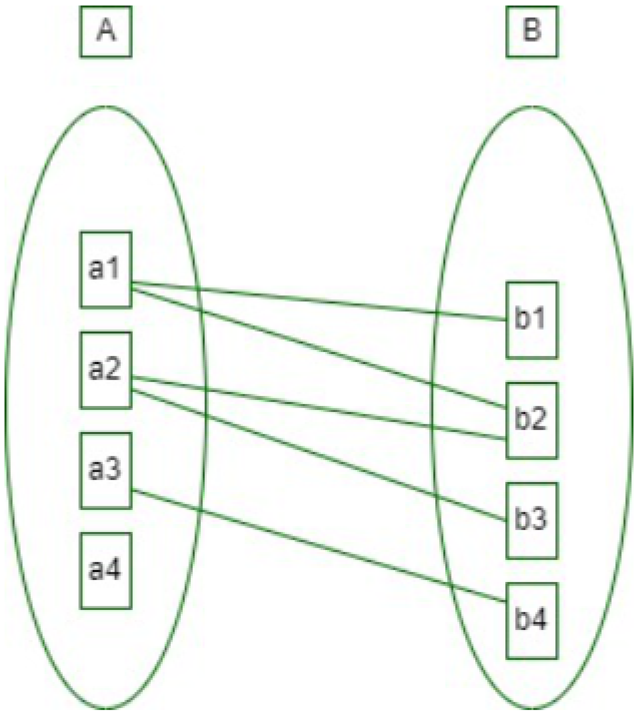


Example:

In a particular hospital, the surgeon department has one head of department. They both serve one-to-one relationships.

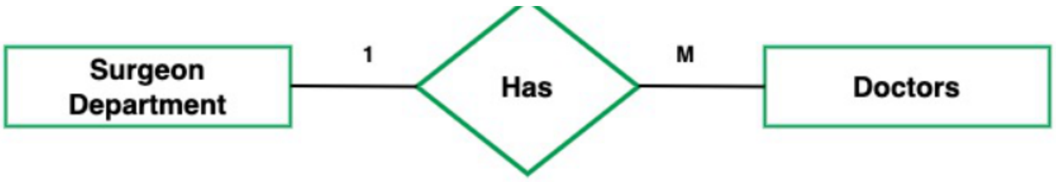


2. **One-to-many(1-N):** In this type of cardinality mapping, an entity in A is associated with any number of entities in B. Or we can say that one unit or item in B can be connected to at most one unit or item in A.

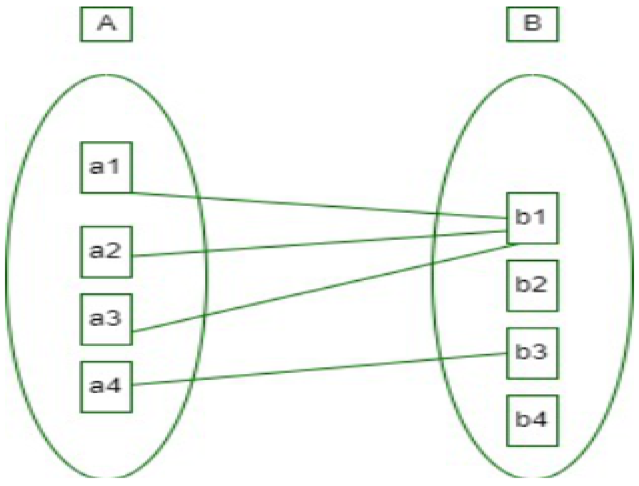


Example:

In a particular hospital, the surgeon department has multiple doctors. They serve one-to-many relationships.

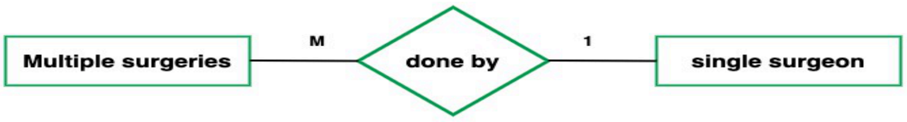


3. **Many-to-one(N-1):** In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say a unit or item in B can be associated with any number (zero or more) of entities or items in A.

