

## Learn : ERD, Types of Attributes, Normalization with their use case

An **Entity-Relationship Diagram (ERD)** is a visual model of a database structure. It illustrates how different entities (things, objects, concepts) in a system relate to one another. ERDs are crucial for designing a relational database.

**Entities:** Represented by **rectangles** (e.g., *Student*, *Course*, *Professor*).

**Attributes:** Represented by **ovals** and linked to the entity (e.g., *Name*, *StudentID*).

**Relationships:** Represented by diamonds (or implied by connecting lines) and describe how entities interact (e.g., a *Professor* teaches a *Course*).

**Cardinality:** Defines the number of instances of one entity that can be associated with the number of instances of another entity (e.g., one-to-one, one-to-many, many-to-many).

### 2. Types of Attributes

Attribute Type	Description	Example
Simple	Cannot be divided into smaller components.	<i>Age, Salary</i>
Composite	Can be divided into smaller meaningful components.	<i>Address</i> (can be divided into <i>Street</i> , <i>City</i> , <i>ZipCode</i> )
Single-Valued	Has only one value for a particular entity instance.	<i>StudentID</i> , <i>DateOfBirth</i>
Multi-Valued	Can have multiple values for a single entity instance.	<i>Phone Number</i> (a person can have multiple)
Derived	Can be calculated or derived from other attributes.	<i>Age</i> (derived from <i>DateOfBirth</i> and the <i>Current Date</i> )
Key	Uniquely identifies an entity instance (e.g., <b>Primary Key</b> ).	<i>SSN</i> , <i>EmployeeID</i>

### 3. Normalization and Use Cases

Normalization is a systematic process of restructuring a relational database to minimize data redundancy and eliminate undesirable characteristics such as insertion, deletion, and update anomalies. The goal is to ensure data dependencies are logical and stored in appropriate tables.

It is organized into levels called Normal Forms (NFs). The most common are 1NF, 2NF, and 3NF.

#### A. First Normal Form (1NF)

- Rule: An attribute must contain only atomic (indivisible) values, and there should be no repeating groups of columns.

- Use Case: This is the basic requirement for any relation to be considered a table. It ensures that each cell in the table holds a single, unique piece of data.

## B. Second Normal Form (2NF)

Rule: Must be in 1NF, and all non-key attributes must be fully functionally dependent on the Primary Key. (This rule primarily applies to tables with composite keys).

Use Case: Eliminates partial dependency, where an attribute depends on only part of a composite primary key. This prevents redundancy and update anomalies.

## C. Third Normal Form (3NF)

Rule: Must be in 2NF, and there should be no transitive dependency. A non-key attribute cannot depend on another non-key attribute.

Use Case: Eliminates dependency between non-key attributes, which is the most common form of redundancy. This is typically the standard for most operational database designs.

## D. Boyce-Codd Normal Form (BCNF)

Rule: A stricter version of 3NF. For every non-trivial functional dependency ( $X \rightarrow Y$ ),  $X$  must be a Superkey (a set of attributes that uniquely identifies a row).

Use Case: Used in complex cases where 3NF doesn't eliminate anomalies, particularly when a table has overlapping candidate keys.

Normal Form	Problem Eliminated	Key Concept	Typical Use Case
1NF	Multi-valued and repeating attributes	Atomic Values	Foundational table structure
2NF	Partial Dependency	Full Functional Dependency	Tables with Composite Keys
3NF	Transitive Dependency	No Non-Key to Non-Key Dependency	Standard design for operational databases
BCNF	Complex overlapping dependencies	Dependencies are determinants. Superkeys	Databases with multiple candidate keys