**Design normal forms with steps.**

**Normalization**

**Aim**

Objectives

**Steps of Normalization**

**Design of Normalization with example**

**Normalization**

**Aim**

To reduce data redundancy and improve data integrity in a relational database by organizing fields and table structures following a series of well-defined rules or forms.

**Objectives**

* To eliminate redundant (repetitive) data.
* To ensure data dependencies make sense (only storing related data in a table).
* To achieve data consistency and reduce anomalies during insert, update, or delete operations.
* To break down complex tables into simpler, smaller, and more manageable ones without losing data.
* To facilitate easier and more efficient database maintenance.

**Steps of Normalization**

Normalization is done through successive stages called **normal forms**.

The commonly used normal forms are:

1. **First Normal Form (1NF):**
   * Ensures each column contains only atomic (indivisible) values.
   * Each record should be unique.
2. **Second Normal Form (2NF):**
   * Must be in 1NF.
   * All non-key attributes must be fully functionally dependent on the **entire** primary key (no partial dependency).
3. **Third Normal Form (3NF):**
   * Must be in 2NF.
   * No transitive dependency (non-key attribute should not depend on another non-key attribute).
4. **Boyce-Codd Normal Form (BCNF):**
   * A stricter version of 3NF.
   * Every determinant must be a candidate key.

**Design of Normalization**

**Student Course Enrollment**

**Unnormalized Form (UNF)**

In UNF, data may have **repeating groups or multivalued attributes**.

| **StudentID** | **StudentName** | **DepartmentID** | **DepartmentName** | **DepartmentHOD** | **Courses** |
| --- | --- | --- | --- | --- | --- |
| 1 | Raj | D001 | Computer Science | Dr. A | DBMS, OS |
| 2 | Priya | D002 | Electronics | Dr. B | Microprocessor |
| 3 | Sam | D001 | Computer Science | Dr. A | DBMS, Java |

**First Normal Form (1NF)**

* Eliminate **multivalued attributes** (i.e., break repeating groups).
* Ensure **atomicity** (one value per cell).

| **StudentID** | **StudentName** | **DepartmentID** | **DepartmentName** | **DepartmentHOD** | **Course** |
| --- | --- | --- | --- | --- | --- |
| 1 | Raj | D001 | Computer Science | Dr. A | DBMS |
| 1 | Raj | D001 | Computer Science | Dr. A | OS |
| 2 | Priya | D002 | Electronics | Dr. B | Microprocessor |
| 3 | Sam | D001 | Computer Science | Dr. A | DBMS |
| 3 | Sam | D001 | Computer Science | Dr. A | Java |

**Second Normal Form (2NF)**

* Must be in **1NF**.
* Remove **partial dependencies** (non-key attributes depending only on part of a composite key).

Assuming the **composite key** is (StudentID, Course):

**Partial dependency exists:** StudentName, DepartmentID, DepartmentName, DepartmentHOD depend only on **StudentID**.

**Split into two tables:**

**1. Student Table**

| **StudentID** | **StudentName** | **DepartmentID** |
| --- | --- | --- |
| 1 | Raj | D001 |
| 2 | Priya | D002 |
| 3 | Sam | D001 |

**2. Enrollment Table**

| **StudentID** | **Course** |
| --- | --- |
| 1 | DBMS |
| 1 | OS |
| 2 | Microprocessor |
| 3 | DBMS |
| 3 | Java |

**3. Department Table (still in raw form)**

| **DepartmentID** | **DepartmentName** | **DepartmentHOD** |
| --- | --- | --- |
| D001 | Computer Science | Dr. A |
| D002 | Electronics | Dr. B |

**Third Normal Form (3NF)**

* Must be in **2NF**.
* Remove **transitive dependencies** (non-key attributes depending on other non-key attributes).

**In Department Table:** DepartmentName and DepartmentHOD depend on DepartmentID → this is fine if DepartmentID is the key.

So now:

**1. Student Table (3NF)**

| **StudentID** | **StudentName** | **DepartmentID** |
| --- | --- | --- |
| 1 | Raj | D001 |
| 2 | Priya | D002 |
| 3 | Sam | D001 |

**2. Department Table (3NF)**

| **DepartmentID** | **DepartmentName** | **DepartmentHOD** |
| --- | --- | --- |
| D001 | Computer Science | Dr. A |
| D002 | Electronics | Dr. B |

**3. Enrollment Table (3NF)**

| **StudentID** | **Course** |
| --- | --- |
| 1 | DBMS |
| 1 | OS |
| 2 | Microprocessor |
| 3 | DBMS |
| 3 | Java |

**Summary of Transformation**

| **Normal Form** | **Action** |
| --- | --- |
| UNF | Raw data, multivalued and repeating fields present |
| 1NF | Remove multivalued attributes (Courses), ensure atomic values |
| 2NF | Remove partial dependencies, separate student and enrollment data |
| 3NF | Remove transitive dependencies, separate department details |

**Conclusion**

Normalization is essential in database design to organize data efficiently, reduce redundancy, and ensure data integrity.

Proper normalization improves database performance and makes data manipulation easier and more reliable.