**Normalization**

1. **What is Normalization in Database.**

Normalization in a database is a process used to organize the structure of data to reduce redundancy and improve data integrity. The main goals of normalization are to eliminate duplicate data and ensure data dependencies are logical and efficient.

1. **When to Normalize the Database.**

**To Eliminate Redundancy**: When you notice that the same piece of data is being repeated multiple times across the database.

**To Ensure Data Integrity**: When it's crucial that updates to data are propagated correctly and consistently throughout the database.

**To Simplify Data Maintenance**: When you want to make your database easier to maintain and avoid complications during insert, update, or delete operations.

**When Designing a New Database**: It’s a best practice to start with a normalized design to ensure a solid foundation.

**When Restructuring or Migrating Data**: If you’re working on a project to clean up or migrate data, normalization helps to set a strong, clean structure.

**Types of Normalization forms.**

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

Ex – Unnormalized Table

|  |  |  |  |
| --- | --- | --- | --- |
| Student ID | Student Name | Courses | Teacher |
| 1 | Tejendra | History, Geography | Mr. Harshal, Dr. Pranay |
| 2 | Ashish | Chemistry, Physics | Dr. isha, Dr. Vekant |
| 3 | Aditya | Maths, Biology | Ms. Akanksha, Mr. Kartik |

1. **First Normal Form (1NF)**:
   * Ensures that the data is stored in tables with rows and columns.
   * Each column must contain atomic (indivisible) values.
   * Each column must contain values of the same type.

Ex - To achieve 1NF, we need to ensure that each column contains atomic values. Here, we'll split the repeating groups into separate rows:

|  |  |  |  |
| --- | --- | --- | --- |
| Student ID | Student Name | Course | Teacher |
| 1 | Tejendra | History | Mr. Harshal |
| 1 | Tejendra | Geography | Dr. Pranay |
| 2 | Ashish | Chemistry | Dr. Isha |
| 2 | Ashish | Physics | Dr. Vekant |
| 3 | Aditya | Maths | Ms. Akanksha |
| 3 | Aditya | Biology | Mr. Kartik |

1. **Second Normal Form (2NF)**:
   * Builds on 1NF by ensuring that all non-key attributes are fully dependent on the primary key.
   * Requires that the table be in 1NF.
   * Eliminates partial dependency, where a non-key attribute is dependent on only part of the primary key.

Ex – To achieve 2NF, we need to ensure that all non-key attributes are fully dependent on the primary key. In our case, Student Name is not dependent on both StudentID and Course. So, we'll split the table into two

**Student Table :**

|  |  |
| --- | --- |
| Student ID | Student Name |
| 1 | Tejendra |
| 2 | Ashish |
| 3 | Aditya |

**Enrollments Table :**

|  |  |  |
| --- | --- | --- |
| Student ID | Course | Teacher |
| 1 | History | Mr. Harshal |
| 1 | Geography | Dr. Pranay |
| 2 | Chemistry | Dr. Isha |
| 2 | Physics | Dr. Vekant |
| 3 | Maths | Ms. Akanksha |
| 3 | Biology | Mr. Kartik |

1. **Third Normal Form (3NF)**:
   * Ensures that all the attributes are only dependent on the primary key and not on any other non-prime attribute.
   * Requires that the table be in 2NF.
   * Eliminates transitive dependency, where a non-key attribute is dependent on another non-key attribute.

**Ex** – To achieve 3NF, we need to ensure that all the attributes are only dependent on the primary key. Here, the Teacher attribute is not dependent on the StudentID and Course combination, so we'll split the table further:

**Students Table :**

|  |  |
| --- | --- |
| Student ID | Student Name |
| 1 | Tejendra |
| 2 | Ashish |
| 3 | Aditya |

**Courses Table :**

|  |  |
| --- | --- |
| Course | Teacher |
| History | Mr. Harshal |
| Geography | Dr. Pranay |
| Chemistry | Dr. Isha |
| Physics | Dr. Vekant |
| Maths | Ms. Akanksha |

|  |  |
| --- | --- |
| Student ID | Course |
| 1 | History |
| 1 | Geography |
| 2 | Chemistry |
| 2 | Physics |
| 3 | Maths |
| 3 | Biology |

**Enrollments Table :**

1. **Boyce-Codd Normal Form (BCNF)**:
   * A stricter version of 3NF.
   * Requires that the table be in 3NF.
   * Ensures that every determinant is a candidate key, which means there are no anomalies that 3NF does not cover.

Ex - Consider a table Student Course

|  |  |  |
| --- | --- | --- |
| Student ID | Course | Teacher |
| 1 | History | Mr. Harshal |
| 2 | Chemistry | Dr. Isha |
| 1 | Geography | Dr. Vekant |
| 2 | History | Mr. Harshal |

* Student ID, Course -> Teacher (A student, when taking a particular course, has a specific Teacher)
* Teacher -> Course (Each Teacher teaches only one course)

To achieve BCNF, the table needs to be decomposed: **Teachers Table**:

|  |  |
| --- | --- |
| Teacher | Course |
| Mr. Harshal | History |
| Dr. Isha | Chemistry |
| Dr. Vekant | Geography |

Student Course Table :

|  |  |
| --- | --- |
| Student ID | Course |
| 1 | History |
| 2 | Chemistry |
| 1 | Geography |
| 2 | History |

1. **Fourth Normal Form (4NF)**:
   * Ensures no multi-valued dependencies, where one attribute can have multiple values independent of other attributes.
   * Requires that the table be in BCNF.
   * Helps to eliminate independent multivalued facts stored in one table.

Ex - Consider a table Employee Skills

|  |  |  |
| --- | --- | --- |
| Employee ID | Skill | Certification |
| 1 | Java | Oracle Certified |
| 1 | Python | Microsoft Certified |
| 2 | Java | Oracle Certified |
| 2 | SQL | Microsoft Certified |

Employee ID has multi-valued dependencies on Skill and Certification.

To achieve 4NF, decompose the table:

**Employee Skills Table**:

|  |  |
| --- | --- |
| Employee ID | Skill |
| 1 | Java |
| 1 | Python |
| 2 | Java |
| 2 | SQL |

**Employee Certifications Table**:

|  |  |
| --- | --- |
| Employee ID | Certification |
| 1 | Oracle Certified |
| 1 | Microsoft Certified |
| 2 | Oracle Certified |
| 2 | Microsoft Certified |

1. **Fifth Normal Form (5NF)**:
   * Ensures that every join dependency in the table is implied by the candidate keys.
   * Requires that the table be in 4NF.
   * Helps to decompose tables into smaller tables without losing information or causing redundancy.

Ex –

Consider a table Project Assignment

|  |  |  |
| --- | --- | --- |
| Employee ID | Project ID | Role |
| 1 | 101 | Developer |
| 1 | 102 | Designer |
| 2 | 101 | Tester |
| 2 | 102 | Developer |

To achieve 5NF, the table needs to be decomposed:

**Employee Project Table**:

|  |  |
| --- | --- |
| Employee ID | Project ID |
| 1 | 101 |
| 1 | 102 |
| 2 | 101 |
| 2 | 102 |

**Project Role Table**

|  |  |
| --- | --- |
| Project ID | Role |
| 101 | Developer |
| 102 | Designer |
| 101 | Tester |
| 102 | Developer |

**Employee Role Table**

|  |  |
| --- | --- |
| Employee ID | Role |
| 1 | Developer |
| 1 | Designer |
| 2 | Tester |
| 2 | Developer |