**NORMALIZATION IN DBMS**

**Normalization** is the process of organizing the data in the database. Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization divides the larger table into the smaller table and links them using relationship. The words normalization and normal form refer to the structure of a database.

The normal form is used to reduce redundancy from the database table. Normalization also simplifies the database design so that it achieves the optimal structure composed of atomic elements (i.e. elements that cannot be broken down into smaller parts). By normalizing a database, you arrange the data into [tables](https://database.guide/what-is-a-table/) and [columns.](https://database.guide/what-is-a-column/) You ensure that each table contains only related data. If data is not directly related, you create a new table for that data.

The concept of normalization was first proposed by Edgar F. Codd in 1970, when he proposed the first normal form (1NF) in his paper [*A Relational Model of Data for Large Shared Data Banks*](https://dl.acm.org/citation.cfm?id=362685) (this is the paper in which he introduced the whole idea of relational databases).

Codd continued his work on normalization and defined the second normal form (2NF) and third normal form (3NF) in 1971.Codd then teamed up with Raymond F. Boyce to define the Boyce-Codd normal form (BCNF) in 1974.Ronald Fagin introduced the fourth normal form (4NF) in 1977.Fagin then introduced the fifth normal form (5NF) in 1979.

**Benefits of Normalization**

* Normalization increases clarity in organizing data in Databases.
* Minimizes data redundancy (duplicate data).
* Minimizes null values.
* Results in a more compact database (due to less data redundancy/null values).
* Minimizes/avoids data modification issues.
* The database structure is cleaner and easier to understand.
* Searching, sorting, and creating indexes can be faster, since tables are narrower, and more rows fit on a data page.

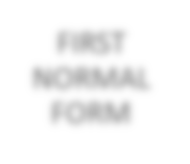
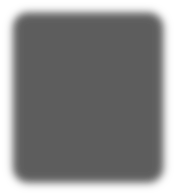
**NEED OF NORMALIZATION**

* To maintain data integrity
* To eliminate redundant data
* To eliminate data duplicacy
* To increase storage efficiency
* To produce a clear and understandable/readable database structure.
* To avoid anomalies(Update/delete/insert)

If database is not normalized, then insertion, updation and deletion anomalies occur. Anomalies are error-prone situation arising when we process the tables. There are three types of anomalies :

1. Insert Anomaly: An insert anomaly occurs when it is not possible to insert certain attributes into the database without the availability of other attributes. i.e We tried to insert data in a record that does not exist at all.
2. Update anomaly : If data items are scattered and are not linked to each other properly, then it could lead to ambiguos situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state. This is called Update anomaly.
3. Delete Anomaly : This anomaly occurs when data of some attributes get lost because of deletion of data of other attributes in the same record.

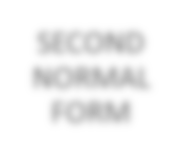
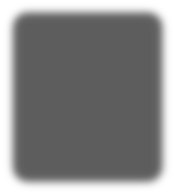
**TYPES OF NORMAL FORM**



FIRST

NORMAL

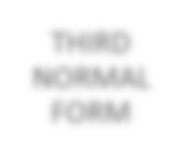
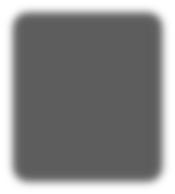
FORM



SECOND

NORMAL

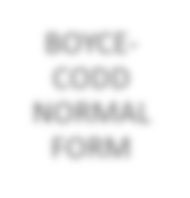
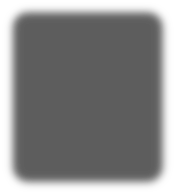
FORM



THIRD

NORMAL

FORM



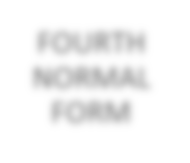
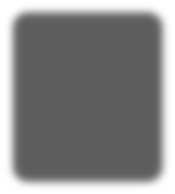
BOYCE

-

CODD

NORMAL

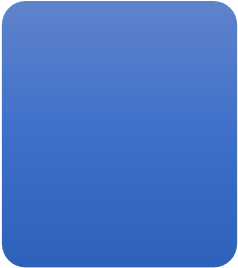
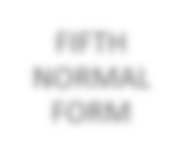
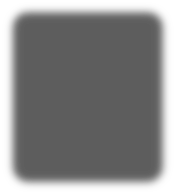
FORM



FOURTH

NORMAL

FORM



FIFTH

NORMAL

FORM

FUNCTIONAL DEPENDENCY :

