# Chapter- 7

# DATABASE NORMALIZATION

## Keys

- o Keys play an important role in the relational database.
- o It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

**For example:** In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

| STUDENT |
|---|
| ID |
| Name |
| Address |
| Course |

| PERSON |
|---|
| Name |
| DOB |
| Passport_Number |
| License_Number |
| SSN |
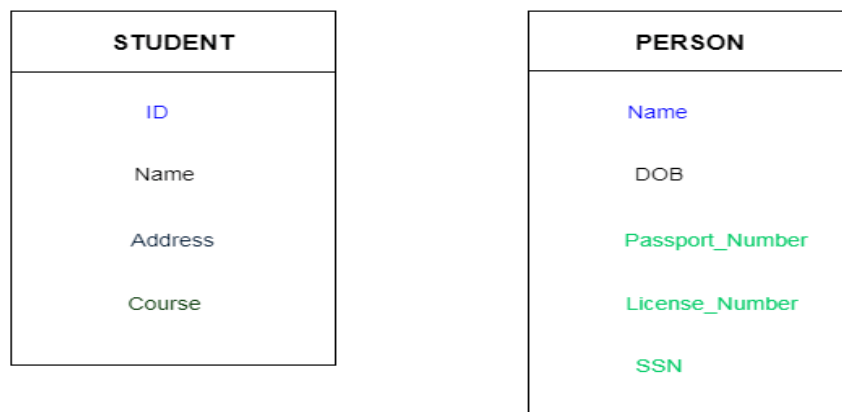
## Types of key:
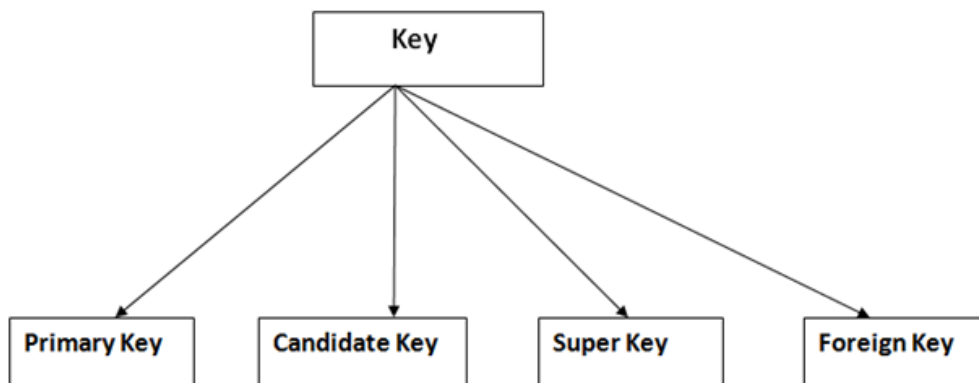


### 1. Primary key

- o It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.

- o In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.
- o For each entity, selection of the primary key is based on requirement and developers.
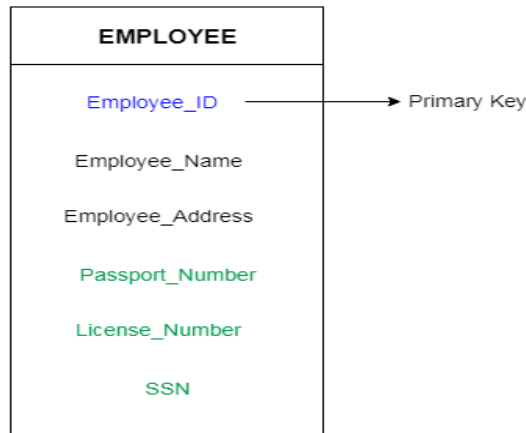


## 2. Candidate key

- o A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- o The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.

**For example:** In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.
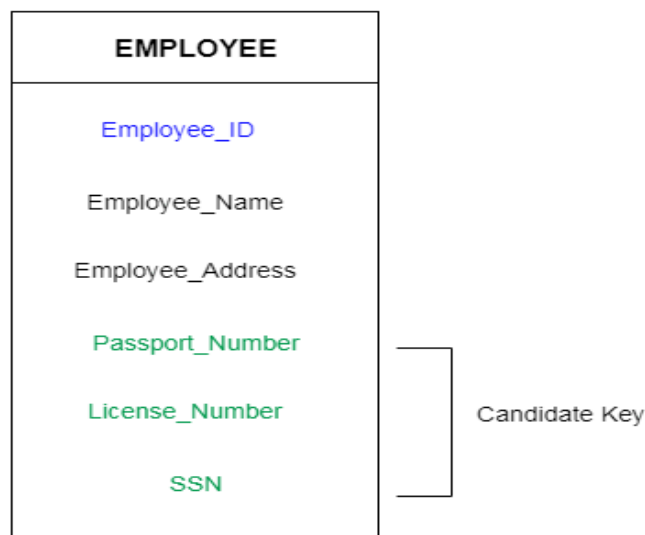


## 3. Super Key

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

**For example:** In the above EMPLOEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

**4. Foreign key**

- o Foreign keys are the column of the table which is used to point to the primary key of another table.

