6.Normalization

Wednesday, February 4, 2026 2:00 PM

Normalization is the process of restructuring a relational database so that data is stored efficiently, without unnecessary repetition or duplication.

Purpose

- Reduce data redundancy (repetition of the same data)
- Improve data consistency
- Make updates easier and safer
- Organize data into well-structured tables
- Ensure each table has a clear purpose

Why Redundancy Is a Problem

Redundancy means storing the same information multiple times.
This leads to:

- Wasted storage
- Update anomalies
- Inconsistent data
- More complex maintenance

Most Commonly Used Normal Form

The most widely applied normal form in real-world databases is **Third Normal Form (3NF)** because it provides a good balance between structure and performance.

Levels of Normalization

1. First Normal Form (1NF)

- Data must be stored in rows and columns
- Each column contains atomic (indivisible) values
- No repeating groups or multi-valued attributes

2. Second Normal Form (2NF)

- Must already be in 1NF
- No partial dependency
- Every non-key column must depend on the whole primary key

3. Third Normal Form (3NF)

- Must already be in 2NF
- No transitive dependency
- Non-key columns should depend only on the primary key

4. Boyce-Codd Normal Form (BCNF)

- A stronger version of 3NF
- Every determinant must be a candidate key

5. Fifth Normal Form (5NF)

- Deals with complex join dependencies
- Ensures data cannot be reconstructed incorrectly from smaller pieces

6. Sixth Normal Form (6NF)

- Used in very specialized systems
- Breaks data into the smallest possible units
- Rarely used in typical business databases

First Normal Form (1NF)

Meaning

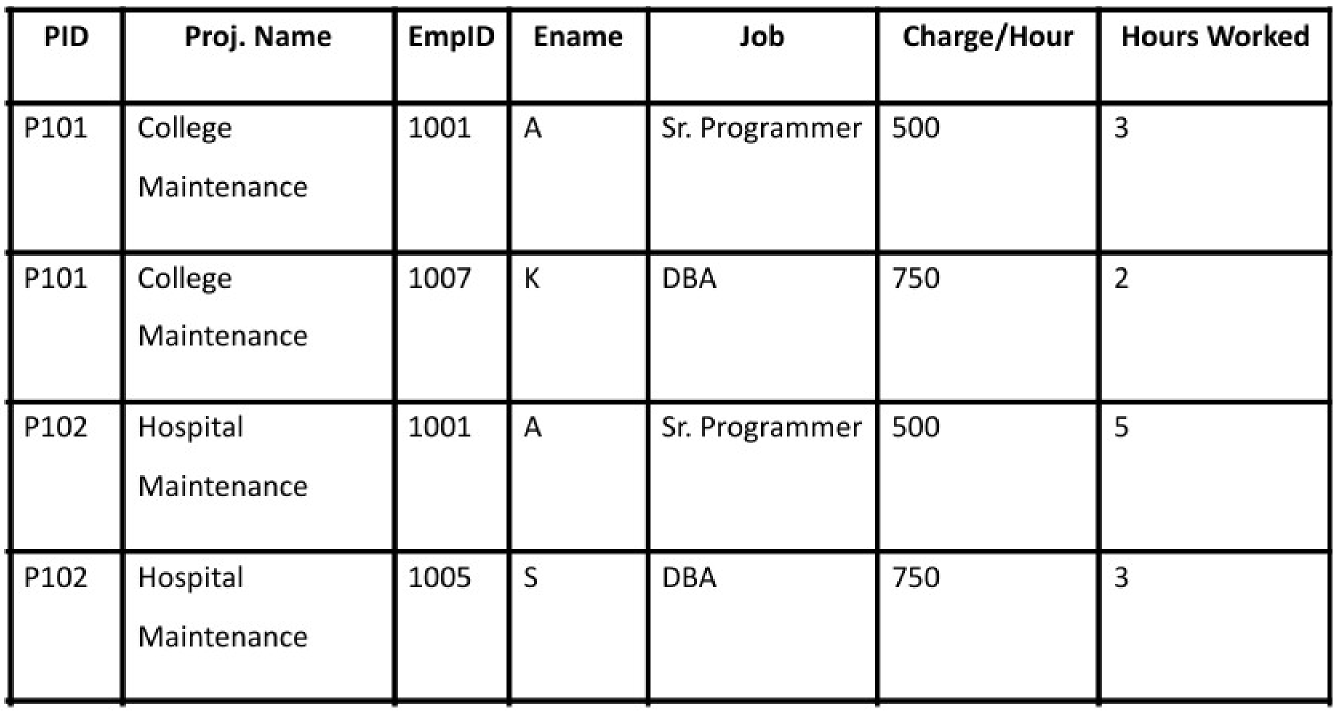
A relation is said to be in First Normal Form (1NF) if and only if a primary key is defined for the relation and all non-key attributes depend on the key attributes.

