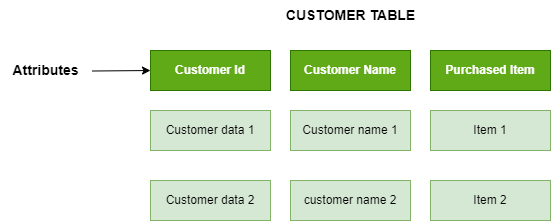
**What is Schema?**

* The Skeleton of the database is created by the attributes and this skeleton is named Schema.
* Schema mentions the logical constraints like table, primary key, etc.
* The schema does not represent the data type of the attributes.



*Details of a Customer*



*Schema of Customer*

**Database Schema**

* A database schema is a **logical representation of data** that shows how the data in a database should be stored logically. It shows how the data is organized and the relationship between the tables.
* Database schema contains table, field, views and relation between different keys like [primary key](https://www.geeksforgeeks.org/primary-key-in-dbms/), [foreign key](https://www.geeksforgeeks.org/foreign-key-constraint-in-sql/).
* Data are stored in the form of files which is unstructured in nature which makes accessing the data difficult. Thus to resolve the issue the data are organized in structured way with the help of database schema.
* Database schema provides the organization of data and the relationship between the stored data.
* Database schema defines a set of guidelines that control the database along with that it provides information about the way of accessing and modifying the data.

**Types of Database Schemas**

**There are 3 types of database schema:**

**Physical Database Schema**

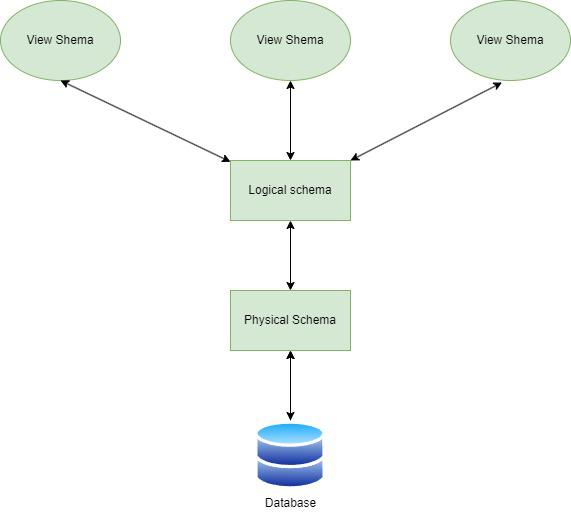
* A Physical schema defines, how the data or information is stored physically in the storage systems in the form of files & indices. This is the actual code or syntax needed to create the structure of a database, we can say that when we design a database at a physical level, it’s called physical schema.
* The Database administrator chooses where and how to store the data in the different blocks of storage.

**Logical Database Schema**

* A logical database schema defines all the logical constraints that need to be applied to the stored data, and also describes tables, views, entity relationships, and integrity constraints.
* The Logical schema describes how the data is stored in the form of tables & how the attributes of a table are connected.
* Using **ER modelling** the relationship between the components of the data is maintained.
* In logical schema different integrity constraints are defined in order to maintain the quality of insertion and update the data.

**View Database Schema**

* It is a view level design which is able to define the interaction between end-user and database.
* User is able to interact with the database with the help of the interface without knowing much about the stored mechanism of data in database.



*Three Layer Schema Design*

**Creating Database Schema**

For creating a schema, the statement “CREATE SCHEMA” is used in every database. But different databases have different meanings for this. Below we’ll be looking at some statements for creating a database schema in different database systems:

**1. MySQL:**In MySQL, we use the “CREATE SCHEMA” statement for creating the database, because, in MySQL CREATE SCHEMA and CREATE DATABASE, both statements are similar.

**2. SQL Server:**In SQL Server, we use the “CREATE SCHEMA” statement for creating a new schema.

**3. Oracle Database:**In Oracle Database, we use “CREATE USER” for creating a new schema, because in the Oracle database, a schema is already created with each database user. The statement “CREATE SCHEMA” does not create a schema, instead, it populates the schema with tables & views and also allows one to access those objects without needing multiple SQL statements for multiple transactions.

**Database Schema Designs**

There are many ways to structure a database, and we should use the best-suited schema design for creating our database because ineffective schema designs are difficult to manage & consume extra memory and resources.