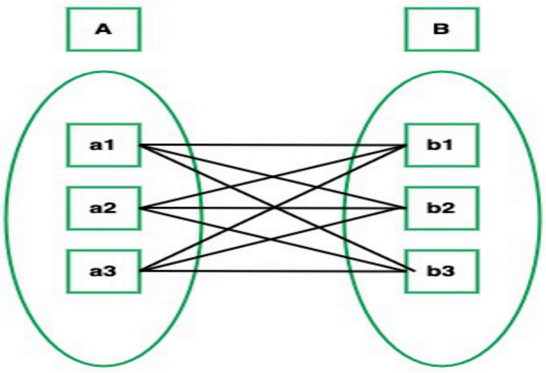


Example:

In a particular hospital, multiple surgeries are done by a single surgeon. Such a type of relationship is known as a many-to-one relationship.

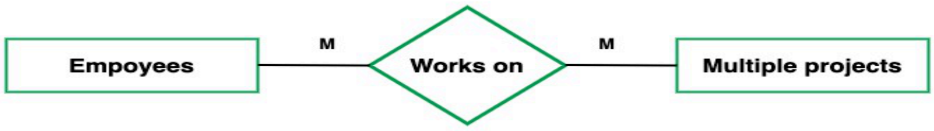


4. **Many-to-many(N-N):** In this type of cardinality mapping, an entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.



Example:

In a particular company, multiple people work on multiple projects. They serve many-to-many relationships.



The appropriate mapping cardinality for a particular relation set obviously depends on the real-world situation in which the relation set is modeled.

- If we have cardinality one-to-many or many to one then, we can mix relational tables with many involved tables.
- If the cardinality is many-to-many we cant mix any two tables.
- If we have a one-to-one relation and we have total participation of one entity then we can mix that entity with a relation table and if we have total participation of both entities then we can make one table by mixing two entities and their relation.

Participation Constraints:

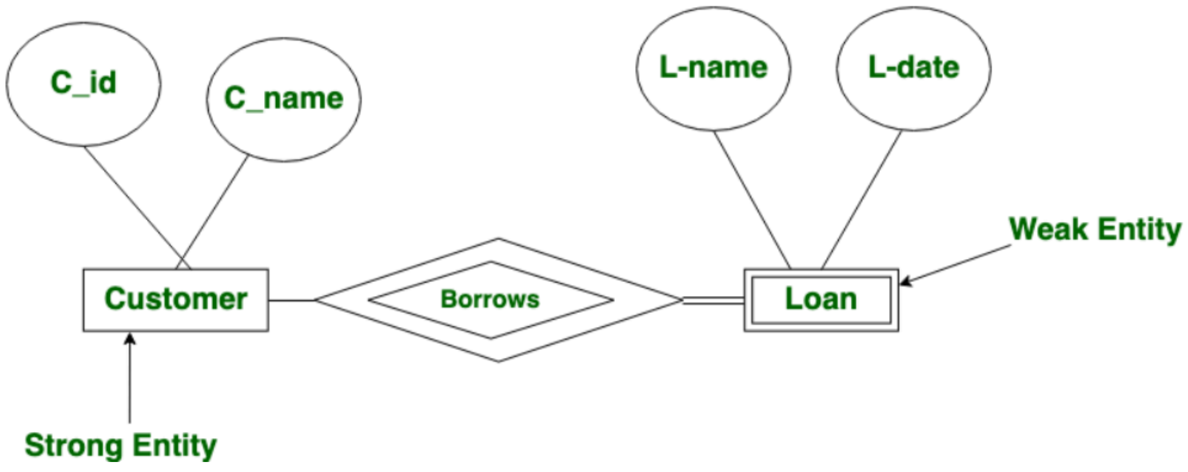
➔ Participation constraints refer to the rules that dictate whether every entity in one entity set must participate in a relationship with another entity set.
➔ They are essential in defining the relationship between entities and ensuring data integrity.

