Unit – 2

Data Modelling Using Entity-Relational Model and Relational Model

High Level Conceptual Data Model For Database Design

- The high-level conceptual data models serve as blueprints for designing databases, providing a common language and framework for communication among stakeholders, designers, and developers during the database design process.
- They help capture the essence of the data requirements and facilitate the creation of detailed logical and physical data models for implementation.
- Some common high-level conceptual data models used in database design are listed below:
  - ✓ **Entity-Relationship Model (ER Model):** It represents the data in terms of entities, attributes, and relationships between entities.
  - ✓ **Object-Oriented Data Model (OODM):** It represents data as objects with attributes and methods, encapsulating data and behavior within a single entity.
  - ✓ **Hierarchical Data Model:** It organizes data in a hierarchical tree-like structure, where each record has a parent-child relationship with other records.
  - ✓ **Network Data Model:** The Network Data Model extends the hierarchical model by allowing each record to have multiple parent and child records, creating a network-like structure.

Database Design

- Database design is the process of creating a detailed conceptual, logical, and physical structure for organizing and storing data in a database management system (DBMS).

Steps involved in Database Design

- Requirements Analysis
- Conceptual Design
- Logical Design
- Physical Design
- Implementation
- Testing and
- Maintenance

Requirement Analysis

- Identify and understand the data requirements of the organization or application.
- Gather input from stakeholders, including end-users, managers, and system developers.
- Define the scope of the database, including the types of data to be stored.

Conceptual Design:

- Create a high-level conceptual model of the database using techniques such as Entity-Relationship (ER) modeling.
- Identify entities (objects), attributes (properties), and relationships between entities.
- Define constraints, such as primary keys, foreign keys, and cardinality constraints.

Logical Design:

- Translate the conceptual model into a logical model that can be implemented in the chosen DBMS.
- Define tables, columns, data types, and relationships based on the conceptual model.
- Normalize the database schema to minimize redundancy and dependency issues.

Physical Design:

- Translate the logical model into a physical database schema optimized for storage and performance.
- Define storage structures, such as tablespaces, data files, and indexes, based on the characteristics of the DBMS and underlying hardware.

Implementation:

- Create the database schema and tables using DDL statements in the chosen DBMS.
- Populate the database with initial data using DML statements.
- Define security policies, access controls, and backup and recovery procedures to protect the database and ensure data integrity.

Testing:

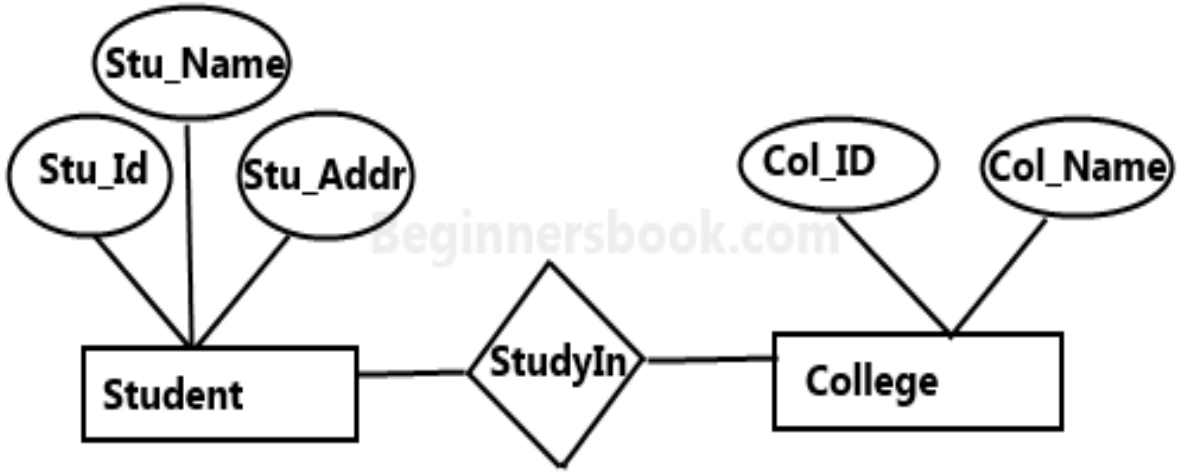
- Test the database design and implementation to ensure that it meets the specified requirements and performs efficiently under expected workloads.

Maintenance:

- Monitor and maintain the database over time to ensure ongoing performance, security, and reliability.
- database design to changing needs and technological advancements.

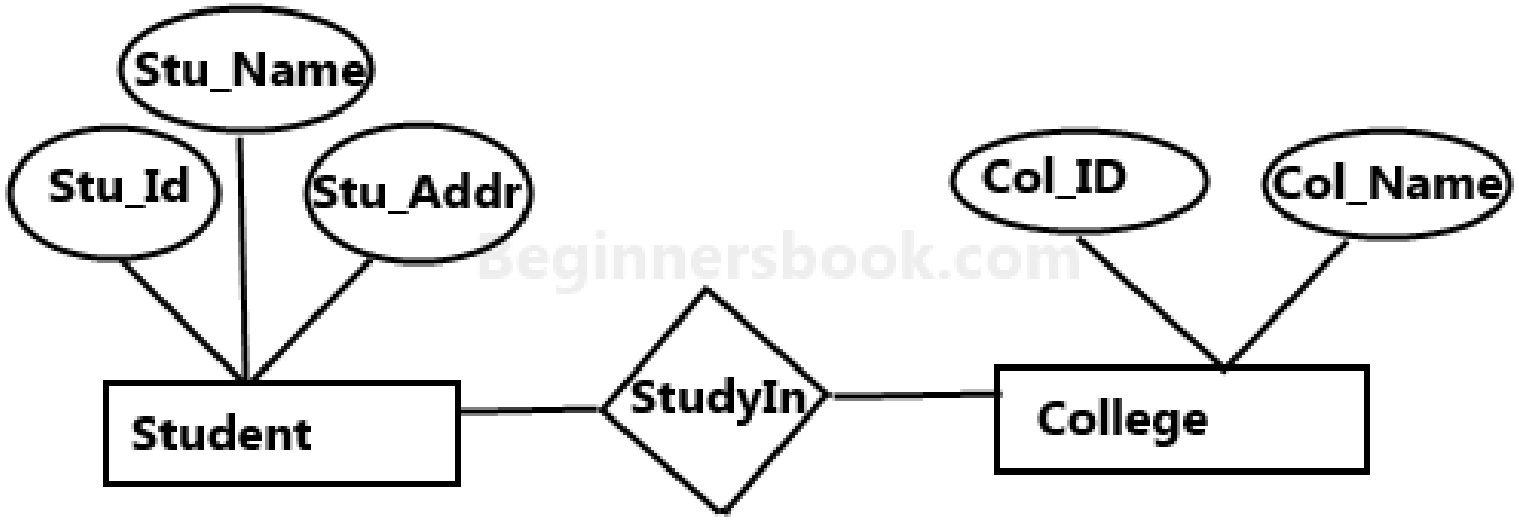
ER Diagram:

- **ER Diagram** stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database.
- In other words, ER diagrams help to explain the logical structure of databases.
- ER diagrams are created based on three basic concepts: entities, attributes and relationships.



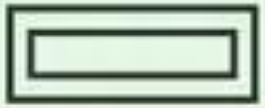
Need for ER diagram:

ER diagram shows the complete logical structure of a database.

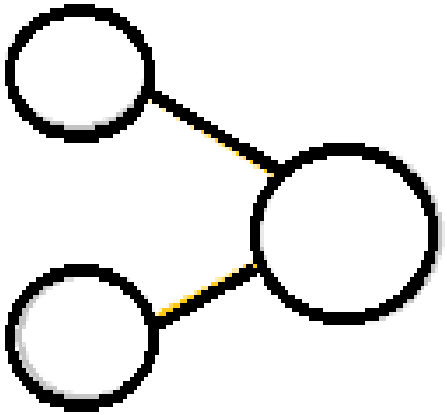


Sample E-R Diagram

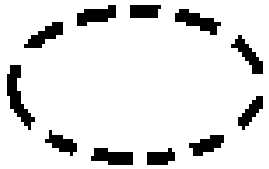
Symbols Used in ER Diagram

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

Key Attribute /
Single-value attribute



Composite Attribute



Derived Attribute

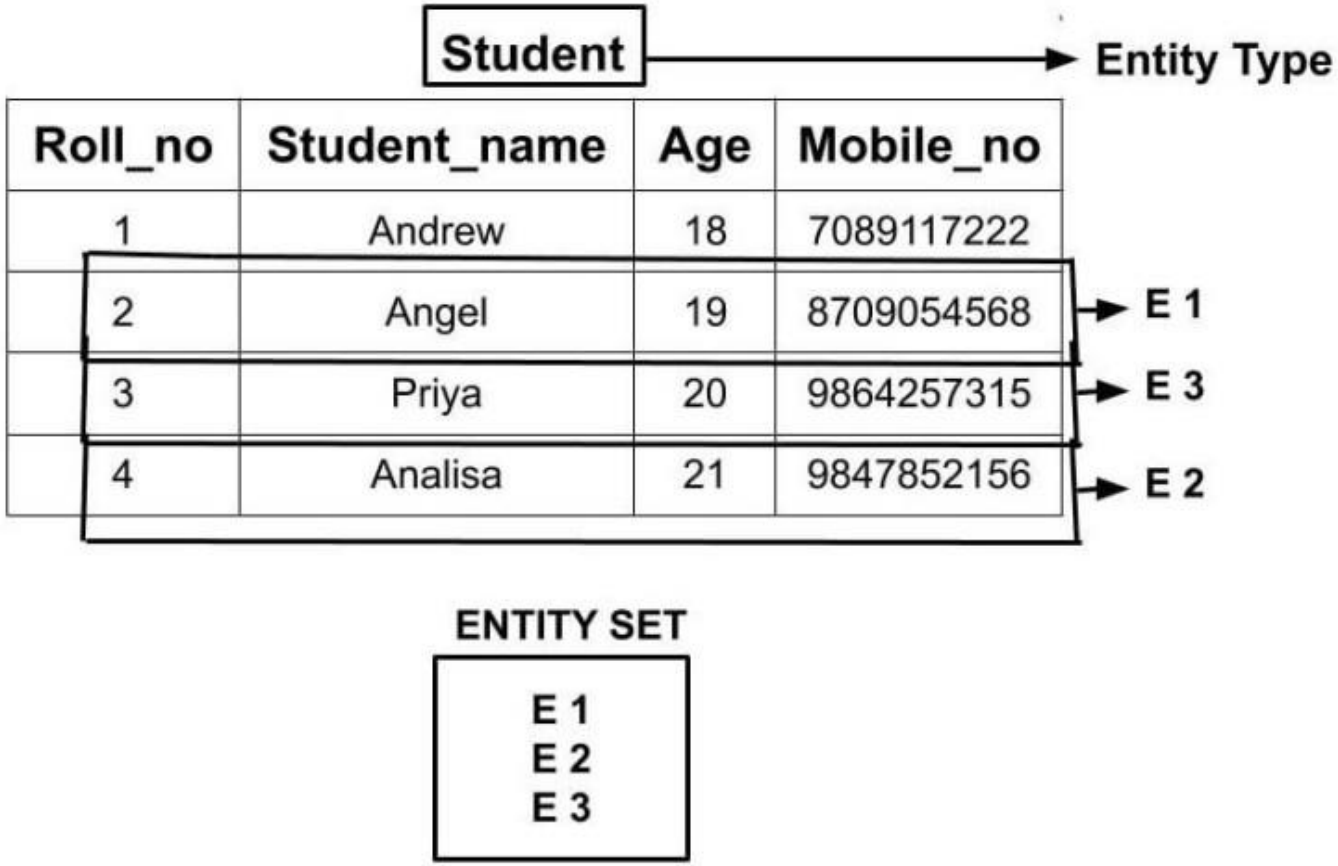
Symbols Used In ER Diagram

Entity

- An entity in DBMS (Database management System) is a real-world thing or a real-world object which is distinguishable from other objects in the real world.
- It is represented by the rectangle shape .

Entity Set

An entity set in DBMS is a set that collectively represents a group of entities of a similar type.



Attributes

- In a database management system (DBMS), an attribute is a piece of data that describes an entity.
- The property of the entity is called attributes.

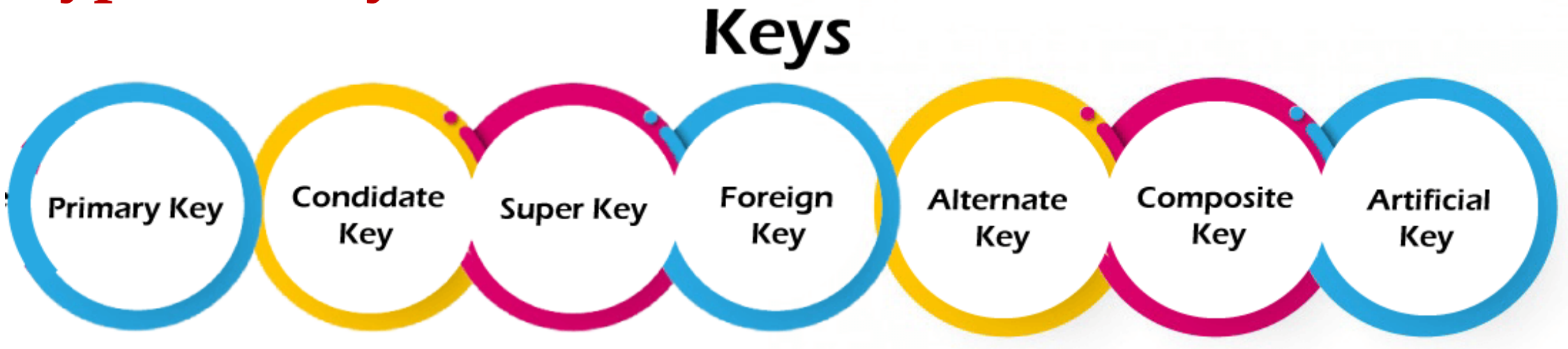
Roll No.	Name	Course
CS08	Steive	Comp. Sci.
EE54	Jhoson	Electronics
B12	Eva	Biology
F32	Jhoson	Finance
M26	Erica	Maths

Student Table

Keys:

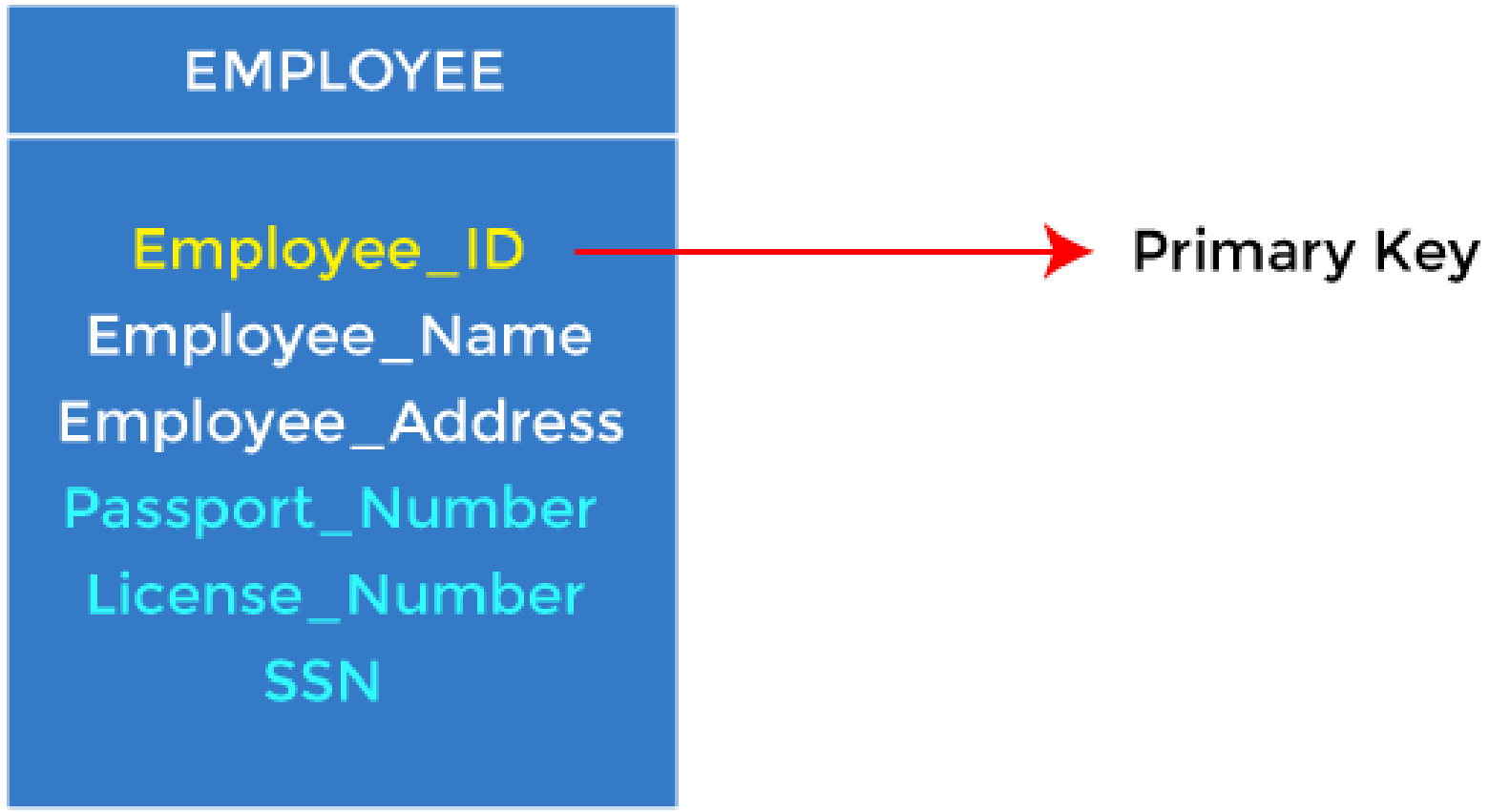
- DBMS keys is an attribute or set of an attribute which helps to identify a row in a relation(table).
- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

Types of Keys:



Primary Key

A primary key is the column or columns that contain values that uniquely identify each row in a table.

