Third Normal Form (3NF)

Last Updated : 25 Jul, 2025

The Third Normal Form (3NF**)** builds on the First (1NF) and Second (2NF) Normal Forms. Achieving 3NF ensures that the database structure is free of transitive dependencies, reducing the chances of data anomalies. Even though tables in 2NF have reduced redundancy compared to 1NF, they may still encounter issues like **update anomalies**.

A relation is in **Third Normal Form (3NF)** if it satisfies the following two conditions:

1. It is in **Second Normal Form (2NF)**: This means the table has no partial dependencies (i.e., no non-prime attribute is dependent on a part of a candidate key).
2. There is **no transitive dependency** for **non-prime attributes**: In simpler terms, no non-key attribute should depend on another non-key attribute. Instead, all non-key attributes should depend directly on the primary key.

Understanding Transitive Dependency

To fully grasp 3NF, it's essential to understand **transitive dependency**. A **transitive dependency** occurs when one non-prime attribute depends on another non-prime attribute rather than depending directly on the primary key. This can create redundancy and inconsistencies in the database.

For example, if we have the following relationship between attributes:

- **A -> B** (A determines B)
- **B -> C** (B determines C)

This means that **A** indirectly determines **C** through **B**, creating a **transitive dependency**. 3NF eliminates these transitive dependencies to ensure that non-key attributes are directly dependent only on the primary key.

Conditions for a Table to be in 3NF

A table is in **Third Normal Form (3NF)** if, for every non-trivial functional dependency X→Y, at least one of the following holds:

- **X is a superkey**: This means that the attribute(s) on the left-hand side of the functional dependency (X) must be a superkey (a key that uniquely identifies a tuple in the table).
- **Y is a prime attribute**: This means that every element of the attribute set Y must be part of a candidate key (i.e., a prime attribute).

Example 1: Third Normal Form (3NF)

Consider the following relation for a **Candidate** table with the following attributes and functional dependencies:

CANDIDATE Table

CAND_NO	CAND_NAME	CAND_STATE	CAND_COUNTRY	CAND_AGE
1	Amayra	West Bengal	India	18
2	Rihaan	Haryana	India	17
3	Manya	Haryana	India	19

1. Functional dependency Set:

The set of functional dependencies is as follows:

- **CAND_NO → CAND_NAME**
- **CAND_NO → CAND_STATE**
- **CAND_STATE → CAND_COUNTRY**
- **CAND_NO → CAND_AGE**

2. Determining the Candidate Key:

The **candidate key** for this relation is **{CAND_NO}**, since **CAND_NO** uniquely identifies all other attributes in the table.

3. Identifying Transitive Dependency:

The issue here arises from the **transitive dependency** between **CAND_NO** and **CAND_COUNTRY**:

- **CAND_NO → CAND_STATE**
- **CAND_STATE → CAND_COUNTRY**

This means that **CAND_COUNTRY** is transitively dependent on **CAND_NO** via **CAND_STATE**, which violates the **Third Normal Form (3NF)** rule that states that no non-prime attribute (non-key attribute) should be transitively dependent on the primary key.

Converting the Relation into 3NF

To remove the transitive dependency and ensure the relation is in **3NF**, we decompose the original **CANDIDATE** relation into two separate relations:

1. **CANDIDATE**: This will store information about the candidates, including their **CAND_NO**, **CAND_NAME**, **CAND_STATE**, and **CAND_AGE**:\text{CANDIDATE (CAND_NO, CAND_NAME, CAND_STATE, CAND_AGE)}
2. **STATE_COUNTRY**: This relation will store information about the states and their respective countries:\text{STATE_COUNTRY (CAND_STATE, CAND_COUNTRY)}

Final Decomposed Relations:

1. **CANDIDATE (CAND_NO, CAND_NAME, CAND_STATE, CAND_AGE)**
2. **STATE_COUNTRY (CAND_STATE, CAND_COUNTRY)**

Why This Decomposition Works:

- The **CANDIDATE** relation now no longer has a transitive dependency. **CAND_STATE** no longer determines **CAND_COUNTRY** within this relation.
- The **STATE_COUNTRY** relation handles the **CAND_STATE → CAND_COUNTRY** dependency separately, ensuring that all data is now organized in a way that satisfies **3NF**.

CANDIDATE Table

CAND_NO	CAND_NAME	CAND_STATE	CAND_AGE
1	Amayra	West Bengal	18
2	Rihaan	Haryana	17
3	Manya	Haryana	19

STATE_COUNTRY Table

CAND_STATE	CAND_COUNTRY
West Bengal	India
Haryana	India

Example 2: Relation R(A, B, C, D, E)

Consider the relation **R(A, B, C, D, E)** with the following functional dependencies:

A → BC
CD → E
B → D
E → A

Step 1: Identify Candidate Keys

A **candidate key** is a minimal set of attributes that can uniquely identify a tuple (row) in the relation. In this case, the possible candidate keys for the relation are **{A, E, CD, BC}**. This means that any of these sets of attributes can uniquely identify all other attributes in the relation.

Step 2: Check Functional Dependencies

Let's analyze the given functional dependencies:

1. **A → BC**: This means that knowing **A** allows us to determine both **B** and **C**.
2. **CD → E**: Knowing **CD** allows us to determine **E**.

3. **B** → **D**: Knowing **B** allows us to determine **D**.
4. **E** → **A**: Knowing **E** allows us to determine **A**.

We observe that all attributes on the right-hand side of the functional dependencies are **prime attributes** (i.e., they are part of some candidate key). This means no non-prime attribute is dependent on another non-prime attribute (which would be a **transitive dependency**).

Step 3: Check for Transitive Dependencies

In **3NF**, a relation must be free of transitive dependencies, where a non-prime attribute depends on another non-prime attribute indirectly via the primary key.

- Here, **A** → **BC** and **B** → **D**, so **B** is a non-prime attribute that determines **D**, and **A** determines **B**. However, since **B** is part of a candidate key, this does not introduce a transitive dependency.
- **E** → **A** and **A** → **BC**, meaning **E** determines **A**, and then **A** determines **B** and **C**. Again, no transitive dependency is formed because **A** is part of a candidate key.

Since there are no transitive dependencies, the relation **R** satisfies the condition of **3NF**.

Step 4: Conclusion

Relation R(A, B, C, D, E) is already in **Third Normal Form (3NF)** because:

- There are no transitive dependencies.
- All non-prime attributes are functionally dependent only on candidate keys.

Why is 3NF Important?

1. Eliminates Redundancy: **3NF** helps to remove unnecessary duplication of data by ensuring that non-prime attributes (attributes not part of any candidate key) depend directly on the primary key, not on other non-prime attributes.

2. Prevents Anomalies: A table in **3NF** is free from common anomalies such as:

- **Insertion Anomaly**: The inability to insert data without having to insert unwanted or redundant data.
- **Update Anomaly**: The need to update multiple rows of data when a change occurs in one place.
- **Deletion Anomaly**: The unintended loss of data when a record is deleted.

3. Preserves Functional Dependencies: **3NF** ensures that all functional dependencies are preserved, meaning that the relationships between attributes are maintained.

4. Lossless Decomposition: When decomposing a relation to achieve **3NF**, the decomposition should be **lossless**, meaning no information is lost in the process of normalization.