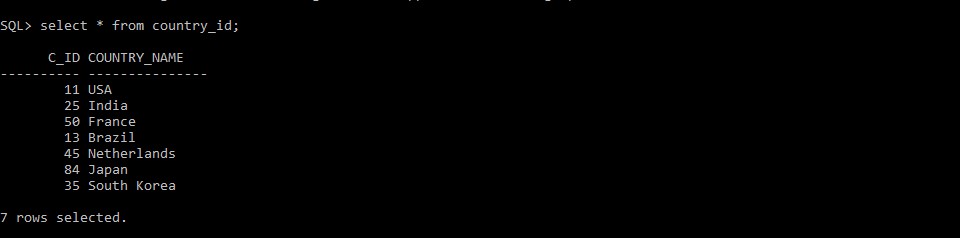
Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign (→) that is, A→B, where A functionally determines B. The left-hand side attributes determine the values of attributes on the right-hand side.

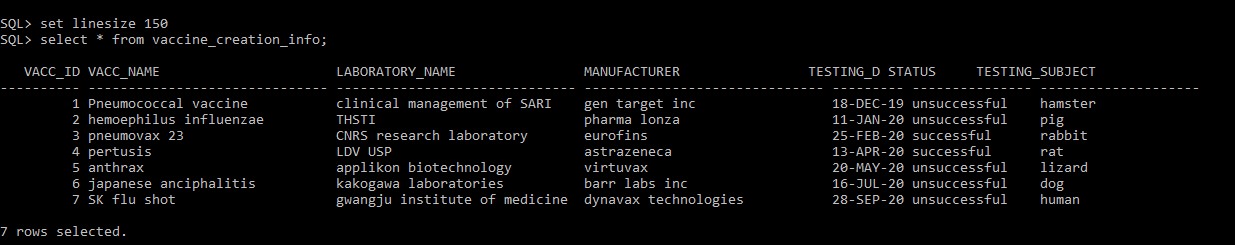
The left-hand side of the FD is called the determinant, and the righthand side is the dependent. If A is the determinant and B is the dependent then we say that B is dependent on A and A functionally determines B.

A functional dependency A->B in a relation holds if two tuples having same value of attribute A also have same value for attribute B.

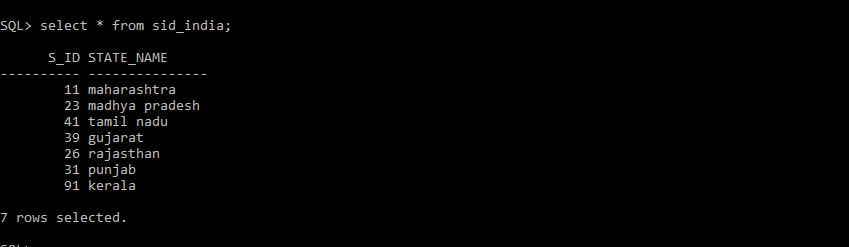
EG.1) In country\_id table country\_name is functionally dependent on c\_id (i.e country id)



EG.2)In vaccine\_creation\_info table all the attributes are functionally dependent on primary key i.e vac\_id



EG.3) In sid\_india table state\_name is functionally dependent on primary key i.e s\_id (state id)



PARTIAL DEPENDENCY :

Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key. i.e B is functionally dependent on A, and A is the part of multipart candidate key.

Eg. If there is a table with following attributes :

|  |  |  |  |
| --- | --- | --- | --- |
| C\_id | country\_name | Vac\_id | Vacc\_name |

In this case, country\_name depends on c\_id and vacc\_name depends on vac\_id. That means there exist a partial dependency in this table.

To remove this partial dependency we split the table as :

|  |  |  |
| --- | --- | --- |
| c\_id | Country\_name | Vac\_id |

|  |  |
| --- | --- |
| vacc\_id | vacc\_name |

TRANSITIVE DEPENDENCY :

Consider attributes A, B, C.

If A functionally determines B (A->B) and B functionally Determines C (B->C) then there exist a transitive dependency i.e C is transitively dependent on A through B (A->C).

