**Data Models and its Type**

A **data model** in a Database Management System (DBMS) is an abstract way to represent how data is structured and organized within a database. It shows the logical arrangement of data and the connections between different data components. Data models are crucial for understanding and designing databases, linking real-world entities to actual data storage.

For instance, before writing an algorithm for “Making Maggi,” having a flowchart makes it easier to implement the algorithm. Similarly, having a data model helps in understanding the relationships between different components in the database.

**Types of Data Models**

* **Hierarchical Data Model**

This model displays data in a tree structure, where each record has a parent-child relationship. It is mainly used in older database systems.

* **Network Data Model**

This model allows records to have multiple parent-child relationships, resembling a graph. It offers more flexibility than the hierarchical model.

For example, the image shows that "Student" has both "Lab" and "School" as parents.

* **Relational Data Model**

The relational model organizes data into tables (known as relations) with rows and columns. It is the most common data model and is based on set theory, using Structured Query Language (SQL) for data manipulation.

* **Entity-Relationship Model (ER Model)**

The ER model is used to design relational databases by representing data as entities (objects), attributes (properties of entities), and the relationships connecting these entities.

* **Object-Oriented Data Model**

This model applies the principles of object-oriented programming to databases. It represents data as objects with attributes and methods, supporting inheritance and encapsulation.

* **NoSQL Data Models**

NoSQL databases offer various data models, including:

* + **Document-oriented** (e.g., MongoDB)
  + **Key-value** (e.g., Redis)
  + **Column-family** (e.g., Cassandra)
  + **Graph** (e.g., Neo4j)

These models are designed for scalability and flexibility, particularly when managing large amounts of unstructured or semi-structured data

**ER Model and its components Entity, Attributes & Relationship**

**Entity and its types**

An Entity is something from the real world, like a person, place, event, or idea. Each entity has specific features or traits that describe it.

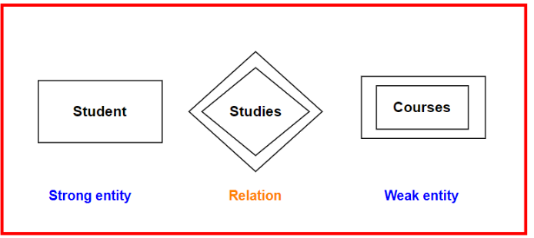
There are two types of entities:

**Strong Entity**

A strong entity is an entity that has its unique identifier (primary key) and is not dependent on any other entity for its existence within the database. Strong entities stand alone and have their own set of attributes. For Example- An entity “person” can exist independently.

**Weak Entity**

A weak entity is an entity that doesn't have a primary key of its own. It relies on a related strong entity known as the "owner" entity for its identity. The weak entity’s existence is defined by being related to the owner entity. For example- In a company, employees can file for dependents under their name. In this case, the entity, “Dependents” is weak.



In the ER Model, the strong entity is represented in a rectangle, the weak entity in a double-edge rectangle and their relationship in a double-edged diamond shape, as shown in the figure

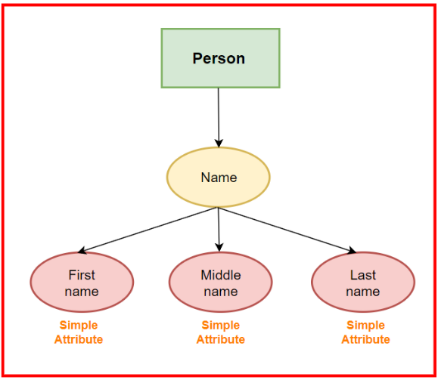
**Attribute and its Types**

Attributes are the distinguishing characteristics that define an entity in a relational database. They provide information about the entities and relationships in the database. They are represented with an elliptic curve in the ER Model.

**Types of attributes:**

**Simple attribute**

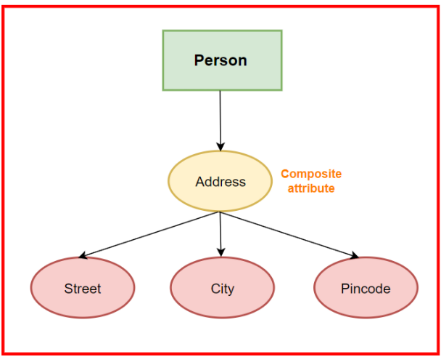
A simple attribute is atomic and cannot be divided any further. For example, in the figure, the “Person” attribute can be further divided into “First name” “Middle name” and “Last name”, making the letter atomic and simple attributes.



