# Database Fundamentals

*Cybersecurity Department*

*Course Code:* ***CBS213***

***Lecture 4: Entity-Relationship Diagram (ERD)***

Halal Abdulrahman Ahmed

## Outline

- SDLC recap & link to database design
- Requirements → SRS → Design
- ERD purpose & components
- Entities & attributes
- Relationships & cardinality

## Learning Outcomes

By the end of this lecture, students will be able to:

- Explain where ERD fits in the SDLC and SRS process
- Understand how requirements transform into ERD models
- Identify entities, attributes, and relationships
- Differentiate strong vs. weak entities and key attributes
- Classify attribute types (simple, composite, multi-valued, derived)
- Recognize unary, binary, and ternary relationships
- Apply cardinality and participation rules (1:1, 1:N, M:N)

## Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a structured process used to transform an idea into a complete working software system. It begins by gathering requirements from the client and users to clearly understand what the system must achieve.

Once the requirements are collected, the team studies and analyzes them in detail to ensure there is no confusion and to plan how the system will operate. This is followed by the design stage, where the structure of the system is planned, including its database, user interface, and overall architecture.

- Before writing any code, it is essential that the development team has a clear and complete understanding of the system requirements. These requirements are obtained from stakeholders such as the client or business owner, system users, business analysts, developers, testers, and project managers.
- Good requirement gathering avoids misunderstandings and ensures everyone shares the same vision for the system, which saves time, cost, and effort during development.

HEY RICHARD, REMEMBER WHEN YOU ASKED US TO MAP OUT OUR SYSTEM? WELL, WE'RE GOING TO NEED A BIGGER OFFICE...

Dataedo /cartoon Piotr@Dataedo

- After gathering and analyzing requirements, an official document is created called the **Software Requirements Specification (SRS)**. The SRS describes in detail what the system must do and works as a contract between the client and the development team. It is usually prepared by the Business Analyst after meeting stakeholders and understanding their needs.

- The SRS includes system objectives, project scope, functional requirements (what the system should do), non-functional requirements (performance, security, usability), user roles, and system constraints. This document guides designers, developers, and testers to ensure the final system meets the client's expectations.

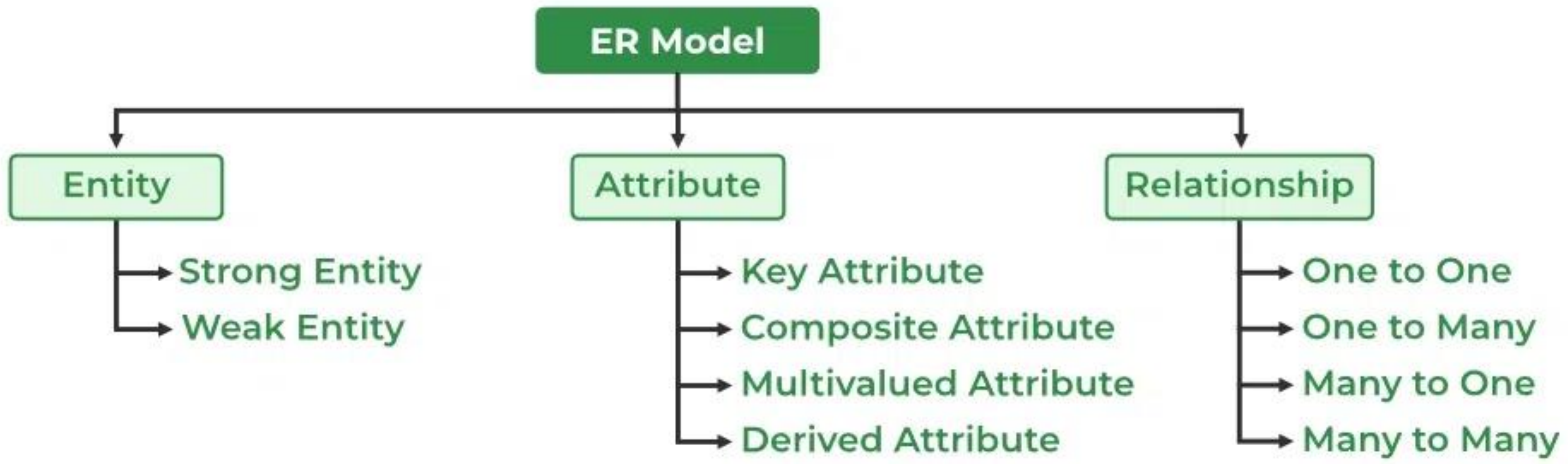
- Once the SRS is complete, the next step is the **design phase**. In this phase, we convert the written requirements into a technical plan. This includes designing the system architecture, planning screens and user interfaces, and most importantly, designing the **database structure**.
- A key part of database design is creating an **Entity-Relationship Diagram (ERD)**. The ERD visually represents the system's data, showing the main entities (tables), their attributes (fields), and the relationships between them. It ensures that the database is correctly organized and supports all requirements defined in the SRS before we start coding.

- A key part of database design is creating an **Entity-Relationship Diagram (ERD)**. The ERD visually represents the system’s data, showing the main entities (tables), their attributes (fields), and the relationships between them. It ensures that the database is correctly organized and supports all requirements defined in the SRS before we start coding.

## Introduction of Entity-Relationship Diagram (ERD)

The Entity-Relationship Model (ER Model) is a conceptual model for designing a databases. This model represents the logical structure of a database, including entities, their attributes and relationships between them.

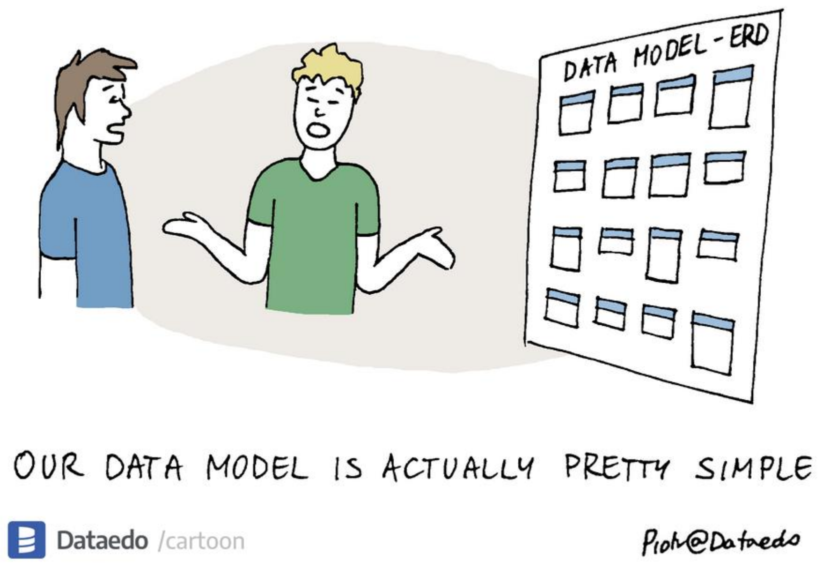
- **Entity:** An objects that is stored as data such as *Student*, *Course* or *Company*.
- **Attribute:** Properties that describes an entity such as *StudentID*, *CourseName*, or *EmployeeEmail*.
- **Relationship:** A connection between entities such as "a *Student* enrolls in a *Course*".


ER Model
Entity
Strong Entity
Weak Entity
Attribute
Key Attribute
Composite Attribute
Multivalued Attribute
Derived Attribute
Relationship
One to One
One to Many
Many to One
Many to Many

## ER Model in Database Design Process

We typically follow the below steps for designing a database for an application.

