Module:4 (Database)

Assignment

Q.1 What do you understand By Database

Ans: - A database is a structured collection of data that is organized and stored in a way that allows for efficient retrieval, management, and manipulation of that data. Databases are essential for storing, managing, and retrieving large volumes of information in a systematic and organized manner. They serve as a central repository for data used by software applications, enabling users and applications to access, update, and query the information stored within them.

1).Data Structure: Databases are designed to store data in a structured format, which means the data is organized into tables, records, and fields. This structure helps maintain consistency and makes it easier to manage and retrieve data.

2).Data Integrity: Databases typically enforce rules and constraints to ensure data accuracy and integrity. These rules can include data validation, referential integrity, and unique constraints.

3).Query Language: Databases often provide a query language, such as SQL (Structured Query Language), that allows users to retrieve and manipulate data using standardized commands. SQL is commonly used to interact with relational databases.

4).Indexing: To improve query performance, databases often use indexing mechanisms that create data structures like B-trees to quickly locate specific records within large datasets.

5).ACID Properties: Databases adhere to ACID (Atomicity, Consistency, Isolation, Durability) properties to guarantee transactional integrity. This ensures that database transactions are processed reliably and without corruption.

6).Data Security: Databases include security features to control access to data. Users and applications must have appropriate permissions to read, write, or modify data.

7).Backup and Recovery: Databases provide mechanisms for backing up data and recovering it in case of failures or data loss.

8).Scalability: Databases can be designed to handle various levels of scalability, from small local databases to large distributed databases that can serve millions of users.

Q.2 What is Normalization?

Ans: - Normalization is the process of organizing the data in the database.

* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
* Normalization divides the larger table into smaller and links them using relationships.
* The normal form is used to reduce redundancy from the database table.
* Types:
* 1).First Normal Form (1NF):
* 2).Second Normal Form (2NF):
* 3).Third Normal Form (3NF):

Q.3 What is Difference between DBMS and RDBMS?

Ans: - DBMS (Database Management System) and RDBMS (Relational Database Management System) are both software systems used to manage databases, but they differ in their capabilities and how they handle data. Here are the key differences between DBMS and RDBMS:

1).Data Structure:

DBMS: DBMS can manage data in various forms, including structured, semi-structured, and unstructured data. It is not limited to the relational data model.

RDBMS: RDBMS specifically manages data using the relational data model. Data is organized into tables with rows and columns, and there are defined relationships between tables.

Data Integrity:

2).DBMS: While DBMS systems may offer some level of data integrity through basic constraints, they may not enforce all the strict rules of relational integrity.

RDBMS: RDBMS systems enforce strict data integrity rules, including referential integrity, unique constraints, and data type constraints. This ensures that data remains accurate and consistent.

Query Language:

3).DBMS: DBMS may or may not provide a standardized query language. SQL (Structured Query Language) support may be limited or nonexistent.

RDBMS: RDBMS systems typically offer robust SQL support, allowing users to query and manipulate data using a standardized and powerful query language.

Normalization:

4).DBMS: DBMS may or may not support data normalization. It depends on the specific system and data model it uses.

RDBMS: RDBMS systems are designed to support data normalization through various normal forms (1NF, 2NF, 3NF, etc.). Normalization is a fundamental concept in relational databases to reduce data redundancy and maintain data integrity.

ACID Compliance:

5).DBMS: Some DBMS systems may not fully adhere to ACID (Atomicity, Consistency, Isolation, Durability) properties, which ensure reliable transaction processing.

RDBMS: RDBMS systems are typically ACID-compliant, guaranteeing the reliability of transactions and data consistency.

Scalability:

6).DBMS: DBMS systems may have limitations when it comes to horizontal scalability (scaling out to handle increased loads), especially for complex data models.

RDBMS: Many RDBMS systems offer solutions for horizontal scalability, such as sharding and distributed databases, to handle large volumes of data and high concurrency.

Examples:

7).DBMS Examples: Microsoft Access, SQLite (though it supports SQL, it's not a full RDBMS), and file-based systems.

RDBMS Examples: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and IBM Db2.

