**Database Design using Normalization:-**

The Database Normalization is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminating data redundancy and Insertion, Update and Deletion Anomalies.

Whenever we are using normalization then it is a multiple method process that puts in data into tabular form removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

1. Eliminating redundant or useless data.
2. Ensuring data dependencies make sense that is data is logically stored.

## Normalization Rule

Normalization rules are divided into the following normalization forms:

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

### 1. First Normal Form (1NF):-

The 1NF specifies that a column of a table cannot hold multiple values for a record. It should hold only atomic values.

The first normalization form should confirm to the following four rules:

1. The table should have only single valued attributes or columns
2. The atomic values should be stored in a column of the same domain
3. All column in a table should have unique names
4. The order in which data is stored doesn’t matter.

**First Normal Form of example:-**

**Example**: The example of 1NF supposes if wants to store the company details like names and contact details of its employees.

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp Id** | **Emp Name** | **Emp Address** | **Emp Phone No.** |
| 701 | Lalit | Pune | 7387380277 |
| 702 | Smith | US | 9955646199  7865663666 |
| 703 | Allen | Hyderabad | 9854785999 |
| 704 | Ford | Bangalore | 9955646200  8954558632 |

The two employees, Smith and Ford have two mobile numbers each, so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF**as each attribute of a table must have a single value. The Emp Phone No. attribute violates the rule for Smith and Ford.

The following version of the same table complies to the 1NF rule:

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp Id** | **Emp Name** | **Emp Address** | **Emp Phone No.** |
| 701 | Lalit | Pune | 7387380277 |
| 702 | Smith | US | 9955646199 |
| 702 | Smith | US | 7865663666 |
| 703 | Allen | Hyderabad | 9854785999 |
| 704 | Ford | Bangalore | 9955646200 |
| 704 | Ford | Bangalore | 8954558632 |

### Second Normalization Form (2NF)

The second Normalization form specifies that a table should hole the following conditions:

* The 2NF it should be in the first normal form and it should not have **partial dependency**.

### Example of Second Normal Form (2NF):-

**In** a training institute database, a table stores the data of teachers and the subjects they teach. Since a teacher can teach more than one subjects the table can have multiple rows for a same teacher.

|  |  |  |
| --- | --- | --- |
| Teacher Id | Subject | Teacher Age |
| 701 | Oracle | 38 |
| 701 | Java | 38 |
| 702 | DevOps | 38 |
| 703 | PHP | 40 |
| 703 | AWS | 40 |

**Candidate Keys:** Teacher id, subject **Non prime attribute:** Teacher age

The table is in 1 NF because each attribute has single values. But it is not in 2NF because the non prime attribute i.e. teacher age is dependent on teacher id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule specifies that, no non-prime attribute be dependent on the proper subset of any candidate key of the table.

Following tables conform to the 2NF rules:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Teacher details table**   |  |  | | --- | --- | | Teacher Id | Teacher Age | | 701 | 38 | | 702 | 38 | | 703 | 40 | | **Teacher subject table**   |  |  | | --- | --- | | Teacher Id | Subject | | 701 | Oracle | | 701 | Java | | 702 | DevOps | | 703 | PHP | | 703 | AWS | |

### Third Normal Form (3NF):-

A table is in 3NF if it is in 2NF and for each functional dependency X to Y at least one of the following conditions hold:

* X is a [super key](https://beginnersbook.com/2015/04/super-key-in-dbms/) of table
* Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

**Example:**

A company wants to store the complete address of each employee in a table named as employee details.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Emp Id | Emp Name | Emp Zip | Emp State | Emp City | Emp District |
| 1111 | Lalit | 228899 | MH | Pune | Pune |
| 2222 | Smith | 204050 | UK | UK | Jalana |
| 3333 | Smith | 200017 | UP | Parola | Jalgaon |
| 4444 | Allen | 227852 | GUJ | Valsad | Surat |
| 5555 | Ford | 275613 | RTH | Tamaswadi | Dhule |

* **The attributes** except emp id remaining are non prime as they are not part of any candidate keys.
* emp state, emp city & emp district dependent on emp zip
* 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.

**Employee Table**

|  |  |  |
| --- | --- | --- |
| Emp Id | Emp Name | Emp Zip |
| 1111 | Lalit | 228899 |
| 2222 | Smith | 204050 |
| 3333 | Smith | 200017 |
| 4444 | Allen | 227852 |
| 5555 | Ford | 275613 |

**Employee zip table**

|  |  |  |  |
| --- | --- | --- | --- |
| Emp Zip | Emp State | Emp City | Emp District |
| 228899 | MH | Pune | Pune |
| 204050 | UK | UK | Jalana |
| 200017 | UP | Parola | Jalgaon |
| 227852 | GUJ | Valsad | Surat |
| 275613 | RTH | Tamaswadi | Dhule |

### Boyce and Codd Normal Form (BCNF):-

The Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by third normal form.

BCNF has these rules:

* R must be in 3rd Normal Form
* and, for each functional dependency ( X → Y )
* X should be a super Key.

**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 |

**Functional dependencies**:  
emp\_id -> emp\_nationality  
emp\_dept -> {dept\_type, dept\_no\_of\_emp}