- Gather the requirements (functional and data) by asking questions to the database users.
- Create a logical or conceptual design of the database. This is where ER model plays a role. It is the most used graphical representation of the conceptual design of a database.
- After this, focus on Physical Database Design (like indexing) and external design (like views)



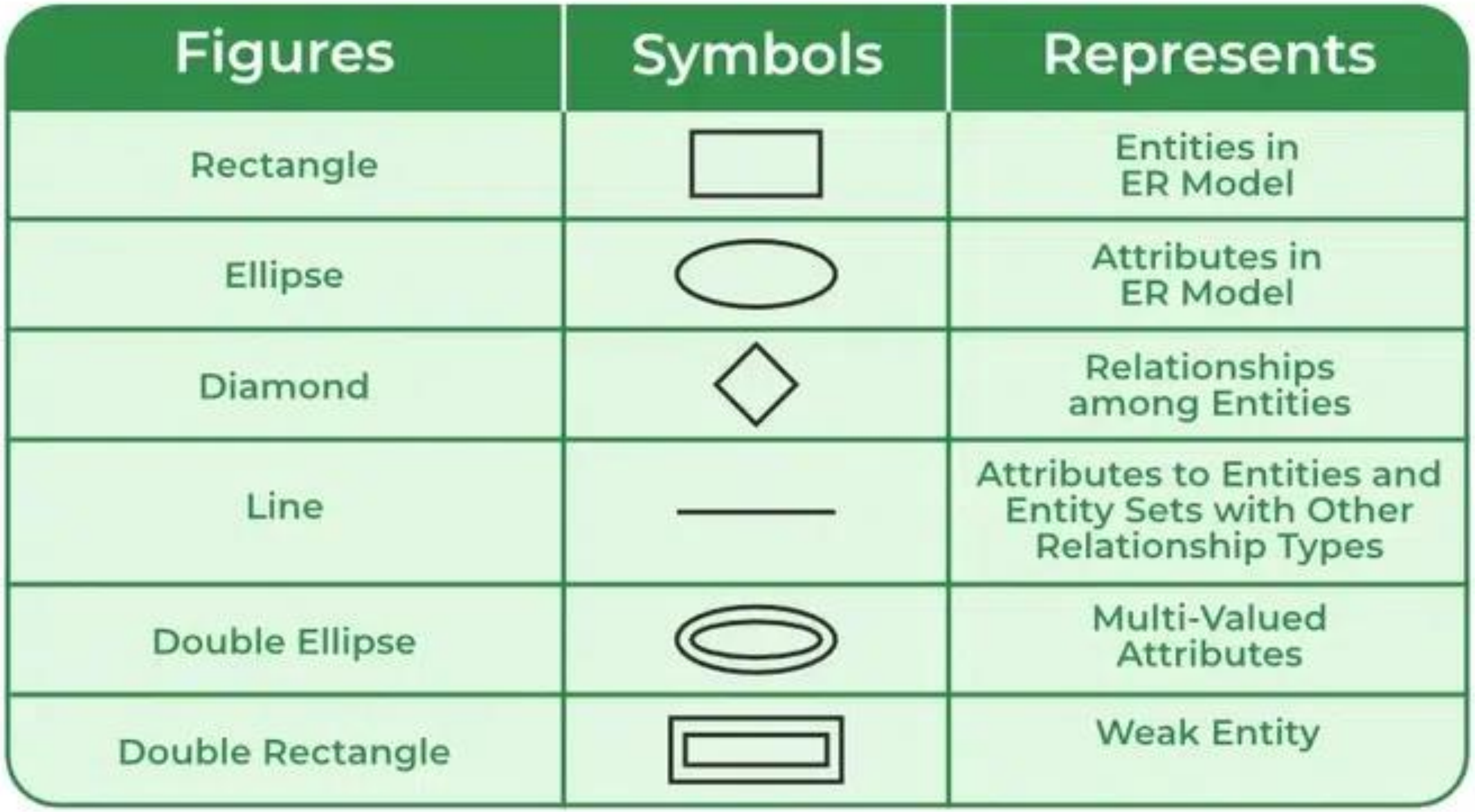
## Why Use ER Diagrams In DBMS?

- ER diagrams represent the E-R model in a database, making them easy to convert into relations (tables).
- These diagrams serve the purpose of real-world modeling of objects which makes them intently useful.
- Unlike technical schemas, ER diagrams require no technical knowledge of the underlying DBMS used.
- They visually model data and its relationships, making complex systems easier to understand.

## Symbols Used in ER Model

ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

- **Rectangles:** Rectangles represent **entities** in the ER Model.
- **Ellipses:** Ellipses represent **attributes** in the ER Model.
- **Diamond:** Diamonds represent **relationships** among Entities.
- **Lines:** Lines represent **attributes to entities** and **entity sets with other relationship types**.
- **Double Ellipse:** Double ellipses represent **multi-valued Attributes**, such as a student's multiple phone numbers
- **Double Rectangle:** Represents **weak entities**, which depend on other entities for identification.



| 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 |

## What is an Entity?

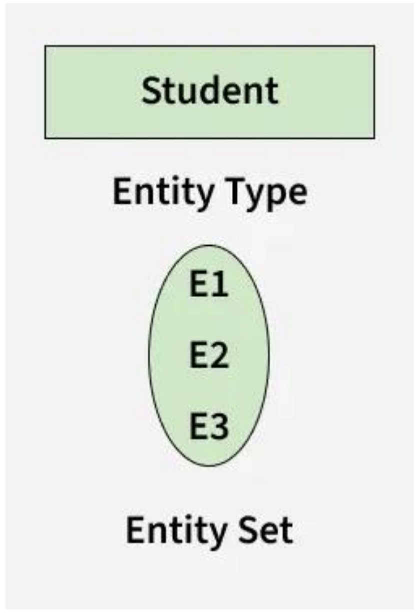
- An Entity represents a real-world object, concept or thing about which data is stored in a database. It act as a building block of a database. Tables in relational database represent these entities.

Example of entities:

- **Real-World Objects:** Person, Car, Employee etc.
- **Concepts:** Course, Event, Reservation etc.
- **Things:** Product, Document, Device etc.
- The entity type defines the structure of an entity, while individual instances of that type represent specific entities.

## What is an Entity Set?

- An entity refers to an individual object of an entity type, and the collection of all entities of a particular type is called an **entity set**. For example, E1 is an entity that belongs to the entity type "Student," and the group of all students forms the entity set.



## Types of Entity

There are two main types of entities:

**1. Strong Entity**

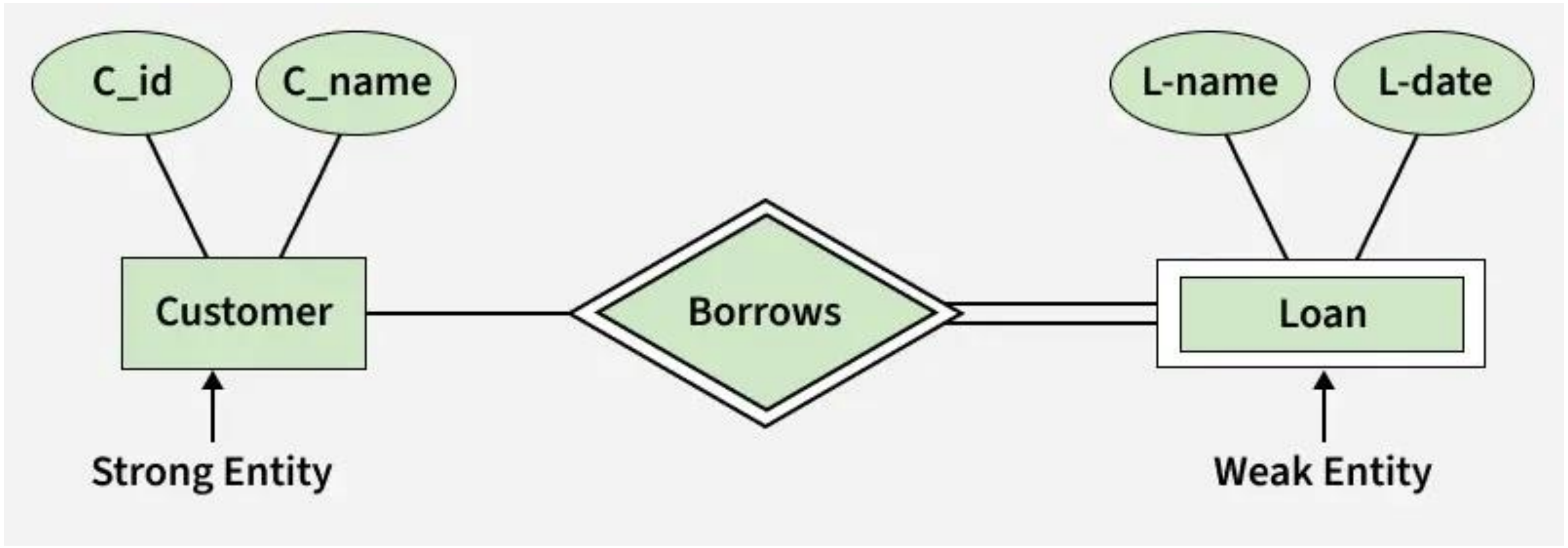
A Strong Entity is a type of entity that has a key Attribute that can uniquely identify each instance of the entity. A Strong Entity does not depend on any other Entity in the Schema for its identification. It has a primary key that ensures its uniqueness and is represented by a rectangle in an ER diagram.