Q.4 What is MF Cod Rule of RDBMS Systems?

Ans: - Rule 1: Information Rule

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Rule 2: Guaranteed Access Rule

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

Rule 3: Systematic Treatment of Null Values

This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate information, unknown data and the primary key should not be null.

Rule 4: Active/Dynamic Online Catalog based on the relational model

It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

Rule 5: Comprehensive Data SubLanguage Rule

The relational database supports various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition, data manipulation, integrity constraints, and limit transaction management operations. If the database allows access to the data without any language, it is considered a violation of the database.

Rule 6: View Updating Rule

All views table can be theoretically updated and must be practically updated by the database systems.

Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule

A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

Rule 8: Physical Data Independence Rule

All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application. If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.

Q.5 What do you understand By Data Redundancy?

Ans: - Data redundancy refers to the duplication of data within a database or across multiple databases in a way that is unnecessary or inefficient. It occurs when the same piece of information is stored multiple times in the same database or in different databases within an organization's data ecosystem. Data redundancy can lead to several issues and challenges, including:

1).Inconsistency: When the same data is stored in multiple places, it becomes challenging to ensure that all copies of the data are consistent. Updates or changes to one copy may not be reflected in others, leading to data inconsistencies.

2).Wasted Storage: Storing redundant data consumes additional storage space, which can be costly, especially in large databases or when using cloud-based storage solutions.

3).Data Integrity Issues: Redundant data increases the risk of data integrity problems. If one copy of the data is corrupted or contains errors, it can propagate those issues to other copies.

4).Complexity: Managing and maintaining redundant data adds complexity to database design and operations. It can make data maintenance and updates more challenging.

5).Data Anomalies: Data anomalies, such as insertion, update, and deletion anomalies, can occur when redundant data is not properly synchronized. For example, an update to one copy of the data might be missed in another, leading to inconsistencies.

6).Performance Issues: Retrieving and manipulating redundant data can impact query performance, as it requires more processing power and time to manage larger datasets.

Q.6 What is DDL Interpreter?

Ans:- A DDL (Data Definition Language) interpreter is a component of a database management system (DBMS) or database software that is responsible for processing and executing DDL statements. DDL is a subset of SQL (Structured Query Language) that is used to define, modify, and manage the structure of a database, including tables, indexes, constraints, and other database objects. The DDL interpreter plays a crucial role in database administration and schema management. Here's what a DDL interpreter does:

1).Schema Creation: The DDL interpreter allows database administrators and users to create new database objects and define their structure. This includes creating tables to store data, specifying columns, data types, and constraints.

2).Schema Modification: It enables users to modify the structure of existing database objects. This can involve adding, modifying, or deleting columns, changing data types, and altering constraints.

3).Schema Deletion: The DDL interpreter handles the removal of database objects when they are no longer needed. This includes dropping tables, indexes, and other schema elements.

4).Constraint Management: DDL statements are used to define and manage various constraints, such as primary keys, foreign keys, unique constraints, and check constraints. The DDL interpreter enforces these constraints to maintain data integrity.

5).Index Creation and Management: Database indexes are essential for optimizing query performance. DDL statements can be used to create, modify, or delete indexes, and the DDL interpreter handles these operations.

6).View and Synonym Management: DDL statements can be used to create and manage database views and synonyms. Views provide virtual tables based on existing data, while synonyms provide alternate names for database objects.

7).User and Permission Management: DDL statements related to user accounts, roles, and permissions are used to control access to the database. The DDL interpreter processes these statements to grant or revoke privileges.

8).Data Dictionary Updates: As DDL statements are executed, the DDL interpreter updates the data dictionary or system catalog, which stores metadata about the database schema. This metadata is crucial for the DBMS to understand the structure of the database.

Q.7 What is DML Compiler in SQL?

Ans:- In SQL (Structured Query Language), there isn't a specific concept called a "DML (Data Manipulation Language) Compiler." Instead, DML is a subset of SQL that deals with the manipulation of data stored in a database. DML operations include inserting, updating, deleting, and querying data within database tables.

