Ques-1:- What do you understand By Database ?

A database is a logical grouping of data. It contains the data associated with one application or with a group of related applications. It includes a collection of related table spaces and index spaces. Databases are designed to support data integrity, security, and concurrency control, ensuring that data remains accurate and accessible to users and applications. Common database systems include relational databases (like MySQL, PostgreSQL), NoSQL databases (like MongoDB, Redis), and others tailored for specific data types and use cases.

Ques-2:- What is Normalization ?

Normalization is the process of organizing data to avoid duplication and redundancy.

1. To minimize duplicate data
2. To minimize or avoid data modification issues
3. To simplify queries

Ques-3:- What is Difference between DBMS and RDBMS ?

A database management system supports the development and administration of database platforms. An RDBMS is an advanced version of a DBMS that saves data in a row-based table structure. The file storage used in RDBMS is different than DBMS.

|  |  |
| --- | --- |
| **DBMS** | **RDBMS** |
| Allows one user at a time | Multiple users at a time |
| Handles a small amount of data. | Takes any amount of data |
| Doesn't Support | Supports |
| DBMS cannot be normalized | RDBMS can be normalized |
| File storage | Tabular structure |
| DBMS will not support | RDBMS provides complete support |
| Lack of security | Good data security due to several log files |

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

Codd's original 12 rules were intended to establish the theoretical foundation and requirements for relational database systems. Here is a summary of some of the key principles from Codd's rules:

1. Information Rule
2. Systematic Treatment of NULL Values
3. Guaranteed Access Rule
4. Comprehensive Data Sublanguage Rule
5. Integrity Independence
6. High-Level Insert, Update, and Delete Rule
7. Non-Subversion Rule
8. Active Online Catalog
9. View Updating Rule
10. Physical Data Independence
11. Logical Data Independence
12. Distribution Independence

Ques-5:- What do you understand By Data Redundancy ?

Data redundancy refers to the situation where the same piece of data is stored in multiple places within a database or across multiple databases. This redundancy can occur intentionally or unintentionally due to poor database design or application architecture.

Data redundancy can lead to several issues:

1. **Wasted Storage Space:** Storing the same data multiple times consumes unnecessary storage space, which can increase costs and resource usage.

2. **Data Inconsistency:** When the same data is stored in multiple locations, there is a risk of inconsistencies arising if the data is updated in one place but not updated in another. This can lead to discrepancies and incorrect information being used in different parts of the system.

3. **Increased Complexity:** Managing redundant data adds complexity to the database design and application logic. It can make it harder to maintain and update the system.

4. **Decreased Data Integrity:** Redundant data can compromise data integrity, as updates or deletions of data may not be properly synchronized across all instances of the data.

Ques-6:- What is DDL Interpreter ?

DDL (Data Definition Language) interpreter is a component of a database management system (DBMS) that processes DDL statements to define or modify the structure of a database. It translates DDL commands like `CREATE`, `ALTER`, `DROP`, and `TRUNCATE` into low-level actions that create, modify, or delete database objects such as tables, indexes, and views. The DDL interpreter updates metadata to reflect these changes and enforces data integrity constraints specified in the DDL statements. Its main role is to manage the schema and structure of the database based on user-defined commands.

Ques-7:- What is DML Compiler in SQL ?

Data Manipulation Language (DML) statements like `SELECT`, `INSERT`, `UPDATE`, and `DELETE`,

- **Query Parser:** Checks the syntax and structure of DML statements.

- **Query Optimizer:**Analyzes and determines the most efficient way to execute DML statements.

- **Query Executor:** Interacts with the database storage engine to perform data manipulation operations specified by the DML statements.

- **Transaction Manager:** Manages transactions to ensure data integrity and consistency during DML operations.

Ques-8:- What is SQL Key Constraints writing an Example of SQL Key Constraints ?

SQL key constraints are rules or conditions applied to columns in a database table to enforce data integrity and define relationships between tables. There are several types of key constraints in SQL:

1. **Primary Key Constraint:**

- A primary key constraint uniquely identifies each record in a table and ensures that there are no duplicate values in the specified column or columns.

- Example:

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

student\_nameVARCHAR(50),

age INT

);

2. **Unique Key Constraint:**

- A unique key constraint ensures that all values in a specified column or columns are unique (i.e., no duplicates), but unlike a primary key, it allows `NULL` values (except for columns defined as `NOT NULL`).

- Example:

CREATE TABLE Employees (

employee\_id INT UNIQUE,

employee\_nameVARCHAR(50),

department\_id INT

);

3. **Foreign Key Constraint:**

- A foreign key constraint establishes a relationship between two tables by enforcing referential integrity. It ensures that values in a column (or columns) of one table match the values in another table's primary key or unique key.

- Example:

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

customer\_nameVARCHAR(50),

email VARCHAR(100)

);

Ques-9:- What is save Point? How to create a save Point write a Query ?

A save point in SQL is a point within a transaction where you can set a marker to which you can later roll back if necessary, while still allowing portions of the transaction to be committed. This feature is useful when you want to divide a transaction into smaller segments.

Here's an example of how to create and use a save point within a transaction:

-- Start a transaction

START TRANSACTION;

-- Execute SQL statements within the transaction

INSERT INTO employees (id, name, salary) VALUES (1, 'John', 50000);

INSERT INTO employees (id, name, salary) VALUES (2, 'Jane', 60000);

-- Create a savepoint named 'before\_update'

SAVEPOINT before\_update;

-- Update salary for employee with ID 1

UPDATE employees SET salary = 55000 WHERE id = 1;

-- Check the updated records

SELECT \* FROM employees;

-- Check the records after rolling back

SELECT \* FROM employees;

-- Commit the transaction

COMMIT;

Ques-10:- What is trigger and how to create a Trigger in SQL ?

In SQL, a trigger is a special type of stored procedure that automatically executes in response to specific events (like `INSERT`, `UPDATE`, `DELETE`) occurring on a database table. Triggers are used to enforce business rules, validate data, maintain integrity, or automate tasks based on data changes.

To create a trigger:

- Use `CREATE TRIGGER` statement with specified timing (`BEFORE` or `AFTER`) and event (`INSERT`, `UPDATE`, `DELETE`).

- Specify the table on which the trigger will act (`ON table\_name`).

- Define the trigger's actions within `BEGIN` and `END` block.

Example:

CREATE TRIGGER update\_last\_updated

AFTER UPDATE

ON employees

FOR EACH ROW

BEGIN

UPDATE employees

SET last\_updated = CURRENT\_TIMESTAMP

WHERE id = :OLD.id;

END;