# DBMS – Unit II

# 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 relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

# Purpose of Normalization

**Normalization** is the process of structuring and handling the relationship between data to minimize redundancy in the relational table and avoid the unnecessary anomalies properties from the database like insertion, update and delete. It helps to divide large database tables into smaller tables and make a relationship between them. It can remove the redundant data and ease to add, manipulate or delete table fields.

A normalization defines rules for the relational table as to whether it satisfies the normal form. A **normal form** is a process that evaluates each relation against defined criteria and removes the multivalued, joins, functional and trivial dependency from a relation. If any data is updated, deleted or inserted, it does not cause any problem for database tables and help to improve the relational table' integrity and efficiency.

## Objective of Normalization

1. It is used to remove the duplicate data and database anomalies from the relational table.
2. Normalization helps to reduce redundancy and complexity by examining new data types used in the table.
3. It is helpful to divide the large database table into smaller tables and link them using relationship.
4. It avoids duplicate data or no repeating groups into a table.
5. It reduces the chances for anomalies to occur in a database.

# Types of Anomalies

Following are the types of anomalies that make the table inconsistency, loss of integrity, and redundant data.

**1. Data redundancy** occurs in a relational database when two or more rows or columns have the same value or repetitive value leading to unnecessary utilization of the memory.

**Student Table:**

| StudRegistration | CourseID | StudName | Address | Course |
|---|---|---|---|---|
| 205 | 6204 | James | Los Angeles | Economics |
| 205 | 6247 | James | Los Angeles | Economics |
| 224 | 6247 | Trent Bolt | New York | Mathematics |
| 230 | 6204 | Ritchie Rich | Egypt | Computer |
| 230 | 6208 | Ritchie Rich | Egypt | Accounts |

There are two students in the above table, 'James' and 'Ritchie Rich', whose records are repetitive when we enter a new CourseID. Hence it repeats the studRegistration, StudName and address attributes.

**2. Insert Anomaly:** An insert anomaly occurs in the relational database when some attributes or data items are to be inserted into the database without existence of other attributes. For example, In the Student table, if we want to insert a new courseID, we need to wait until the student enrolled in a course. In this way, it is difficult to insert new record in the table. Hence, it is called insertion anomalies.

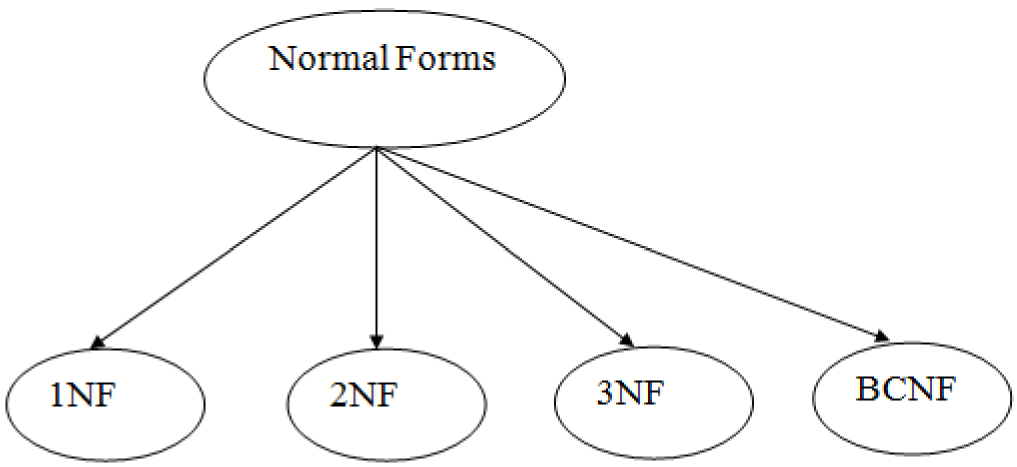
**3. Update Anomalies:** The anomaly occurs when duplicate data is updated only in one place and not in all instances. Hence, it makes our data or table inconsistent state. For example, suppose there is a student 'James' who belongs to Student table. If we want to update the course in the Student, we need to update the same in the course table; otherwise, the data can be **inconsistent**. And it reflects the changes in a table with updated values where some of them will not.

**4. Delete Anomalies:** An anomaly occurs in a database table when some records are lost or deleted from the database table due to the deletion of other records. For example, if we want to remove Trent Bolt from the Student table, it also removes his address, course and other details from the Student table. Therefore, we can say that deleting some attributes can remove other attributes of the database table.