Schema design mostly depends on the application’s requirements. Here we have some effective schema designs to create our applications, let’s take a look at the schema designs:

1. Flat Model
2. Hierarchical Model
3. Network Model
4. Relational Model
5. Star Schema
6. Snowflake Schema

**Flat Model**

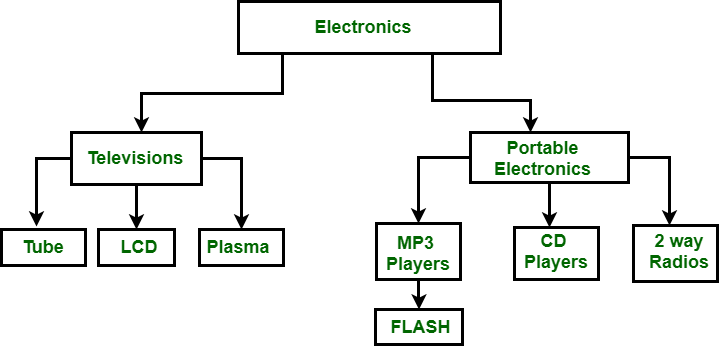
A flat model schema is a 2-D array in which every column contains the same type of data/information and the elements with rows are related to each other. It is just like a table or a spreadsheet. This schema is better for small applications that do not contain complex data.



*Designing Flat Model*

**Hierarchical Model**

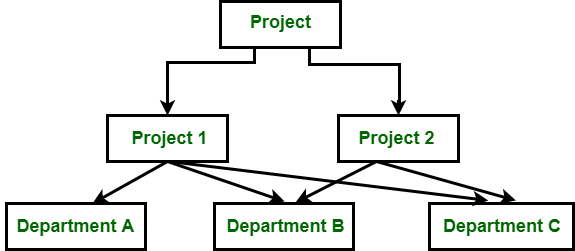
Data is arranged using parent-child relationships and a tree-like structure in the Hierarchical Database Model. Because each record consists of several children and one parent, it can be used to illustrate one-to-many relationships in diagrams such as organizational charts. Although obvious, it might not be as adaptable in complicated partnerships.



*Designing Hierarchical Model*

**Network Model**

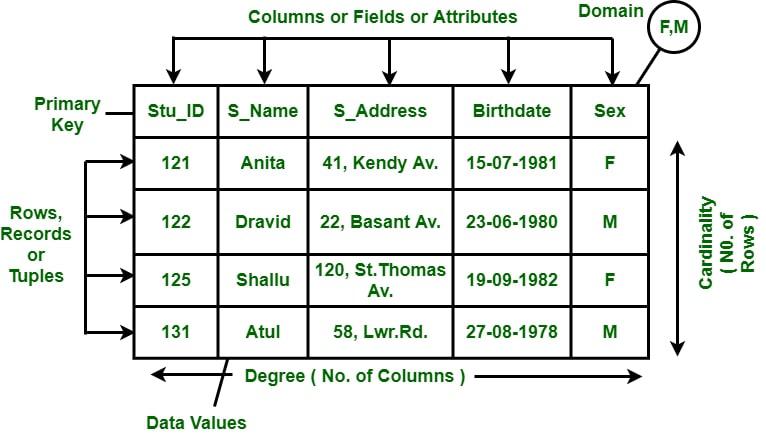
The network model and the hierarchical model are quite similar with an important difference that is related to data relationships. The network model allows many-to-many relationships whereas hierarchical models allow one-to-many relationships.



*Designing Network Model*

**Relational Model**

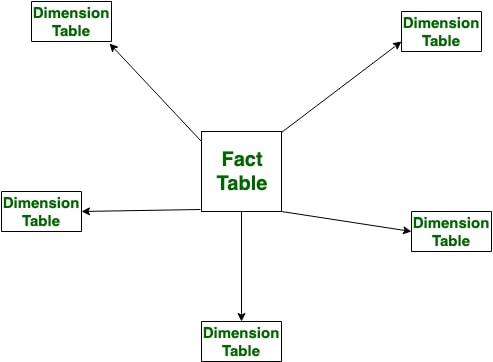
The relational model is mainly used for relational databases, where the data is stored as relations of the table. This [relational model schema](https://www.geeksforgeeks.org/relational-model-in-dbms/) is better for object-oriented programming.