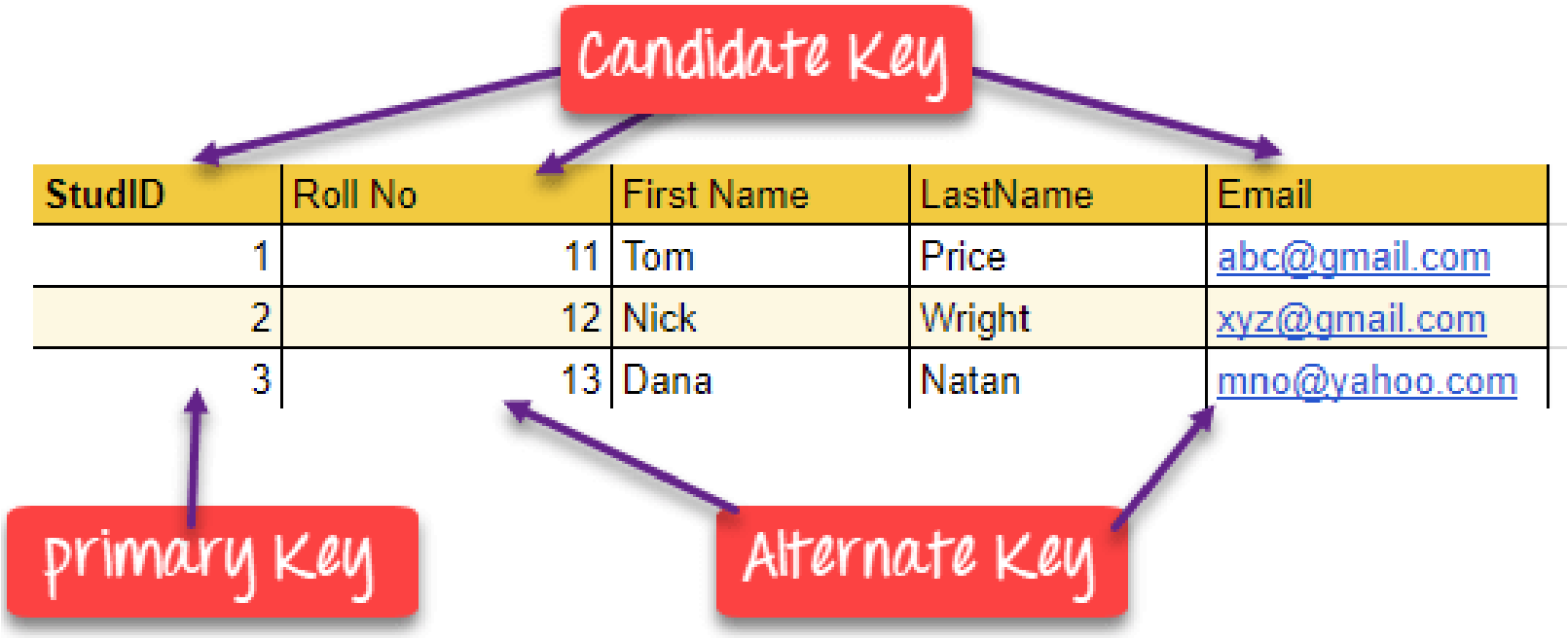


The properties of primary key

- It enforces uniqueness by not accepting any duplicate values.
- A primary key uniquely identifies each field.
- A table can only take one primary key.
- Primary columns have a maximum length of 900 bytes.
- A primary key column cannot accept null values.
- A single-column primary key is a simple one. The one consisting of multiple columns is called a composite primary key
- Attributes which are part of a primary key are known as **Prime attributes**.

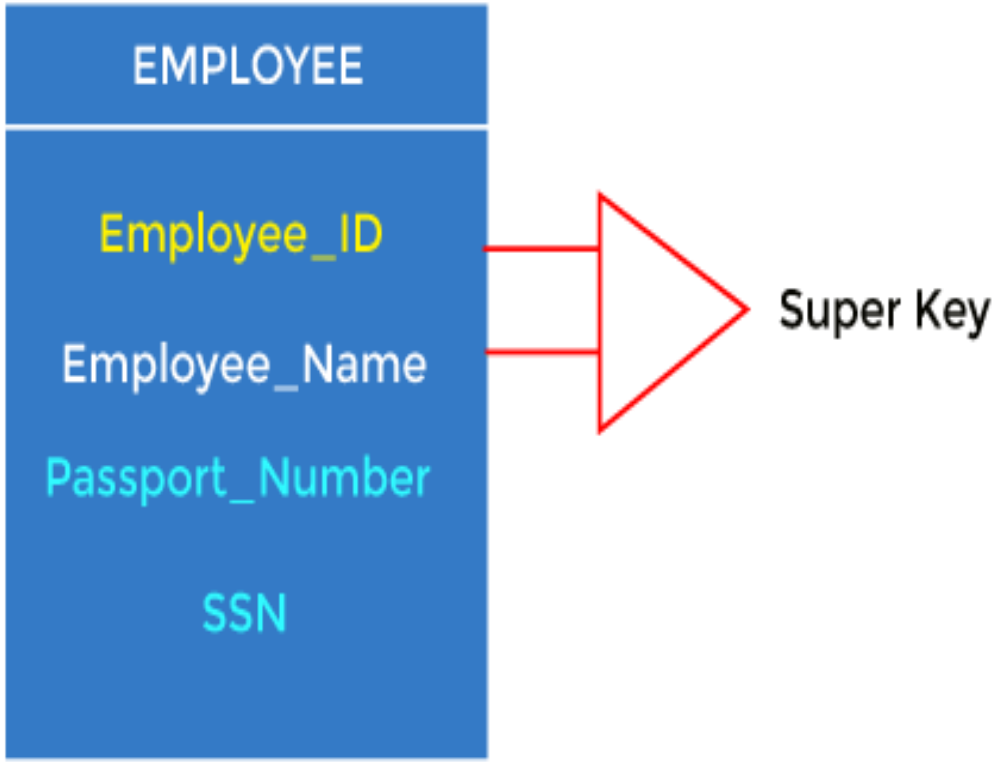
Candidate Key

- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- The candidate keys are as strong as the primary key.



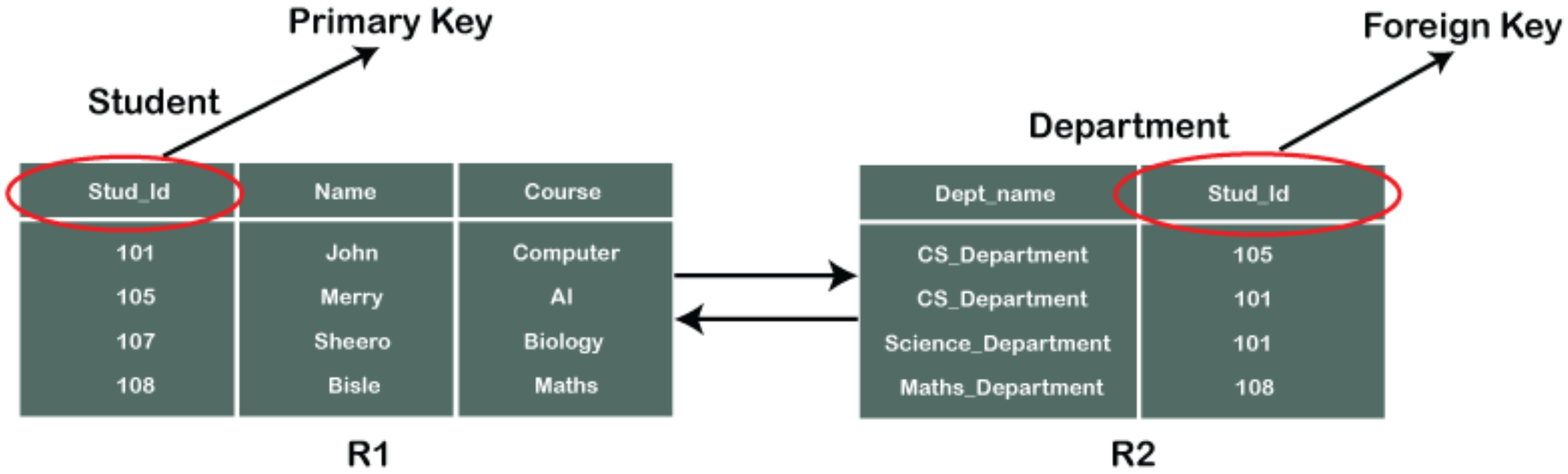
Super Key

- Super key is an attribute set that can uniquely identify a tuple.
- Super Key can contain multiple attributes that might not be able to identify tuples in a table independently, but when grouped with certain keys, they can identify tuples uniquely.



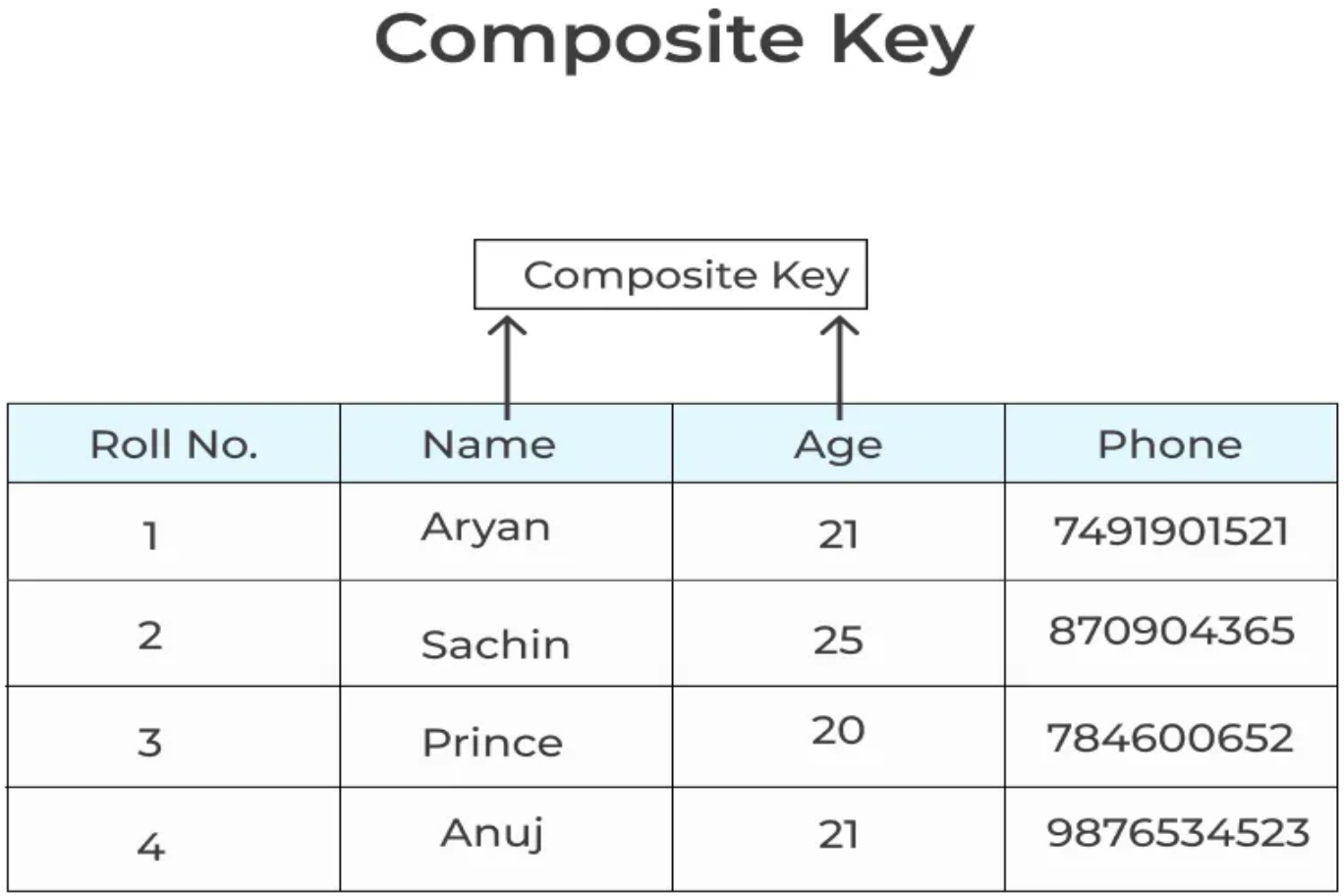
Foreign Key

- A foreign key is a column which is added to create a relationship with another table.
- Foreign keys help us to maintain data integrity and also allows navigation between two different instances of an entity.
- It further means that if any attribute is set as a primary key attribute will work in another table as a foreign key attribute.



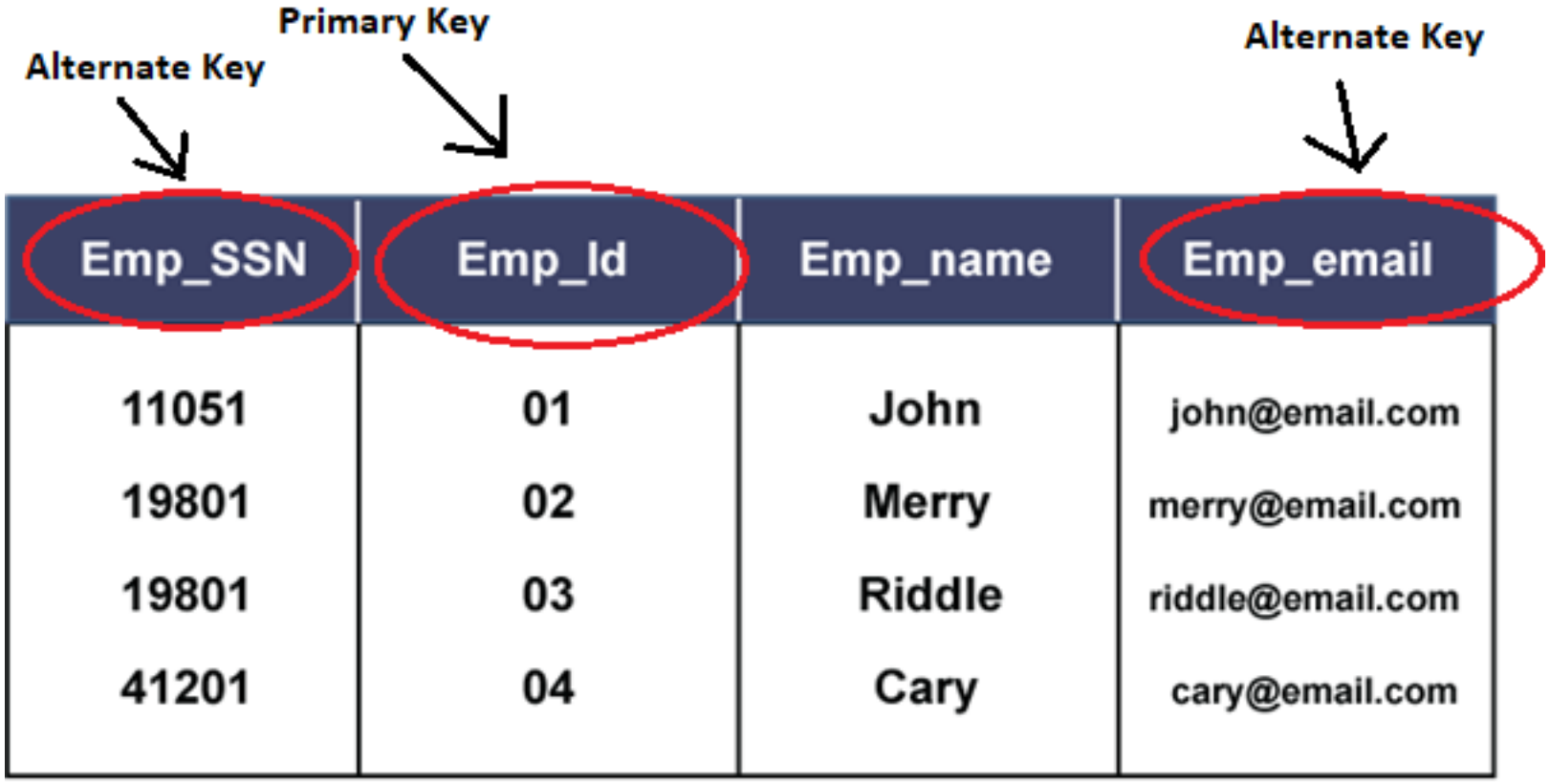
Composite Key

- A key which has multiple attributes to uniquely identify rows in a table is called a composite key.



