# Second Normal Form (2NF)

Last Updated : 12 Jul, 2025

Second Normal Form (2NF) is based on the concept of fully functional dependency. It is a way to organize a database table so that it reduces redundancy and ensures data consistency. Fully Functional Dependency means a non-key attribute depends on the **entire** primary key, not just part of it.

For a table to be in 2NF, it must first meet the following requirements

**1. Meet 1NF Requirements**: The table must first satisfy **First Normal Form (1NF)**, meaning:

- All columns contain single, indivisible values.
- No repeating groups of columns.

**2. Eliminate Partial Dependencies**: A partial dependency occurs when a non-prime attribute (not part of the candidate key) depends only on a part of a composite primary key, rather than the entire key.

By ensuring these steps, a table in 2NF is more efficient and less prone to errors during updates, inserts, and deletes.

## What is Partial Dependency?

The FD (functional dependency) A->B happens to be a partial dependency if B is functionally dependent on A, and also B can be determined by any other proper subset of A.

In other words, if you have a composite key (a primary key made up of more than one attribute), and an attribute depends on only a subset of that composite key, rather than the entire key, that is considered a partial dependency.

A partial dependency would occur whenever a non-prime attribute depends functionally on a part of the given candidate key.

**Example:**

**StaffBranch**

| staffNo | sName | position | salary | branchNo |
|---|---|---|---|---|
| SL21 | Tannu | Manager | 30000 | B005 |
| SG37 | Leela | Assistant | 12000 | B003 |
| SG14 | Henry | Supervisor | 18000 | B003 |
| SA9 | Suresh | Assistant | 9000 | B007 |
| SG5 | Anthony | Manager | 24000 | B003 |
| SL41 | Julie | Assistant | 9000 | B005 |

*StaffBranch Relation*

In the given relation StaffBranch, we have the functional dependency:

- staffNo, sName → branchNo.

This means that the combination of staffNo and sName determines branchNo.

BranchNo is also functionally dependent on a subset of the composite key, specifically staffNo. This means that branchNo can be determined by just staffNo.

- staffNo → branchNo

This is a partial dependency because branchNo depends on only a part of the composite key (staffNo, sName), not the entire key.

## Example of Second Normal Form (2NF)

Consider a table storing information about students, courses, and their fees:

**COURSES Table**

| STUD_NO | COURSE_NO | COURSE_FEE |
|---|---|---|
| 1 | C1 | 1000 |
| 2 | C2 | 1500 |
| 1 | C4 | 2000 |
| 4 | C3 | 1000 |
| 4 | C1 | 1000 |
| 2 | C5 | 2000 |

- There are many courses having the same course fee. Here, COURSE_FEE cannot alone decide the value of COURSE_NO or STUD_NO.
- COURSE_FEE together with STUD_NO cannot decide the value of COURSE_NO.
- COURSE_FEE together with COURSE_NO cannot decide the value of STUD_NO.
- The candidate key for this table is `{STUD_NO, COURSE_NO}` because the combination of these two columns uniquely identifies each row in the table.
- COURSE_FEE is a **non-prime attribute** because it is not part of the candidate key `{STUD_NO, COURSE_NO}`.
- But, COURSE_NO -> COURSE_FEE, i.e., COURSE_FEE is dependent on COURSE_NO, which is a proper subset of the candidate key.
- Therefore, Non-prime attribute COURSE_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF.