**Composite attribute**

A composite attribute is made up of several smaller parts, where each part represents a piece of the whole attribute. In simpler terms, it comprises attributes that can be divided further.

For example, “Name” (First name, middle name, last name), “Address” (Street, Pincode, City) etc.

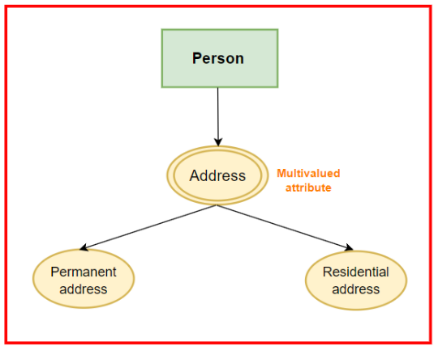


**Single-valued attribute**

A single-value attribute is an attribute that holds a single value for each entity. For example, Age, S.ID etc. They cannot have multiple values.

**Multivalued attribute**

A multi-valued attribute in a database is an attribute that can hold multiple values for a single entity. For example, Address (Permanent and residential address), Phone number (Father’s phone number, Mother’s phone number) etc.



**Stored attribute**

Attribute that is stored as a part of a database record. For example- Date of birth.

**Derived attribute**

A derived attribute is derived from other attributes within the database. For example- Age can be derived from Date of birth, and Salary can be derived from Base Salary, Stocks, Bonus etc.

**Complex attribute**

A complex attribute is an attribute that is made up of multiple smaller attributes. For example- “Name” and “Address” are complex attributes as they can be further divided into simple attributes.

**Key attribute**

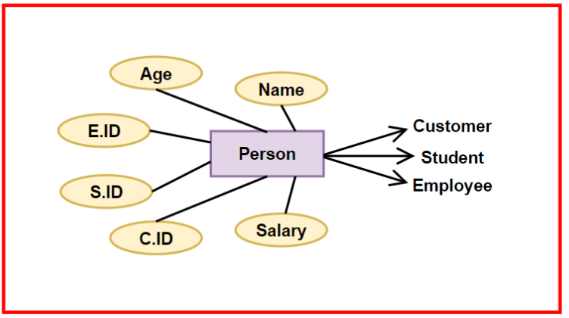
The attribute which uniquely identifies each entity in the entity set is called the key attribute. For example, Roll\_No will be unique for each student. In the ER diagram, the key attribute is represented by an oval with underlying lines.

**Extended ER features**

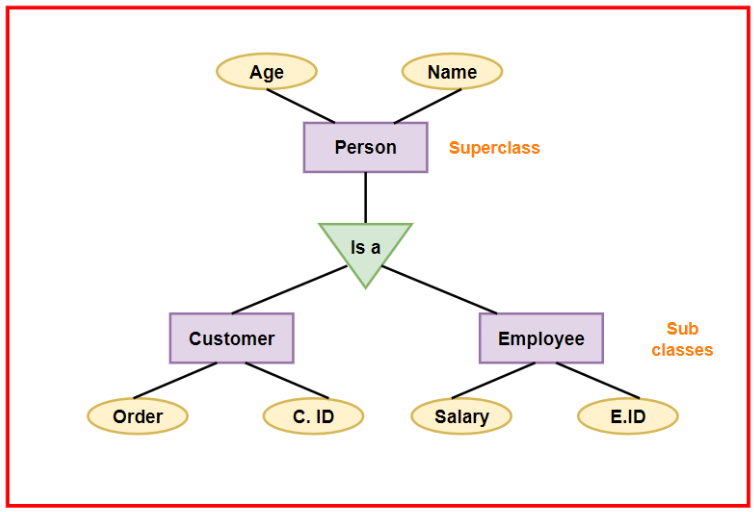
The Extended Entity-Relationship (EER) model is an enhancement of the basic Entity-Relationship (ER) model used in database design. It includes additional features that provide more detail and allow for a more accurate representation of complex real-world scenarios.