Alternate Key

- All the keys which are not primary key are called an alternate key.
- It is a key which is currently not primary key.

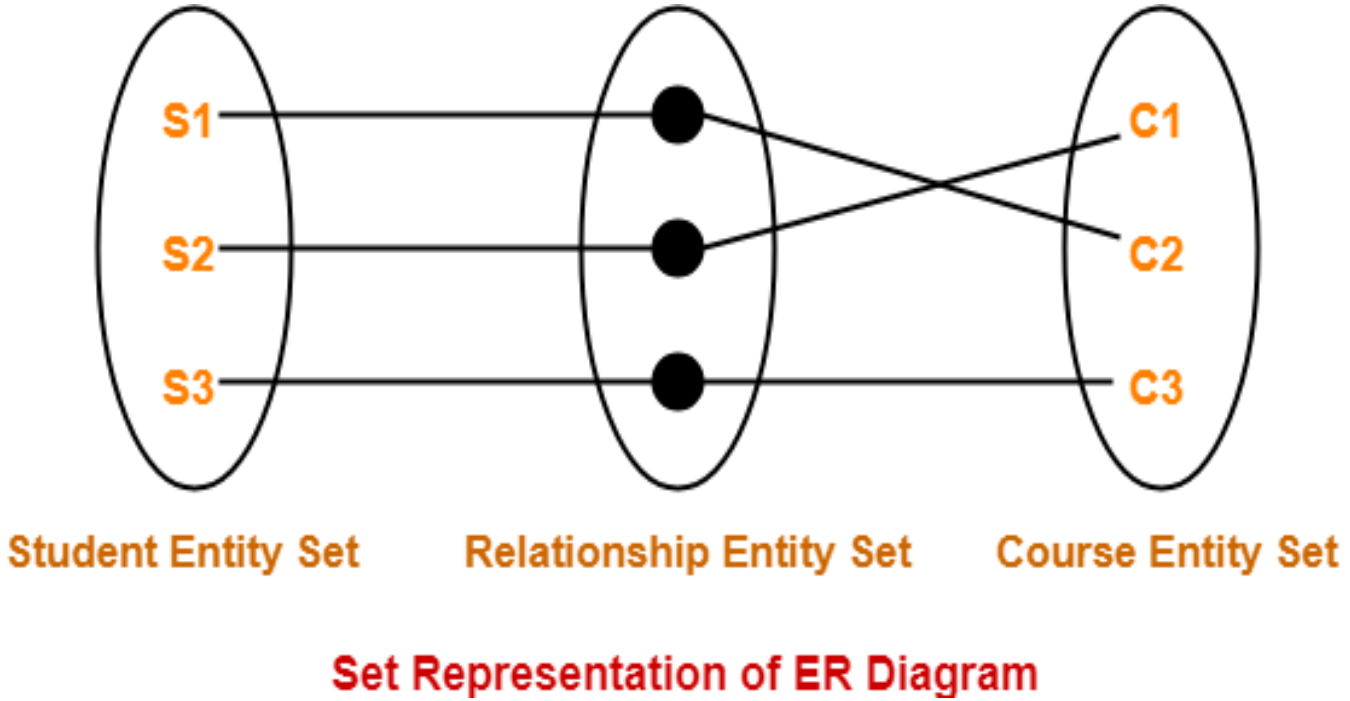


Relationship

- A relationship in DBMS is the way in which two or more data sets are linked, i.e., any association between two entity types is called a relationship.
- The entity takes part in the relationship, and it is represented by a diamond shape.

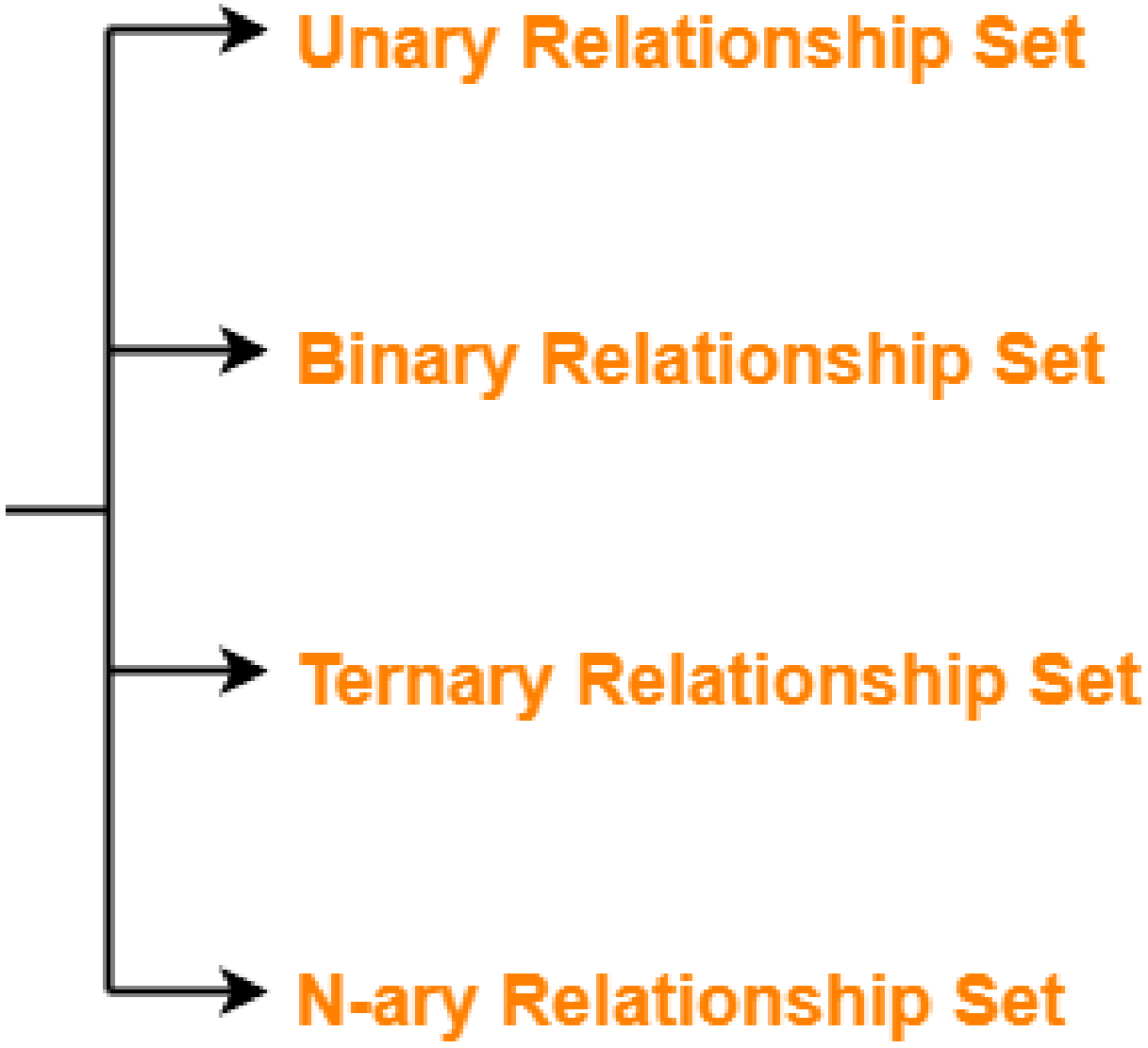
Relationship Set

A relationship set is a set of relationships of the same type.



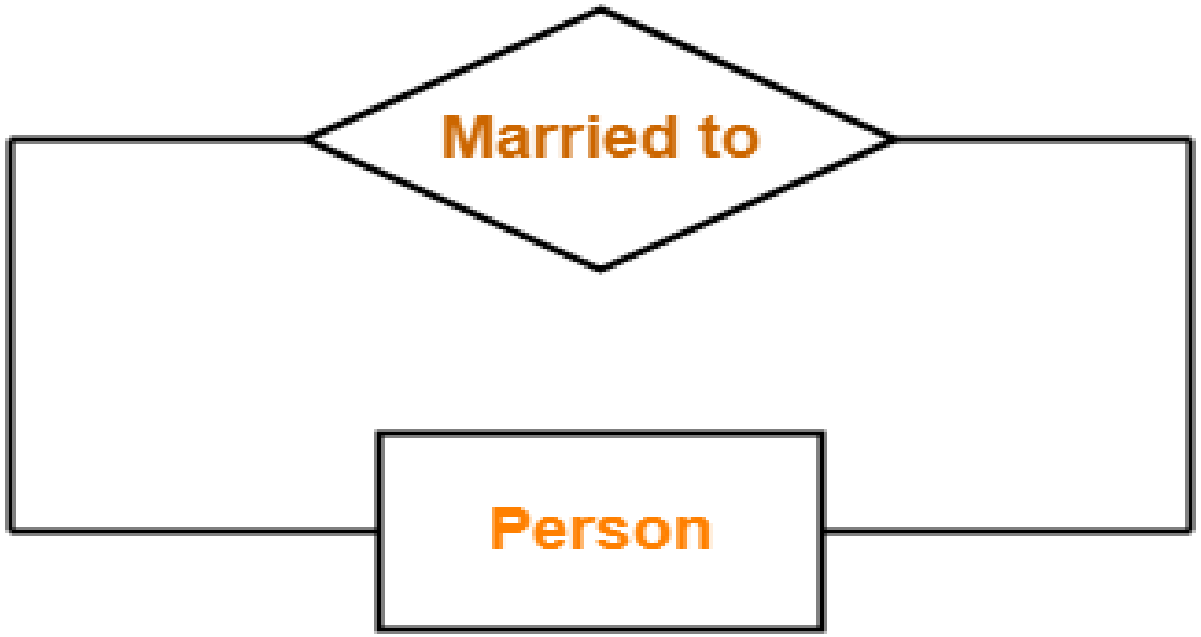
Set Representation of ER Diagram

Type of Relationship Sets
Or
Degree of Relationship



Unary Relationship Set

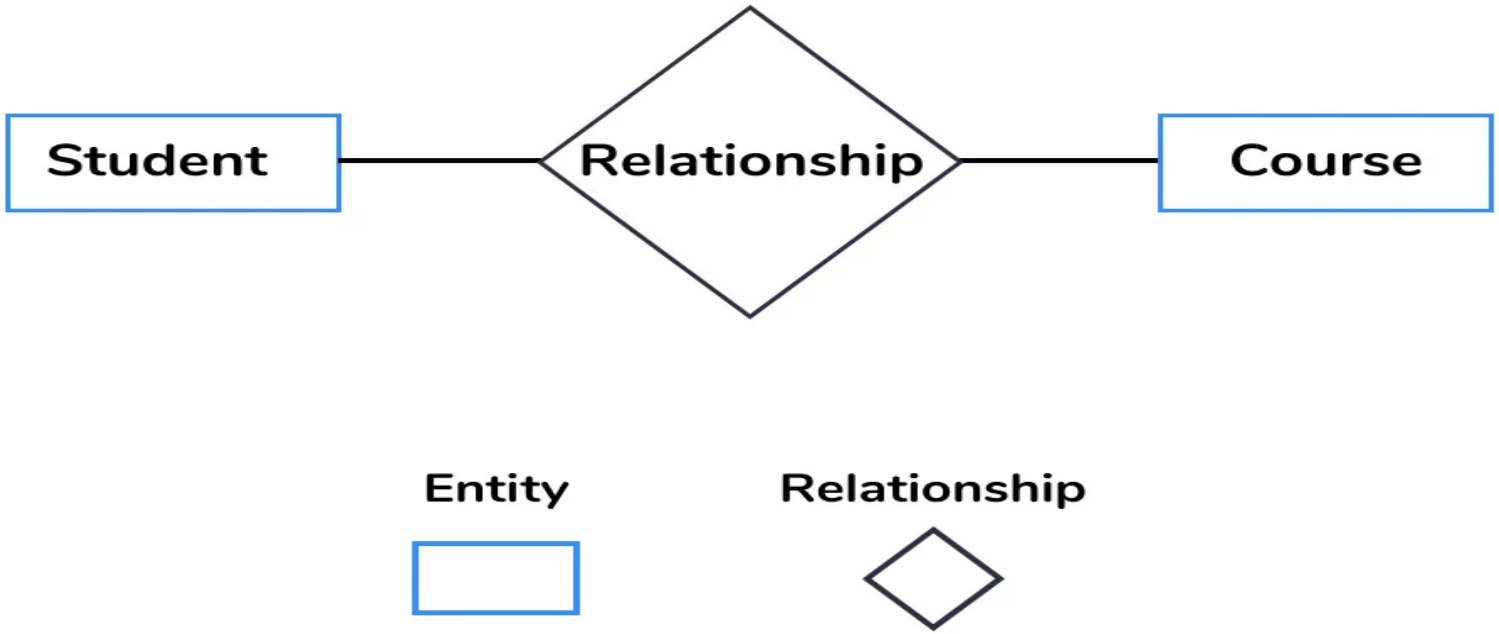
Unary Relationship is the association with the same entity between the same instances represented by same role group.



Unary Relationship Set

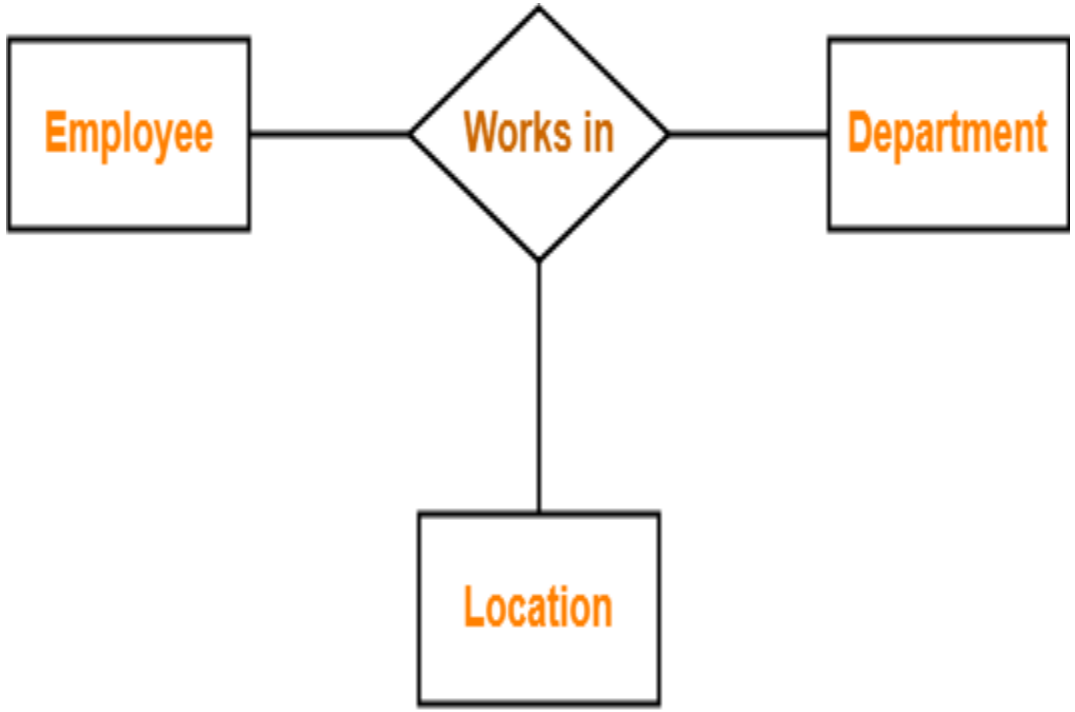
Binary Relationship Set

In a relation when two entity sets are participating then such type of relationship is known as a binary relationship.



Ternary Relationship Set

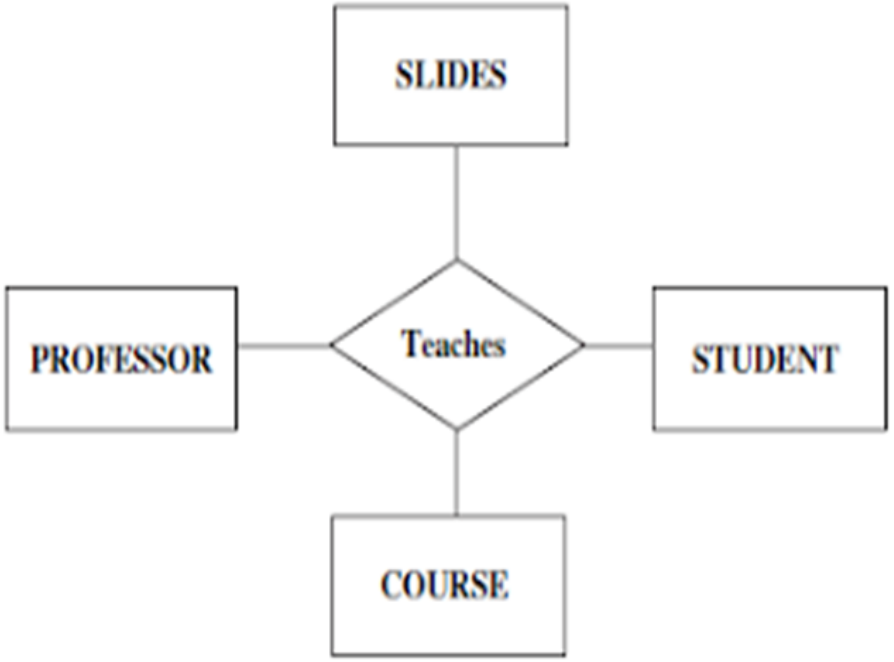
- A ternary relationship is an association among three entities.
- This type of relationship is required when binary relationships are not sufficient to accurately describe the semantics of the association.
- The ternary relationship construct is a single diamond connected to three entities as shown in Figure.



