**SE**

**MODULE: 5 (Database)**

**Q1. What do you understand By Database?**

**A1.** A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a **Database Management System(DBMS)**. Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can be easily accessed, managed, modified, updated, controlled, and organized. Most databases use **Structured Query Language(SQL)** for writing and querying data.

**Q2. What is Normalization?**

**A2. Normalization** is a process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. Thus, normalization helps to minimize redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

**Q3. What is the difference between DBMS and RDBMS?**

**A3.**

| **DBMS** | **RDBMS** |
| --- | --- |
| DBMS stores data as a file. | RDBMS stores data in tabular form. |
| Data elements need to be accessed individually. | Multiple data elements can be accessed at the same time. |
| There is no relationship between data. | Data is stored in the form of tables which are related to each other. |
| Normalization is not present | Normalization is present |
| Does not support distributed databases. | Supports distributed databases. |
| Deals with a small quantity of data. | Deals with large amounts of data. |
| Data redundancy is common in this model. | Keys and indexes do not allow data redundancy. |
| It is used for small organizations and deals with a small amount of data. | It is used to handle large amounts of data. |
| Low software and hardware necessities. | Higher software and hardware necessities. |
| Eg: XML, Forxpro, etc. | Eg: MySQL, PostgreSQL, Oracle, etc. |

**Q4. What is the MF Cod Rule of RDBMS Systems?**

**A4. Codd’s** **12 Principles** of Relational Databases are as follows:

1. Information is represented logically in tables.
2. Data must be logically accessible by table, primary key, and column.
3. Null values must be uniformly treated as “missing information”, and not as empty strings, blanks, or zeros.
4. The database catalog, which contains metadata about the database, must be stored and accessed using the same relational database management system.
5. A single language must be able to define data, views, integrity constraints, authorization, transactions, and data manipulation.
6. The system must also update all views that are theoretically updateable.
7. A successful database system must possess the feature of facilitating high-level insertions, updates, and deletions that can grant users the ability to conduct these operations with ease through a single query.
8. Application programs and activities should remain unaffected when changes are made to the physical storage structures or methods.
9. Application programs and activities should remain unaffected when changes are made to the logical structure of the data, such as adding or modifying tables.
10. Integrity constraints should be specified separately from application programs and stored in the catalog. They should be automatically enforced by the database system.
11. The distribution of data across multiple locations should be invisible to users, and the database system should handle the distribution transparently
12. If the interface of the system is providing access to low-level records, then the interface must not be able to damage the system and bypass security and integrity constraints.

**Q5. What do you understand By Data Redundancy?**

**A5. Data Redundancy** is when multiple copies of the same information are stored in more than one place at a time. It can cause data inconsistency, which can provide a company with unreliable and meaningless data.

**Q6. What is a DDL Interpreter?**

**A6.** A **DDL(Data Definition Language)** interprets the DDL instructions and stores the record in a data dictionary.

**Q7. What is a DML Compiler in SQL?**

**A7.** A **DML(Data Manipulation Language)** compiler in SQL is a component of a database management system (DBMS) that translates DML statements into low-level instructions that the database can execute. DML statements are used to manipulate data within the database and include operations such as **INSERT**, **UPDATE**, **DELETE**, and **SELECT**.

**Q8. What is SQL Key Constraints? Write an Example of SQL Key**

**Constraints.**

**A8.** SQL key constraints are rules applied to table columns to enforce the integrity and uniqueness of data. They ensure that the data in the database adheres to specific rules and relationships, which helps maintain the consistency and accuracy of the data.

**Types of SQL Constraints**

1. **Primary Key Constraint**

* Ensures that each row in a table has a unique identifier.
* A table can have only one primary key, which can consist of one or multiple columns
* **Eg:** CREATE TABLE employees

(emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100),

emp\_salary DECIMAL(10,2));

1. **Foreign Key Constraint**

* Enforces a link between the data in two tables.
* Ensures that the value in a column (or a set of columns) matches the value in another column (or a set of columns) in another table, establishing referential integrity.
* **Eg:** CREATE TABLE department

(dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(100));

CREATE TABLE employees

(emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100),

emp\_salary DECIMAL(10,2),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES department(dept\_id));

1. **Unique Constraint**

* Ensures all values in a column (or a set of columns) are unique across the table.
* **Eg:** CREATE TABLE employees

(emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100),

emp\_salary DECIMAL(10,2)

Email VARCHAR(100) UNIQUE);

1. **Not Null Constraint**

* Ensures that a column cannot have a NULL value.
* **Eg:** CREATE TABLE employees

(emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100) NOT NULL,

emp\_salary DECIMAL(10,2));

SQL key constraints are essential for maintaining data integrity and consistency in relational databases. They define rules to enforce uniqueness, relationships, and mandatory values in database tables. By using key constraints, you can ensure that the data stored in your database is accurate and adheres to the business rules.

**Q9. What is Save Point? How to create a save Point and write a Query?**

**A9.** A **savepoint** in SQL allows you to set a point within a transaction to which you can roll back if necessary. This provides granular control over transactions, enabling partial rollbacks.

**Syntax:** SAVEPOINT savepoint\_name;

**CREATING AND USING SAVEPOINTS**

1. **Begin Transaction:** Start the transaction.

BEGIN TRANSACTION;

1. **Create savepoint:** Create Savepoints.

SAVEPOINT savepoint1;

1. **Execute some operations:** Performing update operation.

UPDATE accounts SET balance = balance - 100 WHERE acc\_id=1;

1. **Create another savepoint:** Create Savepoints.

SAVEPOINT savepoint2;

1. **Execute another operations:** Performing insert operation.

INSERT INTO accounts(acc\_id, balance) VALUES(2, 1500.00);

1. **Rollback to savepoint if needed:** Rollback to ‘savepoint2’ if needed.

ROLLBACK TO SAVEPOINT savepoint2;

1. **Commit the transaction:** Commit the changes.

COMMIT;

**Q10. What is a trigger and how to create a Trigger in SQL?**

**A10.** A **trigger** in SQL is a special kind of stored procedure that automatically executes in response to certain events on a particular table or view. Triggers are often used to enforce business rules, validate input data, maintain audit trails, and implement complex security authorizations.

**Types of Triggers**

1. **Before Triggers**: Execute before an insert, update, or delete operation.
2. **After Triggers**: Execute after an insert, update, or delete operation.
3. **Instead Of Triggers**: Execute instead of an insert, update, or delete operation (often used with views).

**Syntax:** CREATE TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF} {INSERT | DELETE | UPDATE}

ON table\_name [FOR EACH ROW]

BEGIN

--SQL statements

END;

**Eg:** CREATE TRIGGER account\_backup AFTER DELETE ON accounts FOR EACH ROW

INSERT INTO acc\_backup VALUES (old.acc\_id, old.balance);