

# ER Model Explained

## 1. Data Model

A **data model** is a *set of concepts* that helps us describe:

- how data is stored,
- how data is related,
- what the data means (semantics),
- and what rules must be followed to keep the data consistent.

In simple words:

**Data Model = A structured way of organizing and understanding data.**

It gives us a *blueprint* before storing data in a database.

## Example

Let's say we want to store information about **students** in a college database.

### Using a Data Model (Relational Model Example)

1. **Describe the Data (Concepts)**
  - Student has: StudentID, Name, Branch, Age
2. **Describe Relationships**
  - A student can enroll in *many courses*
  - A course can be taken by *many students*  
→ This is a **many-to-many relationship**
3. **Describe Data Semantics**
  - StudentID uniquely identifies each student
  - CourseID uniquely identifies each course
  - Enrollment table connects students and courses
4. **Describe Consistency Rules (Constraints)**
  - **Primary Key:** StudentID cannot be duplicated
  - **Foreign Key:** Enrollment.StudentID must exist in Student table
  - **Age Constraint:** Age must be a positive number
  - **Branch must be one of {CSE, ECE, ME, EE}**

## 2. ER Model

### 1. What is an ER Model?

The **Entity–Relationship (ER) Model** is a **high-level data model** used to describe how real-world objects are connected.

- Real-world data is viewed as **entities** (objects)

- The connections between them are **relationships**

In simple terms:

**ER Model helps us understand the structure of data before creating the database.**

## Entities

Entities are real-world objects that have:

- a name
- properties (attributes)

**Examples:**

Student, Teacher, Course, Department

## Relationships

Relationships show how two entities are connected.

**Examples:**

- A *student* enrolls in a *course*
- A *teacher* teaches a *course*

## 2. ER Diagram – The Blueprint of a Database

The **graphical representation** of an ER Model is called an **ER Diagram (ERD)**.

It shows:

- **Entities** as rectangles
- **Attributes** as ovals
- **Primary key** underlined
- **Relationships** as diamonds
- **Connections** using lines

This diagram acts as the **blueprint of the database**.

Just like engineers draw building plans before construction,  
**database designers draw ER diagrams before creating tables.**

## Example

Let's take a simple college database.

### Entities

1. **Student**
  - StudentID (Primary Key)

- Name
- Age
- Branch

## 2. Course

- CourseID (Primary Key)
- CourseName
- Credits

## Relationship

- **Enrolls** → connects Student and Course
- Type: Many-to-Many  
(Because one student can take many courses, and one course can be taken by many students)

## How the ER Diagram Conceptually Looks

STUDENT ----- ENROLLS ----- COURSE  
 (Entity)              (Relationship)              (Entity)

|                |               |
|----------------|---------------|
| StudentID (PK) | CourseID (PK) |
| Name           | CourseName    |
| Age            | Credits       |
| Branch         |               |

(Attributes appear connected by ovals in an actual ER diagram.)

## Why ER Model is Important

- Gives a clear understanding of data
- Reduces design mistakes
- Helps convert the design into tables
- Maintains logical structure before implementation

## 3. Entity

An **Entity** is any real-world object or “thing” that can be clearly identified from others.

Key Points:

### 1. Has physical or conceptual existence

Example: A student, teacher, book, department, car, etc.

## 2. Each student in a college is an entity

Every student is separate from others.

## 3. Entity can be uniquely identified

This is done using a **primary attribute** → called the **Primary Key**.

Example: StudentID uniquely identifies each student.

## Strong Entity

A **Strong Entity** is an entity that:

- can be uniquely identified on its own
- has a primary key within itself
- does NOT depend on any other entity for identification

## Example

### Loan

- Each loan has a **LoanNo** (primary key)
- LoanNo itself is enough to identify each loan  
So, Loan is a **strong entity**.