Ternary Relationship Set

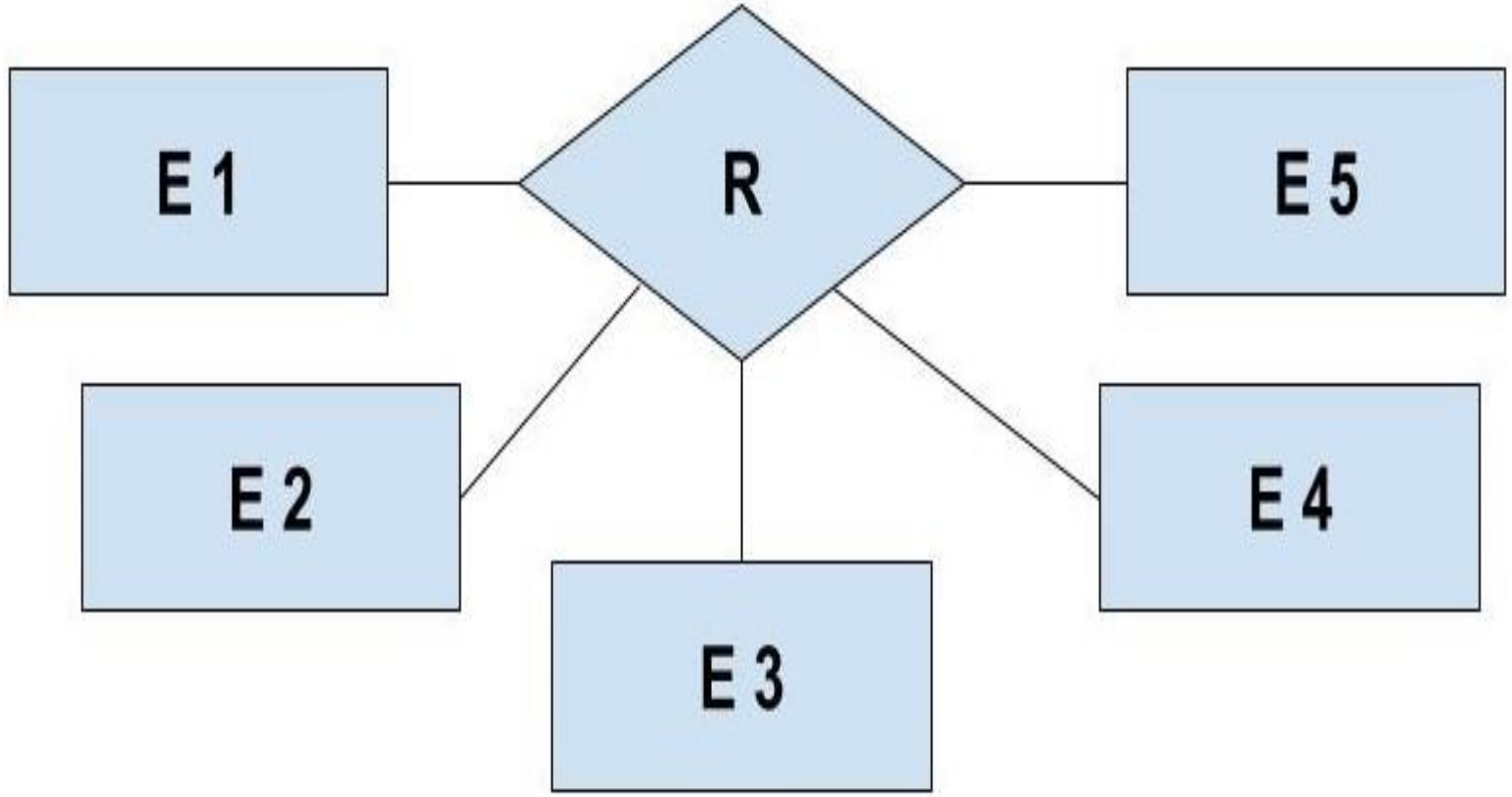
Quaternary Relationship Set

- Quaternary relationship involves four entity sets and represents an association among them.
- It is less common than ternary relationships but can occur in domains where complex interactions need to be modeled.



N-ary Relationship Set

When a large number of entity sets are participating in a relationship, then such type of relationship is called an n-ary relationship.



N-ary Relationship

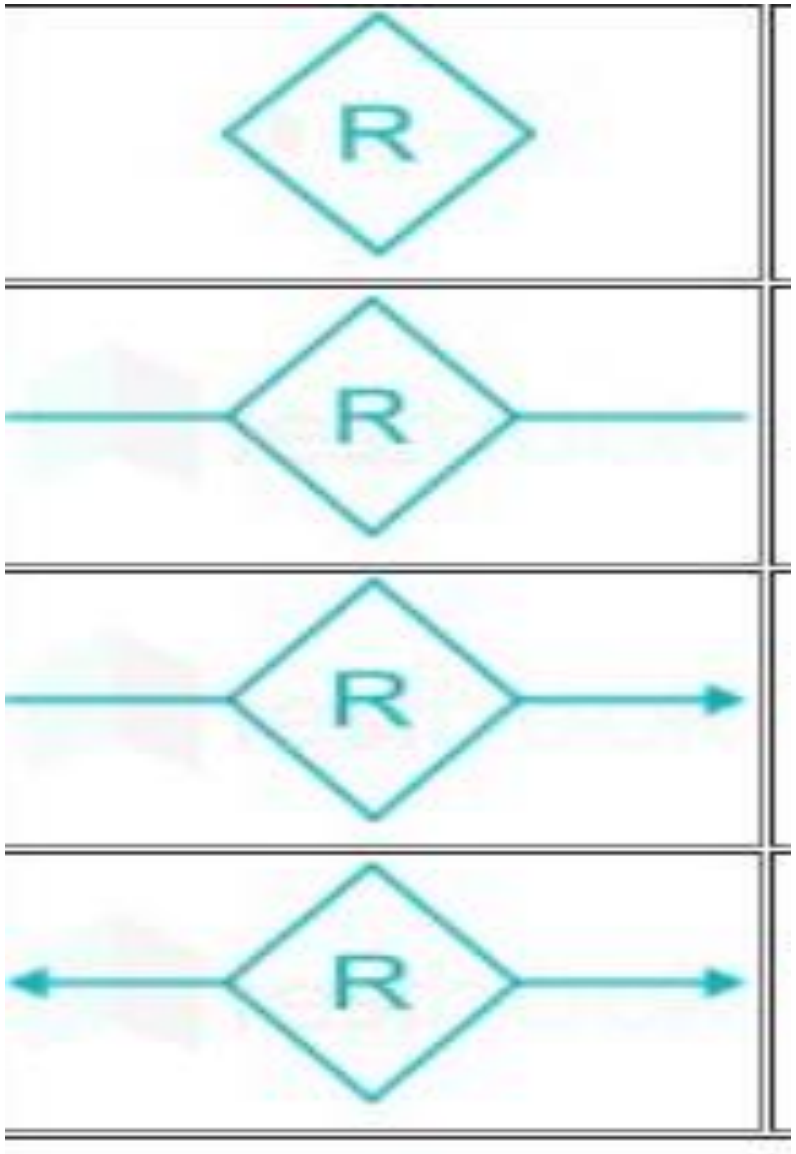
Relationship Types of Degree Higher Than Two

- In ER model, relationships of degree higher than two are less common but still exist and can be represented to capture complex associations between entities.
- Examples:
  - Ternary Relationship Set (Degree – 3 Relationship)
  - Quaternary Relationship (Degree-4 Relationship):
  - N-Ary Relationship Set

Mapping Constraints/ Cardinality Ratio

- Whenever an attribute of one entity type refers to another entity type, then some relationship exist between them.
- A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set. It is most useful in describing the relationship sets that involve more than two entity sets.
- For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:
    - ❑ One to one (1:1)
    - ❑ One to many (1:M)
    - ❑ Many to one (M:1)
    - ❑ Many to many (M:N)

Representation of Mapping Cardinalities in E-R diagram



Relationship

Many-to-Many

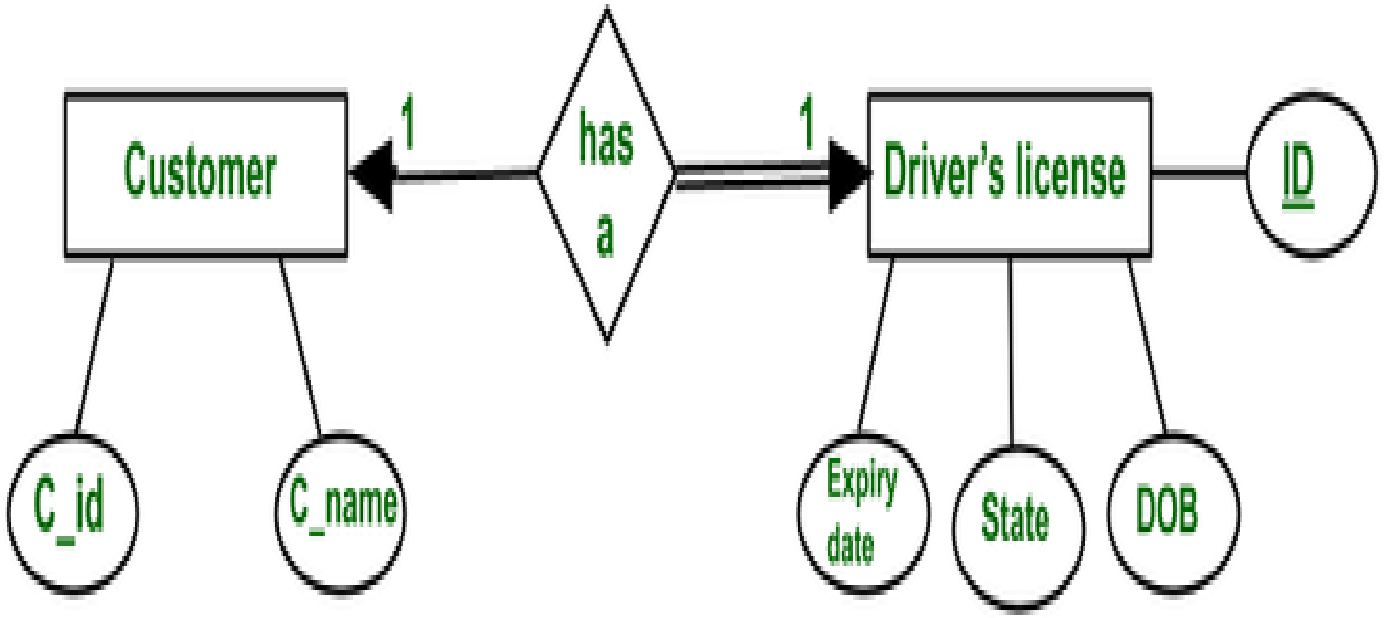
Many-to-One

One-to-One

1. One to One Relationship(1:1) :

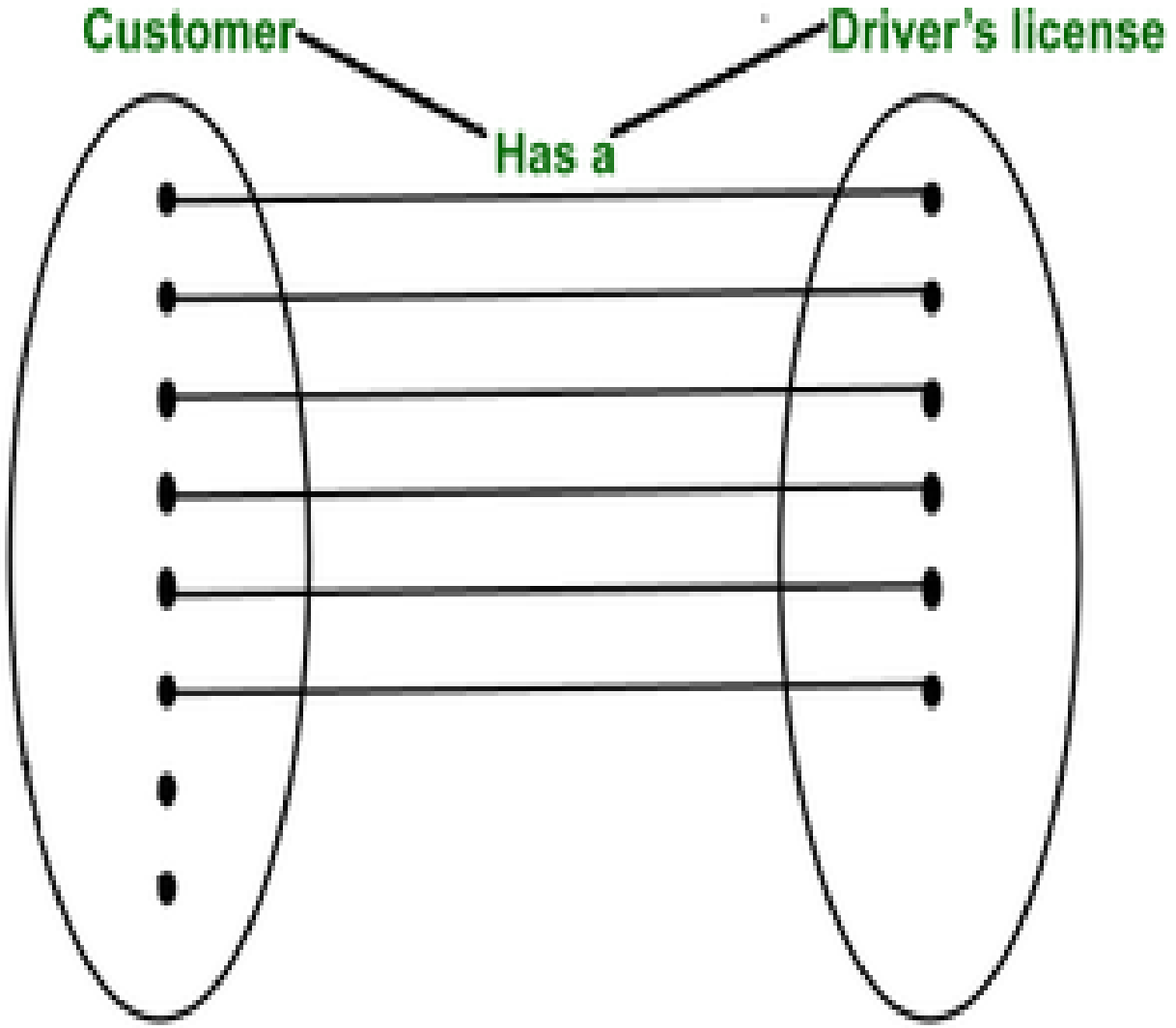
It is represented using an arrow(→←)(There can be many notations possible for the ER diagram).

- In this ER diagram, both entities customer and driving license having an arrow which means the entity Customer is participating in the relation “has a” in a one-to-one fashion.
- It could be read as ‘Each customer has exactly one driving license and every driving license is associated with exactly one customer.



