**Normalization**

* Normalization is the process of organizing the data in the database
* Normalization is used to minimize the redundancy from a relational database or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion anomalies.
* Normalization divides the larger table into smaller tables and links them using relationships.
* The normal form is used to reduce redundancy from the database table.

**Types of Normalization**

1. **1NF**

A relation is in 1NF if it contains an atomic values

1. **2NF**

A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

1. **3NF**

A relation will be in 3NF if it is in 2NF and no transition dependency exists.

1. **4NF**

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

1. **5NF**

A relation is in 5NF if it is in 4NF and does not contain any join dependency and joining should be lossless.

**First Normal Form (1NF)**

* A relation will be in 1NF if it contains an atomic value.
* It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attributes.
* First normal form disallows the multi-valued attribute, composite attribute and their combinations.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP\_PHONE.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **PHONE** | **STATE** |
| 1 | John | 7878787878,  9090909090 | UP |
| 2 | Harry | 8787878787 | Bihar |
| 3 | Sam | 7474747474,  2323232323 | Punjab |

The decomposition of the EMPLOYEE table into 1NF has been shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **PHONE** | **STATE** |
| 1 | John | 7878787878 | UP |
| 1 | John | 9090909090 | UP |
| 2 | Harry | 8787878787 | Bihar |
| 3 | Sam | 7474747474 | Punjab |
| 3 | Sam | 2323232323 | Punjab |

**Second Normal Form (2NF)**

* In the 2NF, the relation must be in 1NF.
* In the second normal form, all non-key attributes are fully functional dependent 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.

|  |  |  |
| --- | --- | --- |
| **ID** | **SUBJECT** | **AGE** |
| 25 | Chemistry | 30 |
| 25 | Biology | 30 |
| 47 | English | 35 |
| 83 | Math | 38 |
| 83 | Computer | 38 |

In the given table, non-prime attribute AGE is dependent on ID which is a proper subset of a candidate key. That’s why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

**TEACHER\_DETAIL**

|  |  |
| --- | --- |
| **ID** | **AGE** |
| 25 | 30 |
| 47 | 35 |
| 83 | 38 |

**TEACHER\_SUBJECT**

|  |  |
| --- | --- |
| **ID** | **SUBJECT** |
| 25 | Chemistry |
| 25 | Biology |
| 47 | English |
| 83 | Math |
| 83 | Computer |

**Third Normal Form (3NF)**

* A relation will be in 3NF if it is in 2NF and not containing any transitive partial dependency.
* 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
* If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial functional dependency X -> Y.

**EMPLOYEE\_DETAIL**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **ZIP** | **STATE** | **CITY** |
| 2 | Harry | 201010 | UP | Noida |
| 3 | Stephan | 02228 | US | Boston |
| 4 | Lan | 60007 | US | Chicago |
| 5 | Katharine | 06389 | UK | Norwich |
| 6 | John | 462007 | MP | Bhopal |

Super key in the table above:

ID, ID - NAME, ID - NAME - ZIP

Candidate key: ID

Non-prime attributes :

In the given table, all attributes except ID are non-prime.

Here, STATE and CITY are dependent on ZIP and ZIP is dependent on ID. The non-prime attributes i.e STATE and CITY transitively dependent on ID. So it violates the rule of 3NF.

So we need to move the CITY and STATE to new ZIP table and with ZIP as a primary key.

**EMPLOYEE\_DETAIL**

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **ZIP** |
| 2 | Harry | 201010 |
| 3 | Stephan | 02228 |
| 4 | Lan | 60007 |
| 5 | Katharine | 06389 |
| 6 | John | 462007 |

**ZIP**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **ZIP** | **STATE** | **CITY** |
| 2 | 201010 | UP | Noida |
| 3 | 02228 | US | Boston |
| 4 | 60007 | US | Chicago |
| 5 | 06389 | UK | Norwich |
| 6 | 462007 | MP | Bhopal |

**Boyce Codd Normal Form (BCNF)**

* BCNF is the advanced version of 3NF. It is stricter than 3NF.
* A table is in BCNF if every functional dependency X -> Y, X is the super key of the table.
* For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let’s assume there is a company where employees work in more than one department.

**EMPLOYEE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **COUNTRY** | **DEPARTMENT** | **TYPE** | **EMP\_DEPT\_NO** |
| 1 | India | Designing | D1 | 1a |
| 2 | India | Testing | D1 | 2a |
| 3 | UK | Stores | D2 | 3a |
| 4 | UK | Development | D2 | 4a |

In the above table we can see:

ID -> Country

DEPARTMENT -> TYPE, EMP\_DEPT\_NO

Candidate key: ID, DEPARTMENT

The table is not in BCNF because neither DEPARTMENT nor ID alone are keys.

To convert the table into BCNF, we decompose it into 3 tables:

**EMPLOYEE\_COUNTRY**

|  |  |
| --- | --- |
| **ID** | **COUNTRY** |
| 1 | India |
| 2 | India |
| 3 | UK |
| 4 | UK |

**DEPARTMENT**

|  |  |  |
| --- | --- | --- |
| **DEPARTMENT** | **TYPE** | **EMP\_DEPT\_NO** |
| Designing | D1 | 1a |
| Testing | D1 | 2a |
| Stores | D2 | 3a |
| Development | D2 | 4a |

**EMPLOYEE\_DEPARTMENT**

|  |  |
| --- | --- |
| **EMP\_ID** | **EMP\_DEPT\_NO** |
| 1 | 1a |
| 2 | 2a |
| 3 | 3a |
| 4 | 4a |

Dependencies: ID -> Country

Candidate: TYPE, DEPT\_NO

Candidate keys: ID, DEPT, ID/DEPT for respective tables