Entity Relationship Model (ER Model)

Entity relationship model is a high-level conceptual model, used to define and design the logical structure of databases.

It visually represents the data, relationships and constraints.

ER modelling can be done using Entity Relationship Diagrams

Terms used in ER Modelling

Entity	Entity Set	Attribute
Relationship	Relationship Set	Key Attribute
Cardinality	Participation	Weak entity
Generalization	Specialization	Constraint
Sub-class	Super-class	Inheritance

Let’s see some important components of ER Models with examples

Entity

Definition	Examples	Representation
• An entity represents a real-world object or concept that can have data stored about it.	• Employee, Customer, Item, etc.	• Using a rectangle*

Types of Entities

- Strong Entity
- Week Entity
- Associative Entity

Strong Entity

Definition

- A **strong entity** is an entity that can be uniquely identified by its own attributes, without relying on any other entity. It has a primary key that uniquely identifies it.

Characteristics

- It has a **primary key** that uniquely identifies each entity.
- It is independent and does not rely on any other entity for its identification.
- It is typically represented by a **single rectangle** in ER diagrams.

Example

- An employee with attributes like employee_id, name, email, etc

Weak Entity

Definition

- A **weak entity** is an entity that cannot be uniquely identified by its own attributes alone. Instead, it relies on a **strong entity** (also called the owner or identifying entity) to form its identification

Characteristics

- A weak entity cannot exist without a corresponding strong entity.
- It uses a **partial key** along with the **key from the strong entity** to be uniquely identified
- It is represented by a **double rectangle in ER diagrams**.

Example

- A **dependent** who is related to an employee
- Here the existence of **dependent** relies on the existence of employee (strong entity).
- So, a composite primary key that contains emp_id, dep_id can be used to uniquely identify the **dependent**

Associative Entity

Definition

- An **associative entity** is a special kind of entity that is used to **represent many-to-many (M:N)** relationships between two or more entities.
- Used in the cases where direct relationship is not feasible between entities.

Characteristics

- It typically involves a **many-to-many** relationship between entities.
- It may have its own attributes that are not found in the participating entities.
- It is represented by a rectangle with a diamond.

Example

- An associative **enrollments** entity connecting **student** and **course** with additional attributes such as **enrollment_date**

Attribute

Definition	Examples	Representation
• An attribute is a property or characteristic of an entity.	• name, salary, age, etc. of an employee (entity)	• Using an oval shape

Types of attributes

Simple (Atomic) Attributes

Composite Attributes

Derived Attributes

Multi-Valued Attributes

Key-Attributes

Simple Attributes

These are attributes that cannot be divided further

They hold a single value for an entity

Example

- Salary
- First name
- Last name

Represented using **oval**

Composite Attributes

Composite attributes are made up of multiple simple attributes that together represent a single entity property

Each part of a composite attribute can be individually useful

Examples

- Full Name (which could be composed of FirstName and LastName)
- Address (could include Street, City, State, Pincode).

Represented using **oval** connected to other **ovals**

Derived Attributes

Derived attributes are those whose values can be **calculated or derived from other attributes**

They are not physically stored in the database but can be computed when needed

Example

- Age (Can be derived from Date of Birth).
- Provident Fund (PF) (can be derived from salary).

Represented using **dotted oval**

Multi-Valued Attributes

These attributes can have multiple values for a single entity

Each value of the multi-valued attribute represents an independent piece of information, but they are all related to a single entity

Examples

- PhoneNumbers (a person can have multiple phone numbers)
- Emails (an employee can have multiple email addresses)

Represented using **double oval**

Key Attributes

Key attributes are those that uniquely identify an entity

Examples

- employee_id in **employee entity** which uniquely identifies each employee
- order_id in **order entity** which uniquely identifies each order made

Represented using **oval with underlined attribute name**

Relationship

Definition	Examples	Representation
• A relationship is **an association between** two or more entities. • It represents how entities interact with one another.	• Employee **works on** a project • User **makes** a submission • Student **enrolls in** a course	• Using a **diamond shape**

Cardinality

Definition

- Cardinality defines **the number of instances of one entity** that can or must be associated with **instance(s) of another entity**

Examples

- **A user** can have only **one account** (1:1)
- **A user** can have **multiple accounts** (1:N)
- **Many users** can participate in **many contests** (M: N)

Representation

- Typically connects two entities with a relationship (diamond symbol)

Types of Cardinality

- One-to-One (1:1) Cardinality
- One-to-Many (1:N) Cardinality
- Many-to-One (N:1) Cardinality
- Many-to-Many (M:N) Cardinality
- Zero or Optional Cardinality (0..1 or 0..N)

One-to-One (1:1) Cardinality

In this type of relationship, one instance of an entity is associated with **exactly one instance** of another entity, and vice versa

A Person and a Passport. Each person has one passport, and each passport is assigned to exactly one person

This is usually represented by a single line between two entities in an ER diagram.

One-to-Many (1:N) Cardinality

In this type of relationship, one instance of an entity is associated with **multiple instances** of another entity, but each instance of the second entity can be associated with **only one instance** of the first entity

A Department and Employee. A department can have many employees, but each employee works in only one department

Represented by a single line from the **one** side (the “1” side) to the **many** side (the “N” side) in the ER diagram

Many-to-One (N:1) Cardinality

This is essentially the reverse of the **One-to-Many (1:N)** relationship. Here, **many instances** of one entity are associated with **exactly one instance** of another entity

Many Employees can work in one Department. Each employee belongs to only one department

Represented by a single line from the **many** side (the “N” side) to the **one** side (the “1” side) in the ER diagram

Many-to-Many (M:N) Cardinality

In this type of relationship, **many instances** of one entity can be associated with **many instances** of another entity

Students and Courses. A student can enroll in multiple courses, and a course can have multiple students enrolled

Represented by a line connecting two entities with “m” on one side and “n” on the other representing many instances from both entities are related to each other

Role / Uses of ER Modelling in Software Development

Visualizing Data Structure

- ER models provide a clear, visual representation of how data entities relate to one another, helping developers and understand the data structure before actual implementation.

Database Design

- ER diagrams serve as a blueprint for creating tables, defining primary and foreign keys, and structuring the database.

Improved System Maintenance

- A well-designed ER model makes it easier to understand the system's data flow and relationships, thus aiding in future updates, debugging, and scaling.

Role / Uses of ER Modelling in Software Development

Code Efficiency

- By defining the data relationships early on, ER modeling minimizes the risk of inefficiencies in the database schema, ensuring that developers can write optimized code that interacts with the database in a streamlined manner.

Communication Tool

- ER diagrams serve as a common language between developers, designers, and non-technical stakeholders. They facilitate discussions about how data will be used and processed in the system.

Ensuring Data Integrity

- By defining the relationships between different entities (such as one-to-many, many-to-many), ER modeling ensures data integrity, making sure that the database schema supports efficient data storage and retrieval without redundancy.