# **MODULE: 5 (Database)**

[**MODULE: 5 (Database)**](#_Toc164976492)

[**Basics of Database** 2](#_Toc164976493)

[**1.** **What do you understand By Database?** 2](#_Toc164976494)

[**2.** **What is Normalization?** 2](#_Toc164976495)

[**3.** **What is Difference between DBMS and RDBMS?** 2](#_Toc164976496)

[**4.** **What is MF Cod Rule of RDBMS Systems?** 3](#_Toc164976497)

[**5.** **What do you understand By Data Redundancy?** 3](#_Toc164976498)

[**6.** **What is DDL Interpreter?** 3](#_Toc164976499)

[**7.** **What is DML Compiler in SQL?** 4](#_Toc164976500)

[**8.** **What is SQL Key Constraints writing an Example of SQL Key Constraints?** 4](#_Toc164976501)

[**9.** **What is save Point? How to create a save Point write a Query?** 6](#_Toc164976502)

[**10.** **What is trigger and how to create a Trigger in SQL?** 6](#_Toc164976503)

## **Basics of Database**

## **What do you understand By Database?**

A database is a structured collection of data organized for efficient retrieval, management, and update. It is designed to manage large volumes of data by storing, retrieving, and updating information in a systematic way.

## **What is Normalization?**

Normalization is a process used in database design to organize tables and minimize redundancy within a relational database. The goal of normalization is to reduce data redundancy and improve data integrity by ensuring that data is stored logically and efficiently. This process involves breaking down large tables into smaller tables and defining relationships between them.

## **What is Difference between DBMS and RDBMS?**

The terms DBMS (Database Management System) and RDBMS (Relational Database Management System) are related but represent different types of database systems. Here are the key differences between DBMS and RDBMS:

* **DBMS:**
* A DBMS can be based on different data models, such as hierarchical, network, or relational. It manages data as files and provides basic functionalities like data storage, retrieval, and organization. Examples include IMS (Information Management System) and CODASYL databases.
* Basic DBMS systems may not enforce strict data integrity rules. They often lack features like primary key constraints, foreign key constraints, and normalization.
* DBMS systems may have their own query languages specific to their data model (e.g., hierarchical query languages for hierarchical databases).
* Basic DBMS systems may have limitations in terms of scalability and performance when handling large volumes of data or complex queries involving multiple tables.
* **RDBMS:**
* An RDBMS specifically uses the relational model to organize data into tables (relations) consisting of rows (tuples) and columns (attributes). It enforces relationships between tables using primary keys and foreign keys. Examples include MySQL, PostgreSQL, Oracle Database, and SQL Server.
* RDBMS systems enforce data integrity through primary key constraints (ensuring each row has a unique identifier), foreign key constraints (enforcing referential integrity between tables), and normalization (reducing data redundancy).
* RDBMS systems generally use SQL (Structured Query Language) as the standard language for querying and managing data. SQL allows for complex queries, joins, and data manipulation operations on relational databases.
* RDBMS systems are designed to efficiently manage large datasets and complex queries. They often include features like indexing, query optimization, and transaction management to ensure scalability and performance.

## **What is MF Cod Rule of RDBMS Systems?**

It appears that there might be a misunderstanding or a typo in your question. The term "MF Cod Rule" does not correspond to a standard or recognized concept in the context of RDBMS (Relational Database Management Systems). It's possible that you meant to refer to a specific rule or principle related to RDBMS, but the exact term "MF Cod Rule" is not familiar in this domain.

## **What do you understand By Data Redundancy?**

Data redundancy refers to the duplication of data within a database or across different databases or data storage locations. It occurs when the same piece of data is stored in multiple places unnecessarily. Redundancy can arise due to various reasons, including design flaws, denormalization, or inefficient data management practices.

Redundant data can lead to various problems such as wasted storage space, data inconsistency, and difficulties in maintaining data integrity.

## **What is DDL Interpreter?**

A DDL (Data Definition Language) interpreter is a component of a database management system (DBMS) responsible for processing and executing DDL commands. DDL is a subset of SQL (Structured Query Language) used to define and manage the structure of the database schema, including tables, indexes, constraints, and other database objects.