Here's a brief explanation of DML and how it relates to SQL:

1).Data Manipulation Language (DML): DML is a category of SQL statements used for interacting with the data stored in a relational database. The primary DML statements are:

* SELECT: Used to retrieve data from one or more tables.
* INSERT: Used to add new rows of data into a table.
* UPDATE: Used to modify existing data in a table.
* DELETE: Used to remove data from a table.

MERGE: Used to perform conditional inserts, updates, or deletions based on specific conditions.

2).SQL Compiler: SQL statements, including DML statements, are typically executed within a database management system (DBMS). The DBMS includes components like a query parser, query optimizer, and query executor.

* The query parser parses SQL statements and checks their syntax.
* The query optimizer analyzes the statement to determine the most efficient execution plan.
* The query executor carries out the actual data manipulation operations.

Q.8 What is SQL Key Constraints writing an Example of SQL Key Constraints?

Ans: - SQL key constraints are rules or conditions applied to columns within database tables to ensure the integrity and uniqueness of data. They define how data in a table is organized and linked. There are several types of key constraints in SQL, including primary keys, unique keys, and foreign keys.

1).Primary Key Constraint:

* A primary key constraint ensures that each row in a table has a unique identifier. It uniquely identifies each record in the table and enforces the uniqueness and non-nullity of the specified column or columns.
* Only one primary key constraint can exist per table.

2).Unique Key Constraint:

* A unique key constraint ensures that the values in the specified column or columns are unique across all rows in the table. It allows one null value but enforces uniqueness for non-null values.
* Multiple unique key constraints can exist in a table.

3).Foreign Key Constraint:

* A foreign key constraint establishes a link between two tables by specifying that values in one column (the foreign key) must match values in another column (the primary key) in a different table.
* Foreign key constraints help maintain referential integrity in a relational database.

Q.9 What is save Point? How to create a save Point write a Query?

Ans: - In SQL, a savepoint is a named point within a transaction to which you can later roll back if needed. Savepoints allow you to create a checkpoint within a transaction, so if you encounter an error or need to undo part of the transaction, you can roll back to the savepoint without affecting the entire transaction. Savepoints are particularly useful in situations where you have a complex transaction with multiple steps, and you want to be able to roll back to a specific point in case of an issue.

* SAVEPOINT is the SQL keyword used to create a savepoint.
* savepoint\_name is the name you choose for the savepoint. It should be a unique identifier for the savepoint within the current transaction.
* Savepoint is a command in SQL that is used with the rollback command.
* It is a command in Transaction Control Language that is used to mark the transaction in a table.
* Consider you are making a very long table, and you want to roll back only to a certain position in a table then; this can be achieved using the savepoint.
* If you made a transaction in a table, you could mark the transaction as a certain name, and later on, if you want to roll back to that point, you can do it easily by using the transaction's name.

Q.10 What is trigger and how to create a Trigger in SQL?

Ans: - A trigger in SQL is a database object that automatically executes a specified set of SQL statements in response to a particular event, such as an INSERT, UPDATE, DELETE, or other data-related action that occurs on a table. Triggers are used to enforce business rules, maintain data integrity, and automate tasks within a database.

* Event: The event that triggers the execution of the trigger. This event is typically a database action like an INSERT, UPDATE, DELETE, or even a table creation.
* Trigger Condition (Optional): A condition or criteria that must be met for the trigger to execute. If the condition is not specified, the trigger will activate for every occurrence of the event.
* Trigger Action: The set of SQL statements that are executed when the trigger is activated. These statements can include data manipulation (e.g., INSERT, UPDATE, DELETE), data validation, logging, and more.
* trigger\_name: The name of the trigger.
* table\_name: The name of the table on which the trigger is defined.
* FOR | AFTER | INSTEAD OF: Specifies when the trigger should fire in relation to the triggering event (e.g., BEFORE or AFTER an INSERT).
* {INSERT | UPDATE | DELETE}: Specifies the event that triggers the execution of the trigger.
* AS: An optional keyword used in some DBMS, such as Microsoft SQL Server.
* BEGIN and END: The block of SQL statements that make up the trigger action. These statements define what the trigger does when it is activated.