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

* DBMS:-

DBMS stands for Data Base Management System. Data + Management System Database is a collection of inter-related data and Management System is a set of programs to store and retrieve those data. DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

DBMS Here is a list of some popular database.

• MySQL

• Microsoft Access

• Oracle

A database is a structured and organized collection of data that is designed to be easily accessed, managed, and updated. It is a fundamental component of information systems and plays a crucial role in storing, retrieving, and manipulating data for various purposes. Databases are used in a wide range of applications, including business, science, government, education, and more. Here are some key characteristics and components of a database:

**Data Structure**:

Databases organize data into structured formats, such as tables, records, and fields. This structure allows for efficient storage and retrieval of information.

**Data Management:**

Databases provide mechanisms for adding, modifying, and deleting data. These operations are typically performed using queries and commands defined by a query language, such as SQL (Structured Query Language).

**Data Security:**

Databases often incorporate security measures to control access to data, protecting it from unauthorized users. This can involve user authentication, role-based access control, and encryption.

**Relational Databases:**

Relational databases are a common type of database that use tables to organize data, and they establish relationships between tables using keys. Examples include MySQL, PostgreSQL, and Oracle Database.

1. **What is Normalization?**

* Normalization is the process of minimizing redundancy (duplicity) from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updating anomalies. So, it helps to minimize the redundancy in relations. Most Commonly used normal forms:

**First Normal Form:**

* First normal form(1NF) Second normal form(2NF) Third normal form(3NF) Boyce & Code normal form (BCNF) If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute.
* A relation is in first normal form if every attribute in that relation is singled valued attribute. Database Normalization

**Second Normal Form:**

• To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency.

• Relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

• Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency. Database Normalization

**Third Normal Form:**

• A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.

• A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y X is a super key. Y is a prime attribute (each element of Y is part of some candidate key).

• Transitive dependency – If A->B and B->C are two FDs then A->C is called transitive dependency.

1. **What is Difference between DBMS and RDBMS?**

* DBMS (Database Management System) and RDBMS (Relational Database Management System) are both types of software systems designed to manage databases, but they have distinct differences in terms of data organization and management. Here's a breakdown of the key differences between DBMS and RDBMS:

**Data Organization:**

* DBMS: A DBMS can manage various types of data, including structured, semi-structured, and unstructured data. It doesn't enforce a specific structure for the data and may not support relationships between data elements.
* RDBMS: An RDBMS is a subset of DBMS that specifically manages structured data. It enforces a tabular structure, where data is organized into tables with rows and columns. RDBMS systems are designed to support relationships between tables through keys.

**Data Integrity:**

* DBMS: DBMS systems may not enforce strong data integrity constraints. They may allow for more flexibility in terms of data entry and relationships.
* RDBMS: RDBMS systems enforce data integrity constraints, such as primary keys, foreign keys, and unique constraints, to maintain data accuracy and consistency. This ensures that data is stored in a structured and standardized manner.

**Query Language:**

* DBMS: DBMS systems may support a variety of query languages and data manipulation techniques, depending on the specific system.
* RDBMS: RDBMS systems typically use SQL (Structured Query Language) as the primary query language. SQL is designed specifically for querying and manipulating structured data in tables.

**Relationships:**

* DBMS: DBMS may not support or enforce relationships between data elements. Data organization is often more flexible, and it may not use the concept of foreign keys to link tables.
* RDBMS: RDBMS systems are built around the concept of relationships between tables. They use foreign keys to establish and maintain these relationships, ensuring referential integrity.

**Examples:**

* DBMS Examples: MongoDB (a NoSQL DBMS), CouchDB, and Redis.
* RDBMS Examples: MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.

**Use Cases:**

* DBMS: DBMS systems are suitable for scenarios where data structure is flexible or when dealing with semi-structured or unstructured data, such as in document-oriented databases or key-value stores.
* RDBMS: RDBMS systems are ideal for applications that require structured and highly normalized data, especially when data relationships and integrity are crucial, such as in business applications, e-commerce, and financial systems.

In summary, while both DBMS and RDBMS are database management systems, RDBMS is a specialized subset of DBMS designed for managing structured data with strict data relationships and integrity constraints, using SQL as the query language. The choice between DBMS and RDBMS depends on the specific data requirements and use cases of the application you are developing.

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

* Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when the same data exists in different formats in multiple tables. Unfortunately, data redundancy can cause data inconsistency, which can provide a company with unreliable and/or meaningless information. Data redundancy can have several implications, both positive and negative:

I**ncreased Storage Space:**

Storing the same data in multiple locations consumes additional storage space, which can be costly, especially when dealing with large datasets.

**Data Inconsistency:**

Redundant data can lead to inconsistencies because when the same data is changed in one place, it may not be updated in all the other places where it's duplicated. This can result in conflicting or outdated information.

