**What is Normalization?**

**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The Theory of Data Normalization in MySQL server is still being developed further. For example, there are discussions even on 6th Normal Form. **However, in most practical applications, normalization achieves its best in 3rd Normal Form**.

**There are 3 "normal" rules of normalization.**  
       **1**. Every table should have:  
        **1a**. A primary key.  
        **1b**. Rows of equal length.  
        **1c**. Only columns where the content is the same all the way down, and no empty fields along the way. (Not a field for a phone number, that's empty if the guy doesn't have a phone.)  
        **1d**. No two columns with the same kind of data. (Not two fields for phones, in case he has two phone numbers.) In reality, this is the kind of table you soon make automatically, so this is a lot of words to describe something banal.

**2**. Every table should have: No columns, only depending on some of the primary key. (This only applies, if the primary key is composite, and there's columns not in the primary key.)

**3**. Every table should have: No columns not depending on the primary key at all.

The evolution of SQL Normalization theories is illustrated below-

Database Normal Forms

**1st NORMAL FORM**

A database is in first normal form if it satisfies the following conditions:

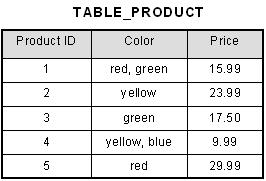
* Contains only atomic values
* There are no repeating groups

An atomic value is a value that cannot be divided. For example, in the table shown below, the values in the [Color] column in the first row can be divided into "red" and "green", hence [TABLE\_PRODUCT] is not in 1NF.

A repeating group means that a table contains two or more columns that are closely related. For example, a table that records data on a book and its author(s) with the following columns: [Book ID], [Author 1], [Author 2], [Author 3] is not in 1NF because [Author 1], [Author 2], and [Author 3] are all repeating the same attribute.

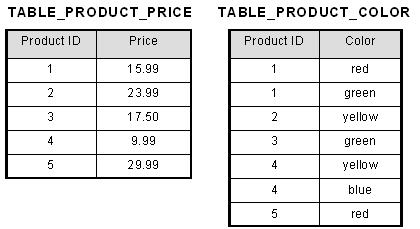
**1st Normal Form Example**

How do we bring an un-normalized table into first normal form? Consider the following example:



This table is not in first normal form because the [Color] column can contain multiple values. For example, the first row includes values "red" and "green."

To bring this table to first normal form, we split the table into two tables and now we have the resulting tables:



Why do we need 1NF?

**1NF** is important because it is much more flexible than 0NF while being much easier to use when inserting, updating and reading data. This is because every type of data element (e.g. customer phone number) has exactly one column in which to find it and that column has only one piece of data for each record.

When can you say that table is in 1NF?

A **table** is in first normal form (**1NF**) if and only if all columns contain only atomic values—that is, each column **can** have only **one** value for each row in the **table**.

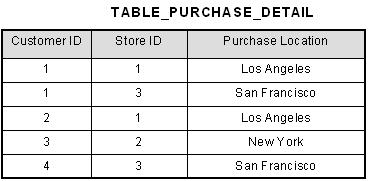
**2nd NORMAL FORM**

A database is in second normal form if it satisfies the following conditions:

* It is in first normal form
* All non-key attributes are fully functional dependent on the primary key

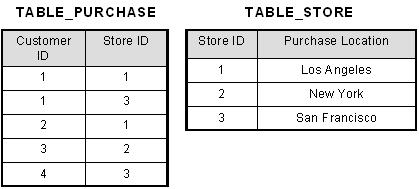
**2nd Normal Form Example**

Consider the following example:



This table has a composite primary key [Customer ID, Store ID]. The non-key attribute is [Purchase Location]. In this case, [Purchase Location] only depends on [Store ID], which is only part of the primary key. Therefore, this table does not satisfy second normal form.

To bring this table to second normal form, we break the table into two tables, and now we have the following:



What we have done is to remove the partial functional dependency that we initially had. Now, in the table [TABLE\_STORE], the column [Purchase Location] is fully dependent on the primary key of that table, which is [Store ID].

Why do we need 2NF?

**2NF** attempts to reduce the amount of redundant data in a table by extracting it, placing it in new table(s) and creating relationships between those tables.

