Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

- An **Insert Anomaly** occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

- A **Delete Anomaly** exists when certain attributes are lost because of the deletion of other attributes.

- An **Update Anomaly** exists when one or more instances of duplicated data is updated, but not all.

Now we will talk about **Normalization techniques** using an example:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

First Normal Form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only **atomic** values.

- Eliminate repeating columns in each table „
- Create a separate table for each set of related data „
- Identify each set of related data with a primary key „
- All attributes are single valued & non-repeating

Student Table :

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

Student	Age	Subject
Adam	15	Biology
Adam	15	Maths
Alex	14	Maths
Stuart	17	Maths

Second Normal Form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

In the above Table in First Normal Form, while the **candidate key is {Student, Subject}**, Age of Student only depends on Student column, which is incorrect as per Second Normal Form.

To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

Student	Age
Adam	15
Alex	14
Stuart	17

In Student Table the candidate key will be Student column, because all other column i.e. Age is dependent on it.

New Subject Table introduced for 2NF will be :

Student	Subject
Adam	Biology
Adam	Maths
Alex	Maths
Stuart	Maths

In Subject Table the candidate key will be {Student, Subject} column. Now, both the above tables qualify for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- **Transitive functional dependency** of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: *{emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on*
Candidate Keys: *{emp_id}*
Non-prime attributes: all attributes except *emp_id* are non-prime as they are not part of any candidate keys.

Here, *emp_state, emp_city & emp_district* dependent on *emp_zip*. And, *emp_zip* is dependent on emp_id that makes non-prime attributes (*emp_state, emp_city & emp_district*) transitively dependent on super key (*emp_id*).

This violates the rule of 3NF.

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Boyce Codd normal form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. 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.

Example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: *{emp_id, emp_dept}*

The table is not in BCNF as neither *emp_id* nor *emp_dept* alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Candidate keys:
For first table: *emp_id*
For second table: *emp_dept*
For third table: *{emp_id, emp_dept}*

This is now in BCNF as in both the functional dependencies left side part is a key.