**Why do we need the EER model?**

We design an ER model for relationships between entities. In the real world, the data may exhibit some hierarchical relationships, and the EER model provides mechanisms to represent these relationships accurately, which helps in code reusability, ensuring data integrity and consistency, and lowering the complexity.



For example, in the figure above, the “Person” entity is overburdened with all the attributes. However, most attributes are specific to a particular category, like “C.ID” to “Customer,” “S.ID” to “Student,” and “E.ID” and “Salary” to “Employee.” “Name” and “Age” are common attributes for the “Customer,” “Student,” and “Employee” entities.





Hence, using EER features, we can reduce the complexity of the ER Model by introducing subclasses. This will also help in reducing the overburdening of just one entity, i.e., “Parent.”

**EER features**

* **Specialisation:** This is a top-down process where a general entity is divided into more specific entities based on certain attributes or relationships. For example, from a general "Person" entity, you might specialise in "Employee" and "Student."

Follows the property of inheritance, i.e., attributes of the parent class are inherited by the child class.

Example: From a general "Person" entity, we might specialise in "Employee" and "Student."

* **Generalisation:** This is a bottom-up process where multiple specific entities are combined into a more general entity. For example, "Car" and "Bike" might be generalised into a single "Vehicle" entity.

Follows the property of Abstraction, i.e., some attributes like “Model” and “Engine type” might be common in the child entities. They can be identified and given to the parent class to reduce redundancy.

Example: "Car" and "Bike" might be generalised into a single "Vehicle" entity.

* **Aggregation:** It can be thought of as stacking things on top of each other to create a structure. It is used to create a hierarchical structure in data modelling, showing how a higher-level entity is composed of lower-level entities.

This follows the property of Abstraction, as multiple lower-level entities are combined into one to form a higher-level entity, thereby abstracting the complexity from the user.

Example: A "Project" entity might be composed of "Tasks," where each task is an entity itself.

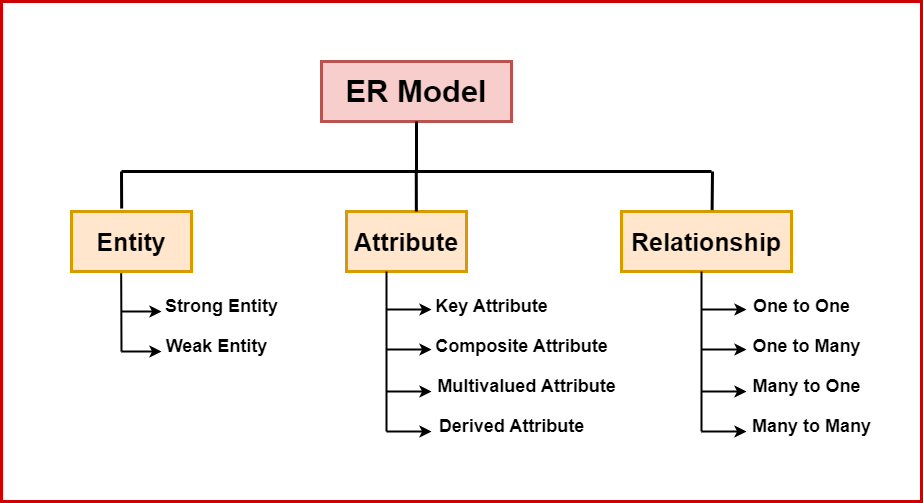
**Benefits of Using the Enhanced Entity-Relationship (EER) Model**

* **Enhanced Clarity and Structure:** The EER model provides a more detailed and hierarchical structure compared to the basic ER model. This detailed nature helps in creating a clear and organised database design. By visually representing entities, relationships, and their specialisations, the EER model makes it easier for designers and users to understand the structure and flow of the database.
* **Improved Maintainability:** The EER model reduces redundancy and complexity in the database design. By organising entities and their relationships more efficiently, it becomes easier to maintain and update the database. Changes can be made at a higher level without affecting the entire structure, thus simplifying the process of updating the database as requirements evolve.
* **Better Representation of Real-World Scenarios:** The EER model includes advanced features such as specialisation, generalisation, and aggregation. These features allow for a more precise and accurate representation of real-world data relationships. For example, a "Vehicle" superclass can have "Car" and "Truck" subclasses, each with specific attributes. This capability helps in modelling complex real-world scenarios more effectively, making the database more relevant and useful

