Unit – 5
Normalization

Y.A.HATHALIYA

Normalization Basics

- Normalization is a process in database design that is used to organize data in such a way that it reduces redundancy and improves data integrity.
- The goal is to divide large, complex tables into smaller, simpler ones, ensuring relationships between the tables are maintained using keys.
- Normalization is typically achieved by applying a series of “Normal Forms," where each subsequent form builds on the previous one with stricter rules.
- Why Normalize?
  - Eliminate Redundant Data: Avoid storing the same data in multiple places.
  - Ensure Data Integrity: Prevent anomalies during data insertion, updating, and deletion.
  - Optimize Query Performance.
  - Reduce Disk Space
  - Make Table Simpler and Manageable.
  - Improve Query Performance.

1NF (First Normal Form)

- In Database Normalization, 1NF is the first step to organize data in a relational database to reduce redundancy and improve data integrity.
- A table is in 1NF if :
  - Table should not have multiple values for a single entry in a column (no multi-valued attributes).
  - Table should not have Composite values for a single entry in a column (no Composite attributes).
  - Table Must Contains Atomic Values (Means a column of a table cannot hold multiple values)

Initial Table (Not in 1NF):

Employee_ID	Employee_Name	Address	Phone_Number
1	John Doe	123 Elm St, New York	555-1234, 555-5678
2	Jane Smith	456 Oak St, Los Angeles	555-2345
3	Bob Johnson	789 Pine St, Chicago	555-3456, 555-7890

Problems:

1. **Composite Attribute**: The "Address" column stores multiple pieces of data (Street, City), which is a composite attribute.
2. **Multivalued Attribute**: The "Phone_Number" column contains multiple phone numbers for some employees, violating the rule that each column should contain atomic (indivisible) values.

Corrected Table (In 1NF):

To convert this table to **1NF**, we must:

1. **Split the composite attribute** into separate columns for each part of the address (e.g., Street, City).
2. **Separate multivalued attributes** into individual rows for each phone number.

Employee Table (1NF):

Employee_ID	Employee_Name	Street	City
1	John Doe	123 Elm St	New York
2	Jane Smith	456 Oak St	Los Angeles
3	Bob Johnson	789 Pine St	Chicago

Phone Numbers Table (1NF):

Employee_ID	Phone_Number
1	555-1234
1	555-5678
2	555-2345
3	555-3456
3	555-7890

2NF (Second Normal Form)

- A table is in 2NF if :
  - It is already in 1NF.
  - All Non-Key attributes are fully functional dependent on the primary key.
    - Means partial dependency must be removed, Partial dependency occurs when a non-key attribute is dependent on part of a composite primary key, rather than the whole key.
    - Non-key attributes should depend on the entire primary key, not just a part of it.
    - Note: In a Database a Fully Functional Dependency Occurs when an attribute is functionally dependent on a composite key but not on any subset of that key.

Example:

Consider a table (in **1NF**, but not in **2NF**) that tracks courses and the instructors who teach them:

Student_ID	Course_ID	Course_Name	Instructor
1	C101	Math	Dr. Smith
2	C102	Physics	Dr. Johnson
3	C101	Math	Dr. Smith
4	C103	Chemistry	Dr. Lee

- **Primary Key**: (Student_ID, Course_ID) (since a student can take multiple courses).
- Here, "Course_Name" and "Instructor" are dependent only on **Course_ID**, not on the full composite key (Student_ID, Course_ID), meaning we have a **partial dependency**.

To convert this into 2NF, we split the table into two separate tables:

Student-Course Table (now in 2NF):

Student_ID	Course_ID
1	C101
2	C102
3	C101
4	C103

Course Table (now in 2NF):

Course_ID	Course_Name	Instructor
C101	Math	Dr. Smith
C102	Physics	Dr. Johnson
C103	Chemistry	Dr. Lee

In this new structure:

- The **Student-Course table** has no partial dependencies. The composite key (Student_ID, Course_ID) fully determines the rows, and there are no attributes dependent on just part of the key.
- The **Course table** is independent and has a single primary key (Course_ID), with all attributes (Course_Name, Instructor) fully dependent on this key.

3NF (Third Normal Form)

- A table is in 3NF if :
  - It is already in 2NF.
  - There are no transitive dependencies available for non prime attributes (All non prime attribute must be dependent on primary key, no non prime attribute can determine other non prime attribute)
    - A transitive dependency occurs in a relational database when a non-key attribute depends on another non-key attribute rather than directly on the primary key, this creates an indirect relationship, where one non-key attribute determines another non-key attribute.
    - To achieve Third Normal Form (3NF), a table must not have any transitive dependencies.
    - Note:
      - when an indirect relationship causes functional dependency it is known as transitive dependency, for example if X->Y and Y->Z then X->Z is a transitive dependency.
      - prime attribute is an attribute that is part of at least one candidate key of a table.
      - non-prime attribute is an attribute that is not part of any candidate key

Example:

Consider a table in **2NF**, but not in **3NF**:

Student_ID	Course_ID	Instructor	Instructor_Dept
1	C101	Dr. Smith	Mathematics
2	C102	Dr. Johnson	Physics
3	C101	Dr. Smith	Mathematics
4	C103	Dr. Lee	Chemistry

- **Primary Key**: (Student_ID, Course_ID).
- **Instructor** and **Instructor_Dept** are both non-key attributes.

Here, "Instructor_Dept" is dependent on "Instructor" (not directly on the composite primary key), meaning there's a **transitive dependency**.

To convert this table into 3NF, we need to remove the transitive dependency by creating a separate table for the instructors:

Student-Course Table (in 3NF):

Student_ID	Course_ID	Instructor
1	C101	Dr. Smith
2	C102	Dr. Johnson
3	C101	Dr. Smith
4	C103	Dr. Lee

Instructor Table (in 3NF):

Instructor	Instructor_Dept
Dr. Smith	Mathematics
Dr. Johnson	Physics
Dr. Lee	Chemistry

Now, the **Student-Course table** has no transitive dependencies—all non-key attributes (Instructor) depend directly on the primary key (Student_ID, Course_ID).

In the **Instructor table**, the department information (Instructor_Dept) depends directly on the primary key (Instructor), ensuring no transitive dependencies. Therefore, both tables are in 3NF.

Advantages of Normalization

- Reduce Data Redundancy
- Reduce Data Inconsistency
- Improve Data Integrity
- Improve Performance of a Query.
- Ensure Minimum NULL Values.
- Remove Database Anomalies.
- Design of a Database become more structured.

Disadvantages of Normalization

- Difficult to Understand
- Expensive to Manage
- Required Skill Man Powers to deal normalization
- Queries might become more complex and slower
- Performance Overhead (Due to Additional Join and Sub Query)
- Time Consuming Process