**FIRST NORMAL FORM (1NF) :**

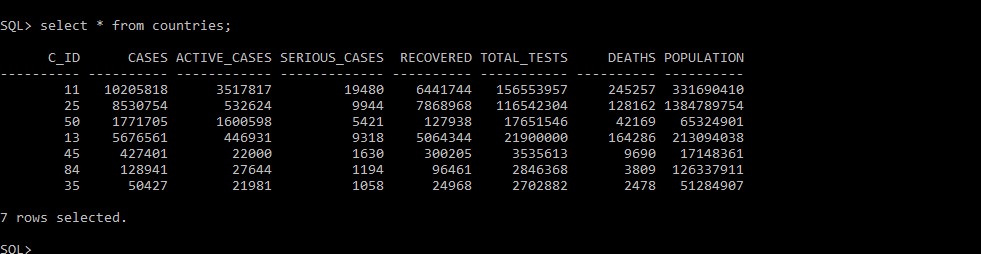
According to first normal form, all the attributes in a relation must have atomic domains.

A domain is the set of all unique values which is permitted for an attribute. Atomic means that cannot be divided further.

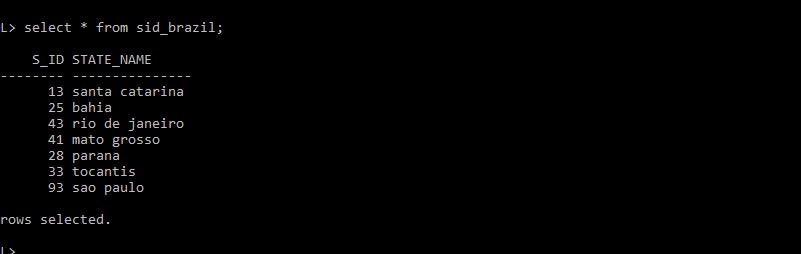
RULES FOR FIRST NF:

* A relation will be in 1NF if it contain atomic values.
* an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
* First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

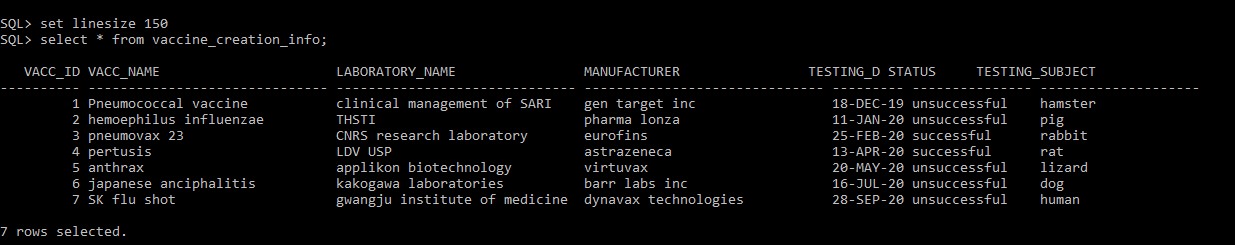
EG.1) In countries table all the attributes are singlevalued and have atomic values



EG.2)In sid\_brazil table all the attributes are singlevalued and have atomic values



EG.3)In vaccine\_creation\_info table all the attributes are single-valued and have atomic values



Similarly, all the tables in our database are single-valued and have atomic values.

**SECOND NORMAL FORM (2NF) :**

For a relation to be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency, i.e all the non-prime attributes must be dependent on prime attributes (primary key).

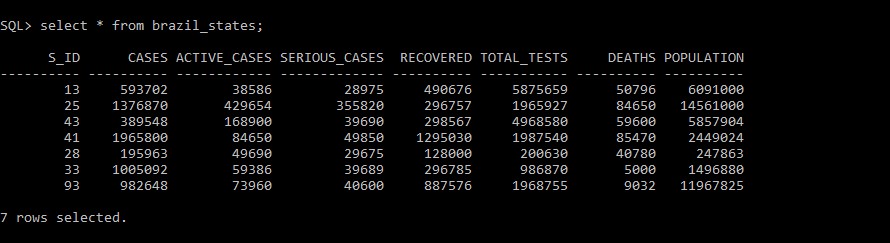
* **Prime attribute** − An attribute, which is a part of the prime-key, is known as a prime attribute.
* **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

RULES FOR SECOND NF:

* In the 2NF, relational must be in 1NF.
* In the second normal form, all non-key attributes are fully functional dependent on the primary key ➢ Partial dependency should not exist.

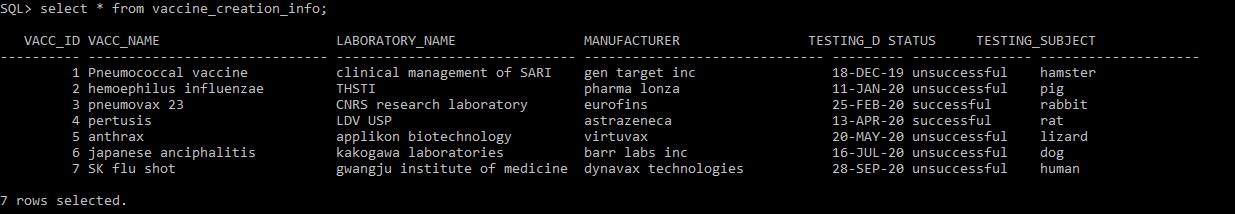
EG.1) According to second NF, all the non-prime attributes must be dependent on prime-key attributes.

