Normalisation and Denormalisation Strategies

LEARN

1. What is Normalization?

Normalization is a systematic approach in database design used to organize data efficiently. It reduces data redundancy, eliminates undesirable characteristics like insertion, update, and deletion anomalies, and ensures logical and consistent data storage.

Objectives of Normalization:

- Reduce redundancy in data storage.
- Improve data integrity.
- Simplify the structure of tables.
- Make databases more efficient and easier to maintain.

2. Normalization Forms (Detailed Explanation):

First Normal Form (1NF)

A table is in 1NF if:

- All rows are unique.
- Each column contains atomic (indivisible) values.
- There are no repeating groups or arrays within a column.

Example:

Before 1NF:

StudentID	Subjects	
1	Maths, Physics	

After 1NF:

StudentID	Subject
1	Maths
1	Physics

Second Normal Form (2NF)

A table is in 2NF if:

- It is already in 1NF.
- There are no partial dependencies, meaning every non-key attribute fully depends on the entire primary key.

Example:

Before 2NF (with partial dependency):

StudentID	CourseID	StudentName	CourseName
1	C101	John Doe	Maths

After 2NF (removal of partial dependency):

- Students(StudentID, StudentName)
- Courses(CourseID, CourseName)
- Enrollment(StudentID, CourseID)

Third Normal Form (3NF)

A table is in 3NF if:

- It is already in 2NF.
- No transitive dependency exists (non-primary attributes do not depend on other non-primary attributes).

Example:

Before 3NF (with transitive dependency):

| StudentID | StudentName | Department | DepartmentHead |

|-----------|-------------|------------|----------------|

| 1 | John Doe | Maths | Dr. Smith |

After 3NF:

- Students(StudentID, StudentName, Department)
- Departments(Department, DepartmentHead)

---

3. Denormalization

Denormalization is the intentional introduction of redundancy into a database to optimize query performance by reducing the need for complex joins.

When to Consider Denormalization?

- Frequent join operations slow query performance.
- When read operations significantly outnumber write operations.
- When database efficiency and speed are prioritized over data redundancy.

Practical Example:

An e-commerce platform may store frequently accessed customer information directly in the order table to avoid complex joins and improve query response time.

PRACTISE

Task 1: Normalize a Given Table

Given:

Order ID	CustomerName	CustomerAddress	Product
101	John Doe	New York	Laptop
102	John Doe	New York	Mouse

Convert the above table into 3NF:

- Customers(CustomerID, CustomerName, CustomerAddress)
- Products(ProductID, Product)
- Orders(OrderID, CustomerID, ProductID)

Task 2: Denormalization Exercise

Identify and justify a suitable scenario for denormalization within an online bookstore database:

- Consider tables like Books, Authors, Categories, and Reviews.

FAQ

- **Q:** Why is normalization critical for database management?
    - **A:** It eliminates redundancy, prevents anomalies, and simplifies maintenance.
- **Q:** What are potential disadvantages of excessive normalization?
    - **A:** It can lead to complex queries with many joins, potentially impacting performance.
- **Q:** In what situations is denormalization recommended?

    - **A:** When quick read access and performance outweigh data redundancy concerns, particularly in reporting databases or data warehouses.
- **Q:** Is achieving 3NF always necessary?
    - **A:** It's commonly used, but sometimes 2NF or denormalization may be adequate depending on performance needs and complexity.