**Normalization in Database Design**

**Normalization** is the process of organizing the attributes (columns) and tables (relations) of a database to minimize redundancy and dependency. The goal is to ensure that the database is free from anomalies like **insertion**, **update**, and **deletion anomalies** while making it more efficient to query.

Normalization involves dividing large tables into smaller ones and defining relationships between them. This process is done using **normal forms**. Each normal form represents a level of organization that eliminates specific types of redundancy and dependency.

There are several **normal forms** (NF), each with its own set of rules that progressively refine the structure of a database. The commonly used normal forms are:

1. **First Normal Form (1NF)**
2. **Second Normal Form (2NF)**
3. **Third Normal Form (3NF)**
4. **Boyce-Codd Normal Form (BCNF)**
5. **Fourth Normal Form (4NF)**
6. **Fifth Normal Form (5NF)**

We have a table that stores information about students, the courses they are enrolled in, the instructors teaching those courses, and the rooms assigned for each course. The data is improperly structured, leading to issues such as **redundancy**, **update anomalies**, **insertion anomalies**, and **deletion anomalies**.

**Initial Unnormalized Table (with transitive and multi-valued dependencies)**

| **StudentID** | **StudentName** | **Courses** | **InstructorName** | **InstructorPhone** | **Room** |
| --- | --- | --- | --- | --- | --- |
| 1 | John | 101: Database Systems, 102: Data Structures | Dr. Smith, Dr. Brown | 123-456-7890, 234-567-8901 | Room 101, Room 102 |
| 2 | Sarah | 101: Database Systems, 103: Networking | Dr. Smith, Dr. Lee | 123-456-7890, 345-678-9012 | Room 101, Room 103 |

**Identifying the Problems:**

1. **Multi-valued dependencies**:
   * The **Courses** column stores multiple values for each student (multiple course IDs).
   * The **InstructorName**, **InstructorPhone**, and **Room** columns also store multiple values because students might be enrolled in multiple courses. This violates the **atomicity** rule of databases, and it causes data redundancy.
2. **Transitive dependencies**:
   * **InstructorPhone** depends on **InstructorName** and **InstructorName** depends on **CourseID** (through the **Instructor**). This results in a **transitive dependency**: **InstructorPhone → InstructorName → CourseID**.
   * So, **InstructorPhone** is indirectly dependent on **CourseID**, violating **3NF**.
3. **Redundancy**:
   * For each student enrolled in multiple courses, the student’s **StudentName**, **InstructorName**, **InstructorPhone**, and **Room** details are repeated, leading to redundancy and the risk of inconsistencies if any information needs updating.

**Step 1: First Normal Form (1NF)**

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

* All columns contain atomic values (no multi-valued attributes).
* Each record is unique.

In our unnormalized table, the **Courses**, **InstructorName**, **InstructorPhone**, and **Room** columns contain multi-valued attributes (e.g., multiple courses in one cell).

**To achieve 1NF:**

* We need to split the multi-valued columns into separate rows for each course. This removes multi-valued attributes and ensures atomicity.

**Table After 1NF:**

| **StudentID** | **StudentName** | **CourseID** | **CourseName** | **InstructorName** | **InstructorPhone** | **Room** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | John | 101 | Database Systems | Dr. Smith | 123-456-7890 | Room 101 |
| 1 | John | 102 | Data Structures | Dr. Brown | 234-567-8901 | Room 102 |
| 2 | Sarah | 101 | Database Systems | Dr. Smith | 123-456-7890 | Room 101 |
| 2 | Sarah | 103 | Networking | Dr. Lee | 345-678-9012 | Room 103 |

**Explanation:**

* The **Courses**, **InstructorName**, **InstructorPhone**, and **Room** values have been broken down into separate rows.
* The table is now in **1NF** because each field contains atomic values and each record is unique.

**Step 2: Second Normal Form (2NF)**

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

* It is in **1NF**.
* All non-key attributes are **fully functionally dependent** on the **entire primary key**.

In our current table, the **primary key** is a composite key: (StudentID, CourseID).

