**SQL Normalization**

# Normalization

Normalization is the process of designing a database effectively such that we can avoid

data redundancy [data duplication], in turn which will help us to avoid anomalies such

insertion, updation and deletion anomalies.

# Different Levels

1. 1NF – First Normal Form
2. 2NF – Second Normal Form
3. 3NF – Third Normal Form
4. 4NF – Fourth Normal Form
5. BCNF – Boyce-Codd Normal Form
6. 5NF – Fifth Normal Form
7. 6NF – Sixth Normal Form

Though we have 6 Normal Form, the golden standard of normalization is 3rd Normal Form.

Most of the companies try to normalize their data till 3rd Normal Form.

Based on the need, the company / project can go beyond the third Normal Form.

# 1NF Rule

1. Every column / attribute needs to have a single value.
2. Each row should be unique. Either through a single or multiple columns. Not mandatory to have primary key.

# 2NF Rule

1. Must be in 1NF
2. All Non-Key attributes must be fully dependent on candidate key. i.e., If a non-key column is partially dependent on candidate key (subset of columns forming candidate key) then split them into separate tables.
3. Every table should have primary key and relationship between the tables should be formed using foreign key.

# Candidate Key

1. Set of columns which uniquely identify a record.
2. A table can have multiple candidate keys because there can be multiple set of columns which uniquely identify a record / row in a table.

# Non-Key Columns

Columns which are not part of the candidate key or primary key.

# Partial Dependency

If your candidate key is a combination of 2 columns (or multiple columns) then every non-key column (columns which are not part of the candidate key) should be fully dependent on all the columns. If there is any non-key column which depends only on one of the candidate key columns then this results in partial dependency.

# 3NF Rule

1. Must be in 2NF.
2. Avoid Transitive Dependencies.

# Transitive Dependency

Let say you have a Table T which has 3 columns such as A, B, and C. If A is functionally dependent on B and B is functionally dependent on C then we can say that A is functionally dependent on C.

# Insertion Anomalies

## Scenarios

### 1. Data Redundancy

In a denormalized dataset, let say we have a customer who purchased 100 different products and those products have different order id and it has 100 records with the same customer details as duplicate information.

But whereas in case of normalized dataset [3NF], we have only one record for the customer detail.

### 2. Missing Data

In a denormalized dataset, let say we have a launched a new product and the detail is updated in database, but only the product related fields are updated as it's a new product and no customer have purchased it. So, order and customer related fields are null.

But whereas in case of normalized dataset [3NF], we have separate table for products and there we

can add any new products.

# Deletion Anomalies

In a denormalized dataset, if we want to delete the order which has been added wrongly. It will delete the order details but along with that, it will also delete the product and customer details as well because we have those data in the record to be deleted.

But whereas in case of normalized dataset [3NF], we can delete the order without affecting the product and customer detail.

# Update Anomalies

In a denormalized dataset, if we want to update the product price of motorcycle from 200 to 250. We have to update more than one record which is not recommended and in real time, it could be 1000 records.

But whereas in case of normalized dataset [3NF], we can update the price of the product in one place under product table.