So, we need to avoid these types of anomalies from the tables and maintain the integrity, accuracy of the database table. Therefore, we use the normalization concept in the database management system.

## Types of Normal Forms

There are the four types of normal forms:




| Normal Form | Description |
| --- | --- |
| 1NF | A relation is in 1NF if it contains an atomic value. |
| 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. |
| 3NF | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |
| 4NF | A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency. |

| | |
|---|---|
| 5NF | A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless. |

# First Normal Form (1NF)

- A relation will be 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 attribute.
- 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.

**EMPLOYEE table:**

| EMP_ID | EMP_NAME | EMP_PHONE | EMP_STATE |
|---|---|---|---|
| 14 | John | 7272826385,<br>9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389,<br>8589830302 | Punjab |

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

| EMP_ID | EMP_NAME | EMP_PHONE | EMP_STATE |
|---|---|---|---|
| 14 | John | 7272826385 | UP |
| 14 | John | 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389 | Punjab |

| | | | |
|---|---|---|---|
| 12 | Sam | 8589830302 | Punjab |

# Second Normal Form (2NF)

- In the 2NF, relational 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.

**TEACHER table**

| TEACHER_ID | SUBJECT | TEACHER_AGE |
|---|---|---|
| 25 | Chemistry | 30 |
| 25 | Biology | 30 |
| 47 | English | 35 |
| 83 | Math | 38 |
| 83 | 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. That's why it violates the rule for 2NF.

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

**TEACHER_DETAIL table:**

| TEACHER_ID | TEACHER_AGE |
|---|---|
| 25 | 30 |
| 47 | 35 |

| | |
|---|---|
| 83 | 38 |

**TEACHER_SUBJECT table:**

| TEACHER_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 contain 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 atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

**Example:**

**EMPLOYEE_DETAIL table:**

| EMP_ID | EMP_NAME | EMP_ZIP | EMP_STATE | EMP_CITY |
|---|---|---|---|---|
| 222 | Harry | 201010 | UP | Noida |

| | | | | |
|---|---|---|---|---|
| 333 | Stephan | 02228 | US | Boston |
| 444 | Lan | 60007 | US | Chicago |
| 555 | Katharine | 06389 | UK | Norwich |
| 666 | John | 462007 | MP | Bhopal |

**Super key in the table above:**

1. {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

**Candidate key:** {EMP_ID}

**Non-prime attributes:** In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

**EMPLOYEE table:**

| EMP_ID | EMP_NAME | EMP_ZIP |
|---|---|---|
| 222 | Harry | 201010 |
| 333 | Stephan | 02228 |
| 444 | Lan | 60007 |
| 555 | Katharine | 06389 |
| 666 | John | 462007 |

**EMPLOYEE_ZIP table:**

| EMP_ZIP | EMP_STATE | EMP_CITY |
|---|---|---|
| 201010 | UP | Noida |
| 02228 | US | Boston |
| 60007 | US | Chicago |
| 06389 | UK | Norwich |
| 462007 | MP | Bhopal |

# Advanced Normalization

# Boyce Codd normal form (BCNF)

- BCNF is the advance 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 table:**

| EMP_ID | EMP_COUNTRY | EMP_DEPT | DEPT_TYPE | EMP_DEPT_NO |
|---|---|---|---|---|
| 264 | India | Designing | D394 | 283 |
| 264 | India | Testing | D394 | 300 |
| 364 | UK | Stores | D283 | 232 |
| 364 | UK | Developing | D283 | 549 |

**In the above table Functional dependencies are as follows:**

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

**Candidate key: {EMP-ID, EMP-DEPT}**

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

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

**EMP_COUNTRY table:**

| EMP_ID | EMP_COUNTRY |
|---|---|
| 264 | India |
| 264 | India |

**EMP_DEPT table:**

| EMP_DEPT | DEPT_TYPE | EMP_DEPT_NO |
|---|---|---|
| Designing | D394 | 283 |
| Testing | D394 | 300 |
| Stores | D283 | 232 |
| Developing | D283 | 549 |

**EMP_DEPT_MAPPING table:**

| EMP_ID | EMP_DEPT |
|---|---|
| D394 | 283 |
| D394 | 300 |

| | |
|---|---|
| D283 | 232 |
| D283 | 549 |

**Functional dependencies:**

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

**Candidate keys:**

**For the first table:** EMP_ID
**For the second table:** EMP_DEPT
**For the third table:** {EMP_ID, EMP_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

# 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 exists, then the relation will be a multi-valued dependency.

# Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).