Key Attributes vs Non-Key Attributes

- **Key Attributes**: Attributes that are part of the primary key
- **Non-Key Attributes**: All remaining attributes in the table

In 1NF, every non-key attribute must depend on the key attribute(s).
Core Idea
1NF ensures that the structure of the table is clean and organized by enforcing:

- A defined primary key
- Atomic (indivisible) values
- No repeating groups
- Proper dependency of non-key attributes on key attributes



PID	Proj. Name	EmpID	Ename	Job	Charge/Hour	Hours Worked
P101	College Maintenance	1001	A	Sr. Programmer	500	3
P101	College Maintenance	1007	K	DBA	750	2
P102	Hospital Maintenance	1001	A	Sr. Programmer	500	5
P102	Hospital Maintenance	1005	S	DBA	750	3

How to Bring the Table into 1NF
To convert the earlier table into First Normal Form (1NF), a proper primary key must be defined, and all non-key attributes must depend on that key.
In the original table, no single column can act as a primary key because every column contains duplicate values. Since no single attribute uniquely identifies each row, we must create a **composite primary key**.
Composite Primary Key
The combination of **PID** (Project ID) and **EMPID** (Employee ID) uniquely identifies each row.
By using both columns together, every row becomes unique.
Dependency Requirement
Once the composite key (PID, EMPID) is defined, all other attributes—such as Pname, Ename, Job, Charge/Hour, and Hours Worked—depend on this key.
This satisfies the requirement of 1NF.
Why the Table Is Now in 1NF

- A primary key is defined (PID + EMPID)
- All values are atomic
- No repeating groups
- Every non-key attribute depends on the key

Final Table Structure in 1NF
Primary Key: (PID, EMPID)
Non-Key Attributes: Pname, Ename, Job, Charge/Hour, Hours Worked
The table is represented as:
(PID, EMPID) → Pname, Ename, Job, Charge/Hour, Hours Worked

Second Normal Form (2NF)
Meaning
A relation is said to be in Second Normal Form (2NF) if and only if:

- The relation is already in **1NF**, and
- **No partial dependencies** exist.

What Is Partial Dependency
A partial dependency occurs when:

- The primary key is made up of **more than one attribute** (composite key), and
- A non-key attribute depends on **only one part** of the composite key instead of the whole key.

When this happens, the table violates 2NF.
Why the 1NF Table Is Not in 2NF
In the 1NF version, the primary key is the combination:
(PID, EMPID)
But the dependencies look like this:

- To get **PNAME**, you only need **PID**
  → PNAME depends only on part of the key
- To get **ENAME, JOB, CHARGE/HOUR**, you only need **EMPID**
  → These attributes depend only on EMPID
- Only **HOURS WORKED** depends on both PID and EMPID together

Because several non-key attributes depend on only one part of the composite key, the table contains **partial dependencies** and is **not in 2NF**.
Original 1NF Relation
PID, EMPID) → Pname, Ename, Job, Charge/Hour, Hours Worked
Converting the Table to 2NF
To remove partial dependencies, the table must be split into three relations:
1. Project Table
Primary Key: (PID)
Dependent Attribute: Pname
2. Employee Table
Primary Key: (EMPID)
Dependent Attributes: Ename, Job, Charge/Hour
3. Project-Employee Work Table
Primary Key: (PID, EMPID)
Dependent Attribute: Hours Worked
Final 2NF Relations

1. **(PID) → Pname**
2. **(EMPID) → Ename, Job, Charge/Hour**
3. **(PID, EMPID) → Hours Worked**

Now all three relations are in **2NF** because:

- They are in 1NF
- They contain **no partial dependencies**

Third Normal Form (3NF)
Meaning
A relation is said to be in Third Normal Form (3NF) if and only if:

- The relation is already in **2NF**, and
- **No transitive (transient) dependencies** exist.

What Is a Transitive Dependency
A transitive dependency occurs when:

- A **non-key attribute depends on another non-key attribute**, instead of depending directly on the primary key.
  This violates 3NF because non-key attributes should depend only on the key.

Why the 2NF Tables Are Not in 3NF

From the 2NF conversion, the second table was:
(EMPID) → Ename, Job, Charge/Hour
But here:

- **Charge/Hour** depends on **Job**, not on EMPID
  This means Charge/Hour is indirectly dependent on the key through another non-key attribute.

This is a **transitive dependency,** so the table is **not in 3NF**.

Converting the Tables to 3NF

To remove the transitive dependency, the table must be split further:

1. **(PID) → Pname**
2. **(EMPID) → Ename, Job**
3. **(JOB) → Charge/Hour**
4. **(PID, EMPID) → Hours Worked**

Now each table contains attributes that depend only on its key, and no non-key attribute depends on another non-key attribute.