1. **What do you understand By Database.**

-------> A database is an organized collection of structured information or data, typically stored electronically in a computer system. It’s usually managed by a database management system (DBMS). Together, the data and the DBMS form a database system. Most databases use Structured Query Language (SQL) for writing and querying data. [Databases allow efficient storage, management, modification, and organization of data, often modeled in rows and columns within tables1](https://www.oracle.com/database/what-is-database/). If you have any specific questions about databases, feel free to ask.

**2) What is Normalization?**

-------> Normalization is a database design technique that organizes data to minimize redundancy and improve data integrity. The process involves dividing a database into two or more tables and defining relationships between them. The main goals of normalization are to eliminate redundant data, ensure data dependencies make sense, and protect the database against certain types of logical or structural problems, such as update anomalies. It typically involves multiple stages, or normal forms, each with specific rules and requirements.

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

**--------- DBMS---------**

-> Stores data in a variety of formats such as hierarchical, network, or object-oriented models.

-> Does not strictly enforce ACID (Atomicity, Consistency, Isolation, Durability) properties.

-> May not enforce a strict schema and does not necessarily support relationships between data elements.

-> May use a variety of query languages or APIs specific to the data model.

-> Typically designed for smaller systems and may not provide robust support for multiple concurrent users.

->Examples include file systems, XML databases, and some NoSQL databases.

**---------RDBMS ---------**

-> Stores data in tables (rows and columns) and uses a relational model to define relationships between these tables.

-> Strictly enforces ACID properties to maintain data integrity and consistency.

-> Requires a predefined schema and uses primary and foreign keys to establish and enforce relationships between tables.

-> Primarily uses SQL (Structured Query Language) for querying and managing data.

->Designed to handle large systems and multiple concurrent users with strong transaction management and concurrency control.

-> Examples include MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.

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

---> **Information Rule**: All information in a relational database is represented explicitly at the logical level and in exactly one way – by values in tables.

 **Guaranteed Access Rule**: Each data item (atomic value) in a relational database is guaranteed to be logically accessible by using a combination of table name, primary key, and column name.

 **Systematic Treatment of Null Values**: Null values (distinct from empty character strings or a string of blank characters and distinct from zero or any other number) are supported for representing missing information and inapplicable information in a systematic way, independent of data type.

 **Dynamic Online Catalog Based on the Relational Model**: The database must support a catalog (data dictionary) that describes the database and is itself relational, allowing authorized users to query the catalog using the same relational language they use to query the database data.

 **Comprehensive Data Sublanguage Rule**: A relational system may support several languages and various modes of terminal use (for example, the fill-in-blanks mode). However, there must be at least one language whose statements are expressible as character strings conforming to a well-defined syntax, and that language must support data definition, data manipulation, and transaction management operations.

 **View Updating Rule**: All views that are theoretically updateable must be updateable through the system.

 **High-Level Insert, Update, and Delete**: The system must support set-at-a-time insert, update, and delete operations. This means that data can be retrieved and manipulated in sets rather than as individual rows.

 **Physical Data Independence**: Changes to the physical level (how the data is stored, for example) must not require a change to an application based on the structure.

 **Logical Data Independence**: Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure.

 **Integrity Independence**: Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and stored in the catalog, not in the application programs.

 **Distribution Independence**: A relational DBMS has distribution independence, which means that applications are not affected by the distribution of the data (distribution of data across different locations).

 **Non-Subversion Rule**: If a relational system has a low-level (single-record-at-a-time) language, that low-level language cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher-level relational language.

 **Rule Zero**: This was added by Codd later, which states that for a system to qualify as an RDBMS, it must be able to manage databases entirely through its relational capabilities.

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

Data redundancy occurs when the same piece of data is stored in multiple places within a database or across different databases. This can lead to several issues, including:

1. **Increased Storage Costs**: Storing the same data multiple times consumes more disk space, leading to higher storage costs.
2. **Data Inconsistency**: If the redundant copies of data are not updated simultaneously, it can result in data inconsistency, where different copies of the data contain different values.
3. **Maintenance Overhead**: Managing and updating multiple copies of the same data requires more effort and increases the complexity of database maintenance.
4. **Slower Performance**: Redundant data can slow down query performance because the database has to handle larger volumes of data.

To minimize data redundancy, normalization techniques are used in database design, which involve organizing data into tables and defining relationships to ensure that each piece of data is stored only once

6) What is DDL Interpreter?

🡪A DDL (Data Definition Language) interpreter is a component of a database management system (DBMS) responsible for interpreting and executing DDL statements. DDL statements are used to define and manage database structures such as tables, indexes, schemas, and views. Common DDL statements include CREATE, ALTER, DROP, and TRUNCATE.

