Database Normalization

Anomalies in Unnormalized Tables

When a database is not normalized, it may suffer from 3 types of anomalies:

✓ Insertion Anomaly

Occurs when you can't insert a valid piece of data unless you also insert unrelated data.

Example:

EmployeeID	EmployeeName	ProjectName	ProjectLocation
1001	Ali Salim	SmartBank App	Muscat
		AI Classifier	Sohar

⚠️ You want to insert a new project (AI Classifier), but you don't yet know which employee will work on it. The table forces you to enter an employee name or ID — which leads to incomplete or incorrect data.

✓ Update Anomaly

Happens when you need to update the same piece of data in multiple places — and if you miss one, your data becomes inconsistent.

Example:

EmployeeID	EmployeeName	Department	DeptPhone
1001	Ali Salim	IT	2410 1010
1002	Ahmed Nasser	IT	2410 1010
1003	Maryam Al Balushi	IT	2410 1010

⚠️ If the IT department's phone number changes, you must update it in all rows. If one row is missed, your data becomes inconsistent.

✓ Deletion Anomaly

Occurs when deleting a row causes loss of important unrelated information.

Example:

StudentID	StudentName	CourseName
S001	Salim Harithi	Databases
S002	Maryam Rashid	Databases
S003	--	Databases

⚠️ If all students drop the "Databases" course and you delete those rows, you will lose information about the course itself, even though it might still be part of the curriculum.

✓ How Normalization Fixes This

By splitting the table into smaller related tables **(like Employees, Departments, Projects, etc.)**, you**:**

- Can insert data without needing unrelated info **(Fixes Insertion anomaly)**
- Update data in one place only **(Fixes Update anomaly)**
- Don’t lose data unintentionally **(Fixes Deletion anomaly)**

What’s the Difference Between Normalization and Database Mapping?

Concept	Database Mapping	Normalization
Definition	Translating the ERD (conceptual model) into tables	Refining tables to reduce redundancy and improve integrity
When Used	During initial design phase of the database	After or during design, especially to improve an existing schema
Goal	Convert entities, relationships, and attributes into a working relational schema	Eliminate anomalies (insertion, update, deletion), redundancy, and improve structure
Example Action	Multivalued attribute → separate table (by rule)	Partial dependency → split tables (by analysis)
Focus	Mapping structure from design into tables (mechanical step)	Optimizing structure based on logic and data behavior
Scenario	A new system being built from ERD	An old database that needs cleanup and better structure

Normalization is a **systematic process** in database systems used to **organize data** to reduce **redundancy** and improve **data integrity (سلامة وتكامل البيانات)**. It involves dividing a database into tables and defining relationships between them to:

- Minimize duplicate data
- Eliminate update anomalies
- Ensure meaningful data dependencies

Purpose of Normalization

1. **Reduce Data Redundancy**
   Prevent duplicate data across tables to save storage and avoid inconsistencies.
2. **Enhance Data Integrity**
   Ensure accuracy and consistency using well-defined relationships and constraints.
3. **Simplify Data Maintenance**
   Make updates, inserts, deletions, and retrievals easier and safer.

1. First Normal Form (1NF)

- **Rule:** Eliminate repeating groups and ensure atomic (indivisible) values in each field.
- **Goal:** No multivalued attributes; each row is unique.

Example (before 1NF):

من الطلاب الي مسجلين في مادة الرياضيات؟!

StudentID	StudentName	Courses
1	Ali Salim	Math, Science
2	Ahmed Nasser	English, History, Math

After 1NF:

StudentID	StudentName	Course
1	Ali Salim	Math
1	Ali Salim	Science
2	Ahmed Nasser	English

2. Second Normal Form (2NF)

- **Rule:** Achieve 1NF and remove partial dependencies.
- **Goal:** All non-key attributes must depend on the **whole** primary key (not part of it).

Example (before 2NF):

Suppose this table uses a **composite key** (StudentID, CourseID):

StudentID	CourseID	StudentName	CourseName
1	101	Ali Salim	Math
1	102	Ali Salim	Science
2	101	Ahmed Nasser	Math

- StudentName depends only on StudentID
- CourseName depends only on CourseID

After 2NF (split into 3 tables):

Students Table

StudentID	StudentName
1	Ali Salim
2	Ahmed Nasser

Courses Table

CourseID	CourseName
101	Math
102	Science

Enrollments Table

StudentID	CourseID
1	101
1	102
2	101

4. Third Normal Form (3NF)

Rule: Achieve 2NF and remove transitive dependencies (non-key attributes should depend only on the key, not other non-key attributes).

Before 3NF:

OrderID	CustomerID	CustomerName	CustomerAddress
1	C001	Salim Al Harithi	Gulf Street 12, Muscat
2	C002	Maryam Al Balushi	Nahdha Street 45, Sohar

CustomerName and CustomerAddress depend on CustomerID, not on OrderID.

After 3NF:

Orders Table

OrderID	CustomerID
1	C001
2	C002

Customers Table

CustomerID	CustomerName	CustomerAddress
C001	Salim Al Harithi	Gulf Street 12, Muscat
C002	Maryam Al Balushi	Nahdha Street 45, Sohar