**Normalization**

Normalization is the process of organizing the data in the database. Normalization is used to minimize the redundancy from a relation or set of relation. Redundancy in relation may cause insertion, deletion, and update anomalies. Normalization divides the larger table into smaller and links them using relationships. Normal forms are used to eliminate or reduce redundancy in database tables.

Normalization is done to remove these anomalies, due to these anomalies the overall integrity of the data stored in the database will eventually degrade.

* **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
* **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
* **Updation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

**Following are the various types of Normal forms:**

**First Normal Form (1NF):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll no** | **Name** | **Course** | **Age** |
| 1 | Navin | c/c++ | 22 |
| 2 | Vipul | Java | 18 |
| 3 | Neeraj | c/c++ | 23 |
| 4 | Chandan | Java | 24 |
| 5 | Ajay | c/c++ | 22 |

If a relation contains composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is **singled valued attribute**.

Table 1

In above student table, we can see that the course column has two values. Thus it does not follow the First Normal Form. If we use First Normal Formal to the above table, you get the below table as a result.

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll no** | **Name** | **Course** | **Age** |
| 1 | Navin | c | 22 |
| 1 | Navin | c++ | 22 |
| 2 | Vipul | java | 18 |
| 3 | Neeraj | c | 23 |
| 3 | Neeraj | c++ | 23 |
| 4 | Chandan | java | 24 |
| 5 | Ajay | c | 22 |
| 5 | Ajay | c++ | 22 |

Table 2

After Applying First Normal Form, we achieve atomicity.

**Second Normal Form (2NF):**

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table

|  |  |  |
| --- | --- | --- |
| **CustId** | **StoreId** | **StoreLocation** |
| 1 | D1 | California |
| 2 | D3 | Texas |
| 3 | T1 | Miami |
| 4 | F2 | Toronto |
| 5 | H3 | Florida |

Table 3

In the above Location table, which contains a composite primary key CustId, StoreId. The non-key attribute is StoreLocation only depends on StoreId, which is a part of primary key, hence, this table is not in second normal form.

To bring the table in Second Normal Form, we need to split the table into two parts. This will give the below tables, as we removed the partial functional dependency from the location table, the column StoreLocation entirely depends on the primary key of the table, StoreId.

|  |  |
| --- | --- |
| **CustId** | **StoreId** |
| 1 | D1 |
| 2 | D3 |
| 3 | T1 |
| 4 | F2 |
| 5 | H3 |

Table 4

|  |  |
| --- | --- |
| StoreId | StoreLocation |
| D1 | California |
| D3 | Texas |
| T1 | Miami |
| F2 | Toronto |
| H3 | Florida |

Table 5

**Third Normal Form (3NF):**

The first condition for the table to be in Third Normal Form is that the table should be in the Second Normal Form.

The second condition is that there should be no transitive dependency for non-prime attributes, which indicates that non-prime attributes (which are not a part of the candidate key) should not depend on other non-prime attributes in a table. Therefore, a transitive dependency is a functional dependency in which A → C (A determines C) indirectly, because of A → B and B → C (where it is not the case that B → A).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **StudId** | **Name** | **SubId** | **Subject** | **Address** |
| 1 | Navin | 11 | SQL | Delhi |
| 2 | Vipul | 12 | java | Bangalore |
| 3 | Neeraj | 13 | c++ | Delhi |
| 4 | Chandan | 12 | java | Kochi |

The third Normal Form ensures the reduction of data duplication. It is also used to achieve data integrity.

Table 6

In above Student table , StudId determines SubId, and SubId determines Subject. Therefore, StudId determines Subject via SubId. This means that table contain transitive functional dependency and 3NF.

Inorder to bring table in 3NF we need to divide the table as shown below

|  |  |  |  |
| --- | --- | --- | --- |
| **StudId** | **Name** | **SubId** | **Address** |
| 1 | Navin | 11 | Delhi |
| 2 | Vipul | 12 | Bangalore |
| 3 | Neeraj | 13 | Delhi |
| 4 | Chandan | 12 | Kochi |

Table 7

|  |  |
| --- | --- |
| **SubId** | **Subject** |
| 11 | SQL |
| 12 | java |
| 13 | c++ |
| 12 | java |

Table 8

As you can see in both the tables, all the non-key attributes are now fully functional, dependent only on the primary key.

**Boyce Codd Normal Form (BCNF):**

A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF if every non-trivial functional dependency

X –> Y, X is a super key.

Consider the below subject table:

|  |  |  |
| --- | --- | --- |
| **StudId** | **Subject** | **Professor** |
| 1 | SQL | Prof. Mishra |
| 2 | java | Prof. Anand |
| 3 | c++ | Prof. James |
| 4 | Java | Prof. Lokesh |

Table 9

The above table is not in BCNF, because the FD(Professor->subject), Professor is not a key.if we try to delete the student id 3, we will loose the information that Prof. james teaches c++.These difficulties are caused by the fact that Professor is not candidate key.

Decomposition for BCNF

* Professor -> subject violates BCNF
* If X->Y violates BCNF then divide R into R1(X, Y) and R2(R-Y)
* So R is divided into two relations R1(Professor, Subject) and

R2(StudId, Professor)

* Another Important point to be noted here is that one professor teaches only one subject, but one subject may have two professors.

|  |  |
| --- | --- |
| **Professor** | **Subject** |
| Prof. Mishra | SQL |
| Prof. Anand | java |
| Prof. James | c++ |
| Prof. Lokesh | Java |

Table 10

|  |  |
| --- | --- |
| **StudId** | **Professor** |
| 1 | Prof. Mishra |
| 2 | Prof. Anand |
| 3 | Prof. James |
| 4 | Prof. Lokesh |

Table 11

**Fourth Normal Form (4NF):**

A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.

For a dependency A → B, if for a single value of A, multiple values of B exist, then the relation will be a multi-valued dependency.

|  |  |  |
| --- | --- | --- |
| **StudId** | **Course** | **Hobby** |
| 1 | Math | Dancing |
| 1 | Chemistry | Singing |
| 2 | Biology | Cricket |
| 3 | Physics | Hockey |

Table 12

The above table is in 3NF, but there is no relationship between Course and Hobby. StudId 1 contains two courses, Math and Chemistry and two hobbies,Dancing and Singing.So there is a Multi-valued dependency on StudId, which leads to unnecessary repetition of data .

To bring the table in 4NF, we decompose it into two tables

|  |  |
| --- | --- |
| **StudId** | **Course** |
| 1 | Math |
| 1 | Chemistry |
| 2 | Biology |
| 3 | Physics |

Table 13

|  |  |
| --- | --- |
| **StudId** | **Hobby** |
| 1 | Dancing |
| 1 | Singing |
| 2 | Cricket |
| 3 | Hockey |

Table 14