**Normalization:**

1. 1-NF
2. 2-NF
3. 3-NF

BCNF (Boyce Code Normal Form)

1. 4-NF
2. 5-NF

**Normalization** is the process of creating a structured database with multiple tables to free the database from *insertion, update, and deletion anomalies*, and at the same time take care of *data integrity and consistency*.



It is important to note that while structuring a database into multiple tables helps avoid data redundancy/duplication, this approach also has a drawback. To extract data from multiple tables at the same time we will have to use SQL-joins, and using joins will downgrade the performance of SELECT statements. So, we must be very cautious while dividing the database into multiple tables. Split the tables only to the extent that is necessary. Different normalizations forms like 1-NF, 2-NF etc. provide guidelines for how to divide a database into multiple tables and at the same time take care of data integrity and consistency.



Third Normal Form (3-NF) is considered adequate for normal relational database design.

With 3-NF most of the database tables are free of insertion, update, and deletion anomalies.

1NF Requirements: 1NF focuses on Atomicity and identifying/creating a Primary key.

1. Atomicity: Each cell in a table must contain only a single value, not a list or a group of values.
2. Unique Rows: Each row in a table should be unique and identifiable by a primary key. In other words “eliminate repeating groups.”

2NF Requirements: After 1NF, in 2NF the focus is on removing partial functional dependencies in the table.

1. 1NF requirements are met, and
2. Remove Partial Functional dependencies if any.

[Partial Functional Dependency: All non-prime attributes must fully depend on the entire primary key, not just a part of it. – This arises only when there is a composite primary key]

3NF Requirements: After 1NF and 2 NF, in 3NF the focus is on removing transitive functional dependencies in the table.

1. 2NF requirements are met, and
2. Remove Transitive Functional Dependency:

[Transitive Dependency: A non-prime attribute (not part of the primary key) depends on another non-prime attribute, **indirectly** through the primary key. In other words, Transitive Dependency is Indirect relationship between attributes of an entity. This creates redundancy and potential update anomalies.]

**NOTE:** In any of the normal forms when splitting a table maintain referential integrity (create foreign key) and take care of the cardinality of the relationship. That is to decide whether 1-to-1, 1-to-M or M-to-M relation has to be created.

**More details about 1NF:**

1. Each cell in a table must contain only a single value, not a list or a group of values.

Look for ***multi-valued attributes***. If a multi-valued attribute is found, create a separate table for this attribute, create or identify a primary key for the new table, and maintain referential integrity with parent table. While maintaining referential integrity take care of the cardinality of the relationship.

For Eg:

1. Employee Qualification would be a multi-valued attribute
2. Employee Address, apart from being a composite attribute, it could also be a multi-valued attribute if each employees Permanent Address and Current Address is being maintained.

**Important activity:** Create multiple tables to eliminate repeating groups.

Look for ***composite attributes***. If a composite attribute is found, spread it across multiple columns of the same table.

For eg:

* 1. Employee Name would be an attribute that is composed of First Name and Middle Name. Split the ENAME field into two columns EMP\_FNAME and EMP\_LNAME.
  2. Address would be an attribute that is composed of address line-1, address line-2, city, state, zipcode and country. Have a separate column in the table for each part of the address.

1. Unique Rows: Each row in a table should be unique and identifiable by a primary key.