### In 2NF, we eliminate such dependencies by breaking the table into two separate tables:

1. A table that links students and courses.
2. A table that stores course fees.

**STUDENT_COURSES Table**

| STUD_NO | COURSE_NO |
|---|---|
| 1 | C1 |
| 2 | C2 |
| 1 | C4 |
| 4 | C3 |
| 4 | C1 |
| 2 | C5 |

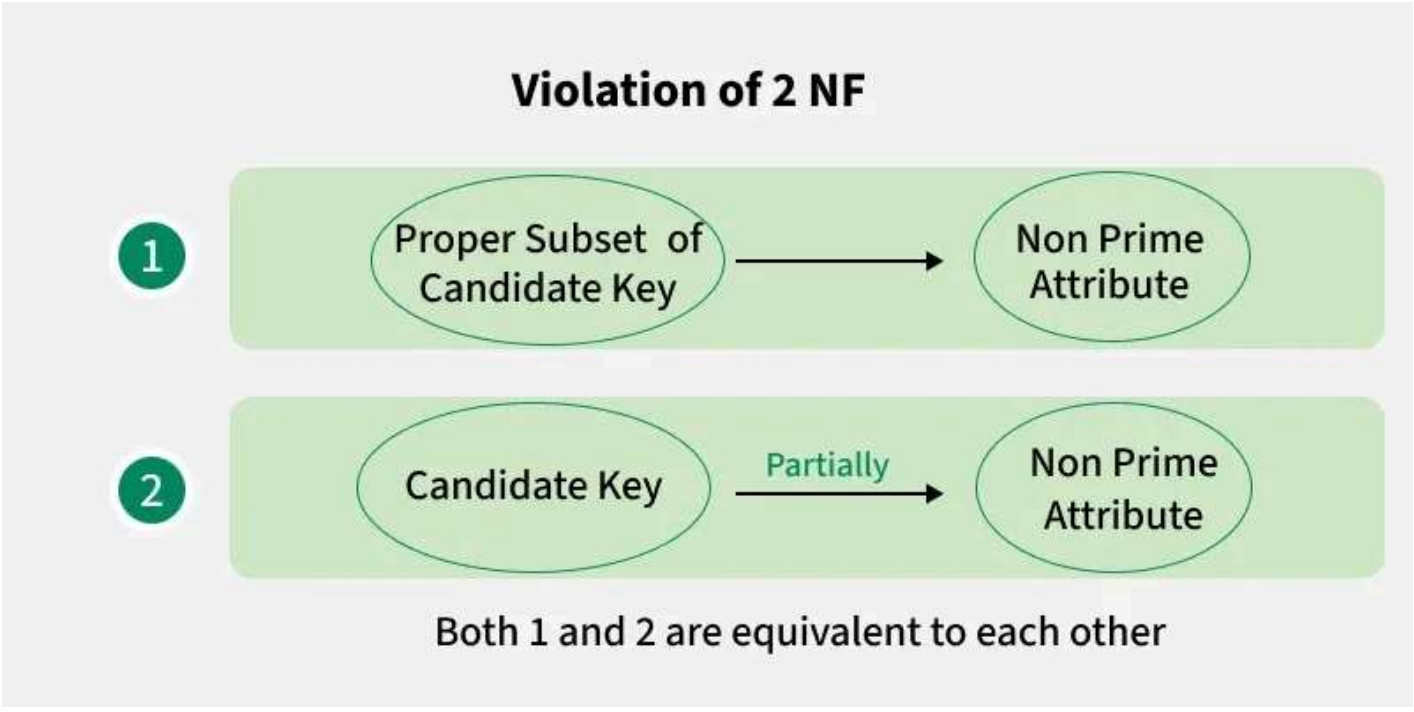
**COURSES Table**

| COURSE_NO | COURSE_FEE |
|---|---|
| C1 | 1000 |
| C2 | 1500 |
| C4 | 2000 |
| C3 | 1000 |
| C5 | 2000 |

### Now, each table is in 2NF:

- The Course Table ensures that `COURSE_FEE` depends only on `COURSE_NO`.
- The Student-Course Table ensures there are no partial dependencies because it only relates students to courses.

Now, the **COURSE_FEE** is no longer repeated in every row, and each table is free from partial dependencies. This makes the database **more efficient** and **easier to maintain**.




*Violation of 2nf*

## Limitations of Second Normal Form (2NF)

While Second Normal Form (2NF) addresses partial dependencies and helps reduce redundancy, it has some limitations:

**1. Doesn't Handle Transitive Dependencies**: 2NF ensures that non-prime attributes are fully dependent on the entire primary key, but it doesn't address transitive dependencies. In a transitive dependency, an attribute depends on another non-key attribute.

For example, if A → B and B → C, then A indirectly determines C. This can lead to further redundancy and anomalies.

**2. Doesn't Ensure Optimization**: Although 2NF eliminates partial dependencies, it may still leave some redundancy in the data, particularly when dealing with larger and more complex datasets. It doesn't guarantee the most efficient or optimized structure for a database.

**3. Complexity in Handling Multi-Attribute Keys**: When dealing with composite primary keys (keys made of multiple attributes), ensuring full dependency can still lead to a complex design. A further step of normalization (Third Normal Form or 3NF) is required to resolve transitive dependencies and achieve better data organization.

**4. Not Sufficient for Some Use Cases**: While 2NF is useful for reducing redundancy in some situations, in real-world applications where data integrity and efficiency are crucial, additional normalization (like 3NF) might be needed to address more complex dependencies and optimize data storage and retrieval.