## Weak Entity

A **Weak Entity** is an entity that:

- **cannot be uniquely identified** by its own attributes
- **depends on a strong entity** for identification
- does not have enough attributes to define a primary key
- uses a **partial key + primary key of strong entity** to uniquely identify each instance

Key Characteristics:

1. It doesn't have sufficient attributes to create its own unique identifier.
2. Needs a strong entity to identify it.
3. Exists only when the strong entity exists.

## Example

### Payment (related to a Loan)

- A loan has multiple payments (installments).
- Each payment has a number like: 1, 2, 3... but these numbers repeat across different loans.
- Payment 1 of Loan A is different from Payment 1 of Loan B.

So, “PaymentNumber” alone **cannot uniquely identify** a payment.

To uniquely identify a payment, we need:

- **LoanNo** (from strong entity Loan)
- **PaymentNumber** (partial key)

Therefore:

**Loan → Strong Entity**

**Payment → Weak Entity**

## Why Weak Entity Depends on Strong Entity

A payment cannot exist without a loan.

If the loan is deleted, its payments should also disappear.

This shows **existence dependency**.

## 4. Entity Set

An **Entity Set** is a **collection of entities of the same type** that share the same attributes or properties.

In simple words:

If an entity is a single object,

then an **entity set** is the group of all such similar objects.

## Key Points

### 1. Collection of same-type entities

All entities in the set have the same attributes.

Example: Every student has attributes like StudentID, Name, Age, Branch.

### 2. Example: Student Entity Set

All students in a college together form the **Student** entity set.

Each individual student is one entity.

### 3. Example: Customer Entity Set

All customers of a bank form the **Customer** entity set.

Each customer is one entity but they share the same properties like CustomerID, Name, Address, AccountType.

## 5. Attributes

Attributes are the **properties** or **characteristics** that describe an entity.

## Key Points

### 1. An entity is represented by a set of attributes.

Example: A student is described using StudentID, Name, Age, etc.

### 2. Each entity has a value for each attribute.

Example:

- StudentID → 101
- Name → Rohan

- Contact → 9876543210

### 3. Each attribute has a domain (value set).

The **domain** is the set of allowed values for that attribute.

- Age domain → positive integers
- Name domain → alphabetic strings
- Contact Number domain → 10-digit numeric values

### 4. Example: Student Entity Attributes

A **Student** entity may have:

- Student\_ID**
- Name**
- Standard** (e.g., 1st year, 2nd year, etc.)
- Course** (CSE, ECE, ME, etc.)
- Batch** (2023–27, 2024–28, etc.)
- Contact Number**
- Address**

Each of these attributes helps uniquely and completely describe a student.

## 6.Types of Attributes

Attributes describe the properties of an entity. These attributes can be classified into different types based on their nature and behavior.

### 1. Simple Attribute

- These attributes **cannot be divided further** into smaller parts.
- They represent a single, indivisible value.

#### Examples:

- Customer's Account Number
- Student's Roll Number
- Employee ID

These values stand alone and have no subcomponents.

### 2. Composite Attribute

- These attributes **can be broken down** into smaller attributes (subparts).
- Useful when a user may need to use the whole attribute or only a part of it.

#### Examples:

- **Name** → First Name, Middle Name, Last Name
- **Address** → Street, City, State, PIN Code

Composite attributes help in organizing detailed data.

### 3. Single-Valued Attribute

- These attributes hold **only one value** for a particular entity.

#### Examples:

- Student ID
- Loan Number
- PAN Number

Every entity has exactly one value for these attributes.

### 4. Multi-Valued Attribute

- These attributes can store **more than one value** for a single entity.
- Limit constraints may be applied (e.g., up to 2 phone numbers).

#### Examples:

- Phone Numbers
- Nominee Names in an insurance policy
- Dependent Names of an employee

These attributes allow multiple valid entries.

### 5. Derived Attribute

- These attributes' values can be **calculated** or **derived** from other attributes.
- They are not stored directly; they depend on related values.

#### Examples:

- Age → derived from Date of Birth
- Loan Age → derived from Loan Start Date
- Membership Period → derived from Join Date

Derived attributes help avoid storing redundant data.