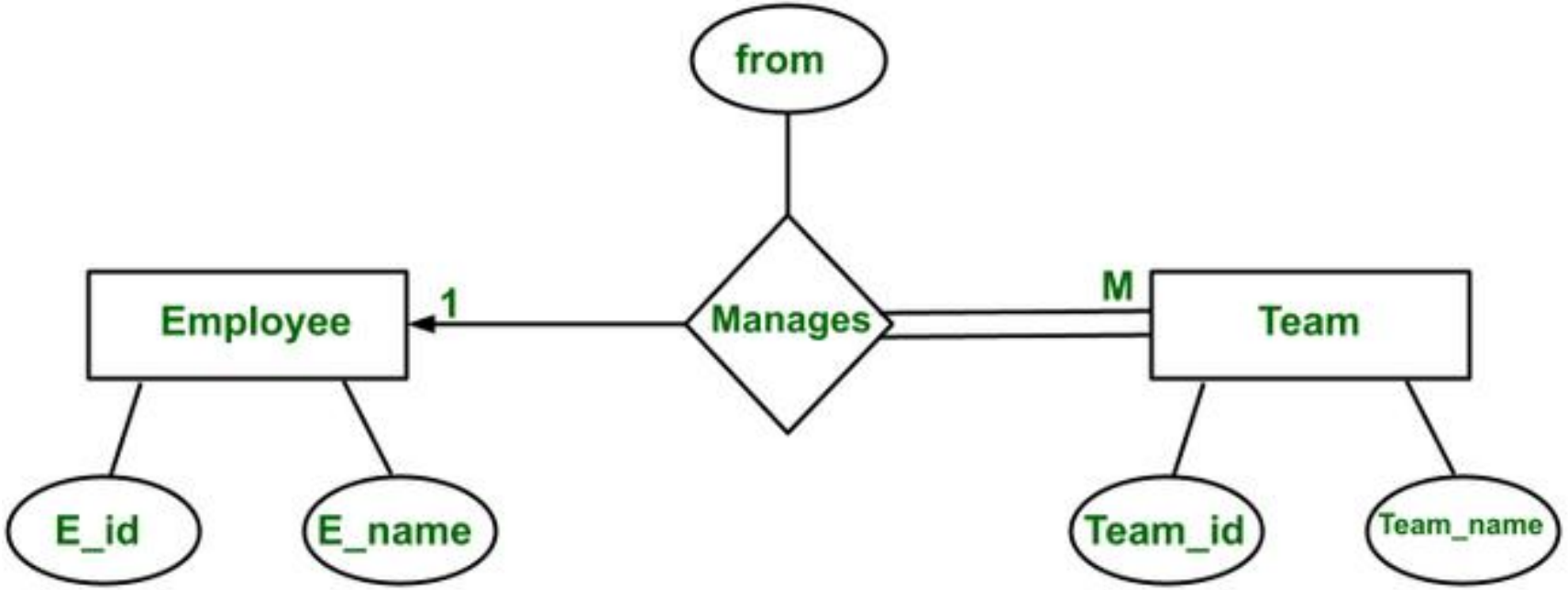
The set-theoretic perspective of the ER diagram is

- There may be customers who do not have a credit card, but every credit card is associated with exactly one customer.
- Therefore, the entity customer has total participation in a relation.



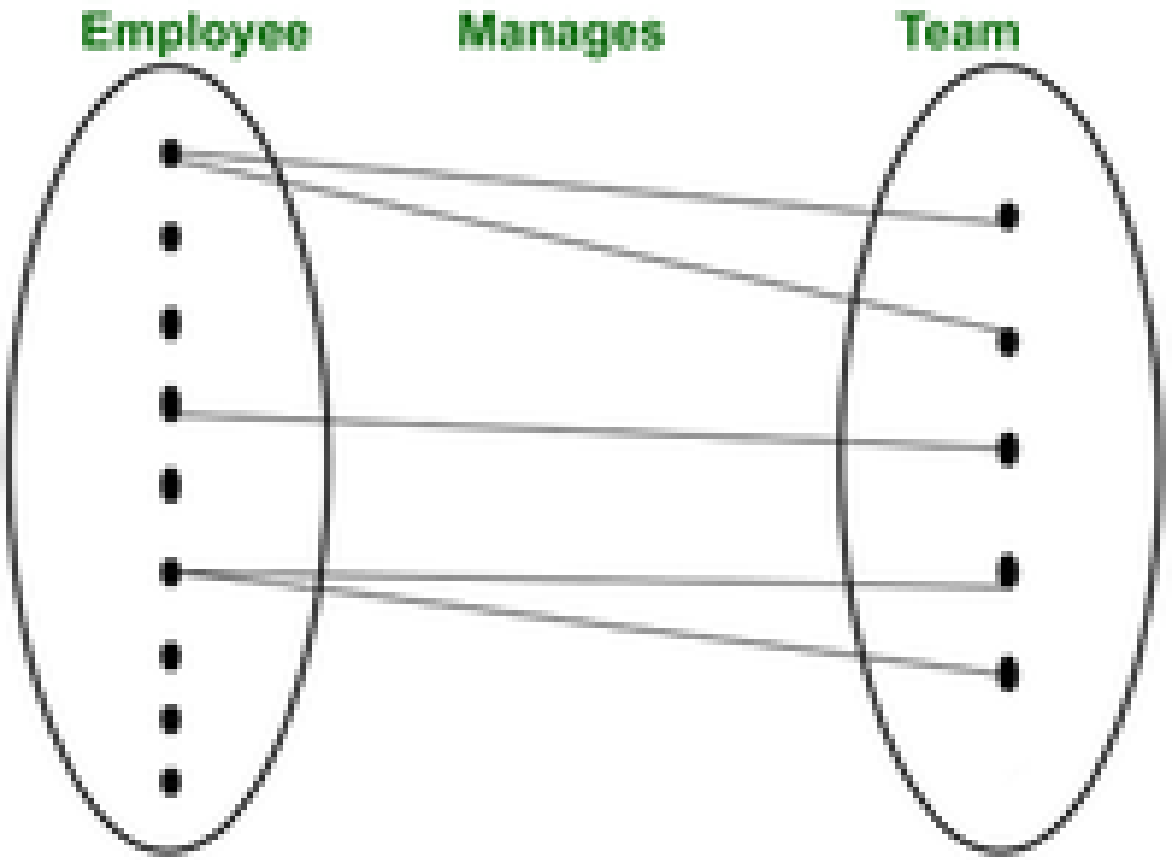
2. One to Many Relationship (1:M) :

Example :



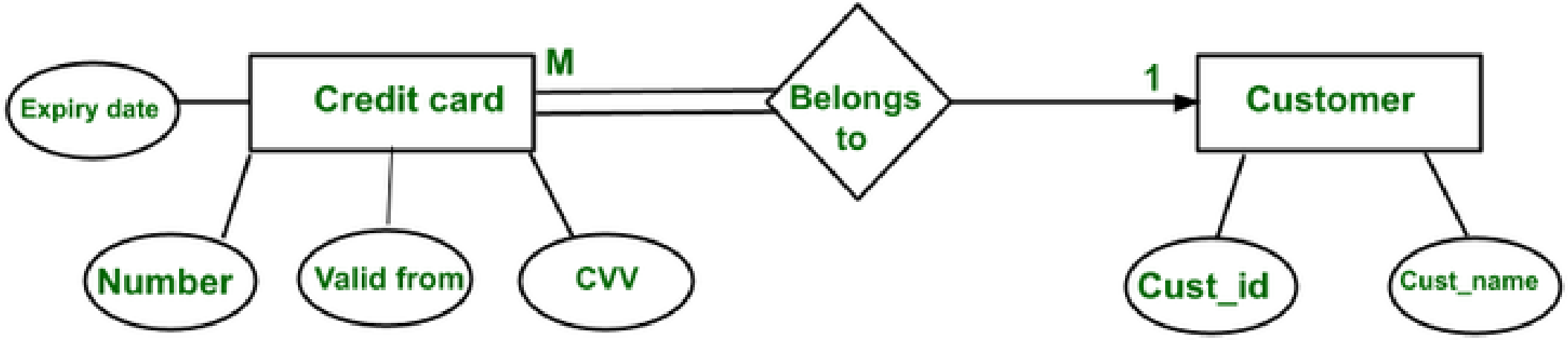
This relationship is one to many because “There are some employees who manage more than one team while there is only one manager to manage a team”.

The set-theoretic perspective of the ER diagram is :



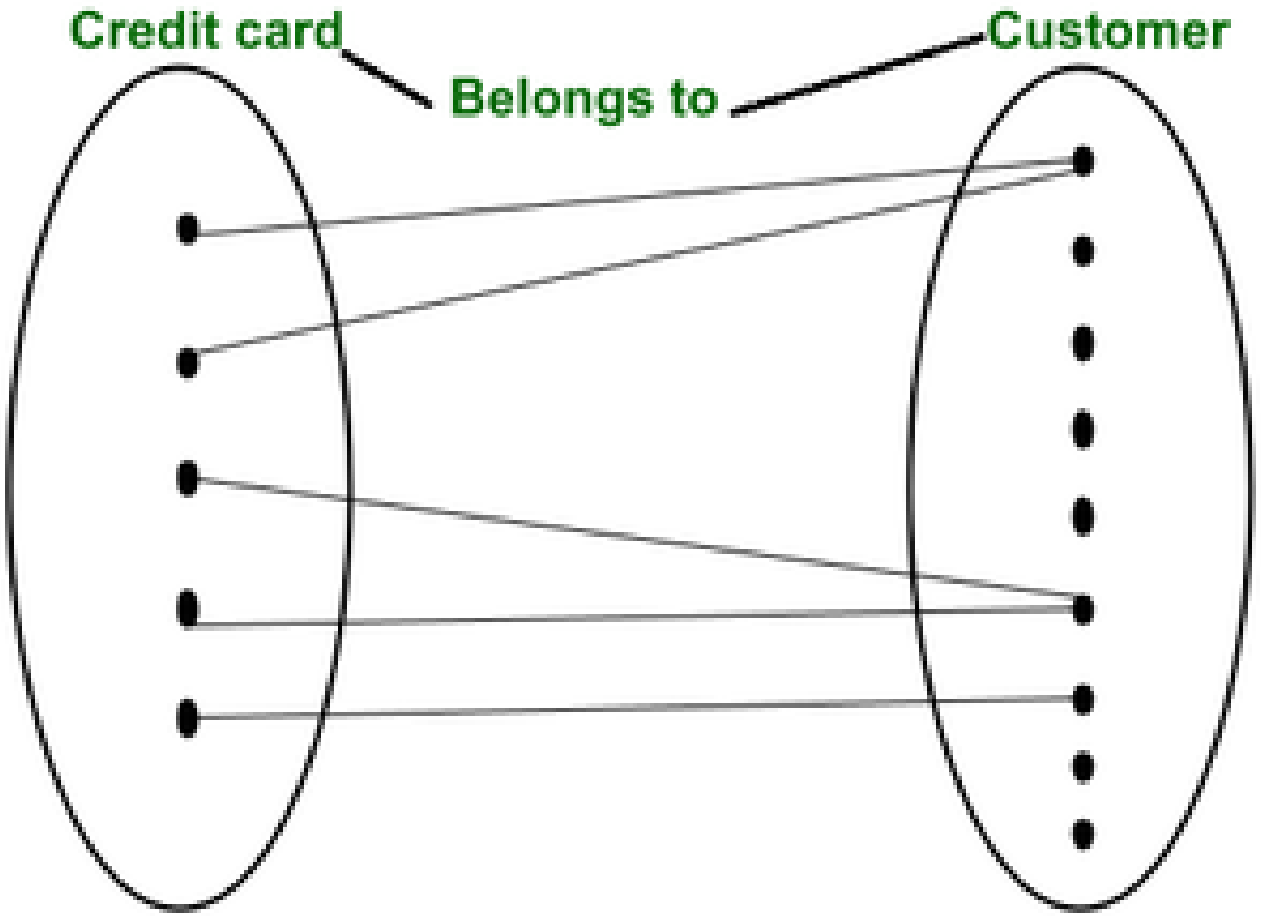
3. Many to One Relationship (M:1) :

Example :



- It is related to a one-to-many relationship but the difference is due to perspective. Any number of credit cards can belong to a customer and there might be some customers who do not have any credit card, but every credit card in a system has to be associated with an employee(i.e. total participation).
- While a single credit card can not belong to multiple customers.

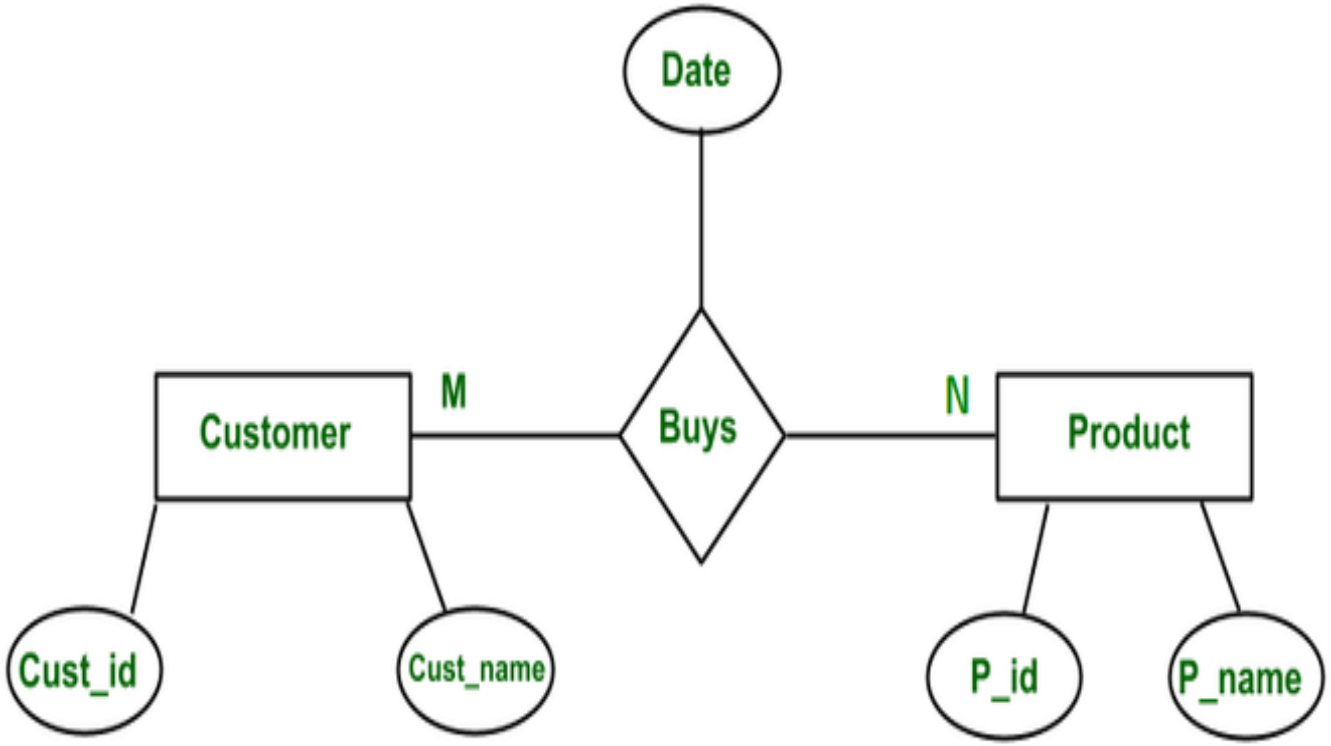
The set-theoretic perspective of the ER diagram is:



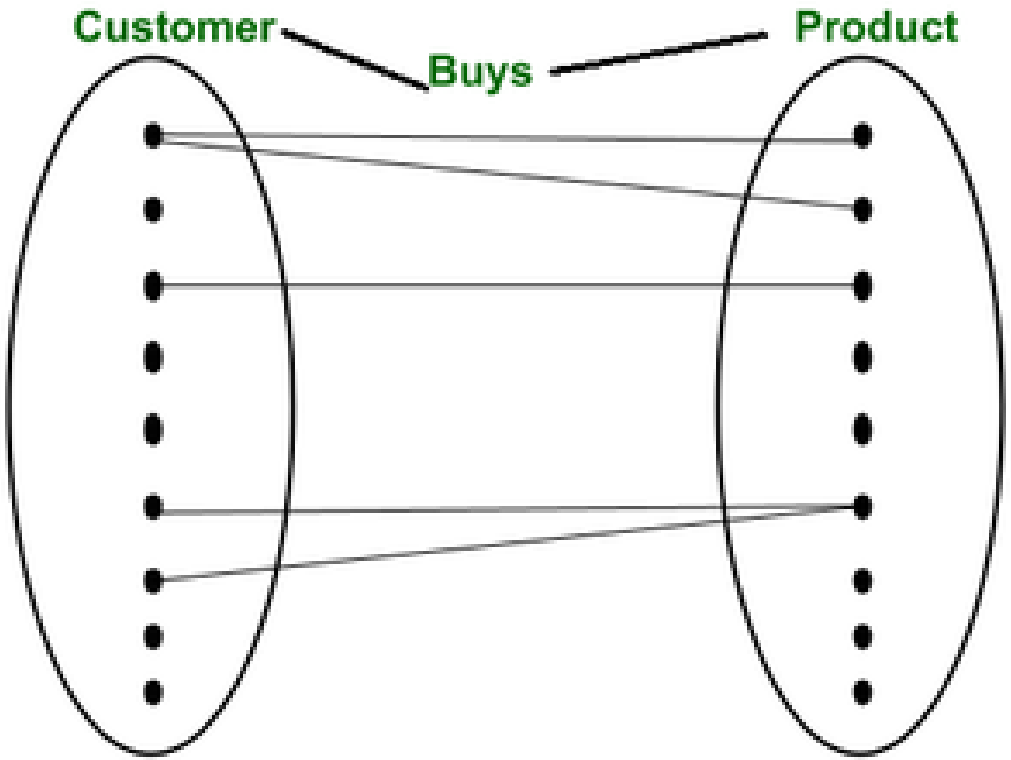
4. Many to Many Relationship (M:N) :

Example :

A customer can buy any number of products and a product can be bought by many customers.