The DDL interpreter performs the following functions:

1. **Parsing**: Analyzes the DDL statements to ensure they are syntactically correct.
2. **Validation**: Checks the validity of the statements, ensuring they adhere to the rules and constraints of the database system.
3. **Execution**: Executes the statements, which involves creating, modifying, or deleting database objects as specified.
4. **Updating Metadata**: Updates the system catalog or data dictionary with information about the new or modified database objects.

For example, when a CREATE TABLE statement is issued, the DDL interpreter parses the statement, validates the table structure and constraints, creates the table in the database, and updates the metadata to reflect the new table's existence.

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

🡪 The Data Manipulation Language (DML) in SQL deals with data manipulation, including common statements like:

SELECT: Retrieves data from one or more tables.

INSERT: Adds data into a table.

UPDATE: Modifies existing data within a table.

DELETE: Removes records from a table.

MERGE: Performs an “upsert” operation (insert or update).

CALL: Invokes a PL/SQL or Java subprogram.

EXPLAIN PLAN: Interprets the data access path.

LOCK TABLE: Manages concurrency control.

[Remember, DML statements are not auto-committed, meaning changes can be rolled back if necessary](https://stackoverflow.com/questions/2578194/what-are-ddl-and-dml)

**8) What is SQL Key Constraints writing an Example of SQL Key Constraints.**

🡪SQL key constraints are rules applied to columns in a relational database table to enforce data integrity and ensure the accuracy and reliability of the data. These constraints include primary keys, foreign keys, unique keys, and more. Here are the main types of key constraints with examples:

Primary Key Constraint:

Ensures that each row in a table has a unique identifier and that no null values are allowed.

Example:

sql

Copy code

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50)

);

Foreign Key Constraint:

Establishes a relationship between two tables by ensuring that the value in a column (or a group of columns) matches a value in the primary key column of another table.

Example:

sql

Copy code

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(50)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);

Unique Constraint:

Ensures that all values in a column (or a group of columns) are unique across the table, allowing null values unless otherwise specified.

Example:

sql

Copy code

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100) UNIQUE

);

Composite Key Constraint:

A composite key is a primary key composed of multiple columns to ensure the uniqueness of the combination of values in these columns.

Example:

sql

Copy code

CREATE TABLE Orders (

OrderID INT,

ProductID INT,

Quantity INT,

PRIMARY KEY (OrderID, ProductID)

);

Check Constraint:

Ensures that the values in a column meet a specific condition.

Example:

sql

Copy code

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Age INT,

CHECK (Age >= 18)

);

These constraints help maintain data integrity by ensuring that the data adheres to specific rules and relationships.

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

🡪 A SAVEPOINT is a marker within a transaction that allows you to create a point to which you can later roll back. It’s useful for partial rollbacks, nested transactions, and better error handling.

-- Start a transaction

BEGIN TRANSACTION;

-- Insert a row into the Employees table

INSERT INTO Employees (EmployeeID, FirstName, LastName) VALUES (1, 'John', 'Doe');

-- Create a savepoint named sp1

SAVEPOINT sp1;

-- Insert another row into the Employees table

INSERT INTO Employees (EmployeeID, FirstName, LastName) VALUES (2, 'Jane', 'Smith');

-- Create another savepoint named sp2

SAVEPOINT sp2;

-- Insert yet another row into the Employees table

INSERT INTO Employees (EmployeeID, FirstName, LastName) VALUES (3, 'Emily', 'Jones');

-- Roll back to the savepoint sp2

ROLLBACK TO SAVEPOINT sp2;

-- Commit the transaction

COMMIT;

>

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

**🡪** A trigger in SQL is a set of instructions that automatically execute (or "trigger") in response to specific events on a particular table or view. These events can include INSERT, UPDATE, or DELETE operations. Triggers are useful for maintaining data integrity, enforcing business rules, and keeping audit trails.

-- Create the EmployeeChanges table to log changes

CREATE TABLE EmployeeChanges (

ChangeID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeID INT,

ChangeType VARCHAR(50),

ChangeDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

OldFirstName VARCHAR(50),

NewFirstName VARCHAR(50),

OldLastName VARCHAR(50),

NewLastName VARCHAR(50)

);

-- Create the trigger

CREATE TRIGGER AfterEmployeeUpdate

AFTER UPDATE ON Employees

FOR EACH ROW

BEGIN

INSERT INTO EmployeeChanges (EmployeeID, ChangeType, OldFirstName, NewFirstName, OldLastName, NewLastName)

VALUES (OLD.EmployeeID, 'UPDATE', OLD.FirstName, NEW.FirstName, OLD.LastName, NEW.LastName);

END;