**2. Weak Entity**

A Weak Entity cannot be uniquely identified by its own attributes alone. It depends on a strong entity to be identified. A weak entity is associated with an identifying entity (strong entity), which helps in its identification. A weak entity are represented by a double rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.

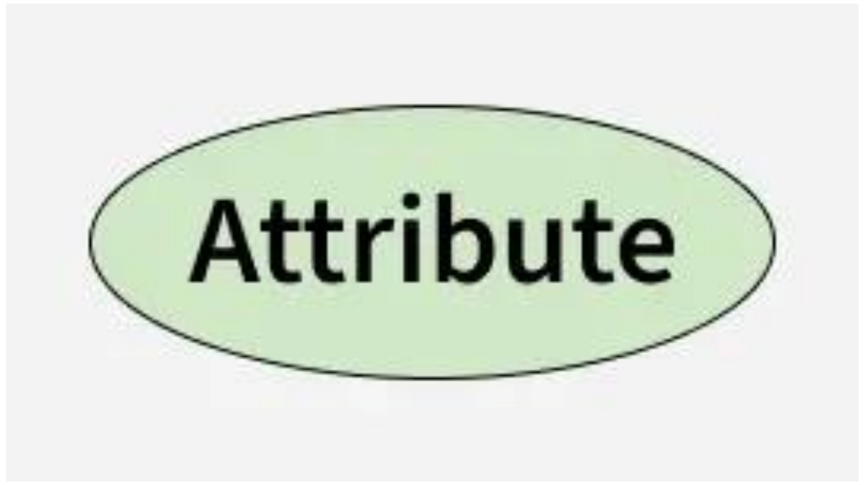
**Example:**

A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents can't exist without the employee. So dependent will be a Weak Entity Type and Employee will be identifying entity type for dependent, which means it is Strong Entity Type.



## Attributes in ER Model

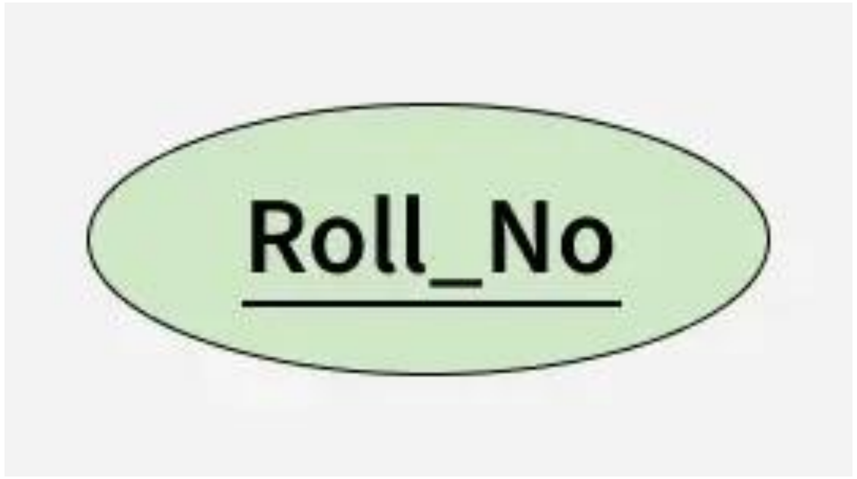
- Attributes are the **properties that define the entity type**. In ER diagram, the attribute is represented by an oval.



## Types of Attributes

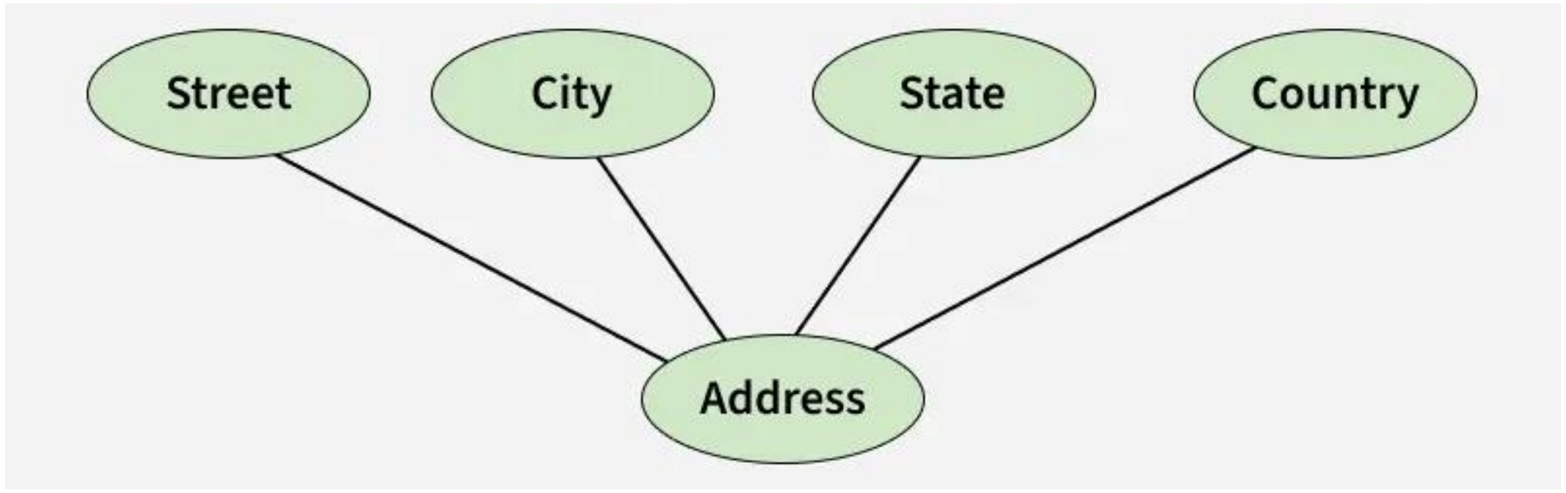
**1. Key Attribute**

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 an underline.



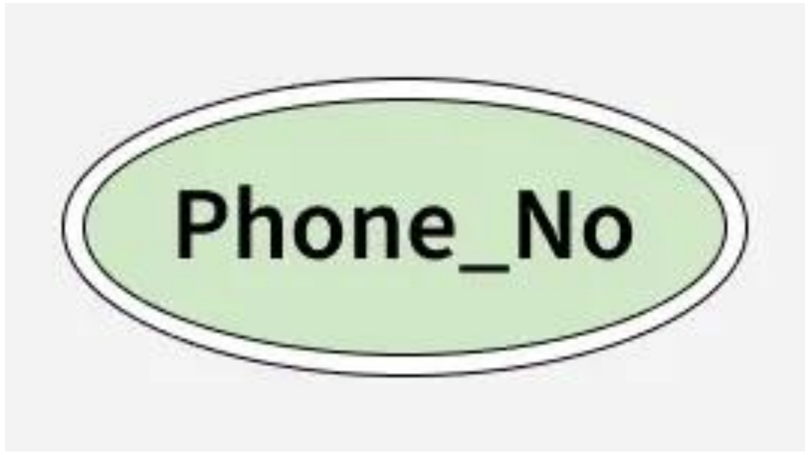
## 2. Composite Attribute

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 ER diagram, the composite attribute is represented by an oval comprising of ovals.