* **StudentName** depends on **StudentID** alone (not the whole primary key), which is a **partial dependency**.
* This violates **2NF**.

**To achieve 2NF:**

* We need to remove the partial dependency by creating separate tables for **Students**, **Courses**, and **Enrollments**.

**Tables After 2NF:**

1. **Students Table:**

| **StudentID** | **StudentName** |
| --- | --- |
| 1 | John |
| 2 | Sarah |

1. **Courses Table:**

| **CourseID** | **CourseName** |
| --- | --- |
| 101 | Database Systems |
| 102 | Data Structures |
| 103 | Networking |

1. **Enrollments Table** (Associates students with courses):

| **StudentID** | **CourseID** |
| --- | --- |
| 1 | 101 |
| 1 | 102 |
| 2 | 101 |
| 2 | 103 |

**Explanation:**

* The **StudentName** is moved to the **Students Table**, and now each non-key attribute in the **Enrollments Table** is fully dependent on the whole primary key (StudentID, CourseID).
* The table is now in **2NF** because all non-key attributes are fully functionally dependent on the whole primary key.

**Step 3: Third Normal Form (3NF)**

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

* It is in **2NF**.
* There are **no transitive dependencies** (non-key attributes should not depend on other non-key attributes).

In the **Courses Table**, **InstructorName** and **InstructorPhone** depend on the **CourseID**, but **InstructorName** can be considered a **non-prime attribute**. This creates a **transitive dependency**:  
**InstructorPhone** → **InstructorName** → **CourseID**.

**To achieve 3NF:**

* We need to remove the transitive dependency by creating a separate table for **Instructors** and link it to the **Courses Table**.

**Tables After 3NF:**

1. **Students Table:**

| **StudentID** | **StudentName** |
| --- | --- |
| 1 | John |
| 2 | Sarah |

1. **Courses Table:**

| **CourseID** | **CourseName** | **InstructorID** |
| --- | --- | --- |
| 101 | Database Systems | 1 |
| 102 | Data Structures | 2 |
| 103 | Networking | 3 |

1. **Instructors Table:**

| **InstructorID** | **InstructorName** | **InstructorPhone** |
| --- | --- | --- |
| 1 | Dr. Smith | 123-456-7890 |
| 2 | Dr. Brown | 234-567-8901 |
| 3 | Dr. Lee | 345-678-9012 |

1. **Enrollments Table:**

| **StudentID** | **CourseID** |
| --- | --- |
| 1 | 101 |
| 1 | 102 |
| 2 | 101 |
| 2 | 103 |

**Explanation:**

* We moved **InstructorName** and **InstructorPhone** to the **Instructors Table**, where they are now related to **InstructorID**.
* The table is now in **3NF** because there are no transitive dependencies.

**Step 4: Boyce-Codd Normal Form (BCNF)**

A table is in **BCNF** if:

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

In the **Instructors Table**, **InstructorName** determines **InstructorPhone**, but **InstructorName** is not a candidate key (only **InstructorID** is a candidate key). This violates **BCNF**.

**To achieve BCNF:**

* We create a new **InstructorPhone Table** to store the phone numbers, removing the non-candidate key dependency.

**Tables After BCNF:**

1. **Students Table:**

| **StudentID** | **StudentName** |
| --- | --- |
| 1 | John |
| 2 | Sarah |

1. **Courses Table:**

| **CourseID** | **CourseName** | **InstructorID** |
| --- | --- | --- |
| 101 | Database Systems | 1 |
| 102 | Data Structures | 2 |
| 103 | Networking | 3 |

1. **Instructors Table:**

| **InstructorID** | **InstructorName** | **Room** |
| --- | --- | --- |
| 1 | Dr. Smith | Room 101 |
| 2 | Dr. Brown | Room 102 |
| 3 | Dr. Lee | Room 103 |

1. **InstructorPhone Table:**

| **InstructorID** | **InstructorPhone** |
| --- | --- |
| 1 | 123-456-7890 |
| 2 | 234-567-8901 |
| 3 | 345-678-9012 |

1. **Enrollments Table:**

| **StudentID** | **CourseID** |
| --- | --- |
| 1 | 101 |
| 1 | 102 |
| 2 | 101 |
| 2 | 103 |

**Explanation:**

* We moved **InstructorPhone** to a separate **InstructorPhone Table** to ensure that every determinant is a candidate key.
* The structure is now in **BCNF**.

