**Module - 5**

**(Database)**

**Ques – 1) What do you understand By Database?**

**Ans – 1)** In the context of SQL (Structured Query Language), a database is a structured collection of data organized and stored in a manner that allows for efficient retrieval, management, and manipulation. Here's a breakdown of what a database encompasses:

1. **Structured Data Storage**: A database stores data in a structured format, typically in tables. Each table consists of rows and columns, where each row represents a record or entry, and each column represents a specific attribute or field of that record.
2. **Data Integrity**: Databases enforce rules to maintain data integrity, ensuring that data remains accurate, consistent, and reliable. This is often achieved through constraints such as primary keys, foreign keys, unique constraints, and data types.
3. **Data Retrieval and Manipulation**: SQL provides a standardized language for querying and manipulating data within a database. With SQL, users can retrieve specific data, insert new records, update existing records, and delete records based on various criteria.
4. **Concurrency Control**: Databases support multiple users accessing and modifying data simultaneously. Concurrency control mechanisms ensure that transactions are executed in a manner that maintains data consistency and prevents conflicts between concurrent transactions.
5. **Security**: Databases offer features to control access to data, ensuring that only authorized users can view or modify sensitive information. This includes user authentication, authorization, and auditing capabilities.
6. **Scalability and Performance**: Databases are designed to scale as data volumes and user loads grow. They employ various optimization techniques such as indexing, query optimization, and data partitioning to ensure efficient performance even with large datasets and high user concurrency.

**Ques – 2) What is Normalization?**

**Ans – 2)** Normalization in SQL is a process of organizing the attributes and tables of a relational database to minimize redundancy and dependency. It involves breaking down a large table into smaller tables and defining relationshipsbetween them. The main goals of normalization are to reduce data redundancy, improve data integrity, and make the database more flexible and adaptable to changes.

There are different normal forms, each addressing different aspects of data redundancy and dependency. The most commonly used normal forms are:

1. **First Normal Form (1NF)**: Ensures that each column in a table contains atomic values and there are no repeating groups of columns.
2. **Second Normal Form (2NF):** Ensures that the table is in 1NF and all non-key attributes are fully functionally dependent on the primary key.
3. **Third Normal Form (3NF**): Ensures that the table is in 2NF and there are no transitive dependencies between non-key attributes.

**Ques – 3) What is Difference between DBMS and RDBMS?**

**Ans – 3)** **DBMS (Database Management System)** and **RDBMS (Relational Database Management System)** are both systems used to manage databases, but they differ in their structure and functionality:

1. **DBMS (Database Management System):**
   * A DBMS is a software system that allows users to interact with a database. It provides tools and interfaces for creating, accessing, and managing databases.
   * In a DBMS, data is typically stored in a hierarchical or navigational structure, and relationships between data are not well-defined.
   * DBMS systems are not necessarily based on the relational model and may not support features like data integrity constraints, transactions, and relational operations.
2. **RDBMS (Relational Database Management System):**
   * An RDBMS is a type of DBMS that is based on the relational model of data.
   * In an RDBMS, data is organized into tables with rows and columns, and relationships between tables are defined using foreign keys.
   * RDBMS systems enforce data integrity through features like primary keys, foreign keys, and referential integrity constraints.
   * RDBMS systems also support SQL (Structured Query Language) for querying and manipulating data in the database.
   * Examples of RDBMS include MySQL, PostgreSQL, Oracle Database, SQL Server, and SQLite.

**Ques - 4) What is MF Cod Rule of RDBMS Systems?**

**Ans – 4) Codd's Rules: A Foundation for Relational Database Integrity and Efficiency**

Developed by E.F. Codd, a computer scientist, these twelve rules establish a framework, also known as the relational model, for defining the core characteristics of a true RDBMS. While not all RDBMS adhere to all twelve rules strictly, they provide a benchmark for ensuring data integrity, consistency, and efficient management in relational databases.

**Essence of Codd's Rules:**

* **Data Organization in Tables**: Information is stored in a structured format called relations (essentially tables) with rows and columns, providing a clear and organized representation of data.
* **Unique Row Identification:** Each row (tuple) within a table is uniquely identified using a primary key, ensuring no duplicate entries and enabling efficient data retrieval**.**
* **Data Manipulation Through a High-Level Language:** A standardized language, typically SQL (Structured Query Language) in most RDBMS, allows users to interact with the database for various operations like querying, modifying, and managing data.
* **Data Independence:** Changes to the physical storage or access methods of the database shouldn't affect the user's logical view of the data or the way they interact with it using the query language.
* **Data Integrity:** Mechanisms are in place to maintain data consistency and validity, preventing invalid or inconsistent data from being stored or manipulated.
* **Data Atomicity and Isolation:** Database transactions (a series of database operations) are treated as atomic units, meaning either they are all completed successfully or none of them.

**Importance of Codd's Rules:**

Codd's rules provide a valuable foundation for designing, implementing, and working with RDBMS effectively. By adhering to these principles, developers and users can:

* **Enhance Data Integrity:** The rules help prevent data inconsistencies and ensure accuracy, reducing the risk of errors and corrupt information.
* **Simplify Data Management:** The structured approach and standardized language (like SQL) make it easier to manage and query data, improving efficiency and productivity.
* **Promote Cross-Platform Compatibility:** Understanding the core tenets of the relational model helps in working with different RDBMS from various vendors, as many of them adhere to these principles to varying degrees.
* **Facilitate Data Sharing and Interoperability:** By adhering to a common framework (the relational model), it becomes easier to share data between different RDBMS, fostering collaboration and data exchange.