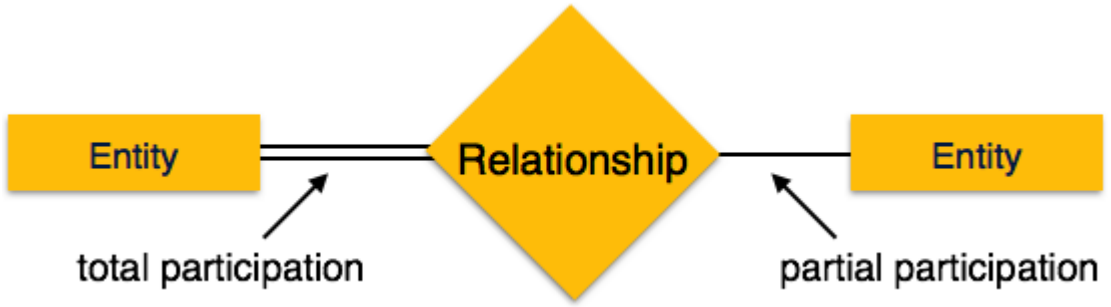


The set-theoretic perspective of the ER diagram is:



Participation or Existence Constraint

- It represents the minimum number of relationship instances that each entity can participate in, and it is also called the minimum cardinality constraint.
- There are two types of participation constraints, which are **total** and **partial**.
- Any of the four cardinalities of a binary relationship can have both sides partial, both total, and one partial, and one total participation, depending on the constraints specified by user requirements.

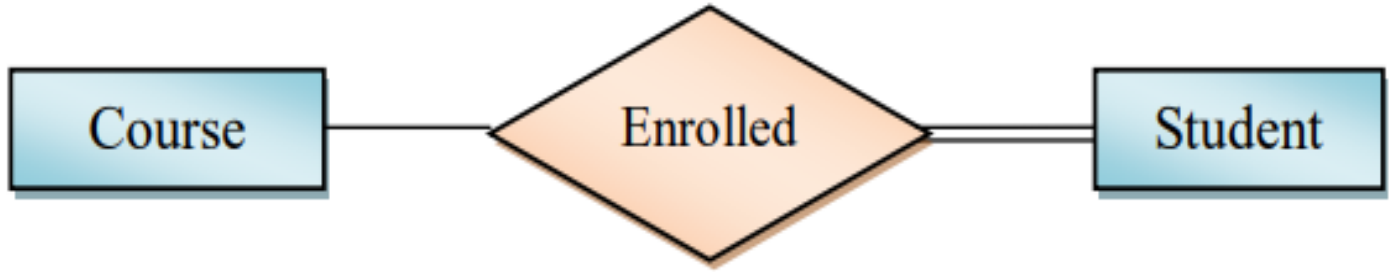


Total Participation (Mandatory Participation)

- In a total participation constraint, every entity instance in one entity set must participate in the relationship with another entity set.
- It is represented using a double line between the entity set and relationship set.

Partial Participation (Optional Participation)

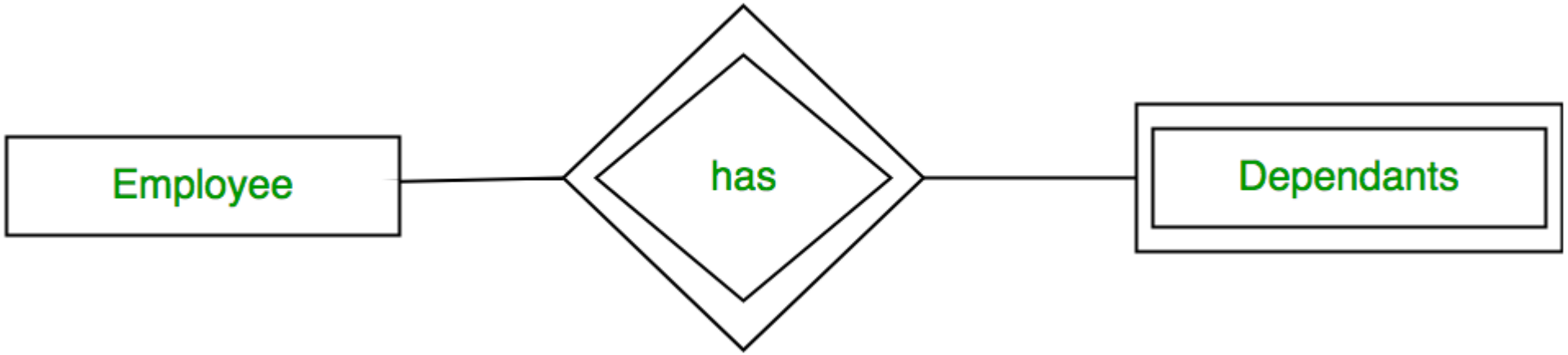
- In a partial participation constraint, some entity instances in one entity set may choose not to participate in the relationship with another entity set.
- It is represented using a single line between the entity set and relationship set.



Participation in Enrolled relationship set: **Partial** Course
Total Student

Representation of Weak Entity Set:

- The entity sets which do not have sufficient attributes to form a primary key are known as weak entity sets and the entity sets which have a primary key are known as strong entity sets.
- The weak entity are dependent on the parent or owner entity.
- Weak entity is represented by double rectangle.
- The relation between one strong and one weak entity is represented by double diamond.



Structural Constraint

Cardinality Ratios and Participation Constraints taken together are called Structural Constraints.

Structural Constraints

- **Multiplicity is made up of two types of restrictions on relationships: cardinality and participation.**
- **Cardinality**
  - **Describes maximum number of possible relationship occurrences for an entity participating in a given relationship type.**
- **Participation**
  - **Determines whether all or only some entity occurrences participate in a relationship.**

Naming Conventions

- Naming conventions in ER diagrams are guidelines or standards used to name entities, attributes, relationships, and other components within the diagram.
- These conventions help ensure consistency, clarity, and understandability in database schema design.
- By following proper naming conventions, database designers can create ER diagrams that are easier to read, understand by the user , developers, and database administrators.
- Some most common naming conventions we must follow are as follows:

❑ **Entity Names:**

- ✓ We must use meaningful names for entities that reflect the real-world objects they represent like "Employee," "Customer," "Product" etc.
- ✓ Avoid using abbreviations or acronyms which may lead to misunderstanding.

❑ **Cardinality and Participation Notations:**

- ✓ Use appropriate symbols to represent cardinality and participation constraints in relationships.

Naming Conventions

- **Attribute Names:**
  - ✓ Use descriptive names for attributes that convey the information they store like "EmployeeID," "FirstName," "LastName".
  - ✓ Follow a consistent naming convention such as camelCase or underscore_case for multi-word attribute names.

- **Relationship Names:**
  - ✓ Choose meaningful names for relationships that indicate the nature of the association between entities like, “Studies," "Purchases," “WorksIn".

- **Primary and Foreign Keys:**
  - ✓ Clearly identify primary key attributes with prefixes such as "PK_" or suffixes like "_ID“. Examples : "PK_EmployeeID," "Customer_ID"
  - ✓ Use prefixes like "FK_" to denote foreign key attributes referencing primary keys of other entities. Examples : "FK_DepartmentID," "ProductID"

Design Issues

- **Choosing or Identifying Entities:**
  - Sometime it may be a challenging to determine the main entities in ER diagram which represent real world object with unique identities and attributes.
- **Defining Attributes:**
  - Identifying the exact attributes is also another important issues because we may assume unnecessary attributes sometimes which may be not needed.
  - Considering whether the attributes shoud be mandatory or optional, single-valued or multi-valued.
- **Establishing Relationships:**
  - Identifying the relationships between entities that shows their association can be quite challegning sometimes.
  - Determining the cardinality and participation constraints for each relationship can be hard.
- **Maintaining Data Integrity:**
  - Establishing foreign key constraints to maintain referential integrity between related entities is hard.

Extended ER Model:

- EER is a high-level data model that incorporates the extensions to the original ER model.
- EER creates a design more accurate to database schemas.
- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model.
- It includes the concept of specialization and generalization.

Super Class and Sub Class

- A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class.
- A subclass is a class derived from the superclass. It inherits the properties of the superclass and contains attributes of its own.

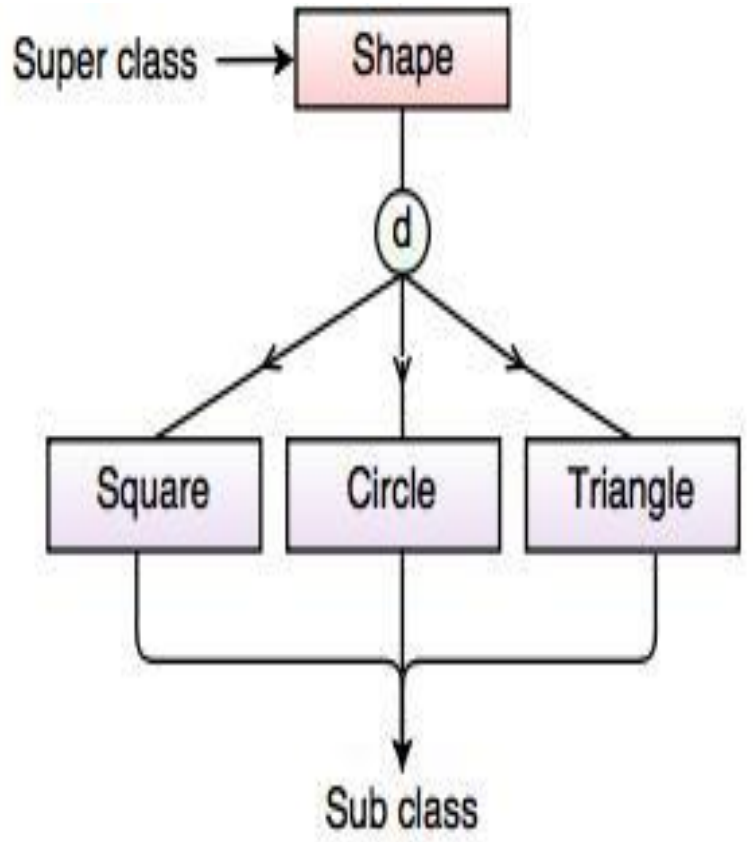
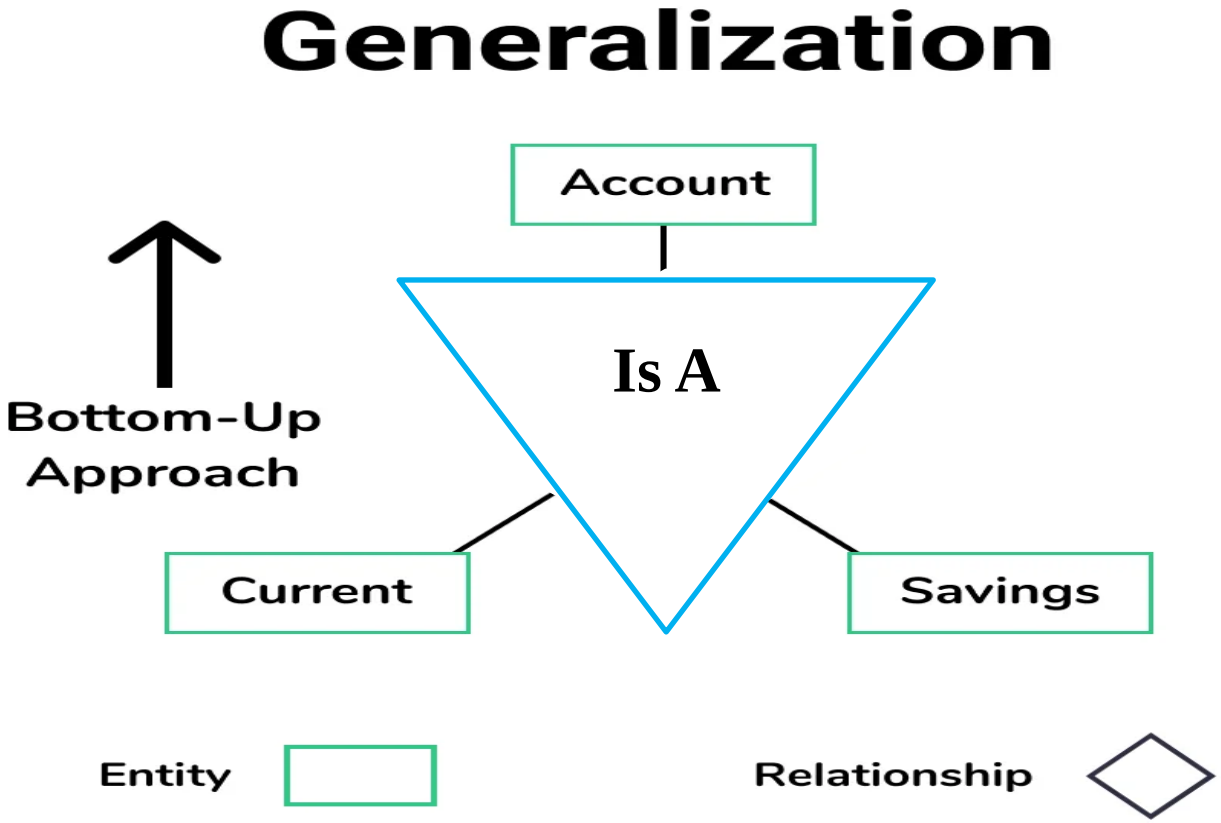
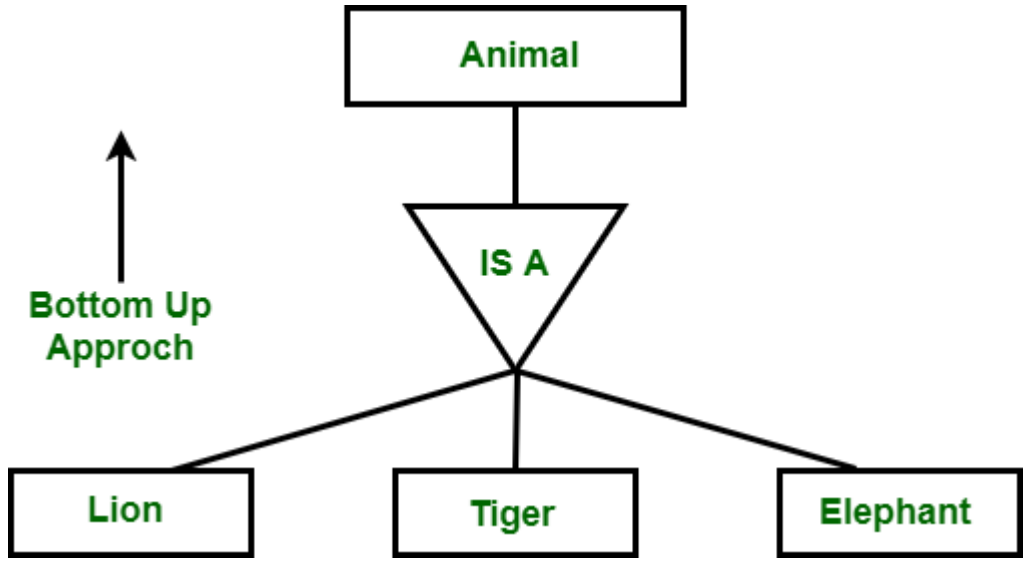


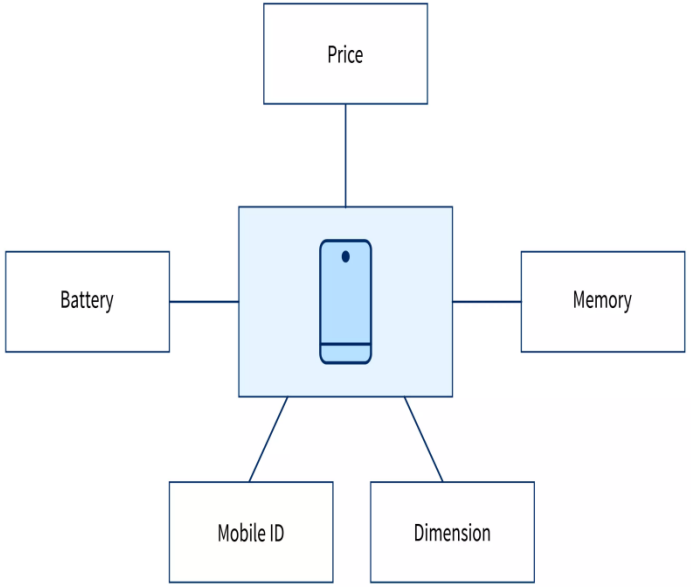
Fig. Super class/Sub class Relationship

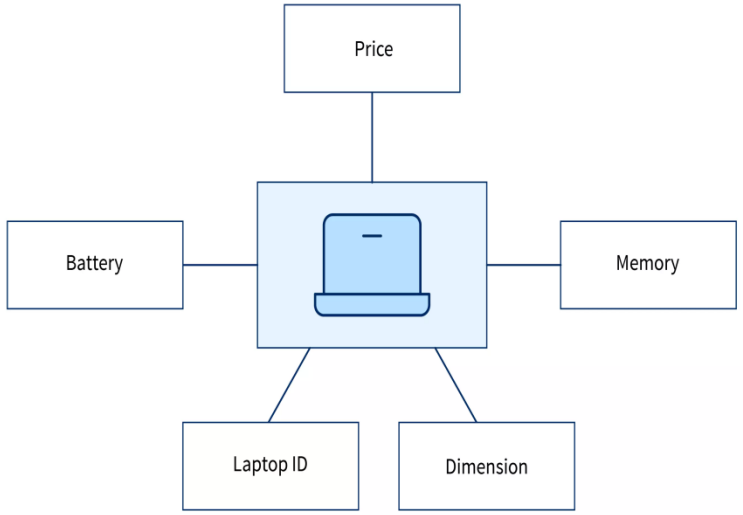
Generalization

- In Database Management System, **Generalization** is a process in which a new entity is formed using the common attributes of two or more entities.
- The newly formed entity is called the generalized entity.