*Designing Relational Model*

**Star Schema**

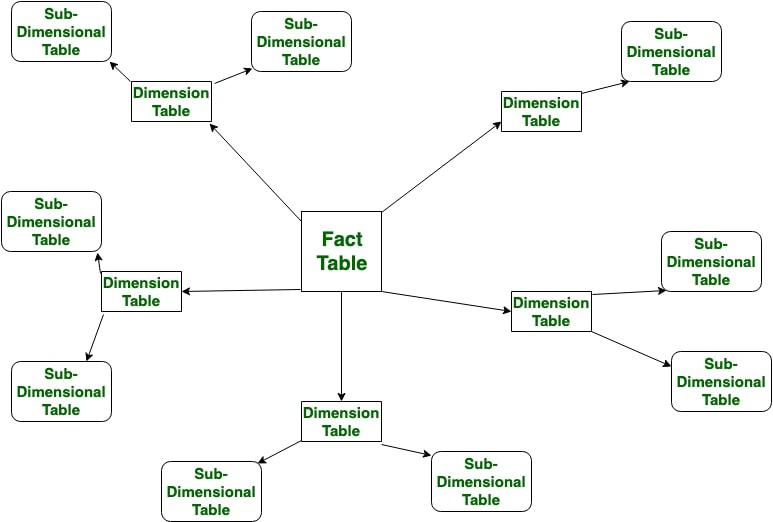
Star schema is better for storing and analyzing large amounts of data. It has a fact table at its center & multiple dimension tables connected to it just like a star, where the fact table contains the numerical data that run business processes and the dimension table contains data related to dimensions such as product, time, people, etc. or we can say, this table contains the description of the fact table. The star schema allows us to structure the data of RDBMS.



*Designing Star Schema*

**Snowflake Schema**

Just like star schema, the snowflake schema also has a fact table at its center and multiple dimension tables connected to it, but the main difference in both models is that in snowflake schema – dimension tables are further normalized into multiple related tables. The snowflake schema is used for analyzing large amounts of data.



*Designing Snowflake Schema*

**Difference between Logical and Physical Database Schema**

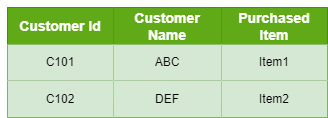
| **Physical Schema** | **Logical Schema** |
| --- | --- |
| Physical schema describes the way of storage of data in the disk. | Logical schema provides the conceptual view that defines the relationship between the data entities. |
| Having Low level of abstraction. | Having a high level of abstraction. |
| The design of database is independent to any database management system. | The design of a database must work with a specific database management system or hardware platform. |
| Changes in Physical schema effects the logical schema | Any changes made in logical schema have minimal effect in the physical schema |
| Physical schema does not include attributes. | Logical schema includes attributes. |
| Physical schema contains the attributes and their data types. | Logical schema does not contain any attributes or data types. |
| **Examples:** Data definition language(DDL), storage structures, indexes. | **Examples:** Entity Relationship diagram, Unified Modeling Language, class diagram. |

**Advantages of Database Schema**

* **Providing Consistency of data:**Database schema ensures the data consistency and prevents the duplicates.
* **Maintaining Scalability:** Well designed database schema helps in maintaining addition of new tables in database along with that it helps in handling large amounts of data in growing tables.
* **Performance Improvement:** Database schema helps in faster data retrieval which is able to reduce operation time on the database tables.
* **Easy Maintenance:** Database schema helps in maintaining the entire database without affecting the rest of the database
* **Security of Data:** Database schema helps in storing the sensitive data and allows only authorized access to the database.

**Database Instance**

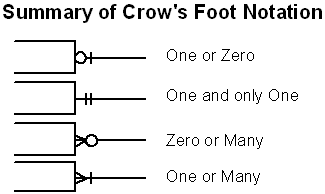
The database schema is defined before the actual database is created, after the database is operational, it is very difficult to modify the schema because the schema represents the fundamental structure of the database. Database instance does not hold any information related to the saved data in database. Therefore database instance represents the data and information that is currently stored in the database at a specific point in time.