The primary role of a DDL interpreter is to understand and execute DDL commands issued by users or applications to define, modify, or remove database objects.

## **What is DML Compiler in SQL?**

In SQL (Structured Query Language), a DML (Data Manipulation Language) compiler is a component responsible for processing and executing DML commands. DML commands are used to retrieve, manipulate, and manage data stored in the database. The DML compiler translates DML statements into low-level instructions that the database system can execute to perform data manipulation operations.

A DML (Data Manipulation Language) compiler in SQL is responsible for processing and executing data manipulation commands such as INSERT, UPDATE, DELETE, and SELECT.

## **What is SQL Key Constraints writing an Example of SQL Key Constraints?**

SQL key constraints are rules applied to columns in database tables to enforce data integrity and define relationships between tables. There are several types of key constraints commonly used in SQL, including primary key, unique key, and foreign key constraints.

* **Primary Key Constraint**

A primary key constraint uniquely identifies each record (row) in a table. It ensures that the values in the specified column(s) are unique and not null. By default, a primary key constraint also enforces the uniqueness of values.

Example:

CREATE TABLE Students (StudentID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), Age INT );

* **StudentID** is defined as the primary key for the **Students** table.
* The **PRIMARY KEY** constraint ensures that each **StudentID** value is unique and not null.
* **Unique Key Constraint**

A unique key constraint ensures that the values in the specified column(s) are unique across all rows in the table. Unlike a primary key, a unique key constraint allows null values (except in cases where the column is also defined as NOT NULL).

**Example:**

CREATE TABLE Employees (EmployeeID INT UNIQUE, LastName VARCHAR(50), FirstName VARCHAR(50), DepartmentID INT );

* **EmployeeID** is defined with a **UNIQUE** constraint.
* The **UNIQUE** constraint ensures that each **EmployeeID** value is unique across all rows in the Employees table.
* **Foreign Key Constraint**

A foreign key constraint establishes a relationship between two tables. It enforces referential integrity by ensuring that the values in a specified column (foreign key) of one table match the values in a primary key or unique key column (referenced key) of another table.

Example:

CREATE TABLE Departments (DepartmentID INT PRIMARY KEY, DepartmentName VARCHAR(50)); CREATE TABLE Employees ( EmployeeID INT PRIMARY KEY, LastName VARCHAR(50), FirstName VARCHAR(50), DepartmentID INT, FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID) );

* The **Departments** table has a primary key defined on **DepartmentID**.
* The **Employees** table has a foreign key constraint (**DepartmentID**) that references the **DepartmentID** column in the Departments table.
* The **FOREIGN KEY** constraint ensures that values in the **DepartmentID** column of the Employees table must exist in the **DepartmentID** column of the Departments table.

## **What is save Point? How to create a save Point write a Query?**

In SQL, a savepoint is a point within a transaction to which you can roll back without rolling back the entire transaction. This means that you can set a savepoint at various points within a transaction, and if later parts of the transaction encounter errors or need to be undone, you can rollback to the specific savepoint rather than rolling back the entire transaction.

To create and use a savepoint in SQL, you can use the SAVEPOINT and ROLLBACK TO SAVEPOINT commands.

* **Syntax to create a savepoint:**

SAVEPOINT savepoint\_name;

* **Example:**

SAVEPOINT my\_savepoint;

1. **What is trigger and how to create a Trigger in SQL?**  
   In SQL, a trigger is a database object associated with a specific table or view that automatically executes in response to certain events (e.g., INSERT, UPDATE, DELETE) occurring on that table or view. Triggers are used to enforce business rules, perform data validation, maintain data integrity, and automate tasks based on changes to the database.

Triggers are classified based on the type of event that activates them:

* **BEFORE Trigger:**

Executes before the triggering event (e.g., INSERT, UPDATE, DELETE) takes place.

* **AFTER Trigger:**

Executes after the triggering event has taken place.

* **Creating a Trigger in SQL**

To create a trigger in SQL, you use the CREATE TRIGGER statement. Here is the general syntax for creating a trigger:

CREATE TRIGGER trigger\_name

{BEFORE | AFTER}

{INSERT | UPDATE | DELETE} ON table\_name [FOR EACH ROW]

BEGIN

======Trigger body =========

END;