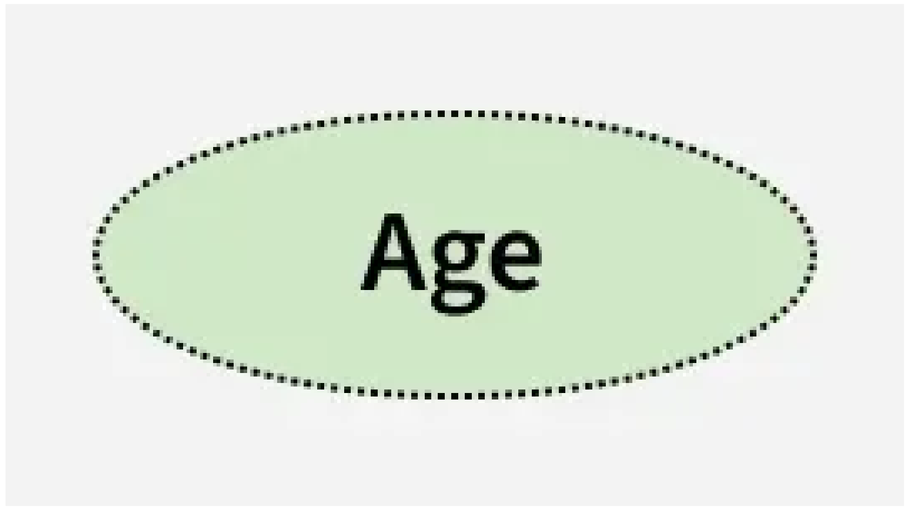
## 3. Multivalued Attribute

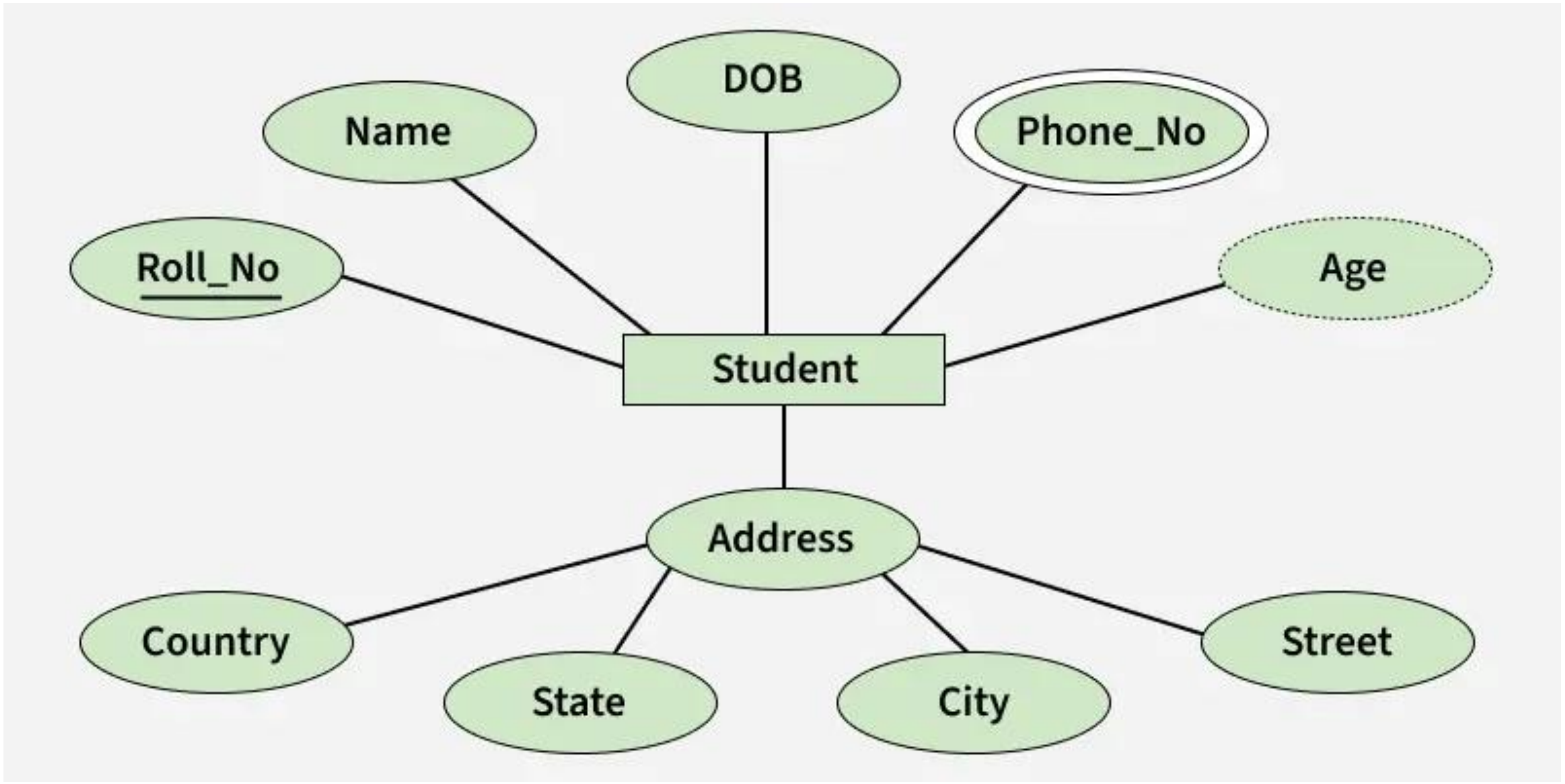
An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



## 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 ER diagram, the derived attribute is represented by a dashed oval.**