*Database instance of Customer table at a specific time*

**Database schema versus database instance**

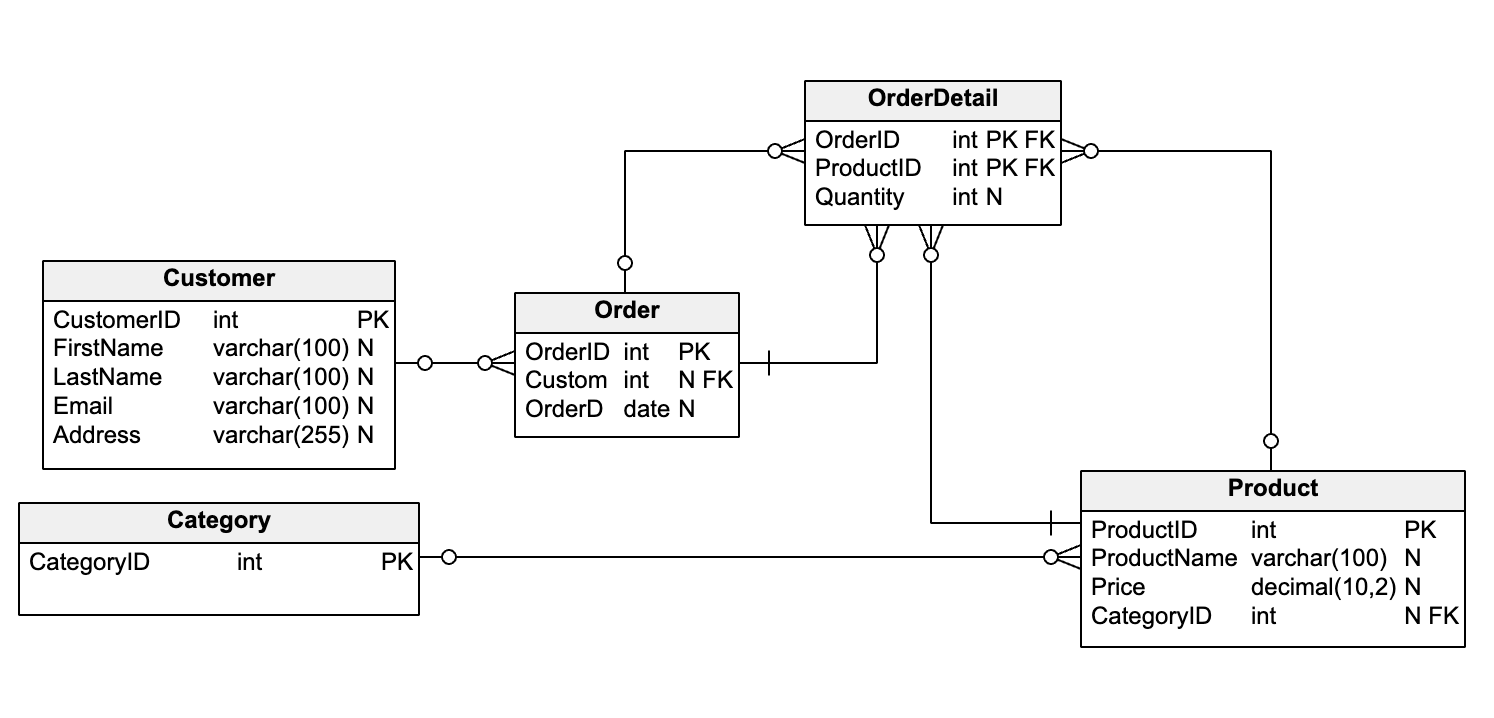
| **Aspect** | **Database Schema** | **Database Instance** |
| --- | --- | --- |
| **Definition** | Blueprint or design of the database structure | Actual data stored in the database at a given time |
| **Nature** | Static (does not change frequently) | Dynamic (changes with every data modification) |
| **Represents** | Structure (tables, columns, data types, relationships) | State of the data in the database |
| **Example** | Table definitions, data types, constraints | Actual rows of data in the tables |
| **Change Frequency** | Changes infrequently (e.g., during schema design changes) | Changes frequently with transactions |



Example 1: Online Shopping System

**Entities:** Customer, Order, Product, Category, OrderDetail.

**Relationships:**Customer 'places' Order, Order 'contains' Product, Product 'belongs to' Category, OrderDetail 'connects' Order and Product.



In this schema:

