**Normalization**

**Definition:**

Normalization in database design is the process of organizing a database to reduce repetition and unnecessary connections between data. This is done by breaking large tables into smaller ones while keeping the data accurate and consistent. In PostgreSQL, normalization is done using different levels called normal forms. These range from 1NF to 5NF, and each level has its own specific rules.

**1. First Normal Form (1NF)**

A table is in **1NF** if it meets the following criteria:

* Each column contains atomic (indivisible) values.
* Each column contains values of a single type.
* Each row must be unique (i.e., no duplicates).
* The order of the data does not matter.

**Example:**

Let's take an example of a table storing customer orders:

| Customer\_ID | Customer\_Name | Product\_Ordered |
| --- | --- | --- |
| 1 | Alice | Laptop, Mouse |
| 2 | Bob | Phone, Headphones |

In **1NF**, we need to remove multi-valued attributes. We can achieve this by creating a new row for each product.

| **Customer\_ID** | **Customer\_Name** | **Product\_Ordered** |
| --- | --- | --- |
| 1 | Alice | Laptop |
| 1 | Alice | Mouse |
| 2 | Bob | Phone |
| 2 | Bob | Headphones |

**Advantages of 1NF:**

* Eliminates multi-valued columns.
* Each value is atomic, which makes querying simpler and more efficient.

**Limitations of 1NF:**

* Redundancy may still exist (repeated customer information).
* Can lead to large data sets when rows are duplicated for each value.

**2. Second Normal Form (2NF)**

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

* It is in **1NF**.
* All non-key attributes are fully dependent on the primary key (i.e., no partial dependency).

**Example:**

Consider the following table:

| **Order\_ID** | **Customer\_ID** | **Customer\_Name** | **Product\_Ordered** |
| --- | --- | --- | --- |
| 101 | 1 | Alice | Laptop |
| 102 | 1 | Alice | Mouse |
| 103 | 2 | Bob | Phone |

In this table, Customer\_Name depends on Customer\_ID, not the composite key (Order\_ID, Customer\_ID). To bring the table to **2NF**, we create a new table for customers and remove the redundancy.

**Orders Table:**

| **Order\_ID** | **Customer\_ID** | **Product\_Ordered** |
| --- | --- | --- |
| 101 | 1 | Laptop |
| 102 | 1 | Mouse |
| 103 | 2 | Phone |

**Customers Table:**

| **Customer\_ID** | **Customer\_Name** |
| --- | --- |
| 1 | Alice |
| 2 | Bob |

**Advantages of 2NF:**

* Eliminates partial dependency, improving data consistency.
* Reduces redundancy in customer data.

**Limitations of 2NF:**

* May still contain transitive dependencies, where non-key attributes depend on other non-key attributes.

**3. Third Normal Form (3NF)**

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

* It is in **2NF**.
* It has no transitive dependency (i.e., non-key attributes depend on other non-key attributes).

**Example:**

Consider the following table:

| **Order\_ID** | **Customer\_ID** | **Customer\_Name** | **Customer\_Country** | **Product\_Ordered** |
| --- | --- | --- | --- | --- |
| 101 | 1 | Alice | USA | Laptop |
| 102 | 2 | Bob | UK | Phone |

Here, Customer\_Country depends on Customer\_ID, which indirectly depends on Order\_ID. To bring this to **3NF**, we create separate tables for orders and customers:

**Orders Table:**

| **Order\_ID** | **Customer\_ID** | **Product\_Ordered** |
| --- | --- | --- |
| 101 | 1 | Laptop |
| 102 | 2 | Phone |

**Customers Table:**

| **Customer\_ID** | **Customer\_Name** | **Customer\_Country** |
| --- | --- | --- |
| 1 | Alice | USA |
| 2 | Bob | UK |

**Advantages of 3NF:**

* Eliminates transitive dependencies.
* Ensures that data is fully dependent on the primary key.

**Limitations of 3NF:**

* Might require more joins when querying complex relationships.
* Data retrieval might become slower in highly normalized tables due to joins.

**4. Boyce-Codd Normal Form (BCNF)**

A table is in **BCNF** if:

* It is in **3NF**.
* For every functional dependency, X → Y, X must be a superkey.

**Example:**

Consider a table that records students and the courses they are assigned to:

| **Student\_ID** | **Course\_Code** | **Instructor** |
| --- | --- | --- |
| 101 | CS101 | Dr. Smith |
| 102 | CS101 | Dr. Smith |
| 103 | CS102 | Dr. Jones |

In this table, Course\_Code → Instructor (i.e., each course is taught by a specific instructor), but Course\_Code is not a superkey. To bring this to **BCNF**, we split the table:

**Student\_Courses Table:**

| **Student\_ID** | **Course\_Code** |
| --- | --- |
| 101 | CS101 |
| 102 | CS101 |
| 103 | CS102 |

**Courses Table:**

| **Course\_Code** | **Instructor** |
| --- | --- |
| CS101 | Dr. Smith |
| CS102 | Dr. Jones |

**Advantages of BCNF:**

* Resolves all potential redundancy issues.
* Improves data integrity.

**Limitations of BCNF:**

* May lead to more tables, increasing complexity.
* Sometimes requires more joins to query related data.

**5. Fourth Normal Form (4NF)**

A table is in **4NF** if:

* It is in **BCNF**.
* It does not contain multi-valued dependencies.

**Example:**

Consider a table with students who may have multiple phone numbers and multiple courses:

| **Student\_ID** | **Phone\_Number** | **Course\_Code** |
| --- | --- | --- |
| 101 | 12345 | CS101 |
| 101 | 67890 | CS102 |
| 102 | 11223 | CS101 |

To bring it into **4NF**, we separate the multi-valued dependencies (phones and courses):

**Student\_Phones Table:**

| **Student\_ID** | **Phone\_Number** |
| --- | --- |
| 101 | 12345 |
| 101 | 67890 |
| 102 | 11223 |

**Student\_Courses Table:**

| **Student\_ID** | **Course\_Code** |
| --- | --- |
| 101 | CS101 |
| 101 | CS102 |
| 102 | CS101 |

**Advantages of 4NF:**

* Resolves multi-valued dependencies, preventing redundancy.
* Eliminates complex relationships.

**Limitations of 4NF:**

* Can lead to a large number of tables, making the schema more complex.
* Queries may require many joins.

**6. Fifth Normal Form (5NF)**

A table is in **5NF** if:

* It is in **4NF**.
* It cannot be decomposed further into smaller tables without losing data.

**Example:**

A table that stores information about orders with multiple products and salespersons:

| **Order\_ID** | **Product\_Code** | **Salesperson** |
| --- | --- | --- |
| 101 | P001 | Alice |
| 101 | P002 | Bob |
| 102 | P001 | Alice |

This table can be decomposed into:

**Orders Table:**

| **Order\_ID** | **Product\_Code** |
| --- | --- |
| 101 | P001 |
| 101 | P002 |
| 102 | P001 |

**Salespersons Table:**

| **Order\_ID** | **Salesperson** |
| --- | --- |
| 101 | Alice |
| 101 | Bob |
| 102 | Alice |

**Advantages of 5NF:**

* Maximizes data efficiency.
* Prevents complex redundancy.

**Limitations of 5NF:**

* Very complex to maintain and design.
* Often overkill for most practical database designs.