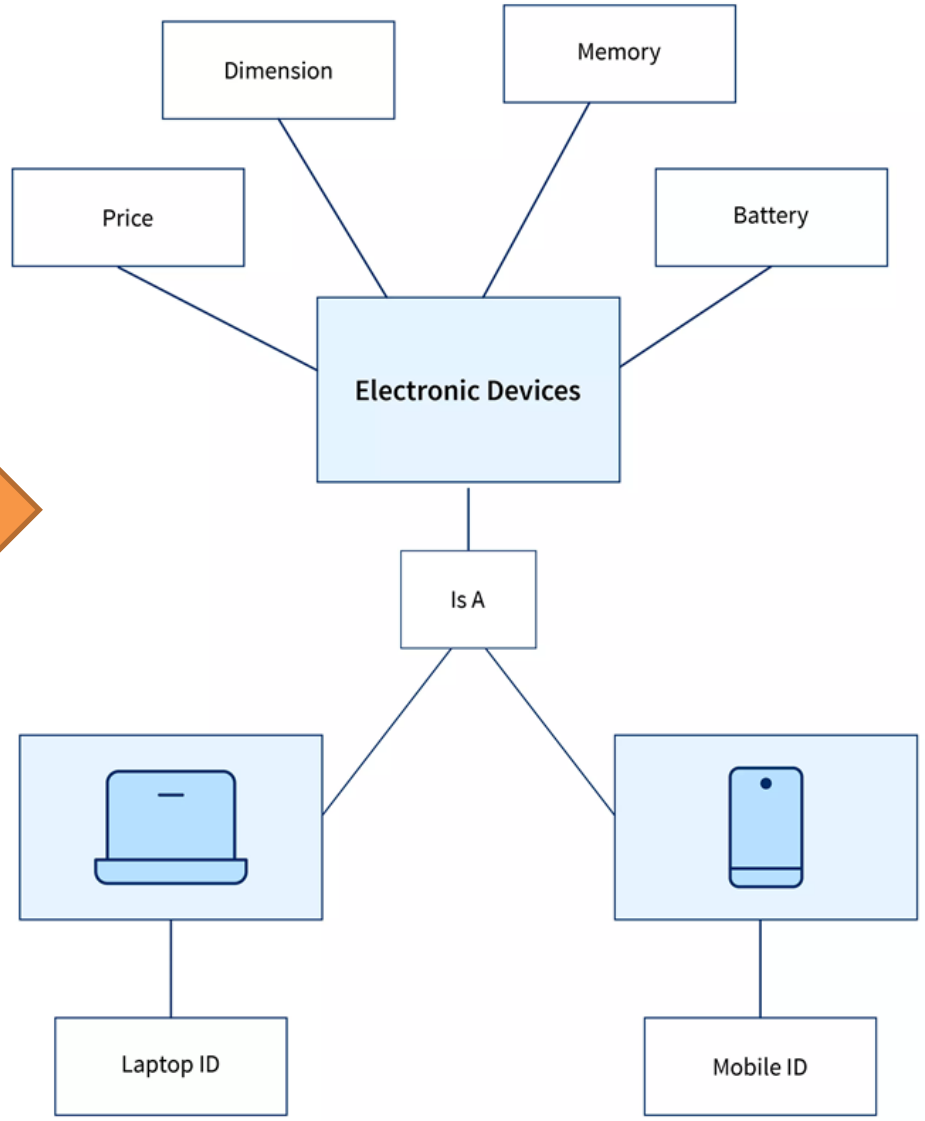






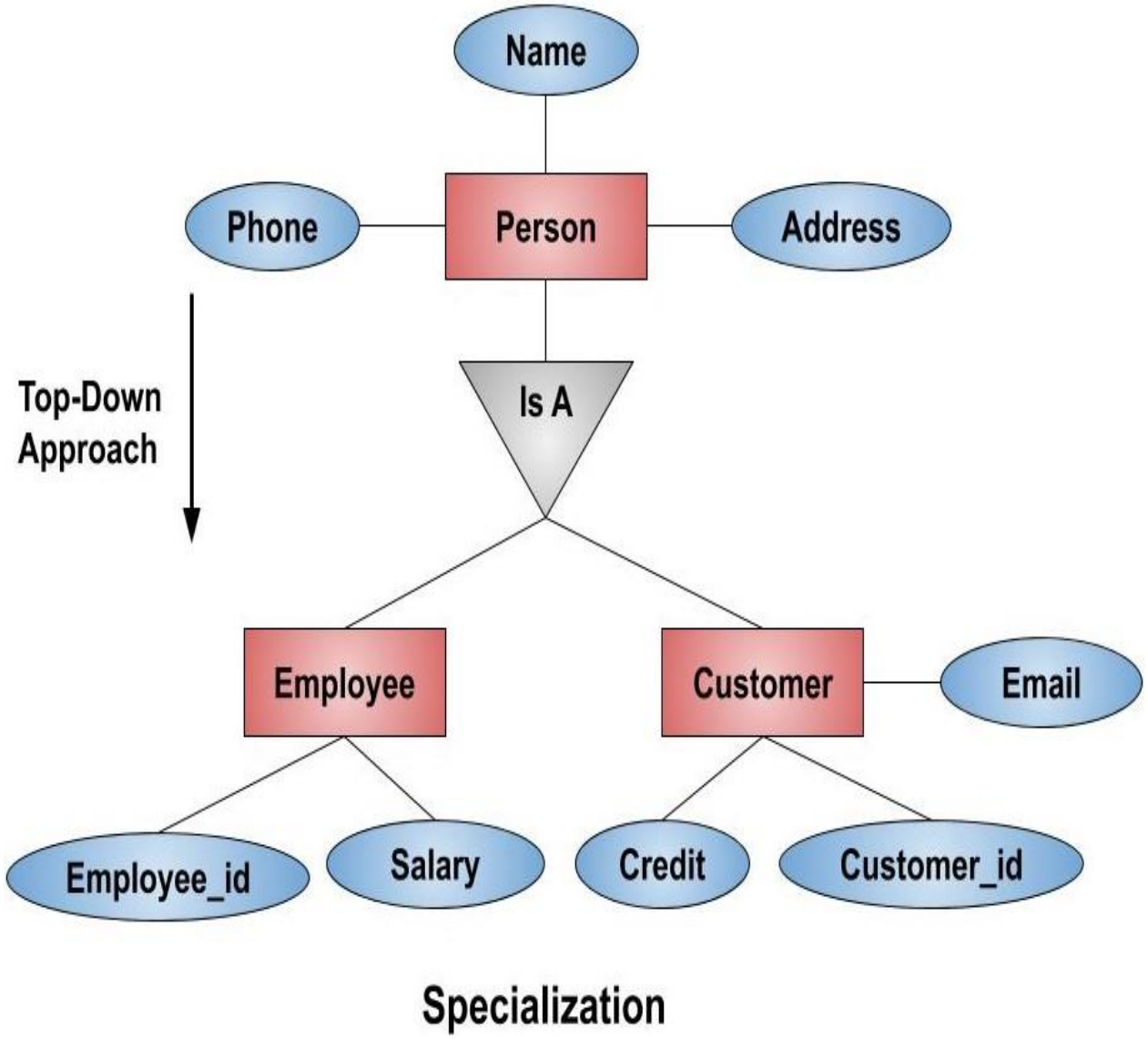


Generalization



Specialization

- Specialization is a top-down approach in which a higher-level entity is divided into multiple specialized lower-level entities.
- In addition to sharing the attributes of the higher-level entity, these lower-level entities have specific attributes of their own.
- For Example, Person entity can be specialized into Employee, Customer etc.



Specialization

Characteristics of Generalization

- The bottom-up technique is used for generalization.
- It simplifies or generalizes the entities.
- Higher-level entities and lower-level entities can be linked.

Characteristics of Specialization

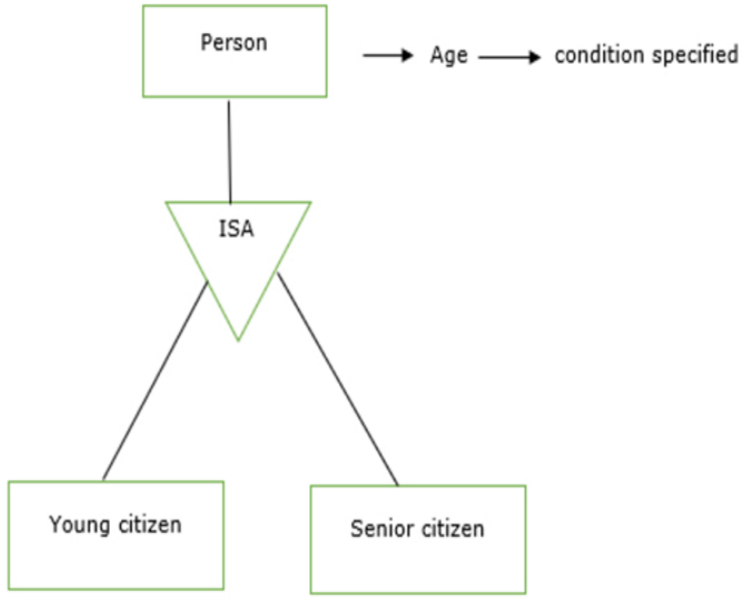
- The top-down technique is used for specialization.
- It divides or specializes in the entity.
- Higher-level entities and lower-level entities can be linked.

Constraints of Specialization and Generalization

There are mainly three types of constraints in specilization and generalization which are given below:

1. Membership Constraint : It determines which entity can be a member of the low-level entity set. It has following two types:

- **Attribute/Condition Defined :** All the lower level attributes are evaluated on the basis of the some attribute.
- **User Defined :** The database user assign entities to a given entity set.



Constraints of Specialization and Generalization

2. Disjointness Constraint : It determines whether or not entities belong to more than one lower-level entity set. It can be further classified as:

- **Disjoint :** An entity bleong to no more than one lowe level entity set.
- **Overlapping :** The same entity may belong to more than one lower level entity set within a single generalization/specialization.

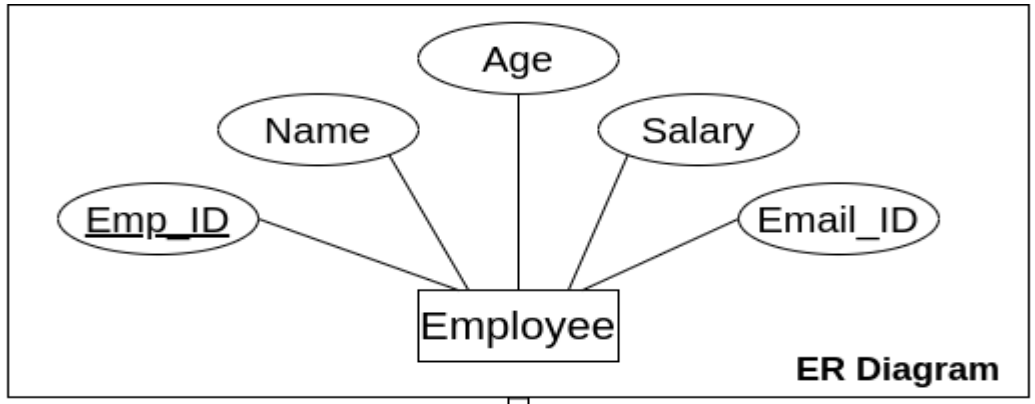
3. Completeness Constraint : It specifies whether or not an entity in the higher level entity set must belong to at least one of the lower level entity set. It can be further divided into two types:

- **Total :** It determines that each higher-level entity must belong to a lower-level entity set.
- **Partial:** It determines that some higher-level entities may not belong to any lower-level entity set.

Converting ER Diagram to Tables

Rule 1: Strong Entity set with only simple attributes

- A strong set with only simple attributes will require only one table in relational model.
  - Attributes of the table will be the attributes of the entity set.
  - The primary key of the table will be the key attribute of the entity set.



Primary Key Emp_ID	Name	Age	Salary	Email_ID

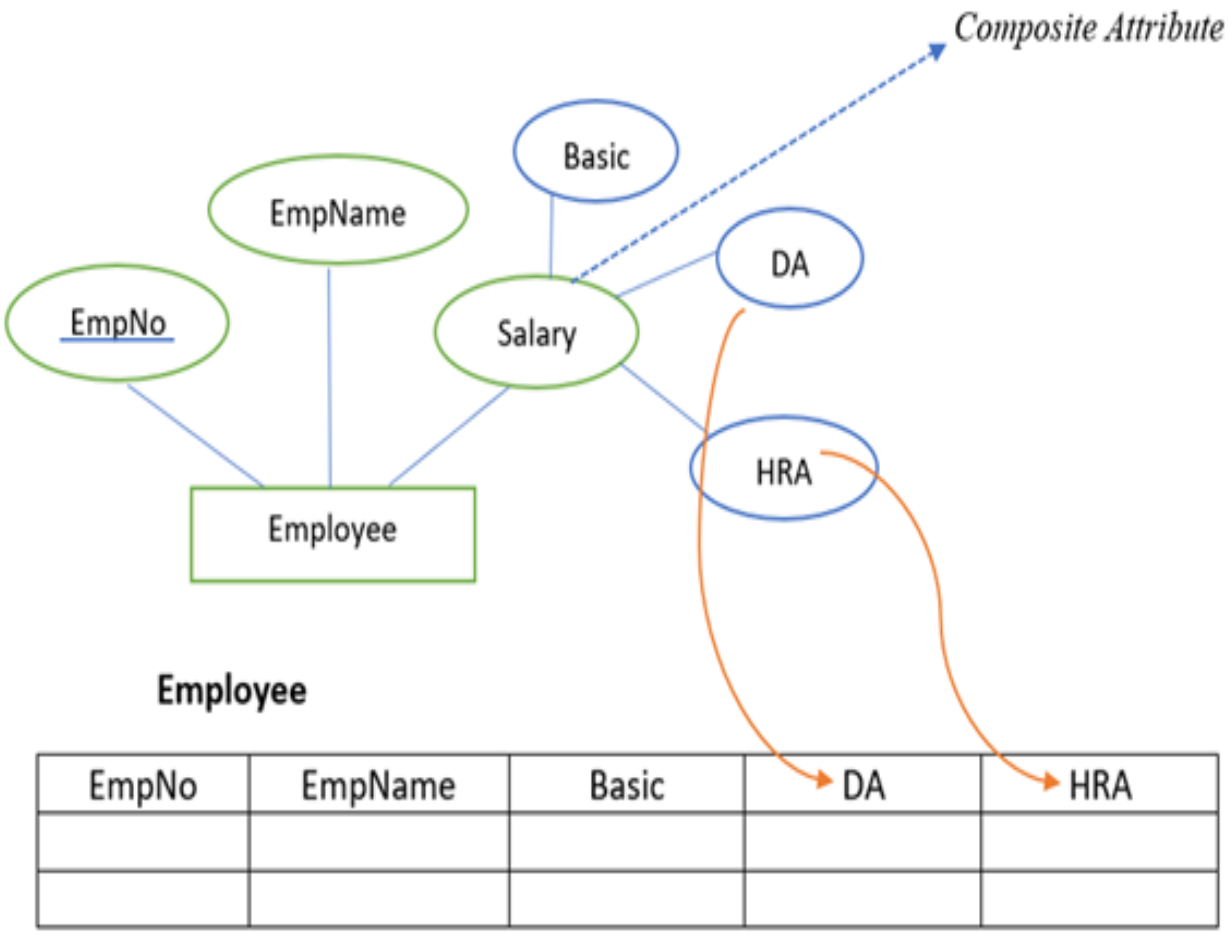
Relation: Employee

Strong Entity With Simple Attributes ⇨ Relational Table

Schema: Employee (Emp_ID, Name, Age, Salary, Email_ID)

Rule 2: Strong Entity set with composite attribute

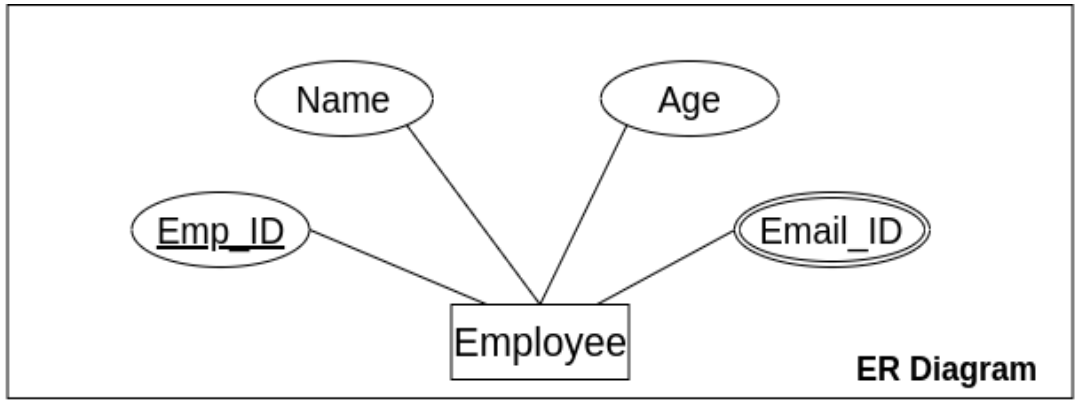
- A strong entity set with any number of composite attributes will require only one table in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.