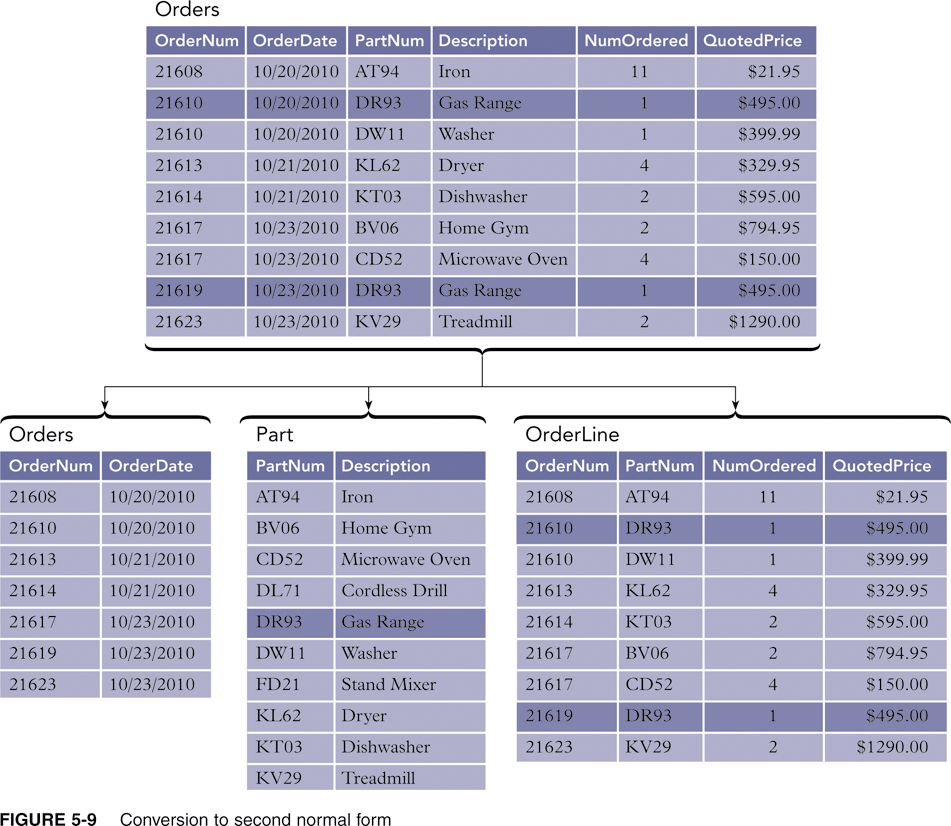
When can you say that table is in 2NF?

A **table is in 2NF**, only if a relation is in 1NF and meet all the rules, and every non-key attribute is fully dependent on primary key.

**Normalization of Tables - From 1NF to 2NF**

There is an algorithm for removing partial dependencies.

1. Form all subsets of the attributes making up the primary key.
2. Begin a new table for each subset, using the subset as the primary key.
3. Now, from the original table, add to each subset the attributes that depend on that subsets primary key.
4. Name each of the new tables appropriately.



**3rd NORMAL FORM**

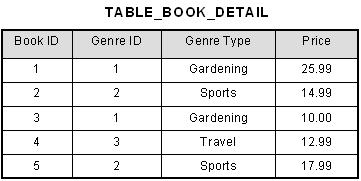
A database is in third normal form if it satisfies the following conditions:

* It is in second normal form
* There is no transitive functional dependency

By transitive functional dependency, we mean we have the following relationships in the table: A is functionally dependent on B, and B is functionally dependent on C. In this case, C is transitively dependent on A via B.

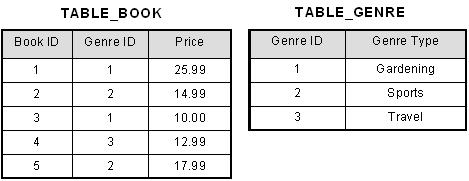
**3rd Normal Form Example**

Consider the following example:



In the table able, [Book ID] determines [Genre ID], and [Genre ID] determines [Genre Type]. Therefore, [Book ID] determines [Genre Type] via [Genre ID] and we have transitive functional dependency, and this structure does not satisfy third normal form.

To bring this table to third normal form, we split the table into two as follows:



Now all non-key attributes are fully functional dependent only on the primary key. In [TABLE\_BOOK], both [Genre ID] and [Price] are only dependent on [Book ID]. In [TABLE\_GENRE], [Genre Type] is only dependent on [Genre ID].

Why do we need 3NF?

Third Normal Form (**3NF**) **is** considered adequate for normal relational database design because most of the **3NF** tables **are** free of insertion, update, and deletion anomalies. Moreover, **3NF** always ensures functional dependency preserving and lossless.

When can you say that table is in 3NF?

