Normalisation

1.What is Normalization?

Normalization is a process in database design where a table is structured in such a way that **data is stored without repetition (redundancy)** and **dependency remains logical and minimal**.
In simple words, normalization organizes data into smaller related tables so that the database remains clean, efficient, and easy to update.

2.Why do we need Normalization?

We normalize a database to:

1. **Avoid duplicate / repeated data**
2. **Reduce unnecessary storage usage**
3. **Ensure data is stored in a proper, organized form**
4. **Make data easy to update, insert, or delete**
5. **Improve data consistency & accuracy**
6. **Maintain relationships properly between tables**

3.What problems does Normalization solve?

Normalization mainly removes **3 types of anomalies (problems)**:

Problem	Meaning	Effect without normalization
Update Anomaly	Changing data in one place requires change in multiple places	Risk of inconsistency / mismatched data
Insert Anomaly	Unable to add new data without adding unnecessary data	Extra unwanted values need to be inserted
Delete Anomaly	Deleting one record deletes important information	Useful data may get lost accidentally

Simple Example

Suppose we store student and course info in one table:

Student	Course	Teacher
Vijay	DBMS	Sharma
Vijay	OS	Sharma
Mikasa	DBMS	Sharma

Problems:

1.Duplicate Teacher name repeated
2.If teacher Sharma leaves DBMS, deleting one row may remove student data
3.To update teacher name, we must update it in many rows

After Normalization

We split into two tables:

Students Table

Student	Course
Vijay	DBMS
Vijay	OS
Mikasa	DBMS

Teacher Table

Course	Teacher
DBMS	Sharma
OS	Sharma

Now:

No duplicate values
Update teacher name once only

Insert / delete becomes safe

Data remains consistent

Normalisation is a step towards DB optimisation.

Functional Dependency (FD)

Functional Dependency means:

If one attribute (or set of attributes) can uniquely determine another attribute in a relation, we say there is a functional dependency between them.

Example:
Roll_No → Student_Name
If we know the Roll_No, we can find the Student_Name uniquely.

Here:

- **Roll_No is Determinant (Left side)**
- **Student_Name is Dependent (Right side)**

Notation

X → Y
X determines Y

🔹 Types of FD

Type	Meaning	Example
Trivial FD	Dependent is part of determinant	A → A, AB → A
Non-Trivial FD	Dependent is not part of determinant	Roll_No → Phone_No

Trivial FD = **Obvious**, nothing new is determined.
Non-trivial FD = **Useful dependency**.

Armstrong's Axioms (FD Rules)

Rule	Meaning	Example
Reflexive	If B is subset of A → A → B	ABC → A
Augmentation	Add extra attribute on both sides	If A → B → AC → BC
Transitivity	Indirect dependency	If A → B & B → C → A → C

These rules help in decomposition and normalization.

Why Normalization?

Because databases should **not store unnecessary repetitive data**.

If redundancy exists, it creates problems:

Anomaly	What happens?
Insertion Anomaly	Can't insert data unless extra unwanted data is also added
Deletion Anomaly	Deleting one record may delete important information
Updation Anomaly	Update must be done in many places → inconsistency risk

Due to redundancy → **DB size increases, performance decreases**.

What is Normalization?

Normalization is a **database optimization technique** used to:

✓ Reduce/Remove redundancy
✓ Avoid anomalies (Insert/Delete/Update)
✓ Break large tables into smaller related tables
✓ Store data in a structured and efficient manner

Simple line:

Normalization = Split big table into smaller tables to remove redundancy and avoid anomalies.

