**Unit I**

**Chapter 1.4**

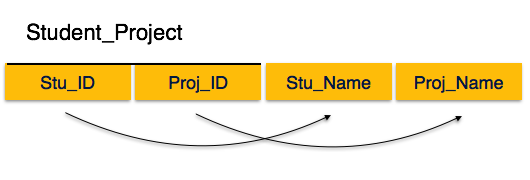
**( Second Normal Form(2NF))**

**Second Normal Form**

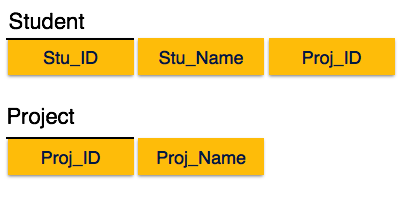
Before we learn about the second normal form, we need to understand the following −

* **Prime attribute**− An attribute, which is a part of the candidate-key, is known as a prime attribute.
* **Non-prime attribute**− An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



We see here in We see here in Student\_Project relation that the prime key attributes are Stu\_ID and Proj\_ID. According to the rule, non-key attributes, i.e. Stu\_Name and Proj\_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu\_Name can be identified by Stu\_ID and Proj\_Name can be identified by Proj\_ID independently. This is called partial dependency, which is not allowed in Second Normal Form.



We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

For a table to be in second normal form, the following 2 conditions are to be met:

1. The table should be in the first normal form.
2. The primary key of the table should compose of exactly 1 column.

The first point is obviously straightforward since we just studied 1NF. Let us understand the first point - 1 column primary key. Well, a primary key is a set of columns that uniquely identifies a row. Basically, no 2 rows have the same primary keys. Let us take an example.

**Course**



Here, in this table, the course code is unique. So, that becomes our primary key.

Let us take another example of storing student enrollment in various courses. Each student may enroll in multiple courses. Similarly, each course may have multiple enrollments. A sample table may look like this (student name and course code):



Here, the first column is the student name and the second column is the course taken by the student.

Clearly, the student name column isn’t unique as we can see that there are 2 entries corresponding to the name ‘Rahul’ in row 1 and row 3. Similarly, the course code column is not unique as we can see that there are 2 entries corresponding to course code CS101 in row 2 and row 4.

However, the tuple (student name, course code) is unique since a student cannot enroll in the same course more than once. So, these 2 columns when combined form the primary key for the database.

As per the second normal form definition, our enrollment table above isn’t in the second normal form. To achieve the same (1NF to 2NF), we can rather break it into 2 tables:



Here the second column is unique and it indicates the enrollment number for the student. Clearly, the enrollment number is unique. Now, we can attach each of these enrollment numbers with course codes.

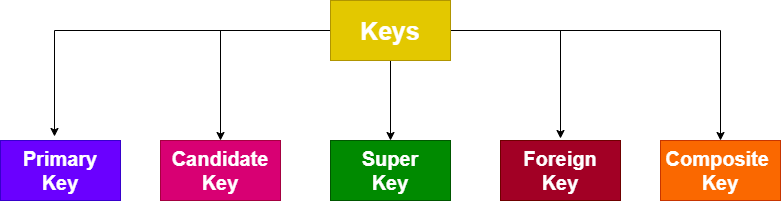
**Courses:**



These 2 tables together provide us with the exact same information as our original table.

**Keys in Database**

The keys in the database play a vital role in identifying the data present in it. Using keys, you can find any data from the table quickly and easily. The large tables are divided into smaller ones, and keys are used to connect the smaller tables.



Let us see the different types of keys used in the DBMS.

**1.Primary Key:**

The **primary key** is an attribute of the table that identifies any row or tuple uniquely. You must choose the primary key that will uniquely find any data from the table.

Consider a relation Employee. It has attributes like **Emp\_ID, Emp\_Name, Emp\_Add, Passport\_Number, and License\_Number.**

The primary key of the Employee relation will be the **Emp\_ID,** as it will uniquely identify every employee’s data. Additionally, the P**assport\_Number and License\_Number** can also serve as primary keys as they are unique for every employee.

**2.Candidate Key:**

The **candidate key** of any table is a set of minimal attributes and can identify any row uniquely in the relation. There can be single or more candidate keys for a single relation.

Consider the above relation of Employee. We saw that the primary key is the **Emp\_ID**, which is unique and non-repetitive for every employee. The other two attributes, **Passport\_Number and License\_Number,** are also non-repetitive. So, they both can serve as candidate keys.

**3.Super Key:**

As the **primary key** and **candidate key** identifies every tuple uniquely, the super key also helps find the table’s unique tuple. The candidate key is the subset of the super key. There can be one or many super keys.

We shall use the same Employee relation to have a clear idea about the super key. The **Emp\_ID** attribute can uniquely find out any employee’s data. The **Emp\_Name** attribute cannot be used as the primary key, as two employees can have the same name. But, the combination of the **Emp\_ID and Emp\_Name** can find employee’s data uniquely. So, **(Epm\_ID, Emp\_Name)** serves as super keys for the Employee relation.

The **Passprt\_Number and License\_Number** are also super keys of the Employee relation.

**4.Foreign Key:**

The foreign key is quite different from the above three keys. It is used to establish a connection between two relations. Consider two relations A and B. Suppose any attribute in the relation A is the primary key of the relation B; that attribute is referred to as the foreign key.

We shall look at the simple example to understand the foreign key concept. Let us take the example of employees in the company. Every employee is assigned to different departments. Hence, we use two relations, Employee and Department.

We define Employee relations as **Employee (Emp\_ID, Emp\_Name, Passport\_Number, License\_Number, Dept\_ID),** and Department relation as **Department (Dept\_ID, Dept\_Name).**

In the Employee relation, Emp\_ID is the primary key, whereas Dept\_ID is the department relation’s primary key. The attribute Dept\_Id is one attribute in the Employee relation, which is the primary key in the Department relation. Hence, the Dept\_ID serves as the foreign key.

**5.Composite Key:**

A composite key is a group of attributes that uniquely finds very employee’s data from the relation. The composite key is the combination of two and more than two attributes.

From the above Employee relation **Emp (Emp\_ID, Emp\_Name, Passport\_Number, License\_Number),** the composite key is **(Emp\_name, Emp\_ID).**

**Other References**

[DBMS 1NF - javatpoint](https://www.javatpoint.com/dbms-first-normal-form)

[Normalization in DBMS: 1NF, 2NF, 3NF and BCNF with Examples (hackr.io)](https://hackr.io/blog/dbms-normalization)

[Introduction to Database Normalization (softwaretesttips.com)](https://www.softwaretesttips.com/database-normalization/)

[Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database (beginnersbook.com)](https://beginnersbook.com/2015/05/normalization-in-dbms/)

**Suggested Book References**

1. J. Date, “An Introduction to Database Systems”,Addison Wesley.
2. Navathe,“Fundamentals of Database System”, The Benjamin / Cummings Publishing Co.