First Normal Form (1NF)

* A relation will be 1NF if it contains an atomic value.
* It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
* First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

**Example:** Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP\_PHONE.

**EMPLOYEE table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_PHONE** | **EMP\_STATE** |
| 14 | John | 7272826385, 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389, 8589830302 | Punjab |

The decomposition of the EMPLOYEE table into 1NF has been shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_PHONE** | **EMP\_STATE** |
| 14 | John | 7272826385 | UP |
| 14 | John | 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389 | Punjab |
| 12 | Sam | 8589830302 | Punjab |

Second Normal Form (2NF)

* In the 2NF, relational must be in 1NF.
* In the second normal form, all non-key attributes are fully functional dependent on the primary key

**Example:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

**TEACHER table**

|  |  |  |
| --- | --- | --- |
| **TEACHER\_ID** | **SUBJECT** | **TEACHER\_AGE** |
| 25 | Chemistry | 30 |
| 25 | Biology | 30 |
| 47 | English | 35 |
| 83 | Math | 38 |
| 83 | Computer | 38 |

In the given table, non-prime attribute TEACHER\_AGE is dependent on TEACHER\_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

**TEACHER\_DETAIL table:**

|  |  |
| --- | --- |
| **TEACHER\_ID** | **TEACHER\_AGE** |
| 25 | 30 |
| 47 | 35 |
| 83 | 38 |

**TEACHER\_SUBJECT table:**

|  |  |
| --- | --- |
| **TEACHER\_ID** | **SUBJECT** |
| 25 | Chemistry |
| 25 | Biology |
| 47 | English |
| 83 | Math |
| 83 | Computer |

Third Normal Form (3NF)

* A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
* 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
* If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| 222 | Harry | 201010 | UP | Noida |
| 333 | Stephan | 02228 | US | Boston |
| 444 | Lan | 60007 | US | Chicago |
| 555 | Katharine | 06389 | UK | Norwich |
| 666 | John | 462007 | MP | Bhopal |

1. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

**Example:**

**EMPLOYEE\_DETAIL table:**

**Super key in the table above:**

* 1. {EMP\_ID}, {EMP\_ID, EMP\_NAME}, {EMP\_ID, EMP\_NAME, EMP\_ZIP}....so on

**Candidate key:** {EMP\_ID}

**Non-prime attributes:** In the given table, all attributes except EMP\_ID are non-prime.

Here, EMP\_STATE & EMP\_CITY dependent on EMP\_ZIP and EMP\_ZIP dependent on EMP\_ID. The non-prime attributes (EMP\_STATE, EMP\_CITY) transitively dependent on super key(EMP\_ID). It violates the rule of third normal form.

That's why we need to move the EMP\_CITY and EMP\_STATE to the new <EMPLOYEE\_ZIP> table, with EMP\_ZIP as a Primary key.

**EMPLOYEE table:**

|  |  |  |
| --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** |
| 222 | Harry | 201010 |
| 333 | Stephan | 02228 |
| 444 | Lan | 60007 |
| 555 | Katharine | 06389 |
| 666 | John | 462007 |

**EMPLOYEE\_ZIP table:**

|  |  |  |
| --- | --- | --- |
| **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| 201010 | UP | Noida |
| 02228 | US | Boston |
| 60007 | US | Chicago |
| 06389 | UK | Norwich |
| 462007 | MP | Bhopal |

Fourth Normal Form (4NF)

**Fourth Normal Form (4NF)** is a level of database normalization designed to reduce redundancy and dependency within a database table. A table is in 4NF if it meets the requirements of **Third Normal Form (3NF)** and has no **multi-valued dependencies**.

**No Multi-Valued Dependencies**:

* A multi-valued dependency occurs when one attribute in a table uniquely determines another set of attributes, and those attributes are independent of each other. This can lead to data redundancy.
* In 4NF, all non-trivial multi-valued dependencies must be removed by breaking the table into smaller ones.
* By eliminating multi-valued dependencies, 4NF ensures that there is no unnecessary duplication of data.

