**Normalization in SQL**

When the table is getting decomposed into two smaller table to normalize that is present on the database, well every table in the database has to be in the normal form so normalization is used mainly for two purpose so the first one is used to eliminate repeated having repeated data in the system not only makes the process flow but will cause trouble during the later part of the transaction.

Then next one ensures that the data dependencies make some logical sense & it usually stored the data in database with certain logic huge data sets without any purpose are completely waste.

Its like having an abundant resource without any application the data that we have should make some logical sense normalization came into existence because of the problems that occurred on data .

* Here we are covering the points are as below:-
* What is Normalization?
* Types of Normalization
* Functional Dependency
* Advantages of Normalization

**What is Normalization ?**

* Normalization is method or technique to remove redundancy or duplicacy from table or relation and ensure that data is stored logically.
* It process to organize the data and the attributes of a database.
* Normalization is a systematic approach of decomposition undesirable characteristics like

Insertion, Update & Delete.

* It is a multi-step process which is done by dividing the large tables into smaller ones and defining relationships among them.

**Types of Normalization**

Normalization has an different stages to perform a transaction is called Normal forms. Here are some important types of normal forms.

1. 1NF :- Removes repeating groups from the table & contains an atomic value.
2. 2NF :- It has to be in 1st normal form & it’s table should not contain partial dependency.
3. 3NF :- It has to be in 2nd normal form & it should be no transitive dependency for non-prime attributes
4. BCNF :- It has to be in 3rd normal form & in this every functional dependency that is A implies B then A has to be the super key.

**NORMAL FORMS**

**1NF**

**BCNF**

**3NF**

**2NF**

* 1. **1NF (First Normal Form) :-** In 1NF relation each table cell should be contain a single value & each record looks like unique.

|  |  |  |
| --- | --- | --- |
| **Player\_ID** | **Player\_name** | **Game** |
| 01 | Sham Shah | kabaddi, Hockey |
| 02 | Ritik Mishra | Football |
| 03 | Virat kohli | Cricket |
| 04 | Aaradhya Yadav | basketball, Golf |
| 05 | Soniya Dabir | Chess, Shooting |

Here in Game row we stored a games so it is multi-valued attribute. It is not 1NF relation.

So, we converted it into a 1NF form :-

|  |  |  |
| --- | --- | --- |
| **Player\_ID** | **Player\_name** | **Game** |
| 01 | Sham Shah | Kabaddi |
| 01 | Sham Shah | Hockey |
| 02 | Ritik Mishra | Football |
| 03 | Virat kohli | Cricket |
| 04 | Aaradhya Yadav | Basketball |
| 04 | Aaradhya Yadav | Golf |
| 05 | Soniya Dabir | Chess |
| 05 | Soniya Dabir | Shooting |

Its simple method to store games separated in 1NF. Now this is first normal form which stores unique information in table without data repeatition.

**b) 2NF( second normal form):-**  In 2NF , all non-key attributes are fully functionally dependant on the primary key.

Example : let’s assume a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

|  |  |  |
| --- | --- | --- |
| **Teacher\_Id** | **Subject** | **Teacher\_age** |
| 1025 | Chemistry | 30 |
| 1025 | Biology | 30 |
| 1047 | English | 35 |
| 1083 | Maths | 38 |
| 1083 | Computer | 38 |

In the given table, non-prime attribute Teacher\_age is dependent on Teacher\_Id which is a proper subset of a candidate key. Thas’s why it violates the rule for 2NF.

|  |  |
| --- | --- |
| **Teacher\_Id** | **Teacher\_age** |
| 1025 | 30 |
| 1047 | 35 |
| 1083 | 38 |

|  |  |
| --- | --- |
| **Teacher\_Id** | **Subject** |
| 1025 | Chemistry |
| 1025 | Biology |
| 1047 | English |
| 1083 | Hindi |
| 1083 | Computer |

The following two tables are satisfy the conditions of 2NF relation. It also in 1NF form and every non-prime attribute is dependent on primary key.

**c) 3NF (Third normal form) :-** A relation will be in 3NF if it is in 2NF and not contain any transaction dependency exist. Non-prime attribute is dependent on the primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student\_Id** | **Student\_name** | **Zip code** | **City** |
| 501 | Shreya | 422602 | Pune |
| 501 | Shreya | 411000 | Kolhapur |
| 502 | Manvi | 400001 | Kalyan |
| 503 | Krishika | 400099 | Mumbai |

Its not 3NF because student\_Id > City occurs transitive dependency.

Zip code is not a super key & city is not prime attribute.

|  |  |  |
| --- | --- | --- |
| **Student\_Id** | **Student\_name** | **Zip code** |
| 501 | Shreya | 422602 |
| 501 | Shreya | 411000 |
| 502 | Manvi | 400001 |
| 503 | Krishika | 400099 |

**<StudentLocation>**

|  |  |
| --- | --- |
| **Zip code** | **City** |
| 422602 | Pune |
| 411000 | Kolhapur |
| 400001 | Kalyan |
| 400099 | Mumbai |

We converted the table into 3NF by converting it into two parts & they don’t have transitive dependency.

Some dependencies cause redundancy in database.

Thus, redundancy removed by BCNF.

**d) BCNF (Boyee-codd normal form):-** boyee-codd normal form is next part of 3NF

* Table must be in 3NF.
* Table in BCNF has every function dependency X 🡪 Y then X is a super key in table.

|  |  |  |
| --- | --- | --- |
| **Employee code** | **Project Id** | **Pro\_Leader** |
| 101 | F03 | Simran |
| 101 | F01 | Soniya |
| 102 | F04 | Rohan |
| 103 | F02 | Anil |

This table in 3NF form but not in BCNF.

So, for non trival function dependency, project leader 🡪Project Id then Project Id is a prime attribute but project leader is not prime attribute.

For BCNF convert table into Three parts.

|  |  |
| --- | --- |
| **Employee code** | **Project Id** |
| 101 | F03 |
| 101 | F01 |
| 102 | F04 |
| 103 | F02 |

|  |  |
| --- | --- |
| **Pro\_Leader** | **Project Id** |
| Simran | F03 |
| Soniya | F01 |
| Rohan | F04 |
| Anil | F02 |

Thus, we converted tables into BCNF by factoring it.

**Data Anomaly**

Normalization came into existence because of the problems that occurred on data and these problems known as Data Anomalies.

If a table is not properly normalized and has data redundancy then it will not only eat up the extra memory space but will also make it difficult to handle and update the data base.

There are Three Data anomalies form Insertion, updation & Deletion.

1. Insertion Anomaly **-** An insertion anomaly occurs in the related database when some attributes or data items are inserted into database without existence of other attributes.
2. Updation Anomaly – Updation anomaly occurs when the same data item are repeated with the same values are not linked to each other.
3. Deletion Anomaly - Deletion Anomaly occurs when deleting one part of the data deletes the other necessary information from the database.

**Functional Dependency**

It is a relationship that exists between the two sets of attributes of a relational table where one set of attributes can determine the value of other sets of attributes.

It is denoted by X > Y.

.:. X is determinant and Y is dependent.

**Advantages of Normalization**

* It reduces the duplicate data.
* It keeps data consistent and accurate.
* It makes queries simpler and faster.
* It is easy to update and manage.
* It saves the space by removing duplicate data.
* Organizes data clearly and logically.
* Make it easy to grow the database efficiently.

These benefits leads to a more efficient, reliable, and scalable database system.