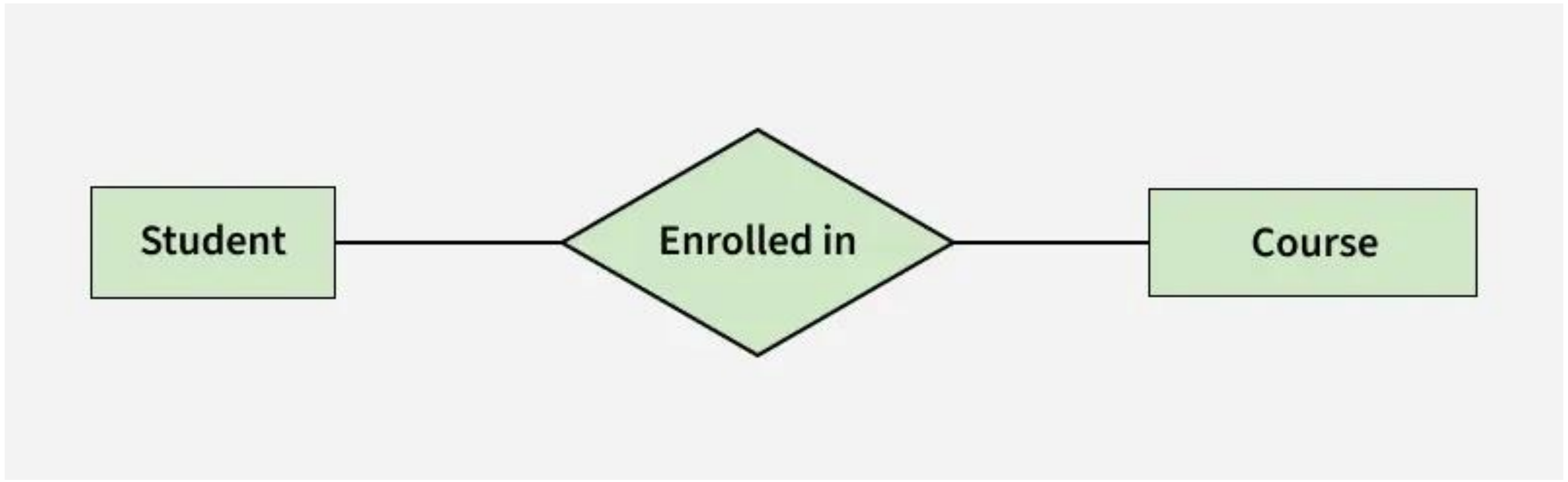




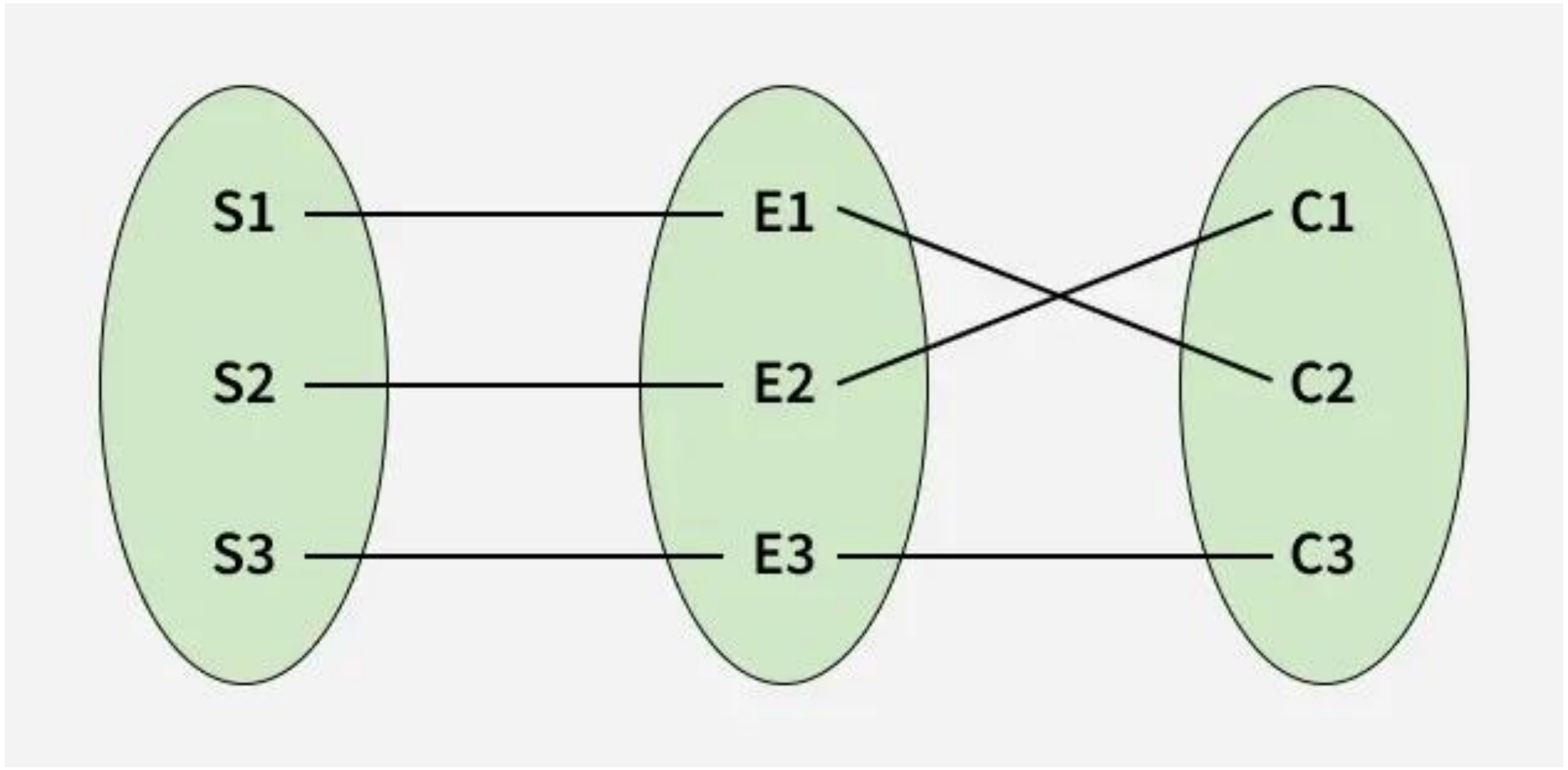
**The Complete Entity Type Student with its Attributes**

## Relationship Type and Relationship Set

A Relationship Type represents the association between entity types. For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.

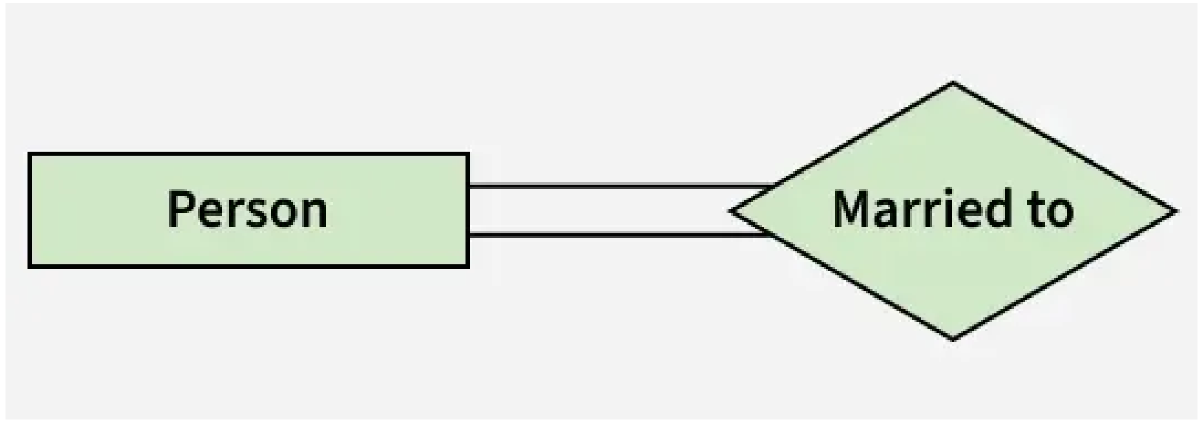


A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

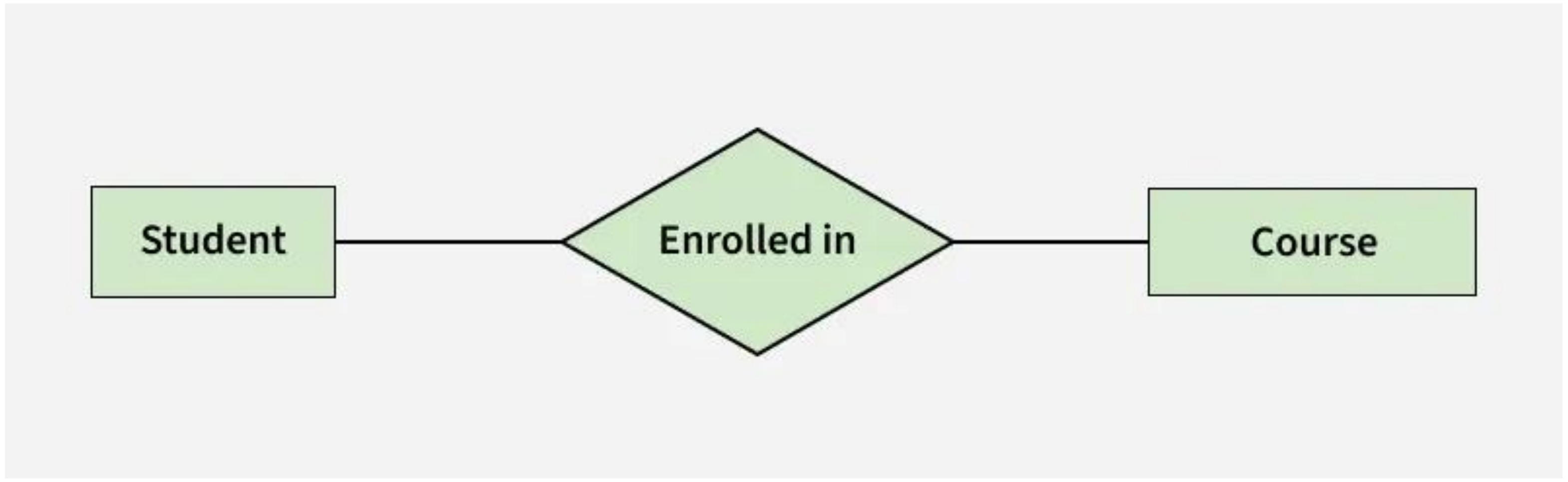


## Degree of a Relationship Set

- The number of different entity sets participating in a relationship set is called the degree of a relationship set.
- **1. Unary/Recursive Relationship:** When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