**Data Integrity Issues:**

Redundant data can increase the risk of data integrity issues, such as data anomalies, where changes in one copy of the data do not get propagated to other copies correctly.

**Complexity:**

Managing redundant data can make the database schema and application code more complex, as developers need to ensure that updates are synchronized across all redundant copies.

1. **What is DDL Interpreter?**

* A DDL (Data Definition Language) interpreter is a component of a database management system (DBMS) responsible for executing and processing DDL statements. DDL statements are used to define and manage the structure and organization of the database itself, including tables, indexes, constraints, and other database schema elements.
* The DDL interpreter processes DDL statements to create and define the database schema, which includes tables, views, indexes, constraints, and other database objects. For example, you can use DDL statements to create tables with specific column definitions, data types, and constraints.
* DDL statements are also used to modify the database schema. You can alter existing tables, add or remove columns, change data types, and apply constraints using DDL statements. Schema modifications ensure that the database structure evolves to meet changing requirements.
* DDL statements can be used to delete or drop database objects such as tables, views, or indexes. Dropping objects removes them from the database entirely.
* DDL interpreters ensure that data integrity constraints specified in DDL statements, such as primary keys, foreign keys, and unique constraints, are enforced properly. This helps maintain data consistency and accuracy.
* Some DDL statements may be used to manage access control and security settings for database objects, specifying who has permission to create, modify, or delete schema elements.

DDL statements are typically executed by database administrators or users with appropriate permissions since they can have a significant impact on the database's structure and integrity. Additionally, the syntax and behaviour of DDL statements may vary between different database management systems (e.g., MySQL, PostgreSQL and Oracle Database).

1. **What is DML Compiler in SQL?**

* SQL is a high-level language used for managing and manipulating relational databases, and it includes two primary types of statements:

**DDL (Data Definition Language):**

DDL statements are used for defining and managing the structure and schema of the database itself. These statements include commands like CREATE TABLE, ALTER TABLE, CREATE INDEX, and DROP TABLE. DDL statements are responsible for creating, modifying, and deleting database objects such as tables, indexes, and views.

**DML (Data Manipulation Language):**

DML statements are used for querying, inserting, updating, and deleting data within the database. Common DML statements include SELECT, INSERT, UPDATE, and DELETE. These statements are used to interact with the data stored in database table

**DML Compiler:**

It processes the DML statements into low level instruction (machine language), so that they can be executed.

**7. What is SQL Key Constraints writing an Example of SQL Key Constraints?**

* **Primary Key:**

• A primary key is a column of table which uniquely identifies each row in that table.

• Primary key enforces integrity constraints to the table.

• Only one primary key is allowed to use in a table.

• The primary key does not accept the any duplicate and NULL values.

• The primary key value in a table changes very rarely so it is chosen with care Where the changes can occur in a seldom manner.

• A primary key of one table can be referenced by foreign key of another table.

Example:

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50)

);

* **Unique Key:**

• Unique key constraints also identifies an individual table uniquely in a relation or table.

• A table can have more than one unique key unlike primary key.

• Unique key constraints can accept only one NULL value for column.

• Unique constraints are also referenced by the foreign key of another table.

Example:

CREATE TABLE Employees (

EmployeeID INT UNIQUE,

FirstName VARCHAR(50),

LastName VARCHAR(50)

);

* **Foreign Key Constraint:**

A foreign key constraint establishes a relationship between two tables. It ensures that the values in a specified column (or columns) of one table match the values in a primary key or unique key column of another table. This constraint is used to enforce referential integrity.

**Example:**

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

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

* **SAVEPOINT:-**

• Creates points within the groups of transactions in which to ROLLBACK.

• A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

• The syntax for a SAVEPOINT command is as shown below.

• SAVEPOINT SAVEPOINT\_NAME;

• This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.

**QUERY:-**

BEGIN;

INSERT INTO Orders (OrderID, CustomerID, OrderDate)

VALUES (1, 101, '2023-09-25');

SAVEPOINT my\_savepoint;

UPDATE Customers SET FirstName = 'John' WHERE CustomerID = 101;

COMMIT;

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

* A trigger in SQL is a database object that automatically executes a specified set of SQL statements in response to certain events or actions that occur within a database. Triggers are typically used to enforce data integrity rules, audit changes to data, or automate certain database actions. They are associated with a specific table and are activated when a particular event occurs, such as an INSERT, UPDATE, DELETE, or other data modification operation.
* To create a trigger in SQL, you use the CREATE TRIGGER statement, specifying the event that triggers the execution of the trigger, the timing (before or after the event), and the actions or SQL statements to be executed when the trigger fires. Here's the basic syntax for creating a trigger:

CREATE TRIGGER trigger\_name

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

ON table\_name

FOR EACH ROW

BEGIN

END;