There are two main types of participation constraints:

1. **Total Participation (Mandatory Participation)**:
   - In a total participation constraint, every entity in the entity set must participate in at least one relationship instance in the related entity set.
   - This is denoted by a double line connecting the participating entity set in an ER diagram.
   - Example: In a university database, every student must be enrolled in at least one course. Therefore, the participation of every student in the "Enrollment" relationship with the "Course" entity set would be total.
2. **Partial Participation (Optional Participation)**:
   - In a partial participation constraint, some entities in the entity set may not participate in any relationship instance with the related entity set.
   - This is denoted by a single line connecting the participating entity set in an ER diagram.
   - Example: In a hospital database, not every patient necessarily has a medical record. Therefore, the participation of patients in the "Has Medical Record" relationship with the "Medical Record" entity set would be partial.

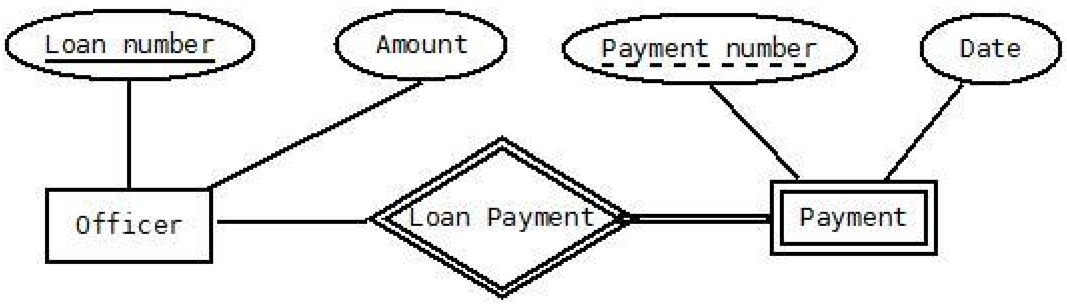
Weak Entity Sets:

- ➔ As the weak entities do not have any primary key, they cannot be identified on their own, so they depend on some other entity.
- ➔ Weak entity depends on a strong entity to ensure the existence of a weak entity.
- ➔ Weak entity does not have any primary key, It has a partial discriminator key(partial key).



- ➔ **Weak entities** are represented with a double **rectangular** box in the ER Diagram and the identifying relationships are represented with **double diamond**. Partial Key attributes are represented with **dotted lines.**

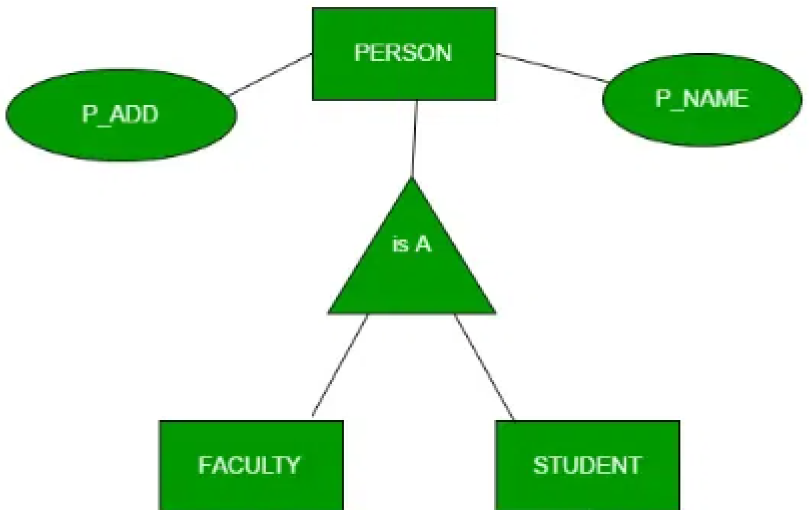
Example-1:
In the below ER Diagram, 'Payment' is the weak entity. 'Loan Payment' is the identifying relationship and 'Payment Number' is the partial key. Primary Key of the Loan along with the partial key would be used to identify the records.



