Normalization

Database normalization is a method of organizing data in a database to avoid duplication and ensure it is stored logically.

Normalization means organizing a big table into smaller, related tables to keep data clear and tidy. The words normalization and normal form refer to the structure of a database. Normalization increases the clarity in organizing data in Database.

Normalization of a Database is achieved by following a set of rules called ‘forms’ in creating the database.

* What is Normalization ?

Normalization is a process of organizing the data in database to avoid **data redundancy, insertion anomaly, update anomaly and deletion anomaly**. Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data. It divides larger tables to smaller tables and links them using relationships.

Normalization means making a database simpler by dividing it into smaller, neat parts.

* Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized.

1. Insertion Anomaly

2. Update Anomaly

3. Deletion Anomaly

Let us assume we have Employee table as given below.

|  |  |  |  |
| --- | --- | --- | --- |
| Employee ID | Employee name | Address | Department |
| 100 | Rusty | Nagpur | 12 |
| 101 | Daisy | Mumbai | 14 |
| 102 | Rani | Pune | 15 |
| 103 | Tom | Nashik | 12 |
| 104 | Joe | Amravati | 14 |

* Update anomaly:

Update anomaly is something when I am trying to update some records in table, and that update is causing data inconsistency.

For example : in the above table I have two records for **Employee Id 100** as he belongs to two departments of the company. If i want to update the address of **Joe** then I have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rock would be having two different addresses, which is not correct and would lead to inconsistent data.

* Insert anomaly:

Insert anomaly is something when I am not able to insert data into tables due to some constraints. Suppose a new employee joins the company, who is under training and currently not assigned to any department then I would not be able to insert the data into the table if Department field doesn’t allow nulls.

* Delete anomaly:

Delete anomaly is something when i delete some data from the table, and due to that delete operation we loss some other useful data.

For example,

if at a point of time the company closes the department 103 then deleting the rows that are having Department as 103 would also delete the information of employee Peter since she is assigned only to this department. Normalization is a method to remove all these anomalies and bring the database to a consistent state.

We have below normal forms which are used to eliminate or reduce redundancy in database tables.

1. First normal form(1NF)

2. Second normal form(2NF)

3. Third normal form(3NF)

4. Boyce-Codd normal form (BCNF)

* *Advantages of Normalization in DBMS*
* *Normalization reduces data redundancy.*
* *Improved database organization in general*
* *Improved data inconsistency*
* *Greater flexibility in database design*
* *Maintains relational integrity*
* *Disadvantages of normalization in DBMS*
* *Before creating the database, you must first determine the demands of the user.*
* *When the relationships are normalized to higher normal forms, the performance degrades.*
* *Careless data decomposition may lead to data loss.*
* *It is a time-consuming process to decompose into higher normalization forms.*

*Link :* [*https://er.yuvayana.org/normalization-in-dbms/*](https://er.yuvayana.org/normalization-in-dbms/)

* First normal form (1NF) :

1 NF (First normal form ) Rules

• Each table cell should contain a single value.

• Each record needs to be unique.

Example :

Sample Employee table, it displays employees are working with multiple departments.

|  |  |  |
| --- | --- | --- |
| Employee | Age | Department |
| Vinisha | 32 | Marketing, Sales |
| Rajaswi | 45 | Quality Assurance |
| Vanashree | 46 | Human Resource |

Employee table following 1NF:

|  |  |  |
| --- | --- | --- |
| Employee | Age | Department |
| Vinisha | 32 | Marketing |
| Vinisha | 32 | Sales |
| Rajaswi | 45 | Quality Assurance |
| Vanashree | 46 | Human Resource |

* Second normal form (2NF) :

1. Table is in 1NF

2. It has no Partial Dependency, i.e., no non-prime attribute is dependent on any proper subset of any candidate key of the table. First we will understand what are Prime and Non-prime attributes.

Prime attribute − An attribute, which is a part of the candidate key, is known as a prime attribute. Non-prime attribute − An attribute, which is not a part of the candidate key, is said to be a non-prime attribute. For example, we have following table which is having employee data.

|  |  |  |  |
| --- | --- | --- | --- |
| Employee Id | Department ID | Employee Name | Department |
| 100 | 12 | Rusty | IT |
| 101 | 14 | Daisy | Finance |
| 100 | 14 | Tom | IT |
| 104 | 12 | Joe | Admin |

Above table is in 1NF as all columns are having atomic values. Here Employee Id and Department Id are the prime attributes. As per 2NF rule Employee Name and Department Name must be dependent upon both prime attributes, but here Employee name can be identified by Employee Id and Department Name can be identified by Department Id alone. So here partial dependency exists. To make this relation in 2NF we have to break above

table as:

|  |  |  |
| --- | --- | --- |
| Employee ID | Employee Name | Department ID |
| 100 | Rusty | 12 |
| 101 | Daisy | 14 |
| 100 | Rusty | 12 |
| 104 | Joe | 14 |

|  |  |
| --- | --- |
| Department ID | Department Name |
| 12 | IT |
| 14 | Finance |
| 14 | IT |
| 12 | Admin |

Now there is no partial dependency exist.

* Third Normal Form (3NF) :

For a relation to be in Third Normal Form it must satisfy the following − 1. It must be in Second Normal form 2. No non-prime attribute is transitively dependent on prime key attribute.

For example,

we have below table for storing employee data :

|  |  |  |  |
| --- | --- | --- | --- |
| Employee ID | Employee Name | City | ZIP |
| 100 | Rusty | Nagpur | 110091 |
| 104 | Joe | Amravati | 560108 |

In above relation Employee Id is the only **prime key attribute**. Now If we see City can be identified by **Employee Id** as well as ZIP. ZIP is not a prime attribute, and also it is not a super key. So we hold below 2 relationships here**. Employee Id -> ZIP** (ZIP can be identified by Employee Id) ZIP -> City (City can be identified by ZIP) Therefore, below transitive dependency is true for above relation. Employee Id -> ZIP -> City To convert this relation into 3NF we wil break this into 2 relations as:

|  |  |  |
| --- | --- | --- |
| Employee ID | Employee Name | ZIP |
| 100 | Rusty | 110091 |
| 104 | Joe | 560108 |

|  |  |
| --- | --- |
| ZIP | City |
| 110091 | Nagpur |
| 560108 | Amravati |

* Boyce and Codd Normal Form (BCNF)

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 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

For a table to be in BCNF, following conditions must be satisfied:

• R must be in 3rd Normal Form

• and, for each functional dependency ( X → Y ), X should be a super Key.