InShort

Topic	Shortcut Memory
FD	One attribute determines another
Determinant	Left side of FD
Dependent	Right side of FD
Trivial FD	B ⊆ A
Normalization	Removes redundancy + anomalies
Anomalies	Insert, Delete, Update problems

Types of Normal Forms

1. First Normal Form (1NF)

A relation is in **1NF** when:

Every cell contains **single (atomic) value**
No multi-valued or repeating groups in a column

Example (NOT 1NF)

Student	Phone_No
Sanya	7876, 7873

After converting to 1NF →

Student	Phone_No
Sanya	7876
Sanya	7873

2. Second Normal Form (2NF)

A table is in **2NF** when:

It is already in **1NF**
There is **no Partial Dependency**

Meaning:
If **Primary Key is composite** (more than one attribute) →
then **no non-prime attribute should depend on part of the key**.

Example of Partial Dependency

(Roll_No, Subject) → Student_Name
Student_Name depends only on Roll_No (part of key).

Remove partial dependency → Normalize into two tables.

Example (Not in 2NF)

Roll_No	Subject	Student_Name
5	Math	Vijay
6	Physics	Vijay
7	Chemistry	Mona

Composite Primary Key = **(Roll_No, Subject)**
But **Student_Name** depends only on **Roll_No**, not on both → Partial Dependency

Convert to 2NF

Table 1: Students

Roll_No	Student_Name
5	Vijay
6	Mona

Table 2: Enrollment

Roll_No	Subject
5	Math
6	Physics
7	Chemistry

Now **every non-prime attribute depends on full key → 2NF achieved.**

3. Third Normal Form (3NF)

A table is in **3NF** when:

It is already in **2NF**
No transitive dependency exists

Meaning:
Non-prime attribute should **not depend on another non-prime attribute**.

Example

Roll_No → Dept
Dept → HOD (Transitive Dependency)

Break into two tables so dependency becomes direct.

Example (Not in 3NF)

Roll_No	Dept	HOD
5	CS	Sharma
6	IT	Singh
7	CS	Sharma

Dependencies:

- Roll_No → Dept
- Dept → HOD (**Non-prime → Non-prime**) = Transitive Dependency

So table is **not in 3NF**

Convert to 3NF

Table 1: Student

Roll_No	Dept
5	CS
6	IT
7	CS

Table 2: Department

Dept	HOD
CS	Sharma
IT	Singh

Now HOD does not depend on Roll_No → Transitivity removed → **3NF achieved.**

4. BCNF (Boyce–Codd Normal Form)

A table is in **BCNF** when:

It is in **3NF**
For every FD **A → B**, A must be a **super key**

BCNF is stronger than 3NF.

Meaning:
No attribute (prime or non-prime) should determine key attribute unless it itself is a key.

If any attribute determines another key attribute → must break table further.

Example (3NF but not BCNF)

Course	Teacher
DBMS	Ajay
CN	Mehra
DBMS	Mehra

Functional Dependencies:

1. Course → Teacher
2. Teacher → Course

Since **Teacher is not a key**, second dependency violates BCNF.

Convert to BCNF

Table 1: Teacher → Course

Teacher	Course
Ajay	DBMS
Mehra	CN

Table 2: Course → Teacher

Course	Teacher
DBMS	Ajay

Now every determinant is a **super key** → **BCNF achieved.**

Advantages of Normalization

Benefit	Why it Matters
Reduces redundancy	No repeated/unnecessary data
Improves storage efficiency	Saves memory space
Avoids anomalies	No insert, delete, update problems
Maintains consistency	Data remains accurate everywhere
Faster updates	Change data at one place only
Better database performance	Query execution becomes smooth

Memory Trick

1NF → Remove multi-values
2NF → Remove partial dependency
3NF → Remove transitive dependency
BCNF → Determinant must be superkey

Normal Form	Condition	Removes	Example Issue
1NF	Atomic values only	Repeating groups	Vijay, Singh in one cell
2NF	In 1NF + No partial dependency	Partial dependency	Student_Name depends only on Roll_No
3NF	In 2NF + No transitive dependency	Transitivity	Dept → HOD from Roll_No
BCNF	LHS of every FD must be super key	All anomalies	Teacher determines Course