**Entity-Relationship Diagrams**

**ER Model in DBMS**

The Entity-Relationship (ER) model is a conceptual data model used in database design to represent the entities, attributes, and relationships within a database system. In the process of database design, the ER model holds significant importance, aiding in the development of an efficient and systematically structured database schema. Let's break down the components of an ER model with the example of your college.



**Entities:**

Entities represent real-world objects or concepts about which data is stored in the database. They are typically nouns. For example: In a university database, the "Student" entity represents individual students enrolled in courses. The “Courses” offered are themselves entities of the ER Model for the university.

**Attributes:**

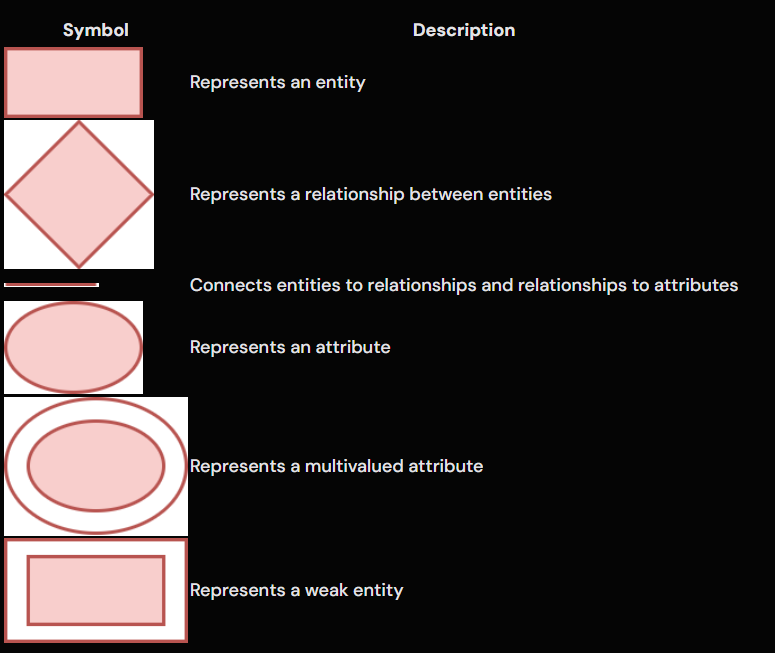
Attributes are the properties or characteristics of entities. They describe the entity. Attributes are typically adjectives or descriptors. For example: StudentID, Name, Age are attributes of the "Student" entity. Each student has a unique ID, a name, and an age.

**Relationships:**

Relationships describe how entities are related to each other. They are typically verbs or phrases that connect entities. For example: In the university database, there is a relationship between the "Student" and "Course" entities called "Enrollment." A student can be enrolled in multiple courses, and a course can have multiple students enrolled in it.