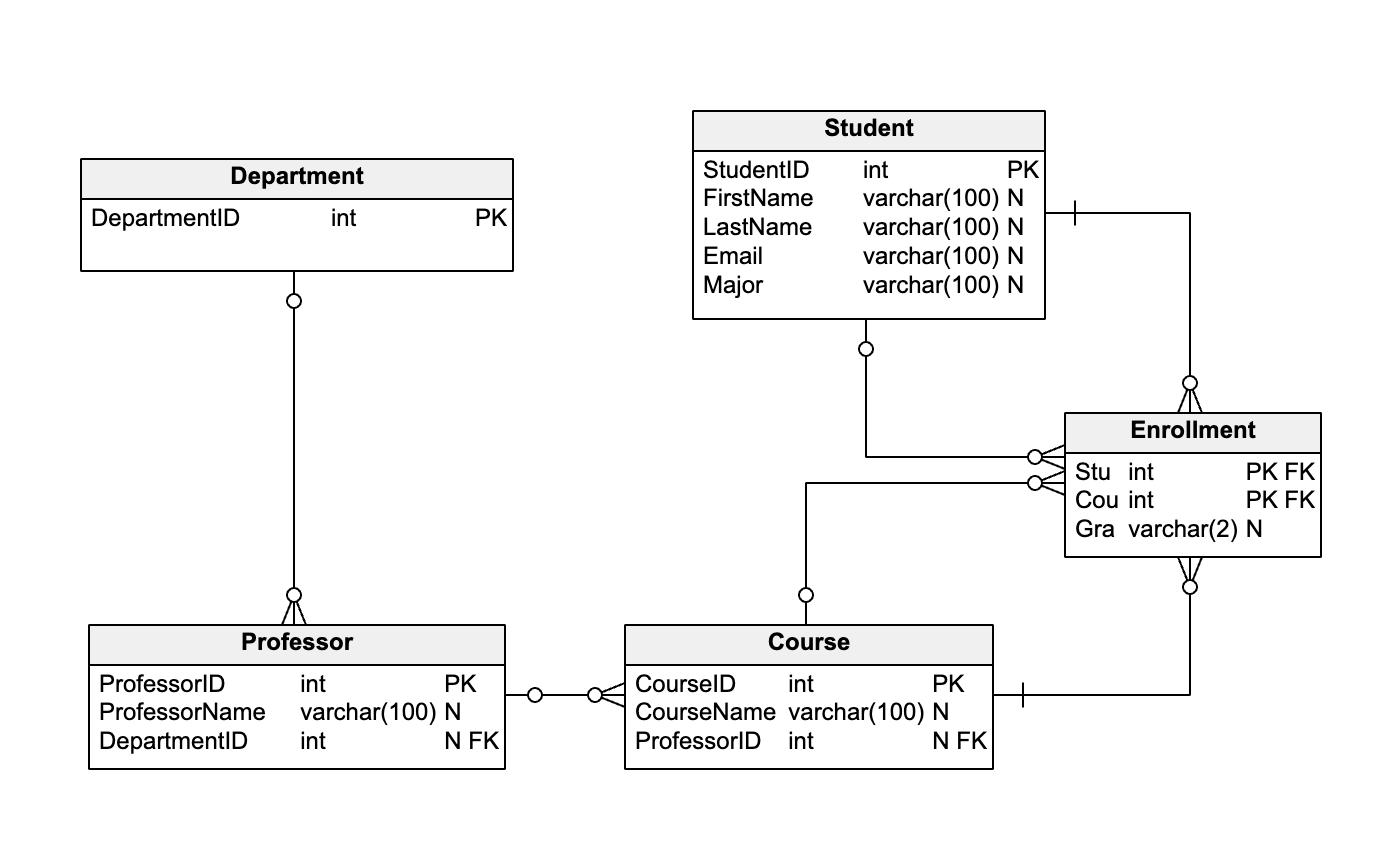
* The Customer table represents the customers who place orders.
* The Category table represents the categories to which products belong.
* The Product table represents the products that can be ordered. Each product belongs to a category, represented by the foreign key CategoryID.
* The Order table represents the orders placed by customers. Each order is placed by a customer, who is represented by the foreign key CustomerID.
* The OrderDetail table represents the many-to-many relationship between orders and products. Each row in this table represents a product contained in an order and the quantity of that product in the order.

**Description:**This diagram models an online shopping system. Customers place orders, each of which contains one or more products. Products belong to categories for easy classification and search

Example 2: University Database

**Entities:**Student, Course, Professor, Department, Enrollment.

**Relationships:** Student 'enrolls in' Course, Course 'taught by' Professor, Professor 'works in' Department.



In this schema:

* The Student table represents the students who enroll in courses.
* The Department table represents the departments in which professors work.
* The Professor table represents the professors who teach courses. Each professor works in a department, which is represented by the foreign key DepartmentID.
* The Course table represents the courses in which students enroll. Each course is taught by a professor, who is represented by the foreign key ProfessorID.
* The Enrollment table represents the many-to-many relationship between students and courses. Each row in this table represents a student enrolled in a course and the grade they received in that course.

**Description:** This diagram models a university database. Students enroll in courses, which are taught by professors. Professors work in departments, indicating their specialties.

