**Database Normalization**

Database normalization is the process of organizing data in a relational database to minimize redundancy and improve data integrity. The goal is to divide large tables into smaller ones and define relationships between them, following specific normal forms.

**Why Normalize?**

1. Avoid data redundancy.
2. Ensure data consistency.
3. Make the database easier to maintain and update.
4. Improve query performance.

**Normal Forms (NF)**

Normalization is categorized into several normal forms, commonly up to **Third Normal Form (3NF)**.

**1. First Normal Form (1NF)**

* Each column should contain atomic (indivisible) values.
* Each row must be unique.

**Example (Unnormalized Table):**

| **StudentID** | **Name** | **Subjects** |
| --- | --- | --- |
| 1 | John Smith | Math, Science |
| 2 | Jane Doe | History, Geography |

**Normalized (1NF):**

| **StudentID** | **Name** | **Subject** |
| --- | --- | --- |
| 1 | John Smith | Math |
| 1 | John Smith | Science |
| 2 | Jane Doe | History |
| 2 | Jane Doe | Geography |

**2. Second Normal Form (2NF)**

* Satisfies 1NF.
* All non-key attributes must depend on the **whole primary key** (no partial dependency).

**Example (1NF Table):**

| **StudentID** | **Subject** | **Teacher** |
| --- | --- | --- |
| 1 | Math | Mr. Brown |
| 1 | Science | Dr. Green |
| 2 | History | Ms. White |
| 2 | Geography | Mr. Gray |

**Issue:** If "Teacher" depends only on "Subject" and not the entire key (StudentID + Subject), it's a partial dependency.

**Normalized (2NF):**

1. **Students Table:**

| **StudentID** | **Name** |
| --- | --- |
| 1 | John Smith |
| 2 | Jane Doe |

1. **Subjects Table:**

| **Subject** | **Teacher** |
| --- | --- |
| Math | Mr. Brown |
| Science | Dr. Green |
| History | Ms. White |
| Geography | Mr. Gray |

1. **Enrollment Table:**

| **StudentID** | **Subject** |
| --- | --- |
| 1 | Math |
| 1 | Science |
| 2 | History |
| 2 | Geography |

**3. Third Normal Form (3NF)**

* Satisfies 2NF.
* No **transitive dependency** (non-key attributes depend on other non-key attributes).

**Example (2NF Table):**

| **Subject** | **Teacher** | **TeacherPhone** |
| --- | --- | --- |
| Math | Mr. Brown | 1234567890 |
| Science | Dr. Green | 9876543210 |

**Issue:** "TeacherPhone" depends on "Teacher," not the key "Subject."

**Normalized (3NF):**

1. **Subjects:**

| **Subject** | **Teacher** |
| --- | --- |
| Math | Mr. Brown |
| Science | Dr. Green |

1. **Teachers Table:**

| **Teacher** | **TeacherPhone** |
| --- | --- |
| Mr. Brown | 1234567890 |
| Dr. Green | 9876543210 |

**Benefits of Normalization**

1. Reduces redundancy and anomalies.
2. Makes updates, inserts, and deletions efficient.
3. Simplifies queries by eliminating repetitive data.

**Drawbacks**

* Complex joins might affect performance.
* Over-normalization can make data retrieval cumbersome.

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

Boyce-Codd Normal Form is an advanced version of the Third Normal Form (3NF). A table is in BCNF if it satisfies the following conditions:

1. **It is in 3NF.**
2. **Every determinant is a candidate key.**

A **determinant** is an attribute (or a set of attributes) that uniquely determines another attribute.

**Why BCNF?**

* BCNF addresses certain anomalies that 3NF cannot resolve, especially when a table has more than one candidate key, and the keys overlap.

**Example:**

**3NF Table (but not BCNF):**

| **StudentID** | **Course** | **Teacher** |
| --- | --- | --- |
| 1 | Mathematics | Mr. Brown |
| 2 | Science | Dr. Green |
| 1 | Science | Dr. Green |

**Candidate Keys:**

* (StudentID, Course)
* (Course, Teacher)

**Issue:**

* Course determines Teacher (Course → Teacher).
* Course is not a candidate key. Hence, the table violates BCNF.

**BCNF Solution:**

Decompose the table into two tables to eliminate the dependency:

1. **Courses Table:**

| **Course** | **Teacher** |
| --- | --- |
| Mathematics | Mr. Brown |
| Science | Dr. Green |

1. **Enrollment Table:**

| **StudentID** | **Course** |
| --- | --- |
| 1 | Mathematics |
| 2 | Science |
| 1 | Science |

**Steps to Achieve BCNF**

1. Identify all candidate keys.
2. Identify dependencies where the determinant is not a candidate key.
3. Decompose the table to ensure every determinant is a candidate key.

**Benefits of BCNF**

1. Eliminates redundancy and anomalies more rigorously than 3NF.
2. Ensures every functional dependency is logical and valid.

**Drawbacks**

* More decomposition can lead to complex joins during query execution.
* May not always be practical for performance-sensitive systems.