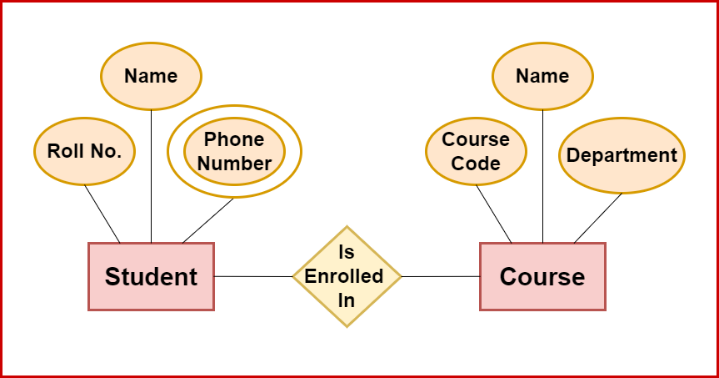
**Symbols used in ER Model:**

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

****

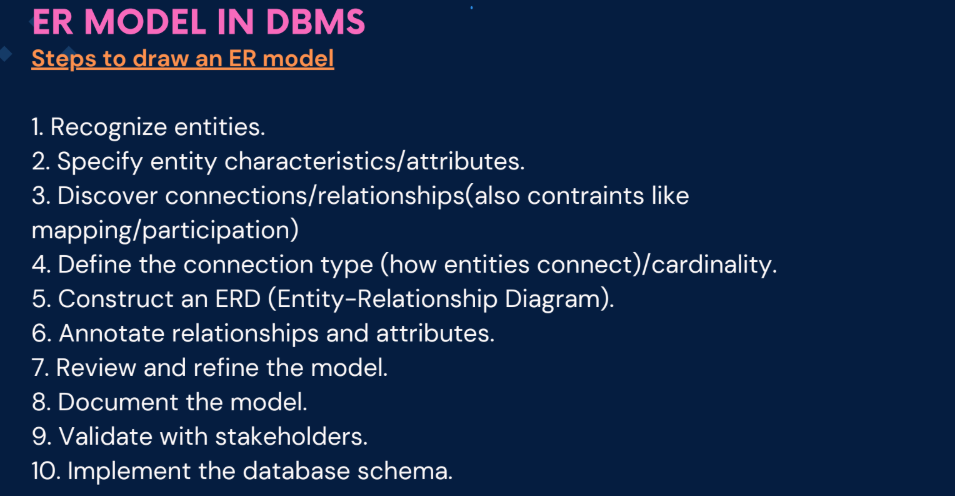
**ER Diagram for College Example:**

Now let’s create a simple ER diagram for your college with two basic entities of “Student” and “Course” using the above-listed symbols.



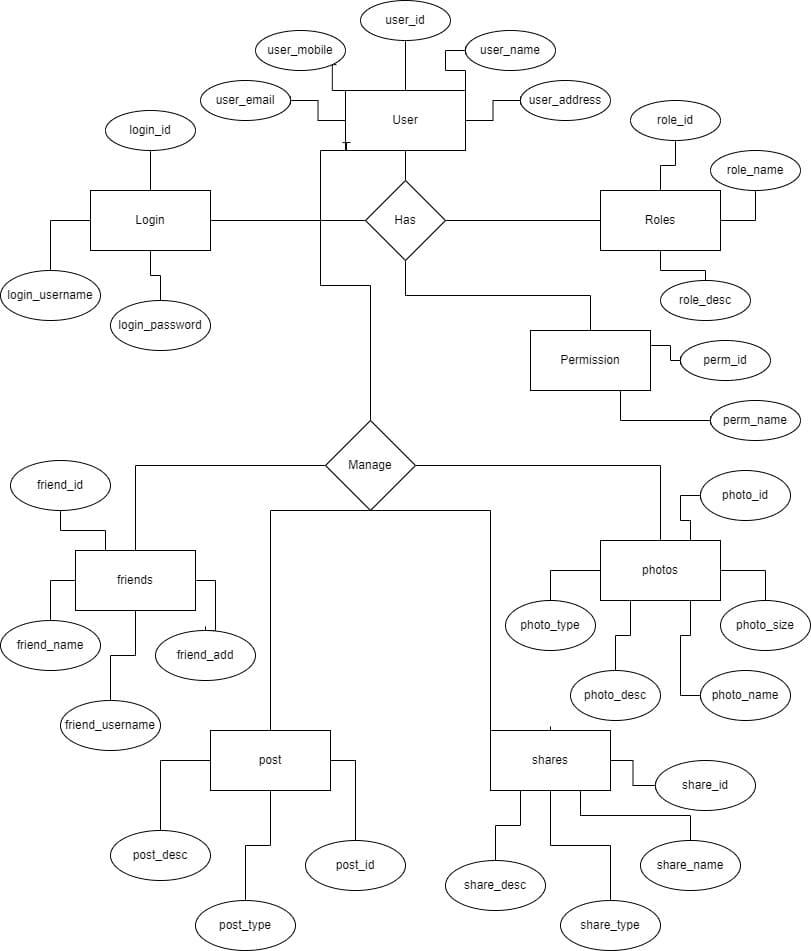
This ERD illustrates that students can enroll in several courses and a course can have many students enrolled in it