Consider a table storing information about employees, their skills, and their certifications:

| **EmployeeID** | **Skill** | **Certification** |
| --- | --- | --- |
| E1 | Java | AWS Certified |
| E1 | Python | AWS Certified |
| E1 | Java | Azure Certified |
| E1 | Python | Azure Certified |

Here, two independent attributes (Skill and Certification) are associated with EmployeeID. This leads to redundancy since each skill gets paired with every certification.

### Steps to Convert to 4NF:

To remove the multi-valued dependency:

1. Decompose the table into two separate tables:
   * One table for EmployeeID and Skill.
   * Another table for EmployeeID and Certification.

Resulting Tables:

**Table 1: Employee Skills**

| **EmployeeID** | **Skill** |
| --- | --- |
| E1 | Java |
| E1 | Python |

**Table 2: Employee Certifications**

| **EmployeeID** | **Certification** |
| --- | --- |
| E1 | AWS Certified |
| E1 | Azure Certified |

This eliminates redundancy, ensuring the tables are now in 4NF.

Fifth Normal Form (5NF)

**Fifth Normal Form (5NF)**, also known as **Project-Join Normal Form (PJNF)**, is the highest level of normalization that ensures a database table is free from **join dependency anomalies**.

### Definition:

A table is in **5NF** if:

1. It is already in **4NF**.
2. It has no **join dependencies** that are not implied by the candidate keys.

This means the table should not contain any redundancy that can only be removed by decomposing the table into smaller tables that can be recombined (joined) without loss of data.

### Key Concept: **Join Dependency**

A **join dependency** occurs when a table can be reconstructed by joining multiple smaller tables, but those smaller tables cannot be joined correctly unless all of them are combined. If such dependencies exist and are not implied by the candidate keys, the table is not in 5NF.

### Example of a Join Dependency:

Consider a table that captures a relationship between **Consultants**, **Projects**, and **Skills**:

| **Consultant** | **Project** | **Skill** |
| --- | --- | --- |
| Alice | Website Redesign | HTML |
| Alice | Website Redesign | CSS |
| Bob | Mobile App | Swift |
| Bob | Mobile App | Kotlin |

Here:

* A consultant can work on multiple projects.
* A consultant has multiple skills.
* Each skill is required for specific projects.

This table has a complex dependency because:

1. **Consultant** relates to **Project**.
2. **Consultant** relates to **Skill**.
3. **Skill** relates to **Project**.

These dependencies are independent but cannot be split without redundancy unless done carefully.

### Steps to Convert to 5NF:

Decompose the table into three smaller tables to eliminate the join dependency:

1. **Consultants and Projects**:

| **Consultant** | **Project** |
| --- | --- |
| Alice | Website Redesign |
| Bob | Mobile App |

1. **Consultants and Skills**:

| **Consultant** | **Skill** |
| --- | --- |
| Alice | HTML |
| Alice | CSS |
| Bob | Swift |
| Bob | Kotlin |

1. **Skills and Projects**:

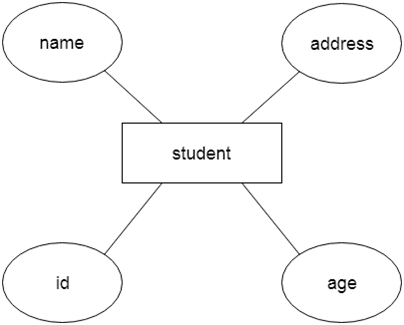
| **Skill** | **Project** |
| --- | --- |
| HTML | Website Redesign |
| CSS | Website Redesign |
| Swift | Mobile App |
| Kotlin | Mobile App |

Now, these tables can be joined to reconstruct the original table without redundancy. This decomposition ensures that the table is in 5NF.