### 6. NULL Value

An attribute is **NULL** when an entity does not have a valid value for it.

NULL may indicate:

- a) **Not Applicable**

- Example: A person without a middle name  
→ Middle-Name attribute will be NULL.

## b) Value Does Not Exist

- Example: A student with no registered phone number yet

## c) Unknown

This has two meanings:

### 1. Missing Entry

- Example: Customer name is NULL → value missing but must exist

### 2. Not Known Yet

- Example: Salary of a new employee is NULL → not known at the moment

NULL does **not** mean zero, blank, or zero-length; it simply means “value not present.”

# 7. Relationships

A **relationship** is an **association** among two or more entities.

## Examples

- Person **has** Vehicle
- Parent **has** Child
- Customer **borrows** Loan

## Types of Relationships

### 1. Strong Relationship

- Relationship between **two strong (independent) entities**.
- Both entities can exist on their own.

### 2. Weak Relationship

- Relationship between a **weak entity** and its **owner (strong entity)**.
- Weak entity **depends** on strong entity for identification and existence.

### Example:

Loan —<instalment-payments>— Payment

- Loan = Strong
- Payment = Weak

## Degree of Relationship

The **degree** shows how many entity sets participate in a relationship.

## 1. Unary Relationship

- Only **one** entity participates.
- Example: Employee **manages** Employee  
(Both manager and subordinate belong to the same entity set.)

## 2. Binary Relationship

- **Two** entity sets participate.
- Most common in databases.

### Examples:

- Student **takes** Course
- Customer **borrow**s Loan

## 3. Ternary Relationship

- **Three** entity sets participate.

### Example:

Employee **works-on** Job **in** Branch

Here: Employee, Job, Branch → all three involved.

# 8. Relationship Constraints

Relationship constraints define **how many entities** participate and **how they must participate**.

## A. Mapping Cardinality (Cardinality Ratio)

Shows **how many entities** of one set associate with entities of another set.

### 1. One-to-One (1:1)

- One entity in A ⇔ at most one entity in B
- Example:  
**Citizen has Aadhar Card**  
One citizen → one Aadhaar  
One Aadhaar → one citizen

### 2. One-to-Many (1:N)

- One entity in A → many entities in B
- Each entity in B → at most one in A

- Example:

### **Citizen has Vehicle**

One citizen → many vehicles  
Each vehicle → one citizen

## **3. Many-to-One (N:1)**

- Many entities in A → one entity in B
- Example:

### **Course taken by Professor**

Many courses → one professor  
One professor → many courses

## **4. Many-to-Many (M:N)**

- Many in A ↔ many in B
- Examples:
  - Customer **buys** Product
  - Student **attends** Course

Many customers can buy many products, and vice versa.

## **B. Participation Constraints**

Also known as **minimum cardinality constraints**.

They define **whether participation in a relationship is optional or mandatory**.

### **1. Partial Participation (Optional)**

- Not all entities must participate.
- Example:
 

#### **Customer borrows loan**

  - Not every customer has a loan  
→ Customer = Partial Participation

### **2. Total Participation (Mandatory)**

- Every entity **must** participate in at least one relationship.
- Example:
 

#### **Loan must belong to a customer**

  - Loan cannot exist without a customer  
→ Loan = Total Participation

### **Important Rule**

- **Weak entity always has total participation**, because it cannot exist without its owner.

## **9. ER Notations**

These are standard symbols used in ER diagrams:

- **Entity (Strong)** → Rectangle
- **Entity (Weak)** → Double Rectangle
- **Relationship (Strong)** → Diamond
- **Relationship (Weak)** → Double Diamond
- **Attribute** → Oval
- **Key Attribute** → Oval with underline
- **Multivalued Attribute** → Double Oval
- **Derived Attribute** → Dashed Oval
- **Cardinality** → (1:1), (1:N), (N:M) written near relationship lines
- **Participation** →
  - Double line (Total)
  - Single line (Partial)

#### Symbols used in ER Diagram