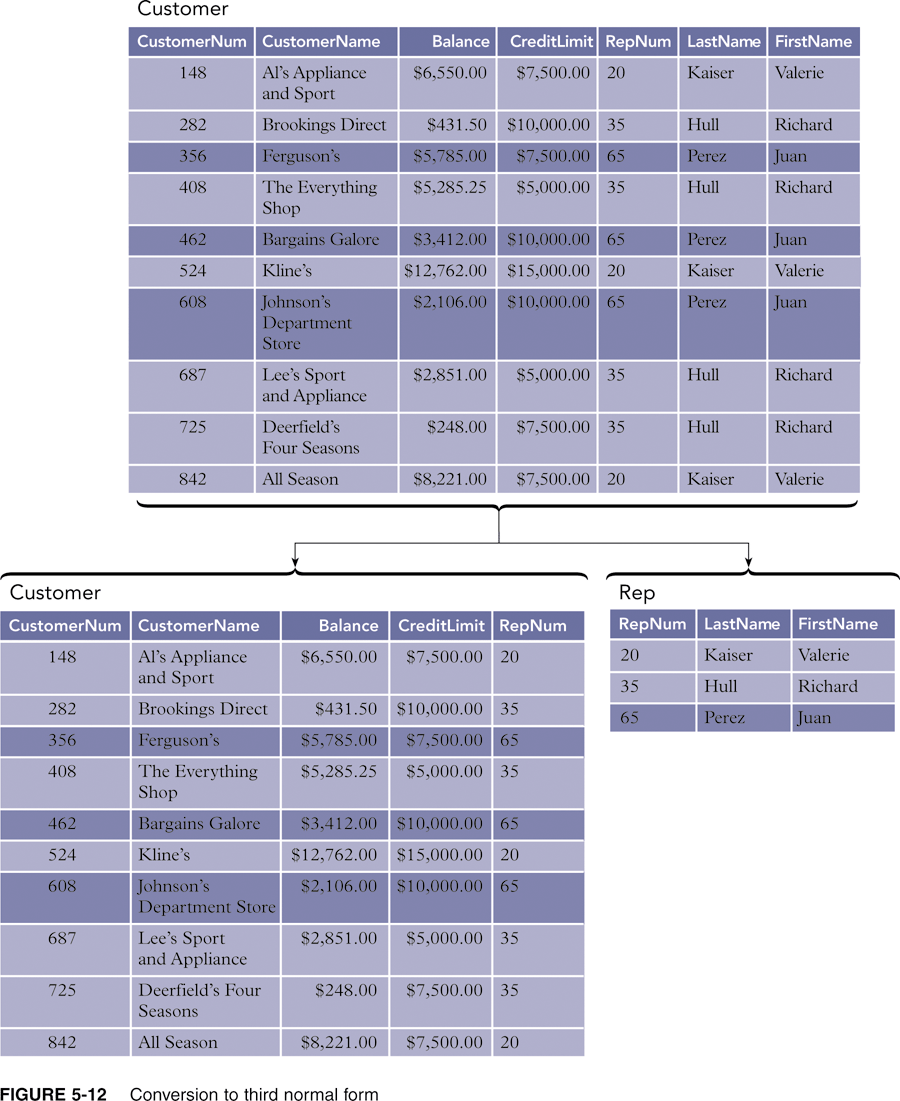
Codd's definition states that a **table is in 3NF** if and only if both of the following conditions hold: The relation R (**table**) is in second normal form (2NF). Every non-prime attribute of R is non-transitively dependent on every key of R.

**Normalization of Tables - From 2NF to 3NF**

To remove transitive dependencies.

1. For each determinant that is not a candidate key\*, remove from the table the columns that depend on this determinant (but don't remove the determinant).
2. Create a new table containing all the columns from the original table that depend on this determinant.
3. Make the determinant the primary key of this new table.
4. Name the new table appropriately.

\* A **candidate key** is any key that could serve as a primary key.



# Boyce Codd normal form (BCNF)

* BCNF is the advance version of 3NF. It is stricter than 3NF.it is also known as 3.5NF
* A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
* For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

EXAMPLE FOR BCNF

|  |  |  |
| --- | --- | --- |
| S.NAME | SUBJECT | TEACHER |
| RAJU | ENGLISH | TEENA |
| RAM | PHYSICS | MEENA |
| SITA | MATHS | RAMESH |
| GEETHA | PHYSICS | MEENA |

To bring this table to third normal form, we split the table into two as follows:

|  |  |
| --- | --- |
| S.NAME | SUBJECT |
| RAJU | ENGLISH |
| RAM | PHYSICS |
| SITA | MATHS |
| GEETHA | PHYSICS |

|  |  |
| --- | --- |
| SUBJECT | TEACHER |
| ENGLISH | TEENA |
| PHYSICS | MEENA |
| MATHS | RAMESH |

Why do we need BCNF?