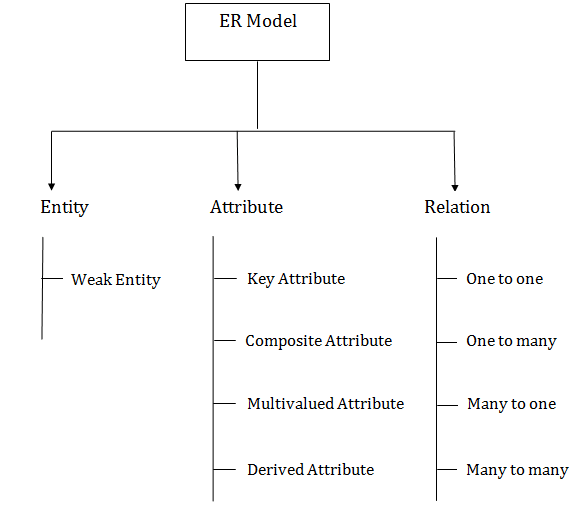
# ER model

* ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
* It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
* In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

**For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.

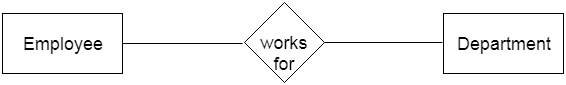
## **Component of ER Diagram**

### **1. Entity:**

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

**a. Weak Entity**

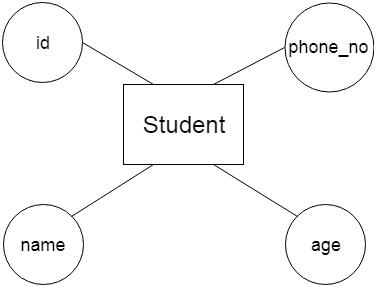
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

### **2. Attribute**

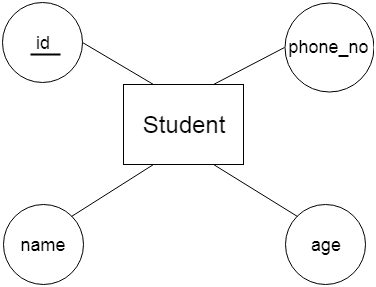
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

**For example,** id, age, contact number, name, etc. can be attributes of a student.

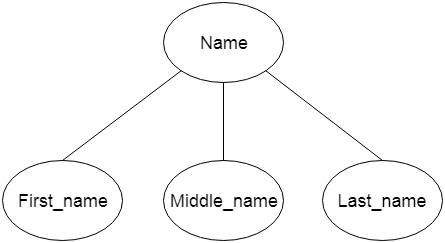
**a. Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

**b. Composite Attribute**

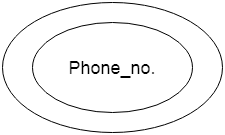
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

**c. Multivalued Attribute**

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

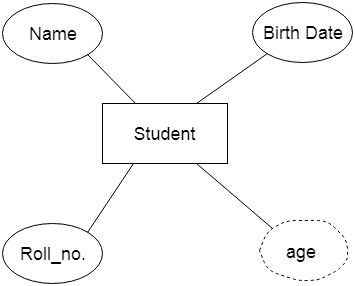
**For example,** a student can have more than one phone number.

**d. Derived Attribute**

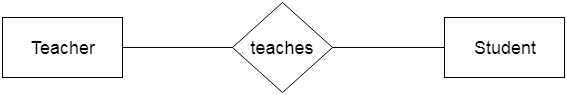
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

**For example,** A person's age changes over time and can be derived from another attribute like Date of birth.

### **3. Relationship**

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

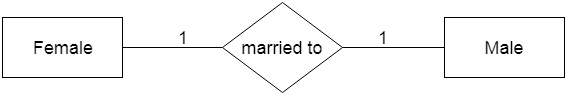
 

Types of relationship are as follows:

**a. One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

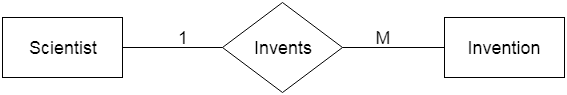
**For example,** A female can marry to one male, and a male can marry to one female.

**b. One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

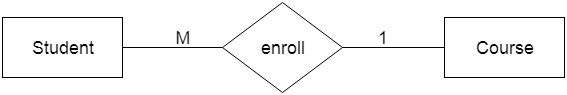
**For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.

**c. Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

**For example,** Student enrolls for only one course, but a course can have many students.

**d. Many-to-many relationship**

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

**For example,** Employee can assign by many projects and project can have many employees.