**Ques – 5) What do you understand By Data Redundancy?**

**Ans – 5)** In SQL, data redundancy occurs when the same data is stored in multiple places within a database. This can happen due to unnormalized database design, where tables contain duplicate information. For example, if a database has multiple tables with a column for ‘Customer Address’, and this address is stored in each table, that’s data redundancy.

To address this, SQL databases use normalization, which involves organizing data into related tables and establishing relationships between them using keys. This minimizes redundancy, ensuring that each piece of data is stored only once.

**Que – 6) What is DDL Interpreter?**

**Ans – 6)** A DDL Interpreter is a component of a database management system that processes Data Definition Language (DDL) statements. These statements are used to define and modify the database schema, which includes creating, altering, and dropping tables and other database objects.

[For example, when you execute a CREATE TABLE statement in SQL, the DDL Interpreter processes this statement and creates a new table in the database, along with its metadata that defines the table’s structure, such as column names, data types, and constraints](https://www.w3schools.in/mysql/ddl-dml-dcl).This allows the database to understand how data is stored and how it should be accessed and manipulated.

**Que – 7) What is DML Compiler in SQL?**

**Ans – 7)** In SQL, DML stands for Data Manipulation Language, and a DML compiler is a component of a database management system (DBMS) that is responsible for processing and executing data manipulation commands. These commands include INSERT, UPDATE, DELETE, and sometimes SELECT statements that modify data stored in the database.

The DML compiler takes the SQL statements written by users or applications and converts them into an internal representation that the DBMS can understand and execute. This process involves parsing the SQL statements, optimizing them for efficient execution, and generating an execution plan that specifies how the data manipulation should be carried out.

The execution plan produced by the DML compiler is then passed to other components of the DBMS, such as the query executor, which actually performs the data manipulation operations on the database. The DML compiler plays a crucial role in ensuring that data manipulation commands are executed accurately and efficiently, contributing to the overall performance and reliability of the database system.

**Que – 8) What is SQL Key Constraints writing an Example of SQL Key Constraints?**

**Ans)** SQL Key Constraints are rules that enforce the integrity and uniqueness of data within a relational database table. These constraints ensure that certain columns or combinations of columns adhere to specific rules, such as uniqueness or referential integrity. The main types of key constraints in SQL are:

1. Primary Key Constraint: This constraint ensures that a column or a combination of columns uniquely identifies each row in a table. It enforces the uniqueness of values within the specified column(s) and also ensures that the column(s) cannot contain NULL values.
2. Unique Constraint: Similar to the primary key constraint, a unique constraint ensures that the values in a specified column or combination of columns are unique across all rows in the table. However, unlike a primary key constraint, unique constraints allow NULL values in the column(s).
3. Foreign Key Constraint: This constraint establishes a relationship between two tables by enforcing referential integrity. It ensures that the values in a column (or set of columns) in one table match the values in another table's referenced column (usually the primary key). This constraint helps maintain consistency and prevents orphaned records.

-- Creating a table with primary key constraint

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT);

-- Creating a table with unique constraint

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(50) UNIQUE,

ManagerID INT);

-- Creating a table with foreign key constraint

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

OrderDate DATE,

CustomerID INT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) );

**Que – 9) What is save Point? How to create a save Point write a Query?**

**Ans – 9)** A save point in SQL is a point within a transaction where you can mark a specific point in the transaction to which you can later rollback if needed. Save points provide a way to break a transaction into smaller parts and rollback to a specific point if an error occurs or if you need to undo part of the transaction.

To create a save point in SQL, you typically use the SAVEPOINT statement followed by the name you want to give to the save point.

SAVEPOINT savepoint\_name;

START TRANSACTION; -- Starting a transaction

-- Performing some operations within the transaction

INSERT INTO employees (employee\_id, first\_name, last\_name) VALUES (101, 'John', 'Doe');

-- Creating a savepoint

SAVEPOINT my\_savepoint;

-- Performing more operations within the transaction

UPDATE employees SET last\_name = 'Smith' WHERE employee\_id = 101;

-- If needed, rollback to the savepoint

ROLLBACK TO SAVEPOINT my\_savepoint;

-- Performing additional operations or committing the transaction

COMMIT;

**Que – 10) What is trigger and how to create a Trigger in SQL?**

**Ans)** In SQL, a trigger is a special type of stored procedure that automatically executes in response to certain events or actions occurring in the database. These events can include INSERT, UPDATE, DELETE operations on a table, or even database schema changes. Triggers are useful for enforcing business rules, maintaining data integrity, and automating tasks within the database.

**Example :**

CREATE [OR REPLACE] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF} {INSERT | UPDATE | DELETE} ON table\_name

[FOR EACH ROW]

[WHEN condition]

BEGIN

-- SQL statements to execute when the trigger fires

END;

CREATE OR REPLACE TRIGGER employee\_audit

BEFORE INSERT OR UPDATE OR DELETE ON employees

FOR EACH ROW

BEGIN

IF INSERTING THEN

INSERT INTO audit\_table (action, employee\_id, action\_date)

VALUES ('INSERT', :NEW.employee\_id, SYSDATE);

ELSIF UPDATING THEN

INSERT INTO audit\_table (action, employee\_id, action\_date)

VALUES ('UPDATE', :OLD.employee\_id, SYSDATE);

ELSIF DELETING THEN

INSERT INTO audit\_table (action, employee\_id, action\_date)

VALUES ('DELETE', :OLD.employee\_id, SYSDATE);

END IF;

END;