**2. Binary Relationship:** When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.

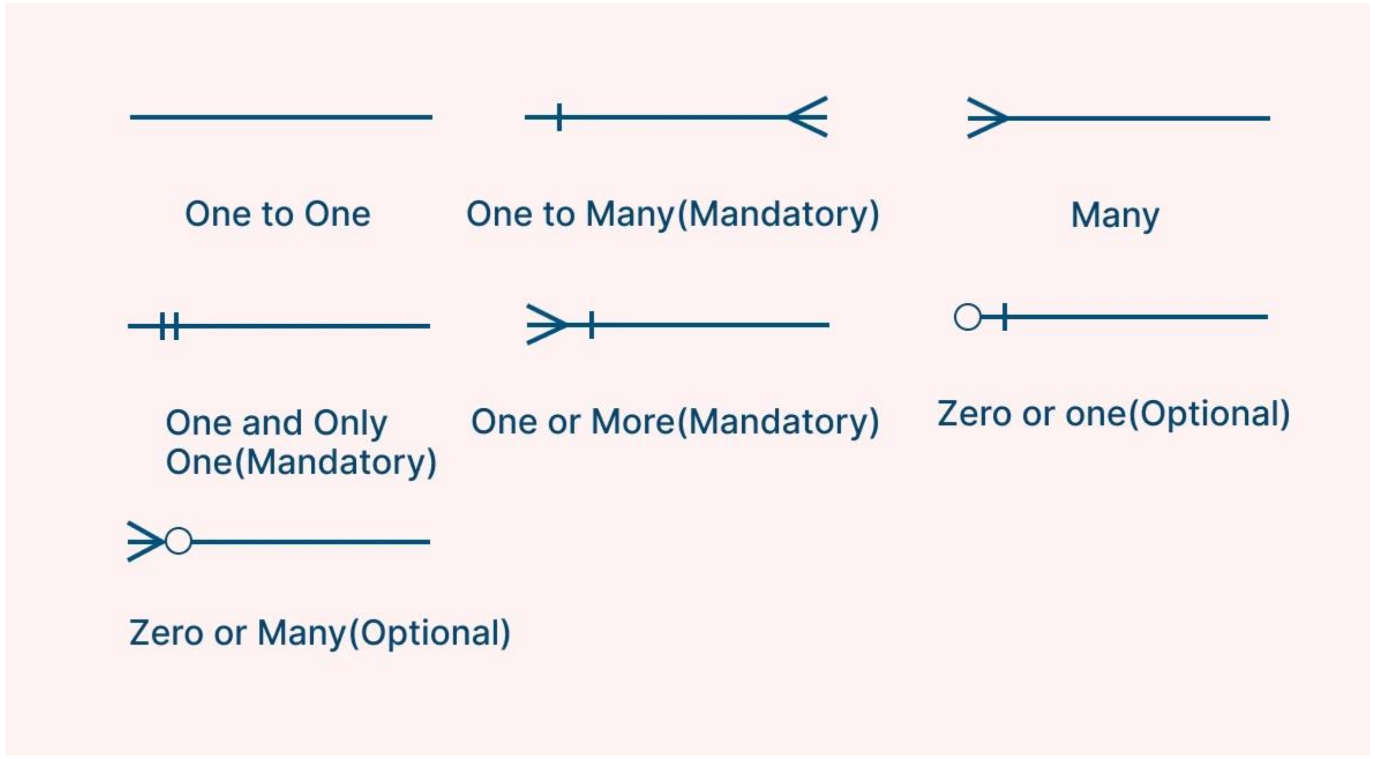


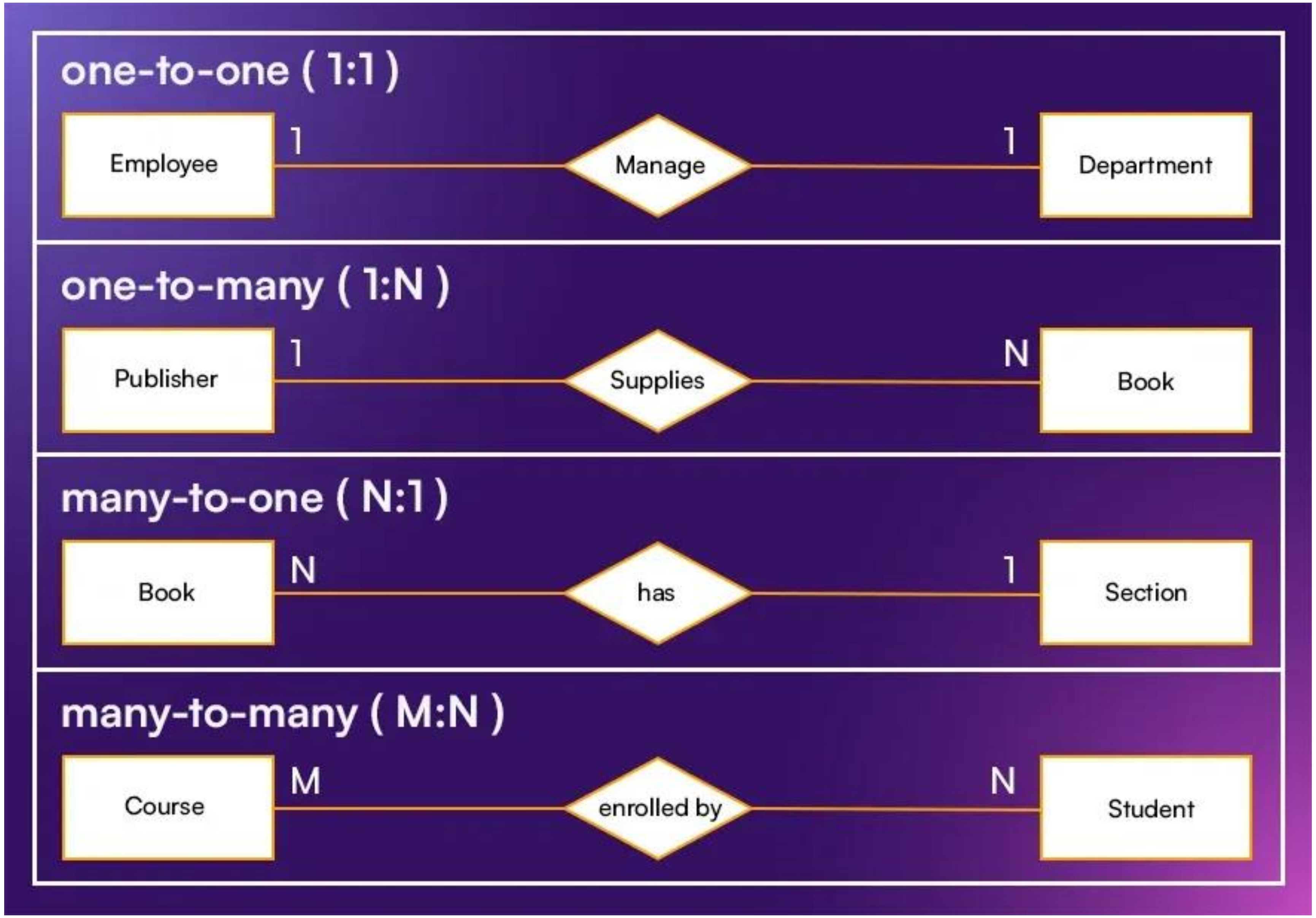
**3. Ternary Relationship:** When there are three entity sets participating in a relationship, the relationship is called a ternary relationship.

**4. N-ary Relationship:** When there are n entities set participating in a relationship, the relationship is called an n-ary relationship.

## Cardinality in ER Model

- The maximum number of times an entity of an entity set participates in a relationship set is known as **cardinality**.

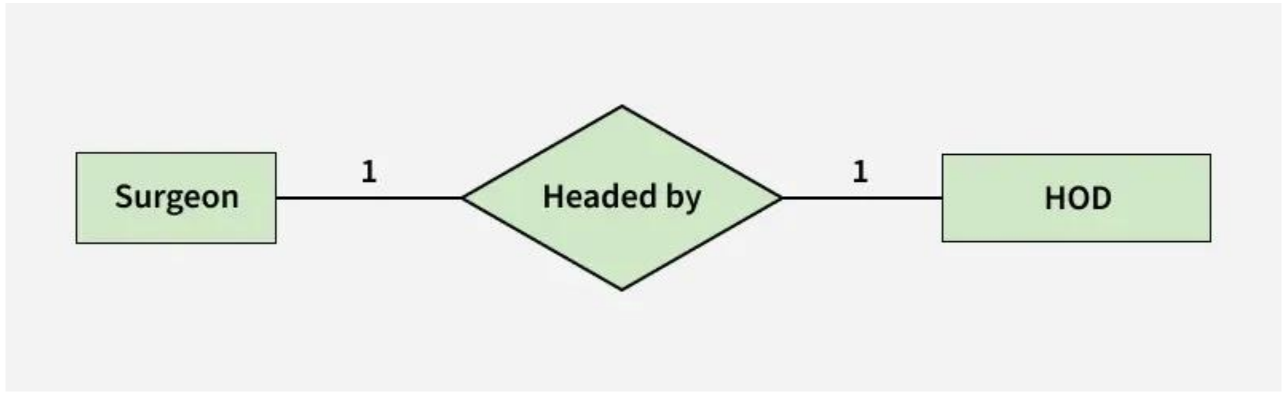



one-to-one ( 1:1 )
Employee
1
Manage
1
Department
one-to-many ( 1:N )
Publisher
1
Supplies
N
Book
many-to-one ( N:1 )
Book
N
has
1
Section
many-to-many ( M:N )
Course
M
enrolled by
N
Student

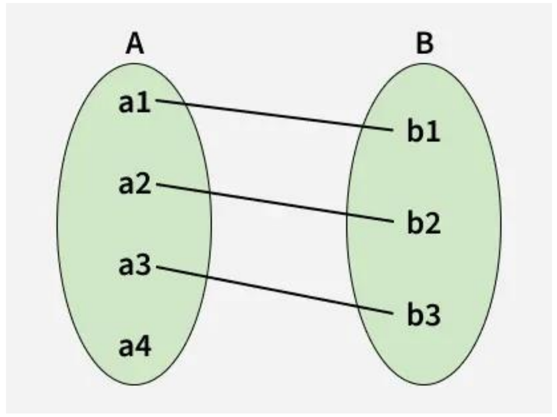
## Types of Cardinality

### 1. One-to-One

When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.