Generalization:

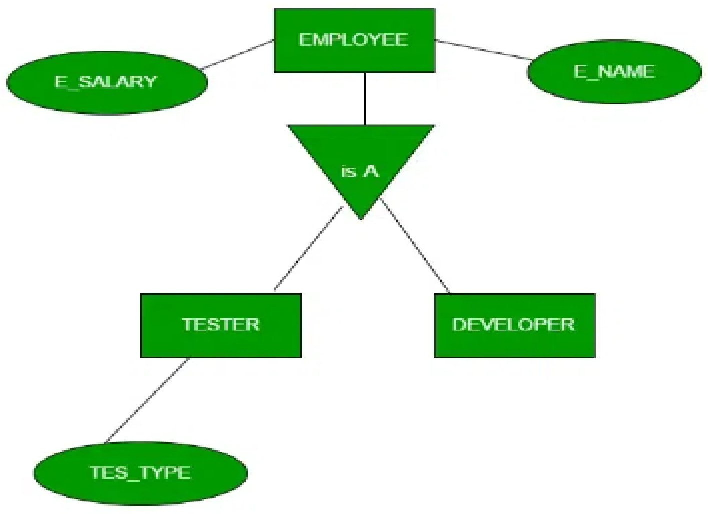
- ➔ Generalization is the process of extracting common properties from a set of entities and creating a generalized entity from it.
- ➔ It is a bottom-up approach in which two or more entities can be generalized to a higher-level entity if they have some attributes in common.

➔ For Example, STUDENT and FACULTY can be generalized to a higher-level entity called PERSON



<u>**Specialization:**</u>

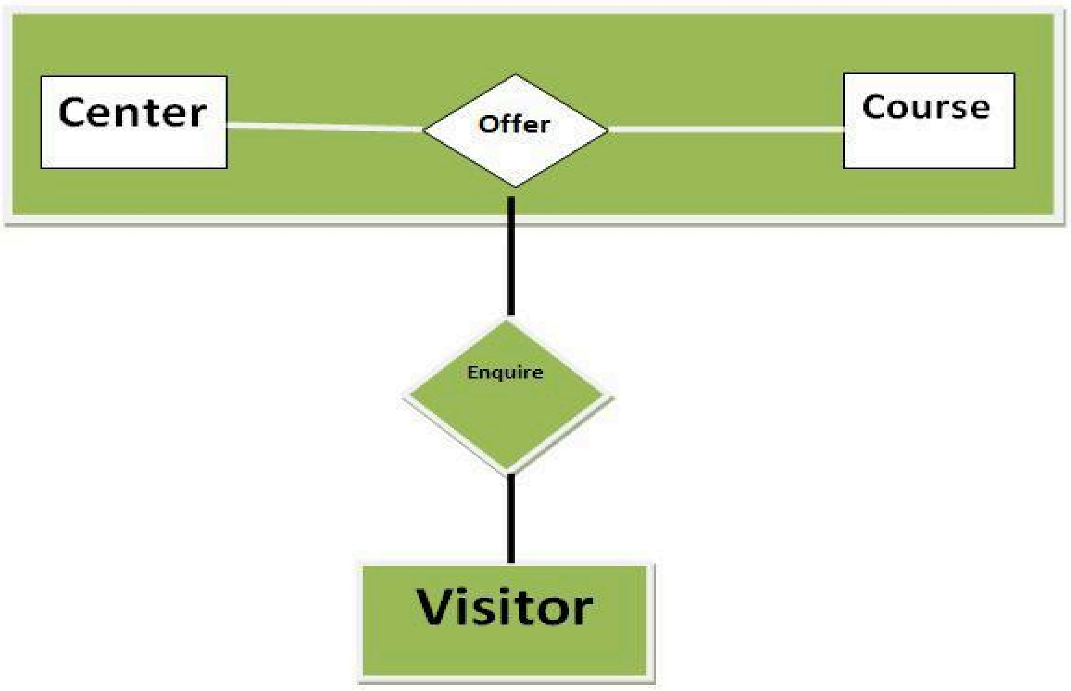
➔ In specialization, an entity is divided into sub-entities based on its characteristics.
➔ It is a top-down approach where the higher-level entity is specialized into two or more lower-level entities.
➔ For Example, an EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER, etc. In this case, common attributes like E_NAME, E_SAL, etc. become part of a higher entity (EMPLOYEE), and specialized attributes like TES_TYPE become part of a specialized entity (TESTER).