In the following table non-prime attributes like cases, active\_cases, serious\_cases, recovered, total\_tests, deaths , population are dependent prime-key attribute i.e s\_id.



✓ But in case of weak entity non-prime attributes are dependent on foreign key

EG.2) All the non-prime attributes like vacc\_name,laboratory\_name, manufacturer, testing\_date, status, testing\_subject are dependent on primary key i.e vacc\_id



Similrly, all the tables in the database are in second NF without any partial dependency.

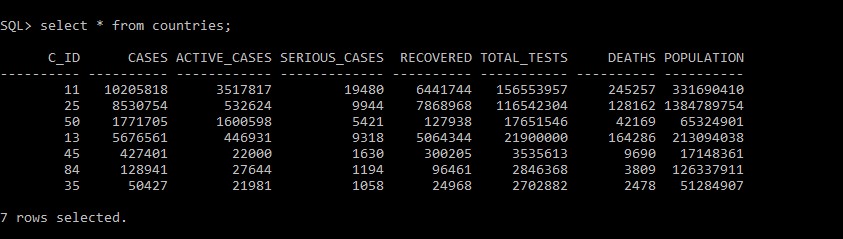
**THIRD NORMAL FORM (3NF) :**

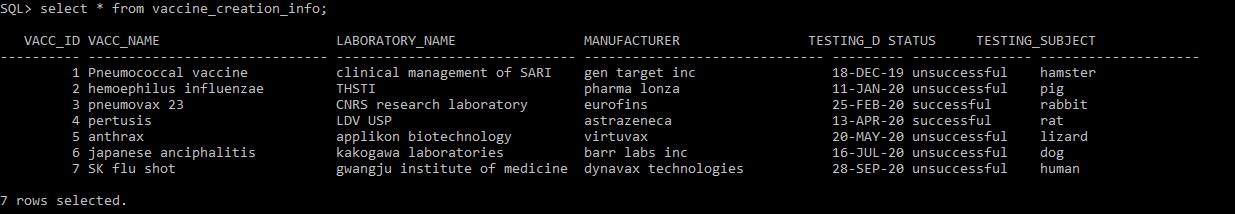
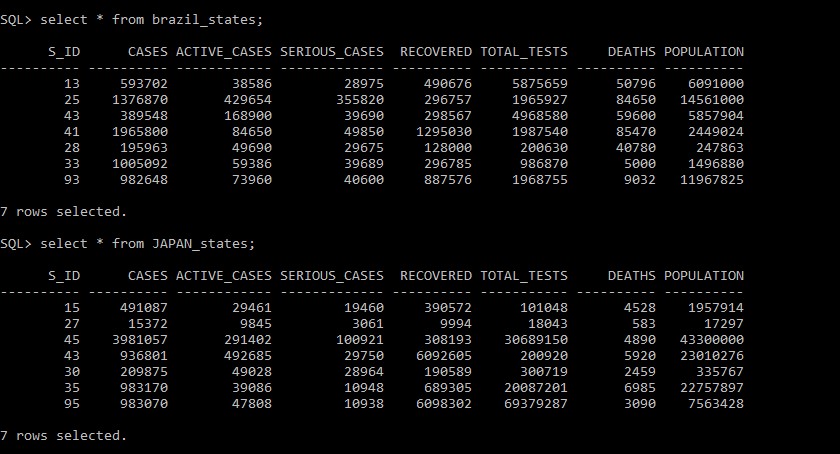
A relation is in third normal form, if there is **no transitive dependency** for non-prime attributes as well as it is in second normal form. 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

RULES FOR THIRD NF:

* A relation will be in 3NF if it is in 2NF.
* It should not contain any transitive dependency.

EG.1) The following table is in second NF as well as there is no transitive dependency. i.e the table is in 3NF.





✓ MANY-TO-MANY RELATIONSHIP :

This is a very common situation , it occurs when the table has only foreign-keys. If we add further columns in future there will be no problem due to the existing columns all being foreign keys.

EG.) The following table consist of only foreign-keys. Therefore, there exist a many-to-many relationship.