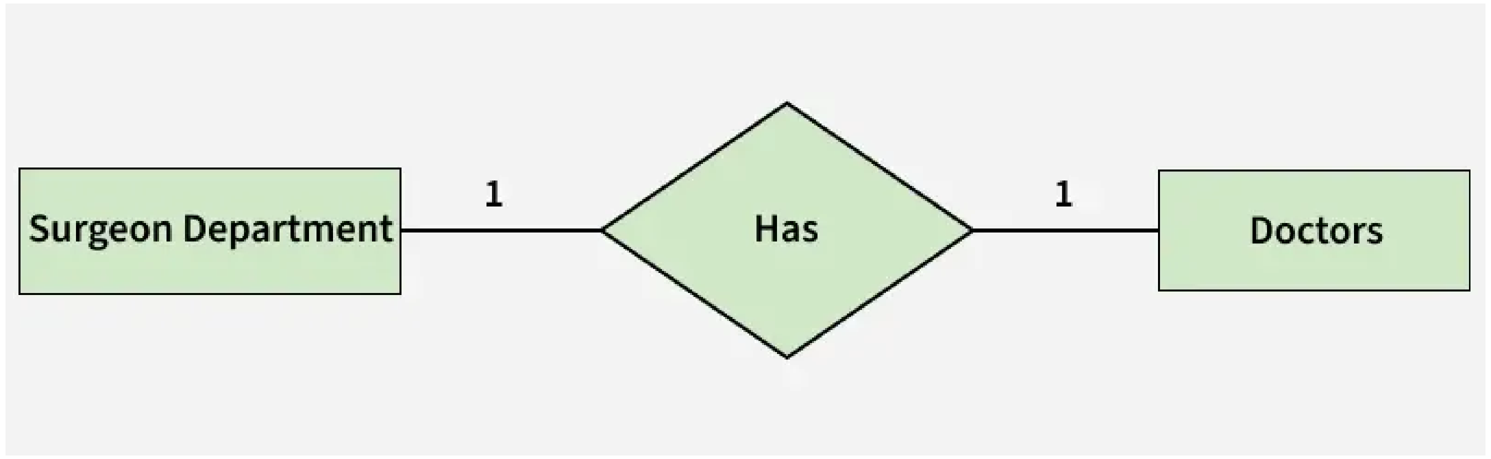
*One to One Cardinality*



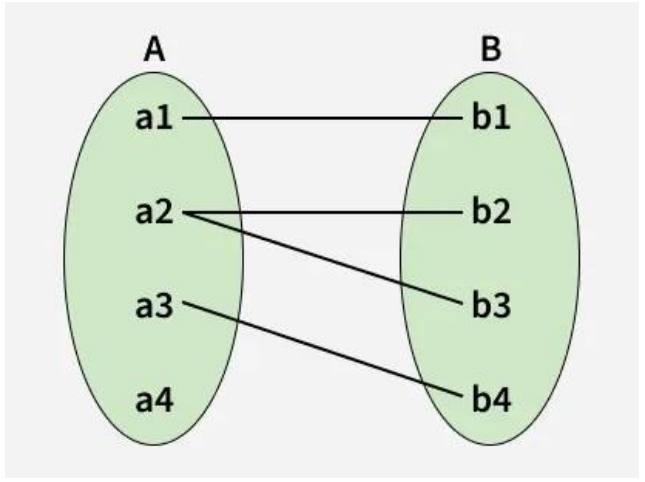
*Set Representation of One-to-One*

## 2. One-to-Many

In one-to-many mapping as well where each entity can be related to more than one entity. Let us assume that one surgeon department can accommodate many doctors. So the Cardinality will be 1 to M. It means one department has many Doctors.



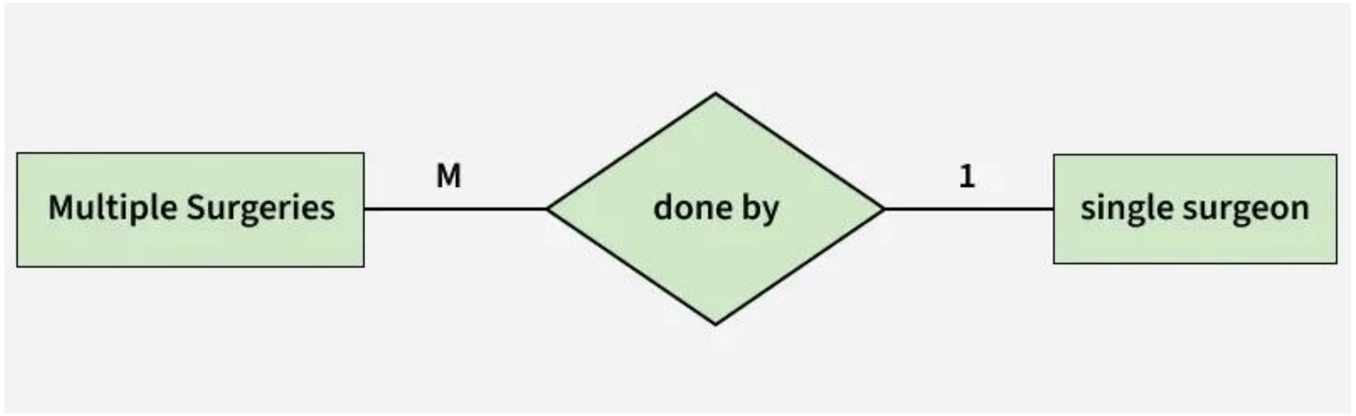
*one to many cardinality*



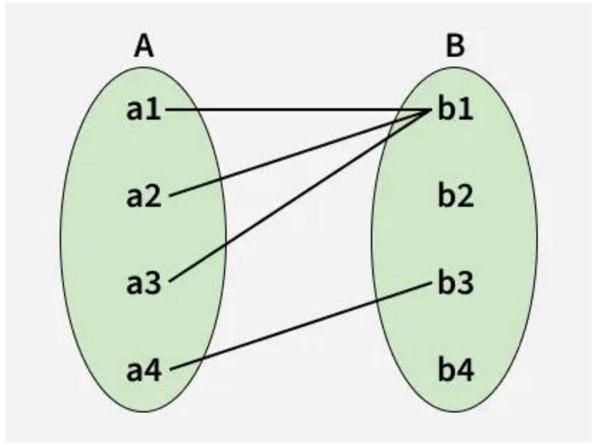
*Set Representation of One-to-Many*

## 3. Many-to-One

- When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one.
- Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course. In this case, each student is taking only 1 course but 1 course has been taken by many students.



