SQL

NORMALIZATION

SQL Normalization

Normalization is a database design technique used to minimize redundancy and dependency by organizing the attributes and relations of a database. The goal is to ensure that the database structure is efficient and free from anomalies during insertion, deletion, and update operations. The process involves dividing large tables into smaller ones and defining relationships between them.

SQL Normalization categorized by type

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

1. First Normal Form (1NF)

A table is in **First Normal Form (1NF)** if:

All columns contain atomic (indivisible) values. There should be no repeating groups or arrays in a column.

Each record (row) is unique, i.e., there must be a primary key to identify each row.

Example: Before 1NF:

EmployeeID	E_Name	PhoneNumbers
101	John	1234567890, 0987654321
102	Jane	1112233445

After 1NF:

PhoneNumberssplit into multiple rows column was

EmployeeID	E_Name	PhoneNumbers
101	John	1234567890
101	john	0987654321
102	Jane	1112233445

2. Second Normal Form (2NF)

A table is in **Second Normal Form (2NF)** if:

It is already in **1NF**. **There are no partial dependencies.** This means that non-key attributes must depend on the **entire primary key** and not just a part of it. If a table has a composite primary key, all non-key attributes must be fully functionally dependent on the entire composite key.

Example: Before 2NF:

EmployeeID	DepartmentID	E_Name	DepartmentName
101	10	John	HR
102	20	Jane	IT

Here, **DepartmentName** is only dependent on **DepartmentID**, not on the whole primary key (EmployeeID, DepartmentID). This is a partial dependency.

After 2NF:

- Create a separate table for **Departments**:
  - **Employees Table:** Store EmployeeID and DepartmentID.

- **Departments Table**: Store DepartmentID and DepartmentName.

Employees Table:

EmployeeID	DepartmentID	E_Name
101	10	John
102	20	Jane

Departments Table:

DepartmentID	DepartmentName
10	HR
20	IT

Now, there are no partial dependencies as each non-key attribute depends on the whole primary key.

3. Third Normal Form (3NF)

A table is in **Third Normal Form (3NF)** if:

It is already in **2NF**. **There are no transitive dependencies.** This means that non-key attributes should not depend on other non-key attributes. Every non- key attribute must depend only on the primary key, not on any other non-key attribute.

Example: Before 3NF:

EmployeeID	E_name	DepartmentID	DepartmentHead
101	John	10	Sarah
102	Jane	20	Mike

Here, **DepartmentHead** depends on **DepartmentID**, not directly on the primary key **EmployeeID**. This is a transitive dependency.

After 3NF:

Split the table into two tables: one for **Employees** and one for **Departments**.

Employees Table:

EmployeeID	E_Name	DepartmentID
101	John	10
102	Jane	20

Departments Table:

DepartmentID	DepartmentHead
10	Sarah
20	Mike

Now, there are no transitive dependencies, as the **DepartmentHead** is stored in the **Departments** table, where it should be.

4. Boyce-Codd Normal Form (BCNF)

A table is in **Boyce-Codd Normal Form (BCNF)** if:

It is already in **3NF**. **Every determinant is a candidate key.** In other words, if any non- prime attribute (i.e., attribute not part of a candidate

key) determines another attribute, the determinant must be a candidate key.

Example: Before BCNF:

StudentID	CourseID	Instructor
1	CS101	Prof. A
2	CS101	Prof. A
1	CS102	Prof. B

Here, **Instructor** is determined by **CourseID**, but **CourseID** is not a candidate key because **StudentID**, **CourseID** together form a composite key.

Therefore, this table violates BCNF.

After BCNF:

Split the table into two:

Student-Course Table:

StudentID	CourseID

1	CS101
2	CS101
1	CS102

Course-Instructor Table:

CourseID	Instructor
CS101	Prof. A
CS102	Prof. B

Now, the **Instructor** is determined by **CourseID**, which is a candidate key in the second table, satisfying BCNF.