**Normalization in DBMS**

**🎯 Definition**

**Normalization** is the process of organizing data in a database to:

* Eliminate **data redundancy** (duplicate data),
* Ensure **data integrity**, and
* Simplify **data maintenance**.

It divides large, complex tables into smaller, related tables and links them using **foreign keys**.

**💡 Why Normalize a Database?**

| **Reason** | **Description** |
| --- | --- |
| **Avoid Redundancy** | Prevents storing the same data multiple times. |
| **Ensure Consistency** | Updates to data happen in one place only. |
| **Save Space** | Removes duplicate and unnecessary data. |
| **Ease of Maintenance** | Simplifies data modification and insertion. |
| **Improved Query Efficiency** | Reduces anomalies and increases performance. |

**⚠️ Types of Data Anomalies Before Normalization**

If a database is not normalized, three types of anomalies can occur:

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| **Insertion Anomaly** | Cannot insert data due to missing attributes. | Can’t add a new student unless they’re enrolled in a course. |
| **Update Anomaly** | Updating data in one place but forgetting others. | Changing a teacher’s name in one row but not others. |
| **Deletion Anomaly** | Deleting one record removes important data. | Removing a course also deletes teacher info. |

**🧠 Normalization Forms**

Normalization is performed in **stages (Normal Forms)**. Each form builds on the previous one.

**1️⃣ First Normal Form (1NF)**

A table is in **1NF** if:

* All columns contain **atomic values** (no repeating groups or arrays).
* Each record is unique.

**Example (Before 1NF):**

| **StudentID** | **Name** | **Courses** |
| --- | --- | --- |
| 1 | Arjun | Math, Science |
| 2 | Meena | English |

**After 1NF:**

| **StudentID** | **Name** | **Course** |
| --- | --- | --- |
| 1 | Arjun | Math |
| 1 | Arjun | Science |
| 2 | Meena | English |

**2️⃣ Second Normal Form (2NF)**

A table is in **2NF** if:

1. It is already in **1NF**.
2. There are **no partial dependencies** — meaning, no non-key column depends on a part of a composite key.

**Example:**

| **StudentID** | **CourseID** | **StudentName** | **CourseName** |
| --- | --- | --- | --- |
| 1 | 101 | Arjun | Math |
| 1 | 102 | Arjun | Science |

**Issue:** StudentName depends only on StudentID, not on the full key (StudentID, CourseID).

**After 2NF:**

**Student Table:**

| **StudentID** | **StudentName** |
| --- | --- |
| 1 | Arjun |

**Course Table:**

| **CourseID** | **CourseName** |
| --- | --- |
| 101 | Math |
| 102 | Science |

**Enrollment Table:**

| **StudentID** | **CourseID** |
| --- | --- |
| 1 | 101 |
| 1 | 102 |

**3️⃣ Third Normal Form (3NF)**

A table is in **3NF** if:

1. It is in **2NF**, and
2. There are **no transitive dependencies** (no non-key attribute depends on another non-key attribute).

**Example:**

| **StudentID** | **StudentName** | **DeptID** | **DeptName** |
| --- | --- | --- | --- |
| 1 | Arjun | D01 | Science |

**Issue:** DeptName depends on DeptID, not directly on StudentID.

**After 3NF:**

**Student Table:**

| **StudentID** | **StudentName** | **DeptID** |
| --- | --- | --- |
| 1 | Arjun | D01 |

**Department Table:**

| **DeptID** | **DeptName** |
| --- | --- |
| D01 | Science |

**4️⃣ Boyce–Codd Normal Form (BCNF)**

A table is in **BCNF** if:

* For every **functional dependency (X → Y)**, X should be a **super key**.

BCNF is a **stronger version of 3NF**.

**Example:**

| **Professor** | **Subject** | **Department** |
| --- | --- | --- |
| John | DBMS | CS |
| Mary | OS | CS |
| John | Java | IT |

Here,

* Each **Professor** teaches only one **Department**,
* But a **Department** can have multiple **Subjects**.

So, Professor → Department is a dependency, but Professor is **not a superkey**.

John

D1(S1,S2)----john D2 (SS1,SS2,)--- D3(SSS1,SSS2)

**Solution (BCNF):**  
Split the table:

**Table 1:** Professor → Department

| **Professor** | **Department** |
| --- | --- |
| John | CS |
| Mary | CS |

**Table 2:** Department → Subject

| **Department** | **Subject** |
| --- | --- |
| CS | DBMS |
| CS | OS |
| IT | Java |

**5️⃣ Fourth Normal Form (4NF)**

* It removes **multi-valued dependencies**.
* A table is in **4NF** if it is in **BCNF** and has **no multi-valued dependencies**.

**Example:**  
If a student can have multiple skills and hobbies, those should be stored in separate tables, not in the same one.

**6️⃣ Fifth Normal Form (5NF)**

* Deals with **join dependencies**.
* Ensures that no data can be reconstructed incorrectly by joining tables.

This is rarely used in practical systems but is important in **data warehousing and complex relational design**.

**🏁 Summary Table**

| **Normal Form** | **Main Rule** | **Removes** |
| --- | --- | --- |
| **1NF** | Atomic values only | Repeating groups |
| **2NF** | No partial dependency | Partial dependency |
| **3NF** | No transitive dependency | Transitive dependency |
| **BCNF** | Every determinant is a candidate key | Anomalies due to non-superkey dependencies |
| **4NF** | No multi-valued dependency | Multi-valued dependency |
| **5NF** | No join dependency | Complex join dependency |