**Step 5: Fourth Normal Form (4NF)**

A table is in **4NF** if:

* It is in **BCNF**.
* It has no **multi-valued dependencies**.

Currently, there are no multi-valued dependencies in the schema, so it already complies with **4NF**.

A **multi-valued dependency** (MVD) occurs when a **single attribute** determines multiple independent attributes, and the determination applies to every combination of values in those independent attributes. In simpler terms, when one attribute in a table can independently determine two or more attributes, and the values of those attributes are independent of each other, a multi-valued dependency exists.

Consider a table that stores data about **students**, their **skills**, and their **hobbies**:

| **StudentID** | **StudentName** | **Skills** | **Hobbies** |
| --- | --- | --- | --- |
| 1 | John | Java, Python | Reading, Gaming |
| 2 | Sarah | C++, Java | Drawing, Reading |

* In this case:
  + **StudentID** determines both **Skills** and **Hobbies**.
  + **Skills** and **Hobbies** are independent of each other.
  + For **StudentID = 1**, John has the skills **Java, Python**, and the hobbies **Reading, Gaming**.
  + For **StudentID = 2**, Sarah has the skills **C++, Java**, and the hobbies **Drawing, Reading**.

This creates **multi-valued dependencies** where:

* **StudentID →→ Skills**
* **StudentID →→ Hobbies**

**Explanation**:

* For each student (i.e., **StudentID**), there are multiple skills and multiple hobbies, and each combination of skills and hobbies must appear independently for every student. But the set of skills is independent of the set of hobbies, which makes this a **multi-valued dependency**.

**How to Resolve Multi-Valued Dependencies (Achieve 4NF):**

To resolve multi-valued dependencies and bring the table into **4NF**, we **decompose** the table into two or more tables so that each table no longer has multiple independent sets of values that depend on a single determinant.

For example, in our table, we can break it down into two separate tables:

**1. Skills Table:**

| **StudentID** | **Skills** |
| --- | --- |
| 1 | Java |
| 1 | Python |
| 2 | C++ |
| 2 | Java |

**2. Hobbies Table:**

| **StudentID** | **Hobbies** |
| --- | --- |
| 1 | Reading |
| 1 | Gaming |
| 2 | Drawing |
| 2 | Reading |

**Explanation:**

* Now, the **Skills Table** and **Hobbies Table** each contain independent sets of values for **StudentID**, and they no longer contain multi-valued dependencies.
* This satisfies **4NF** because there are no multi-valued dependencies left in either table.

**Step 6: Fifth Normal Form (5NF)**

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

* It is in **4NF**.
* It cannot be decomposed further without losing information.

At this stage, the design cannot be decomposed further without breaking meaningful relationships between the data, so the schema is already in **5NF**.

**Final Normalized Schema (After 5NF)**

1. **Students Table:**

| **StudentID** | **StudentName** |
| --- | --- |
| 1 | John |
| 2 | Sarah |

1. **Courses Table:**

| **CourseID** | **CourseName** | **InstructorID** |
| --- | --- | --- |
| 101 | Database Systems | 1 |
| 102 | Data Structures | 2 |
| 103 | Networking | 3 |

1. **Instructors Table:**

| **InstructorID** | **InstructorName** | **Room** |
| --- | --- | --- |
| 1 | Dr. Smith | Room 101 |
| 2 | Dr. Brown | Room 102 |
| 3 | Dr. Lee | Room 103 |

1. **InstructorPhone Table:**

| **InstructorID** | **InstructorPhone** |
| --- | --- |
| 1 | 123-456-7890 |
| 2 | 234-567-8901 |
| 3 | 345-678-9012 |

1. **Enrollments Table:**

| **StudentID** | **CourseID** |
| --- | --- |
| 1 | 101 |
| 1 | 102 |
| 2 | 101 |
| 2 | 103 |