For better understanding please read this article:

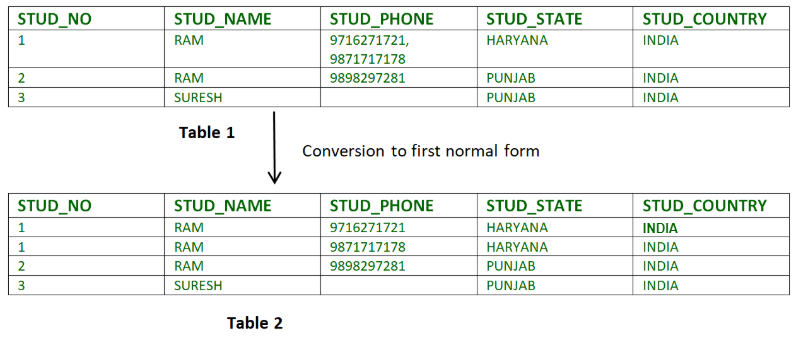
<https://365datascience.com/tutorials/sql-tutorials/sql-primary-key/>

##### **Normalization**

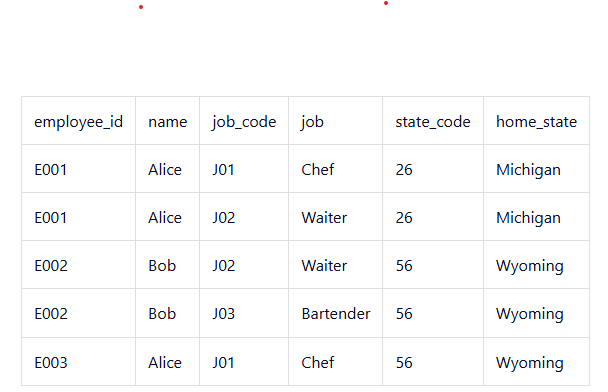
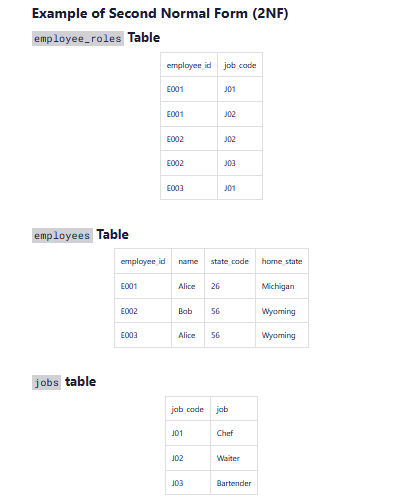
**Normalization** is the process of minimizing **redundancy** from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. **Normal forms** are used to eliminate or reduce redundancy in database tables.

**First Normal Form (1NF)**: A table is 1NF if every cell contains a single value, not a list of values. This properties is known as atomic. 1NF also prohibits repeating group of columns such as item1, item2,.., itemN. Instead, you should create another table using one-to-many relationship.

* **Example 1 –** Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD\_PHONE. Its decomposition into 1NF has been shown in table 2.

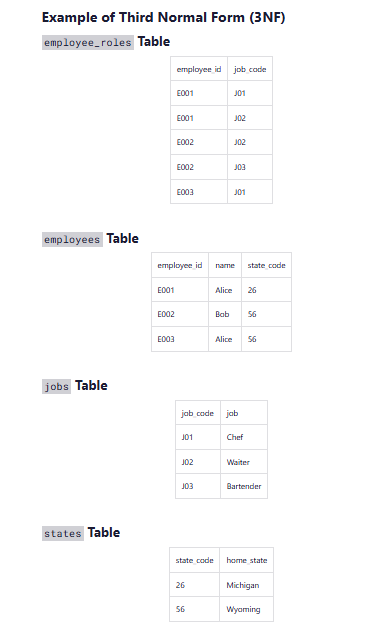


**Second Normal Form (2NF)**: A table is 2NF, if it is 1NF and every non-key column is fully dependent on the primary key. Furthermore, if the primary key is made up of several columns, every non-key column shall depend on the entire set and not part of it.

**Third Normal Form (3NF)**: A table is 3NF, if it is 2NF and the non-key columns are independent of each others. In other words, the non-key columns are dependent on primary key, only on the primary key and nothing else. for a table to be in 3NF, it must:

* be in 2NF
* have no transitive partial dependency.



**Higher Normal Form**: 3NF has its inadequacies, which leads to higher Normal form, such as Boyce/Codd Normal form, Fourth Normal Form (4NF) and Fifth Normal Form (5NF)