*many to one cardinality*

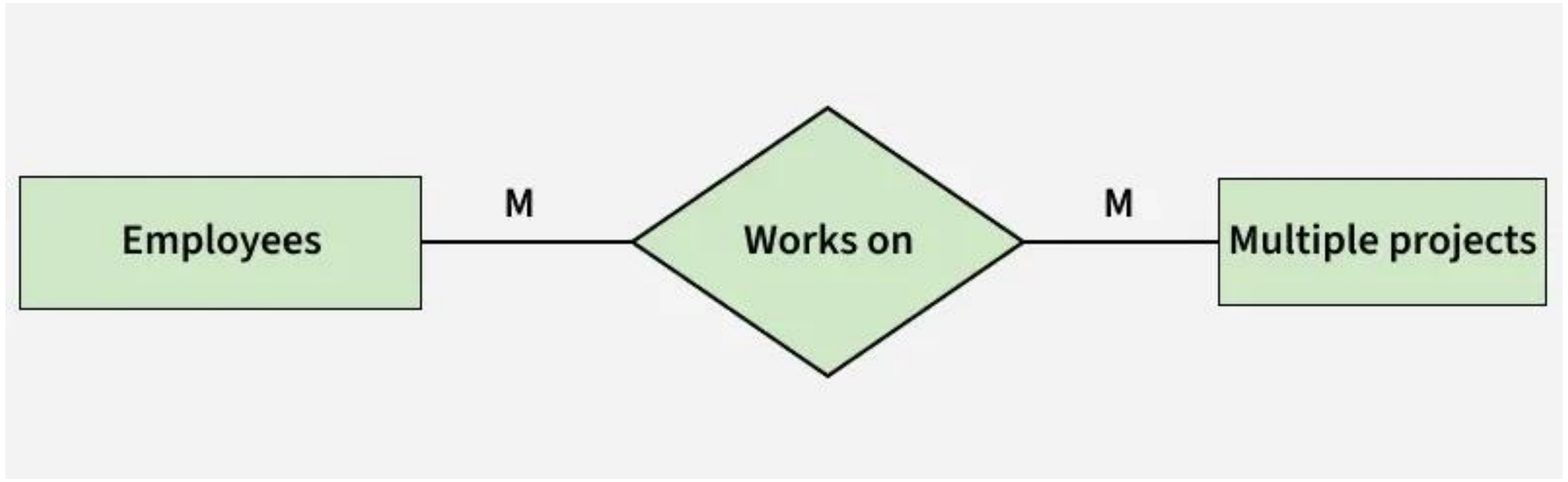


*Set Representation of Many-to-One*

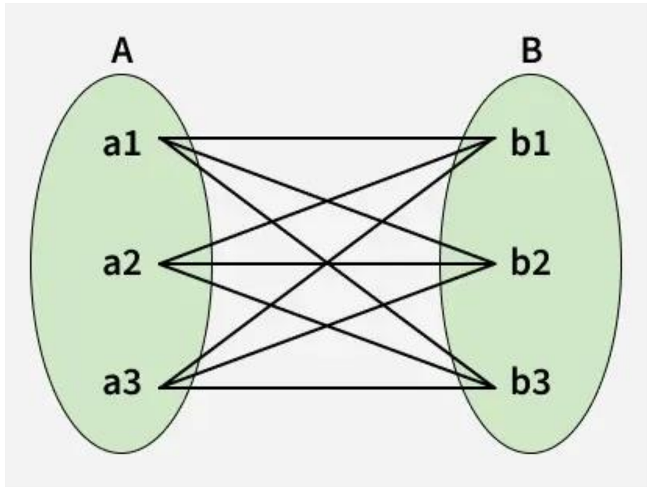
## 4. Many-to-Many

When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3, and S4.



many to many cardinality



Many-to-Many Set Representation

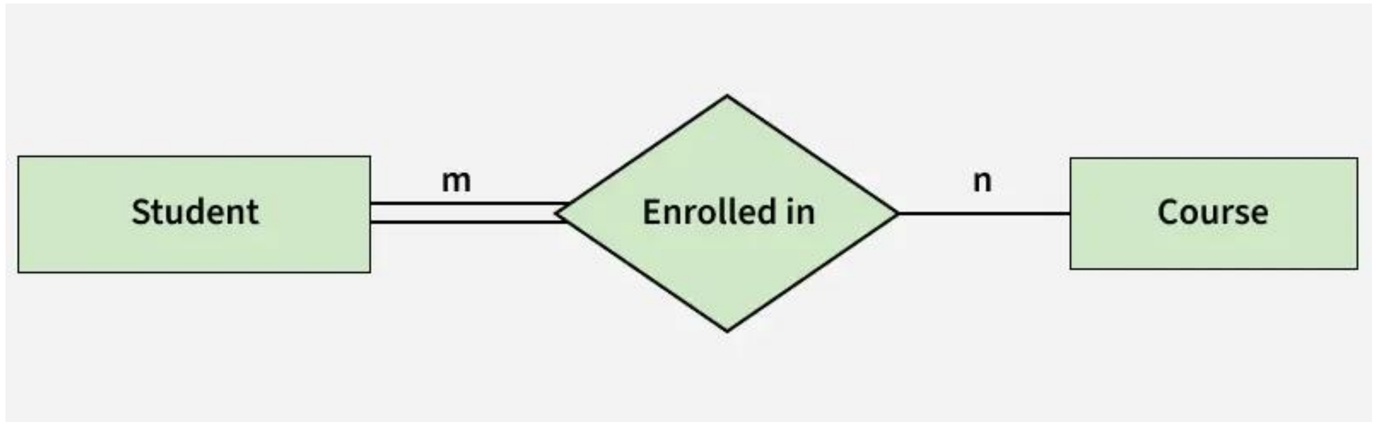
## Participation Constraint

Participation Constraint is applied to the entity participating in the relationship set.

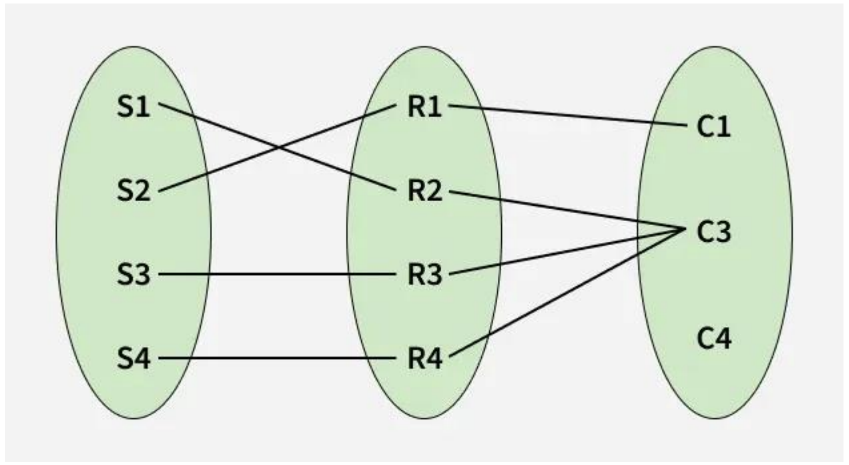
- **1. Total Participation:** Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.

- **2. Partial Participation:** The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.



*Total Participation and Partial Participation*



*Set representation of Total Participation and Partial Participation*

## How to Draw an ER Diagram

**1. Identify Entities:** The very first step is to identify all the Entities. Represent these entities in a Rectangle and label them accordingly.

**2. Identify Relationships**: The next step is to identify the relationship between them and represent them accordingly using the Diamond shape. Ensure that relationships are not directly connected to each other.

**3. Add Attributes**: Attach attributes to the entities by using ovals. Each entity can have multiple attributes (such as name, age, etc.), which are connected to the respective entity.

**4. Define Primary Keys:** Assign primary keys to each entity. These are unique identifiers that help distinguish each instance of the entity. Represent them with underlined attributes.

**5. Remove Redundancies**: Review the diagram and eliminate unnecessary or repetitive entities and relationships.

**6. Review for Clarity**: Review the diagram make sure it is clear and effectively conveys the relationships between the entities.

## Homework

- Draw an ERD for: **Clinic** with Doctor, Patient, Appointment, Room.
  - Each appointment has **date_time**, **reason**, and **status**; every appointment must have exactly one doctor, one patient, one room.
  - A room may have zero appointments; each doctor may have zero or more appointments; each patient may have zero or more appointments.
- Identify attribute types (simple/derived/etc.).
- Mark cardinalities and participation.
- Bonus: Which attributes (if any) belong on relationships?

## References

- Hernandez, M. J. (2013). *Database design for mere mortals: A hands-on guide to relational database design* (3rd ed.). Addison-Wesley Professional.
- Stallings, W. (2022). *Computer organization and architecture: Designing for performance* (11th ed.). Pearson.
- GeeksforGeeks. (2025, November 6). *Introduction of ER Model – DBMS.* https://www.geeksforgeeks.org/dbms/introduction-of-er-model/

Any
Question