**Normalization in DBMS**

**Definition:**  
Normalization is a process in Database Management Systems (DBMS) used to organize data efficiently by eliminating redundancy and ensuring data integrity. It involves dividing a database into smaller, related tables and applying rules to minimize data duplication.

**Types of Normal Forms with Examples**

**Unnormalized Form (UNF)**

A table that contains redundant data and does not follow any normalization rules.

**Example (UNF Table)**

| **Student\_ID** | **Student\_Name** | **Subjects** |
| --- | --- | --- |
| 1 | John | Math, Science |
| 2 | Alice | Math, English |
| 3 | Bob | Science |

* The "Subjects" column contains multiple values (violating atomicity).

**First Normal Form (1NF)**

A table is in **1NF** if:  
✅ Each column contains atomic (indivisible) values.  
✅ Each row has a unique identifier (Primary Key).

**1NF Table**

| **Student\_ID** | **Student\_Name** | **Subject** |
| --- | --- | --- |
| 1 | John | Math |
| 1 | John | Science |
| 2 | Alice | Math |
| 2 | Alice | English |
| 3 | Bob | Science |

* The "Subjects" column is now split into individual rows (Atomicity achieved).
* Still contains redundancy (student names are repeated).

**Second Normal Form (2NF)**

A table is in **2NF** if:  
✅ It is in **1NF**.  
✅ All non-key attributes are fully dependent on the primary key.

**1NF Issues**

* In the previous table, **Student\_Name** depends only on **Student\_ID** and not on **Student\_ID + Subject** (partial dependency).

**2NF Table (Dividing into Two Tables)**

**Student Table:**

| **Student\_ID** | **Student\_Name** |
| --- | --- |
| 1 | John |
| 2 | Alice |
| 3 | Bob |

**Enrollment Table:**

| **Student\_ID** | **Subject** |
| --- | --- |
| 1 | Math |
| 1 | Science |
| 2 | Math |
| 2 | English |
| 3 | Science |

* Now, **Student\_Name** is in a separate table, fully dependent on **Student\_ID**.
* Eliminated **partial dependency**.

**Third Normal Form (3NF)**

A table is in **3NF** if:  
✅ It is in **2NF**.  
✅ It has **no transitive dependency** (non-key attributes must not depend on other non-key attributes).

**2NF Issues**

* Suppose we add a column **Department** to the Student table.
* If **Department** depends on **Student\_Name**, we have a transitive dependency.

**3NF Table (Further Dividing)**

**Student Table:**

| **Student\_ID** | **Student\_Name** |
| --- | --- |
| 1 | John |
| 2 | Alice |
| 3 | Bob |

**Department Table:**

| **Student\_ID** | **Department** |
| --- | --- |
| 1 | Science |
| 2 | Arts |
| 3 | Science |

* Now, **Department** depends only on **Student\_ID**, not on **Student\_Name**.
* **Transitive dependency removed**.

**Boyce-Codd Normal Form (BCNF)**

A table is in **BCNF** if:  
✅ It is in **3NF**.  
✅ Every determinant is a candidate key (No anomalies in functional dependencies).

* Used when a table has **more than one candidate key**.
* Not required in many cases unless there are **complex dependencies**.

**Conclusion**

Normalization **reduces redundancy**, **improves efficiency**, and **ensures consistency** in databases.  
Most databases follow up to **3NF**, while **BCNF** is used in special cases.

Would you like an example of **BCNF** as well? 😊