## Database Anomaly

Database anomaly is normally the flaw in databases which occurs because of poor planning and storing everything in a flat database. Generally this is removed by the process of normalization which is performed by splitting/joining of tables.

| **Prof_ID** | **Prof_Name** | **Dept.** | **Course Group** |
|---|---|---|---|
| 39404 | Ashish | Marketing | Sec A |
| 39445 | Sonam | Product | Sec B |
| 43576 | Anu Priya | Finance | Sec C |
| 54325 | Anu Priya | Finance | Sec C |
| 99823 | Anushka | HR | Sec D |
| 14325 | Anushka | HR | Sec E |

There are three types of database anomalies:

**a) Insertion anomaly:** **A**n insertion anomaly occurs when we are not able to insert certain attribute in the database without the presence of other attribute. For example suppose any professor is hired but not immediately assigned any course group or any department may not get his/her place in such type of flat database mentioned above, if null entries are not allowed in the database. So in the case mentioned above removing such type of problems requires splitting of the database which is done by normalization.

**b) Update anomaly:** This occurs in case of data redundancy and partial update. In other words a correct update of database needs other actions such as addition, deletion or both. For example in the above table the department assigned to Anushka is an error because it needs to be updated at two different place to maintain consistency.

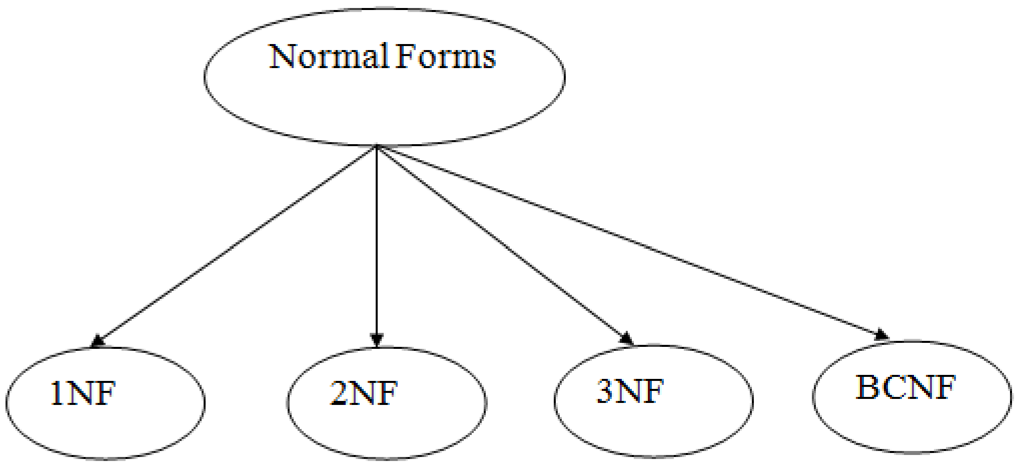
**c) Deletion Anomaly:** Deletion anomaly occurs where deletion some data is deleted because of deletion of some other data. For example if Section B is to be deleted then un-necessarily Sonam’s detail has to be deleted. So normalization is generally done before deleting any record from a flat database.

## 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.

## 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 | 727282638 | UP |

| | | | |
|---|---|---|---|
| | | 5, 9064738238 | |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389, 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 |

## 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 |
| 36 | UK | Devel | D283 | 549 |

|  |  |  |  |  |
|---|---|---|---|---|
| 4 |  | oping |  |  |

**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.

## Example

**STUDENT**

| STU_ID | COURSE | HOBBY |
|---|---|---|
| 21 | Computer | Dancing |
| 21 | Math | Singing |
| 34 | Chemistry | Dancing |
| 74 | Biology | Cricket |
| 59 | Physics | Hockey |

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

**STUDENT_COURSE**

| STU_ID | COURSE |
|---|---|
| 21 | Computer |
| 21 | Math |

| | |
|---|---|
| 34 | Chemistry |
| 74 | Biology |
| 59 | Physics |

**STUDENT_HOBBY**

| STU_ID | HOBBY |
|---|---|
| 21 | Dancing |
| 21 | Singing |
| 34 | Dancing |
| 74 | Cricket |
| 59 | Hockey |

## 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).

### Example

| SUBJECT | LECTURER | SEMESTER |
|---|---|---|

| | | |
|---|---|---|
| Computer | Anshika | Semester 1 |
| Computer | John | Semester 1 |
| Math | John | Semester 1 |
| Math | Akash | Semester 2 |
| Chemistry | Praveen | Semester 1 |

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

**P1**

| SEMESTER | SUBJECT |
|---|---|
| Semester 1 | Computer |
| Semester 1 | Math |
| Semester 1 | Chemistry |

| | |
|---|---|
| Semester 2 | Math |

**P2**

| SUBJECT | LECTURER |
|---|---|
| Computer | Anshika |
| Computer | John |
| Math | John |
| Math | Akash |
| Chemistry | Praveen |

**P3**

| SEMSTER | LECTURER |
|---|---|
| Semester 1 | Anshika |
| Semester 1 | John |
| Semester 1 | John |
| Semester 2 | Akash |
| Semester 1 | Praveen |