**BCNF is** a stronger form of normalization than 3NF because it eliminates the second condition for 3NF, which allowed the right side of the FD to be a prime attribute. Thus, every left side of an FD in a table **must** be a super-key.

When can you say that table is in BCNF?

A **table** complies with **BCNF** if it is in 3NF and for every functional dependency X->Y, X **should** be the super key of the **table**.

**4th Normal Form**

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1. It should be in the **Boyce-Codd Normal Form**.
2. And, the table should not have any **Multi-valued Dependency**.

**What is Multi-valued Dependency?**

A table is said to have multi-valued dependency, if the following conditions are true,

1. For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.
2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.
3. And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

Example

Below we have a college enrolment table with columns s\_id, course and hobby.

|  |  |  |
| --- | --- | --- |
| **s\_id** | **course** | **hobby** |
| 1 | Science | Cricket |
| 1 | Maths | Hockey |
| 2 | C# | Cricket |
| 2 | Php | Hockey |

As you can see in the table above, student with s\_id **1** has opted for two courses, **Science** and **Maths**, and has two hobbies, **Cricket** and **Hockey**.

You must be thinking what problem this can lead to, right?

Well the two records for student with s\_id **1**, will give rise to two more records, as shown below, because for one student, two hobbies exists, hence along with both the courses, these hobbies should be specified.

|  |  |  |
| --- | --- | --- |
| **s\_id** | **course** | **hobby** |
| 1 | Science | Cricket |
| 1 | Maths | Hockey |
| 1 | Science | Hockey |
| 1 | Maths | Cricket |

And, in the table above, there is no relationship between the columns course and hobby. They are independent of each other.

So there is multi-value dependency, which leads to un-necessary repetition of data and other anomalies as well.

### How to satisfy 4th Normal Form?

To make the above relation satify the 4th normal form, we can decompose the table into 2 tables.

**Course Table**

|  |  |
| --- | --- |
| **s\_id** | **course** |
| 1 | Science |
| 1 | Maths |
| 2 | C# |
| 2 | Php |

And, **Hobbies Table**,

|  |  |
| --- | --- |
| **s\_id** | **hobby** |
| 1 | Cricket |
| 1 | Hockey |
| 2 | Cricket |
| 2 | Hockey |

Now this relation satisfies the fourth normal form.

A table can also have functional dependency along with multi-valued dependency. In that case, the functionally dependent columns are moved in a separate table and the multi-valued dependent columns are moved to separate tables.

**5th NORMAL FORM**

The 5NF (Fifth Normal Form) is also known as project-join normal form.

A relation is in Fifth Normal Form (5NF),

* It is in 4NF.
* And won’t have lossless decomposition into smaller tables(Join dependency).

You can also consider that a relation is in 5NF, if the candidate key implies every join dependency in it.

**EXAMPLE FOR 5TH NORMAL FORM**

**Example:**Consider the relation **R** below having the schema R(supplier, product, consumer). The primary key is a combination of all three attributes of the relation.

|  |  |  |
| --- | --- | --- |
| supplier | product | consumer |
| S1 | P1 | C1 |
| S1 | P2 | C1 |
| S2 | P1 | C1 |
| S3 | P3 | C3 |

TABLE-2

|  |  |
| --- | --- |
| supplier | product |
| S1 | P1 |
| S1 | P2 |
| S2 | P1 |
| S3 | P3 |

**Table 3**

|  |  |
| --- | --- |
| consumer | product |
| C1 | P1 |
| C1 | P2 |
| C3 | P3 |

**Table 4**

|  |  |
| --- | --- |
| supplier | consumer |
| S1 | C1 |
| S2 | C1 |
| S3 | C3 |

**Explanation:**  
Table 2, Table 3 and Table 4 when joined yield the original table (Table 1). Hence join dependency exists in Table 1, therefore Table 1 is not in 5NF or PJNF. However Table 2, Table 3 and Table 4 satisfy 5NF as it has no multi-valued dependency and cannot be decomposed further (join dependency does not exists). But this might not be true in all cases i.e., when we combine the decomposed tables, the resultant table may not be equivalent to the original table, in that case the original table is said to be in 5NF provided it is already in 4NF. However, 5NF is not applied in practical scenarios and remains limited to theoretical concepts.

**6TH NORMAL FORM**

A relation is in 6NF only if when it doesn’t support any nontrivial join dependencies. Any relation which is in 6NF should also be in 5NF. Though Some authors used the term sixth normal form as a synonym for DKNF.

It is important when we are dealing with the temporal variables