Aggregation:

➔ Aggregation in DBMS is the concept where the relation between 2 different entities is considered as a single entity.

➔ We can say that Aggregation is used to define relationships among relationships.

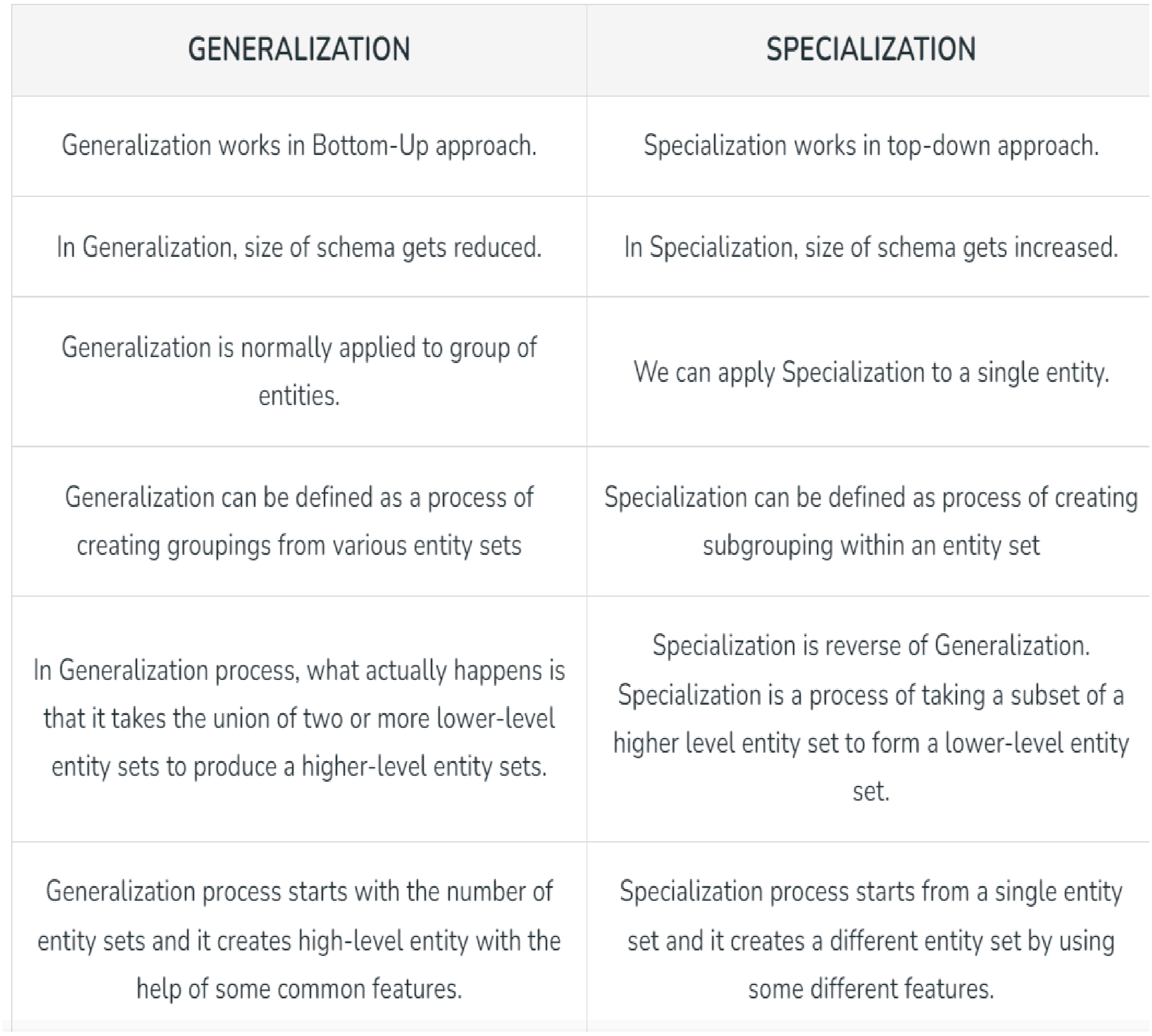


IMPORTANT NOTES:

1. **ER MODEL OF HOSPITAL MANAGEMENT SYSTEM**
2. **ER MODEL OF LIBRARY SYSTEM**

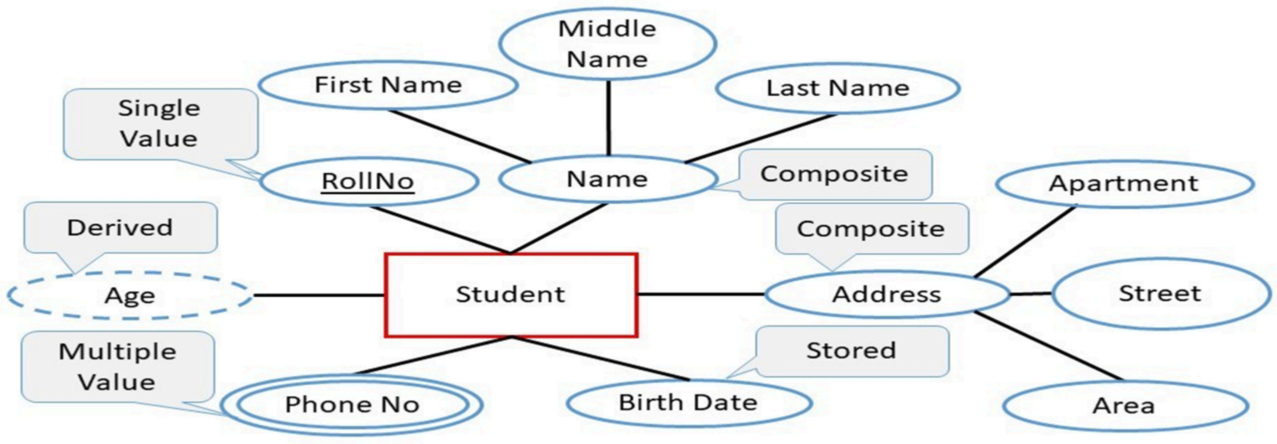
Faculty: Shubham Upadhyay

Difference between Generalization and Specialization :



GENERALIZATION	SPECIALIZATION
Generalization works in Bottom-Up approach.	Specialization works in top-down approach.
In Generalization, size of schema gets reduced.	In Specialization, size of schema gets increased.
Generalization is normally applied to group of entities.	We can apply Specialization to a single entity.
Generalization can be defined as a process of creating groupings from various entity sets	Specialization can be defined as process of creating subgrouping within an entity set
In Generalization process, what actually happens is that it takes the union of two or more lower-level entity sets to produce a higher-level entity sets.	Specialization is reverse of Generalization. Specialization is a process of taking a subset of a higher level entity set to form a lower-level entity set.
Generalization process starts with the number of entity sets and it creates high-level entity with the help of some common features.	Specialization process starts from a single entity set and it creates a different entity set by using some different features.

Entity with all types of Attributes



Faculty: Shubham Upadhyay