Schema: Employee (EmpNo, EmpName, Basic, DA, HRA)

Rule 3: Strong Entity set with multi valued attributes

- A strong entity set with any number of multi-valued attributes will require two tables in relational model.
  - One table will contain all the simple attributes with the primary keys.
  - Other table will contain the primary key and all the multi valued attributes.



Primary Key Emp_ID	Name	Age

Relation: Employee1

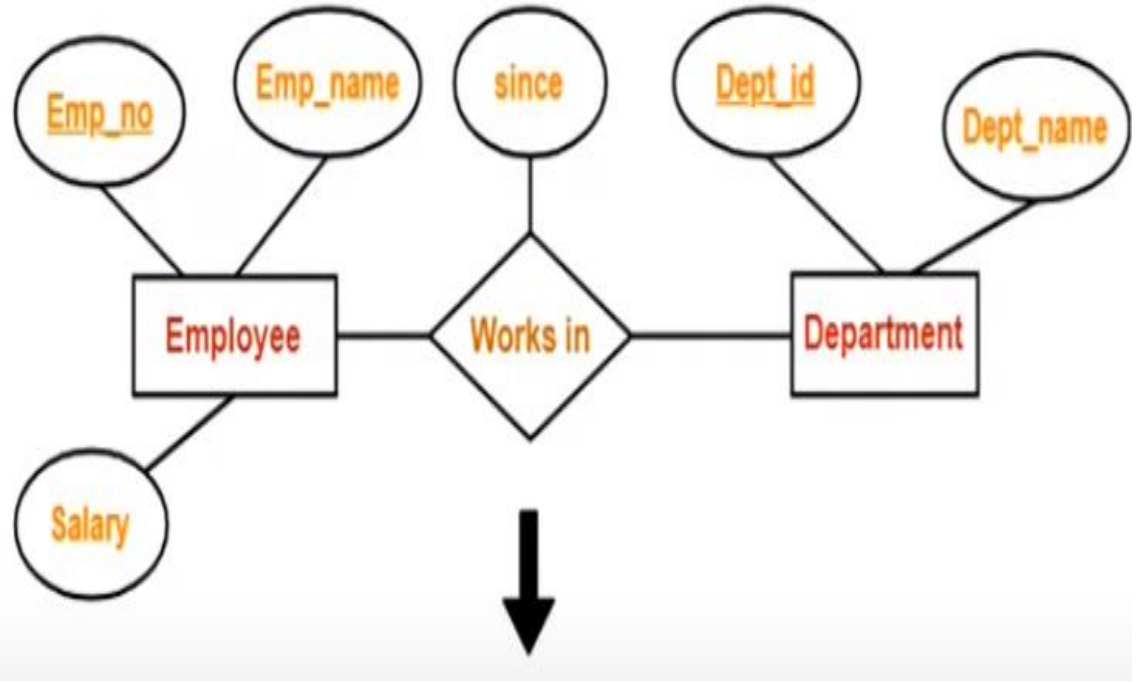
Primary Key Emp_ID	Email_ID

Relation: Employee2

Strong Entity With Multi-valued Attributes → Relational Table

Rule 4: Translating relationship set into a table

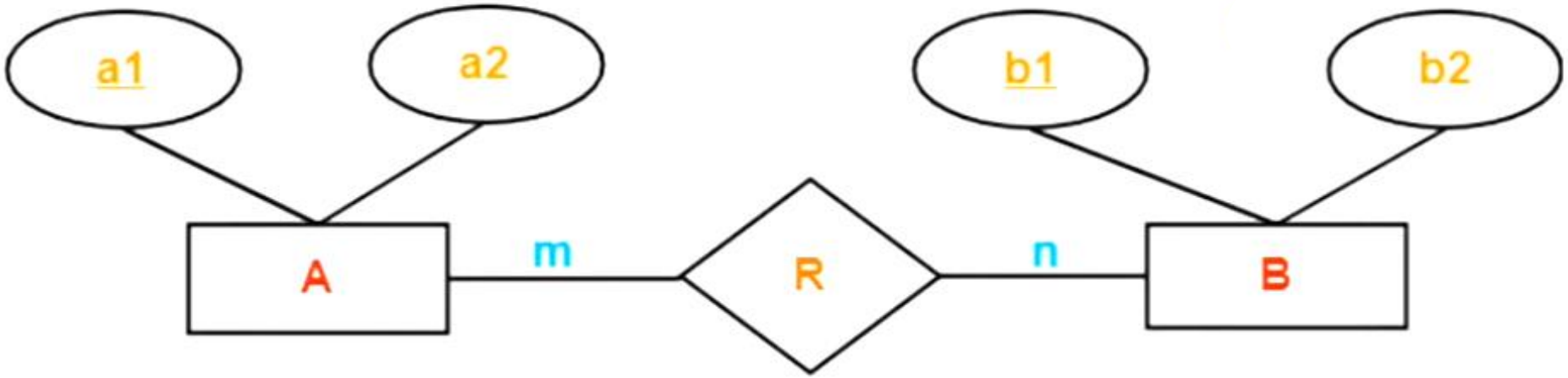
- A relationship set will require one table in the relational model.
- Attributes of the table are:
  - Primary key attributes of the participating entity sets
  - Its own descriptive attributes if any
- Set of non-descriptive attributes will be the primary key.
- For a given ER diagram attributes will be required in relational model:
  - One table for the entity set “Employee”
  - One table for the entity set “Department”
  - One table for the relationship set “Works in”



Rule 5: For Binary relationships with cardinality ratios

- The following four cases are possible:
  - ✓ Case – 1 : Binary relationship with cardinality ration M: N
  - ✓ Case – 2 : Binary relationship with cardinality ration 1: N
  - ✓ Case – 3 : Binary relationship with cardinality ration M : 1
  - ✓ Case – 4 : Binary relationship with cardinality ration 1: 1

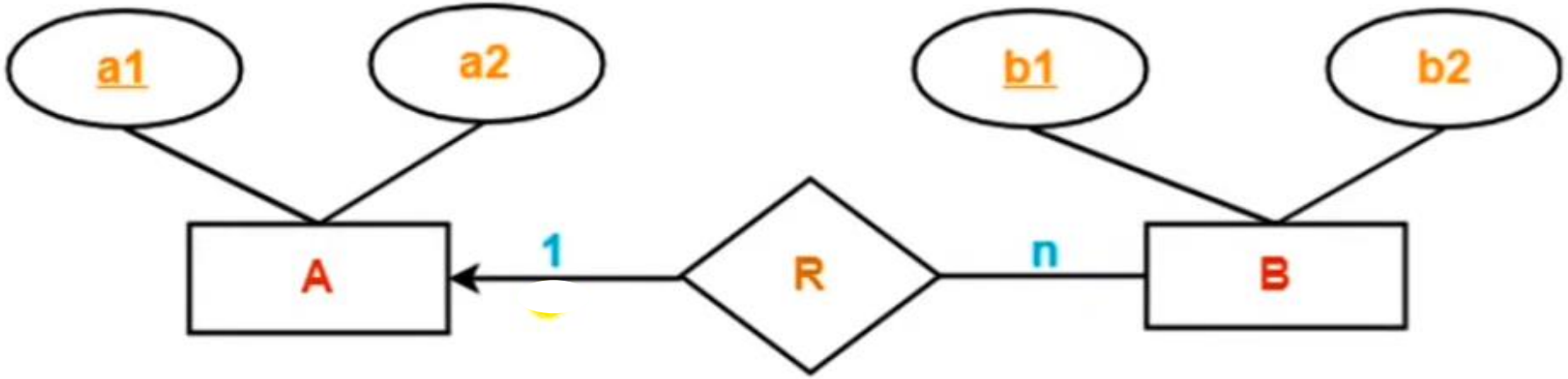
Case – 1: For Binary relationships with cardinality ratio M:N



In ***Many-to-Many relationship***, **three tables** will be required-

1. **A (a1 , a2)**
2. **R (a1 , b1)**
3. **B (b1 , b2)**

Case – 2: For Binary relationships with cardinality ratio 1:N

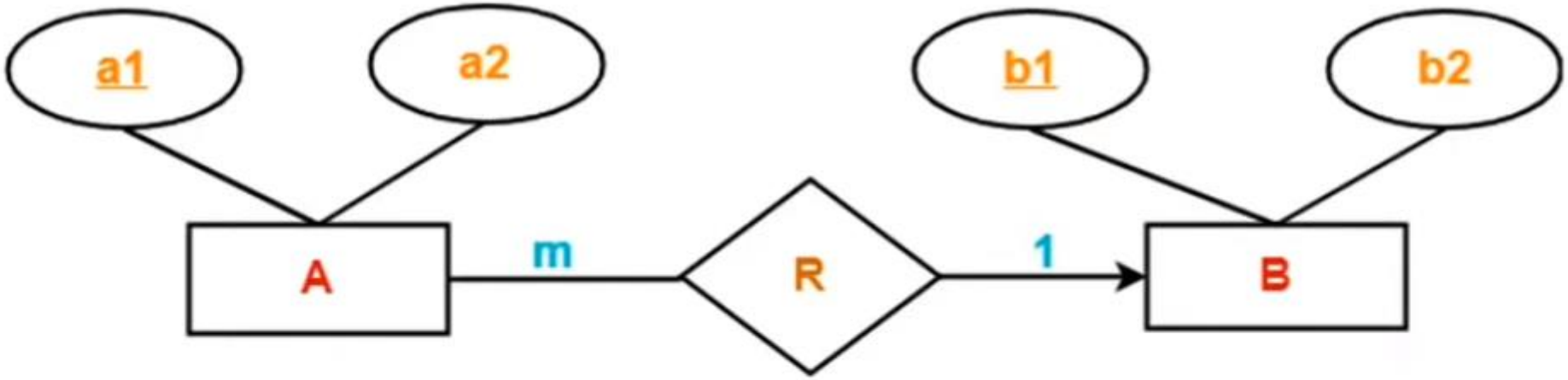


In ***One-to-Many relationship***, **two tables** will be required-

1. **A (a1 , a2)**
2. **BR (b1 , b2, a1)**

NOTE - Here, combined table will be drawn for the entity set B and relationship set R.

Case – 3: For Binary relationships with cardinality ratio M:1

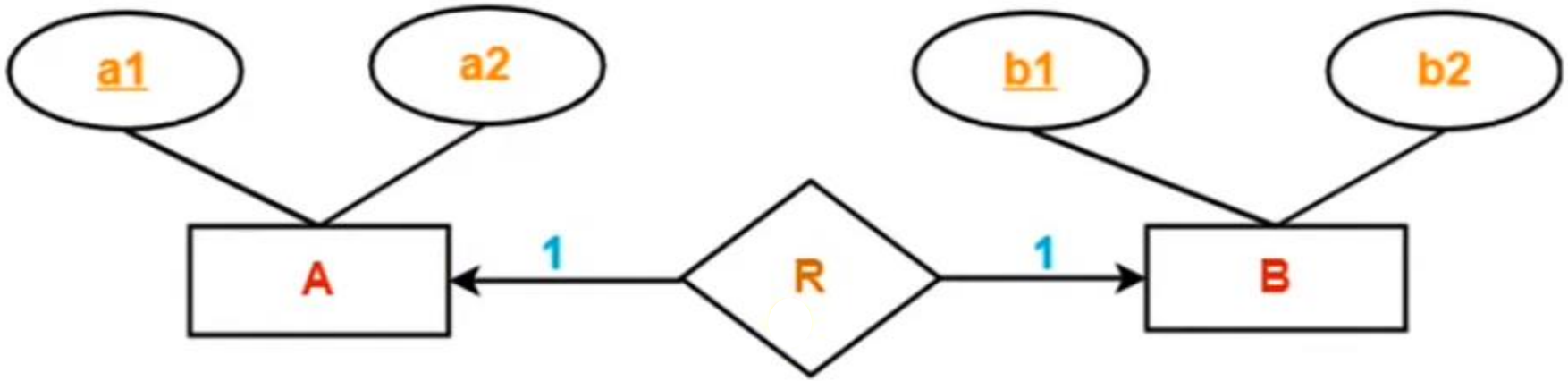


In ***Many-to-One relationship***, **two tables** will be required-

1. **AR (a1 , a2 , b1)**
2. **B (b1 , b2)**

NOTE - Here, combined table will be drawn for the entity set A and relationship set R.

Case – 4: For Binary relationships with cardinality ratio 1:1



In ***One-to-One relationship***, **two tables** will be required. Either combine 'R' with 'A' or 'B'

Way-01:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

1. A (a1 , a2)
2. BR (b1 , b2, a1)

Thumb rules to remember:

(for determining minimum number of tables)

While determining the minimum number of tables required for binary relationships with given cardinality ratios, following thumb rules must be kept in mind:

- **For binary relationship with cardinality ratio m:n**
  - Separate and individual tables will be drawn for each entity set and relationship (i.e. three tables will be required)
- **For binary relationship with cardinality ratio either m:1 or 1:n,**
  - Always remember “many side will consume the relationship” i.e. a combined table will be drawn for many side entity set and relationship set (i.e. Two tables will be required)
- **For binary relationship with cardinality ratio 1:1**
  - Two tables will be required. You can combine the relationship set with any of the entity sets.

Rule 6: For Binary relationships with both cardinality constraints and Participation constraints

- Cardinality constraints will be implemented as discussed in Rule-5
- Because of the total participation constraint, foreign key acquires NOT NULL constraint i.e. now foreign key cannot be null.
- Two cases:
  - Case – 1 : For binary relationship with cardinality constraint and total participation constraint from one side
  - Case – 2 : For binary relationship with cardinality constraint and total participation constraint from both sides.

Case - 1: For Binary relationships with both cardinality constraints and Participation constraint from one side

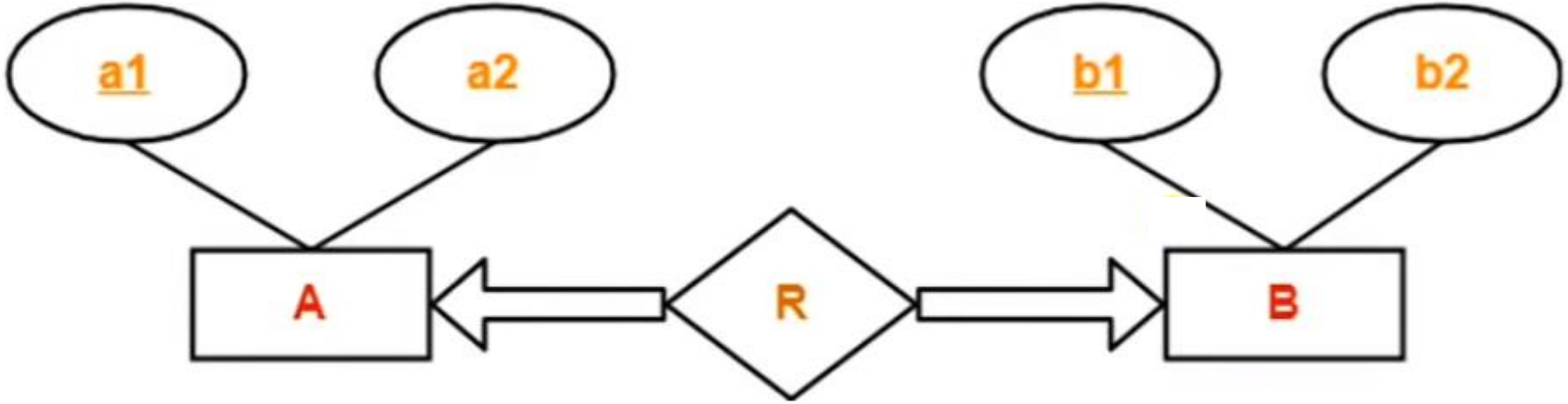
- Because cardinality ratio = 1 : n , we will combine the entity set B and relationship set R.
- Then, **two tables** will be required-

1. A (a1 , a2)
2. BR (b1 , b2 , a1)

Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can't be null now.

Case - 2: For Binary relationships with both cardinality constraints and Participation constraint from both side

- If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using **only single table.**



- Here, Only **one table** is required.
  1. **ARB (a1 , a2 , b1 , b2)**

Rule 7: For Binary relationships with Weak Entity Set

- Weak entity set always appears in association with identifying relationship with **total** participation constraint and there is always **1: n** relationship from identifying entity set to weak entity set.

- Here, **two tables** will be required-

1. **A (a1 , a2)**
2. **BR (a1 , b1 , b2)**

Finished!