- o In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.

- o We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.

- o Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.
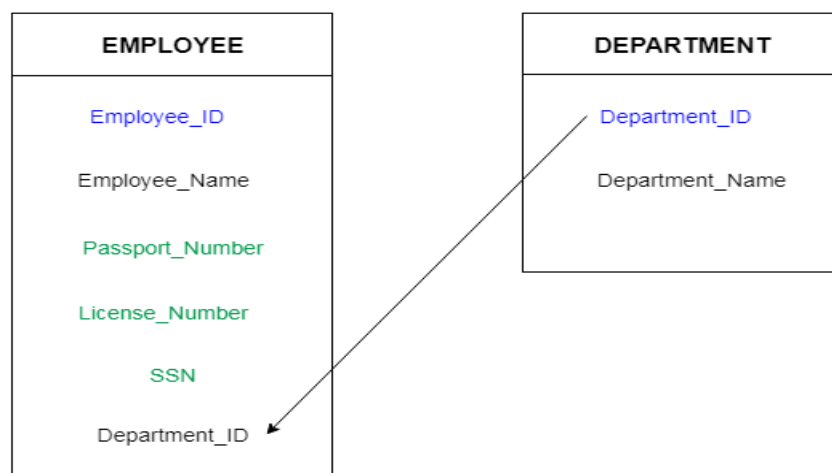


**Well-Structured Relations**

• A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies

• Goal is to avoid (minimize) anomalies

**– Insertion Anomaly** – adding new rows forces user to create duplicate data

**– Deletion Anomaly** – deleting a row may cause loss of other data representing completely different facts

**– Modification Anomaly** – changing data in a row forces changes to other rows because of duplication

# Normalization

- o Normalization is the process of organizing the data in the database.
- o 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.
- o Normalization divides the larger table into the smaller table and links them using relationship.
- o The normal form is used to reduce redundancy from the database table.

## Types of Normal Forms

There are the four types of normal forms:
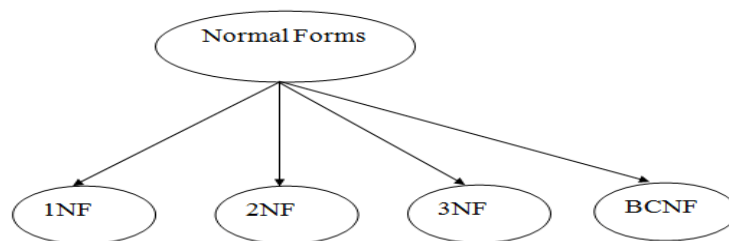


| Normal Form | Description |
|---|---|
| 1NF | A relation is in 1NF if it contains an atomic value. |
| 2NF | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| 3NF | A relation will be in 3NF if it is in 2NF and no transition 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)

- o A relation will be 1NF if it contains an atomic value.
- o It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- o 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, 9064738238 | UP |
| 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 |

## Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

X → Y

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

**For example:**

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id.

**Types of Functional dependency**
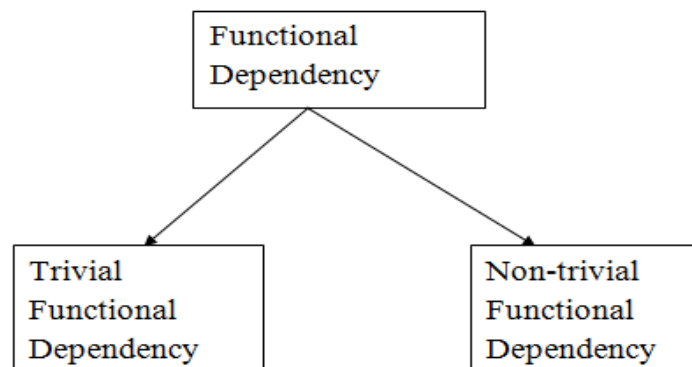


**1. Trivial functional dependency**

- o   A → B has trivial functional dependency if B is a subset of A.
- o   The following dependencies are also trivial like: A → A, B → B

**Example:**
1. Consider a table with two columns Employee_Id and Employee_Name.
2. {Employee_id, Employee_Name}   →   Employee_Id is a trivial functional dependency as
3. Employee_Id is a subset of {Employee_Id, Employee_Name}.
4. Also, Employee_Id → Employee_Id and Employee_Name   →   Employee_Name are trivial dependencies too.

**2. Non-trivial functional dependency**

- o   A → B has a non-trivial functional dependency if B is not a subset of A.
- o   When A intersection B is NULL, then A → B is called as complete non-trivial.

**Example:**
ID   →   Name,
Name   →   DOB

# Second Normal Form (2NF)

- o   In the 2NF, relational must be in 1NF.
- o   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)

- o A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- o 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- o 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)

- o BCNF is the advance version of 3NF. It is stricter than 3NF.
- o A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- o 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.

| 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 |

**EMPLOYEE Table:**

**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:**
EMP_ID  →  EMP_COUNTRY
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  |