Steps to draw ER MODEL





ER model of social Media Network





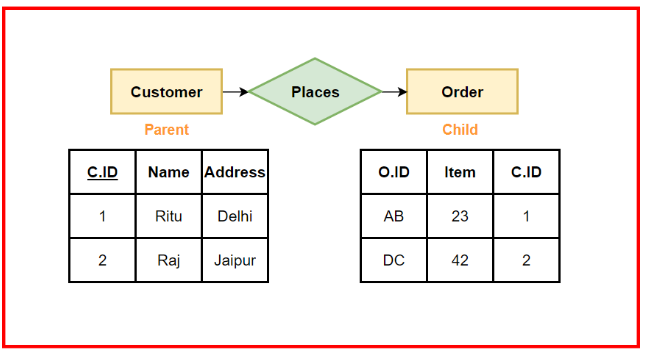
**Relationship and Degree in ER Model**

Relationship in the ER Model is the connection between entities (tables) based on related data. It is represented by the diamond in the ER model.

**Types of Relationship:**

**Strong Relationship**

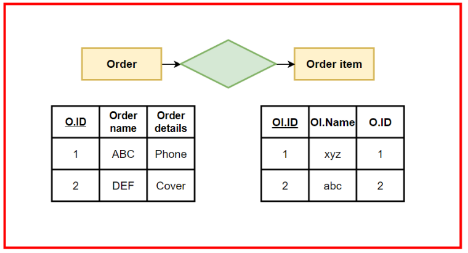
A strong relationship exists when two entities are highly dependent on each other, and one entity cannot exist without the other.



From the figure, we can see that C.ID in the “Order” table is a foreign key referencing the “Customer” table. The “Order” entity holds no value in case the “Customer” table does not exist. It will not satisfy the Referential integrity rule in an ER Model. Therefore, the figure displays a strong relationship between the “Customer” and the “Order” entity.

**Weak Relationship**

A weak relationship, on the other hand, exists when two entities are related, but one entity can exist without the other. For example, in the “Order item” table we can see that “OI.ID” is a primary key. This table also has a foreign key “O.ID” referenced from the “Order” table. Even though there is a foreign key in the “Order item” table, this table can exist alone also as it has its own primary key.



**Degree in DBMS**

A degree in DBMS refers to the number of attributes/columns that a relation/table has. For example, in the figure above, for the “Order” table we have attributes, “O.ID”, “Order name” and “Order details”, hence the degree is 3.

**Degree Types:**

|  |  |  |
| --- | --- | --- |
| **Degree** | **Name** | **Definition** |
| 1 | Unary degree | A relation with a single attribute |
| 2 | Binary degree | A relation with two attributes |
| 3 | Ternary degree | A relation with three attributes |
| 4 | N-ary degree | A relation with more than three attributes n>3 |

**Null Value**

In databases, a null value can occur for various reasons that are:

* **Not Needed Information**

Sometimes, some details are asked, but they don't apply to everyone. For instance, asking for a "Spouse Name" from someone who isn't married.

* **Don't Know the Answer**

Now and then, we’re asked a question, but we don't have an answer yet. For example, when you are filling out a quiz form in college, you might not know the answer to a question.

* **Forgot to Fill In**

When you’re filling out a form, you accidentally miss putting in some important information

**Relational model**

It is a way of organizing data in tables.

Some terms used in relational model

* Table - Relation
* Row - Tuple
* Column - Attribute
* Record - Each row in a table
* Domain - The type of value an attribute can hold
* Degree - No. of columns in a relation
* Cardinality - No of tuples

Relational model is all about:

* Data being organized into tables
* Establishing Relationships between tables using Foreign key
* Maintaining data Integrity
* A flexible and efficient way to store(SQL) and retrieve data

In relational model we take care of different things like:

1. Maintaining integrity constraints like domain, entity, referential integrity.

2. The values to be atomic i.e can’t be divided further.

3. Each row must be unique, here keys comes into picture i.e candiate, super,

primary etc

Converting ER Model to Relational Model:

