**Module(Database)-4**

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

**Ans.**

* A database is an organized collection of data, so that it can be easily accessed and managed.
* Database handlers create a database in such a way that only one set of software program provides access of data to all the users.
* The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data.
* There are many dynamic websites on the World Wide Web nowadays which are handled through databases.
* You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.
* There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.
* Modern databases are managed by the database management system (DBMS).

**2). What is Normalization?**

**Ans.**

* The normal form is used to reduce redundancy from the database table.
* 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.
* Normalization is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
* There are Types Of Normalization :-
* **First Normal Form :**
* A table is in 1NF if it does not contain repeating groups or arrays, and each column contains only atomic (indivisible) values.
* **Second Normal Form :**
* A table is in 2NF if it's in 1NF, and all non-key attributes are functionally dependent on the entire primary key. This form is mainly concerned with eliminating partial dependencies.
* **Third Normal Form :**
* A table is in 3NF if it's in 2NF and all transitive dependencies are removed. Transitive dependencies occur when a non-key attribute depends on another non-key attribute, which itself depends on the primary key.
* **Boyce-Codd Normal Form :**
* BCNF is a more stringent form of normalization that further eliminates anomalies by ensuring that non-key attributes are functionally dependent only on the primary key.
* **Fourth Normal From :**
* 4NF deals with multi-valued dependencies. A table is in 4NF when it is in BCNF and has no multi-valued dependencies.

## 3). What is Difference between DBMS and RDBMS?

## Ans.

|  |  |  |
| --- | --- | --- |
| **PARAMETERS** | **DBMS** | **RDBMS** |
| **Type of Program** | Manages the data in a computer | Maintains the relationship of tables in a database |
| Storage | Stores data in the form of a file | Stores data in the form of tables |
| **Database structure** | Hierarchical arrangement of data | Stores data in the form of rows and columns within tables |
| **Number of Users** | Allows one user at a time | Allows more than one user at a time |
| **ACID** | Does not use the ACID form of data storage | Uses the ACID model |
| **Hardware and Software Needs** | Not many hardware and software requirements | Needs a good set of hardware and software requirements |
| **Normalization** | Cannot be normalization | Supports normalization |
| **Data Handling Capacity** | Cannot handle large amounts of data | Able to handle high amounts of data |
| **Distributed Databases** | No support for distributed databases | Allows distributed databases |
| **Integrity Constraints** | Does not support integrity constraints | Supports integrity constrains |
| **Data Relationship** | No relationships defined for the data | Defines relationships using foreign keys |
| **Data Security** | Lack of data security | Good data security due to several log files |
| **Data Access** | Individual data access | Easy and straightforward data access |

## 4). What is MF Cod Rule of RDBMS Systems?

## Ans.

## Rule 1: The Information Rule

## All information, whether it is user information or metadata, that is stored in a database must be entered as a value in a cell of a table. It is said that everything within the database is organized in a table layout.

## Rule 2: The Guaranteed Access Rule

## Each data element is guaranteed to be accessible logically with a combination of the table name, primary key (row value), and attribute name (column value).

## Rule 3: Systematic Treatment of NULL Values

## Every Null value in a database must be given a systematic and uniform treatment.

## Rule 4: Active Online Catalog Rule

## The database catalog, which contains metadata about the database, must be stored and accessed using the same relational database management system.

## Rule 5: The Comprehensive Data Sublanguage Rule

## A crucial component of any efficient database system is its ability to offer an easily understandable data manipulation language (DML) that facilitates defining, querying, and modifying information within the database.

## Rule 6: The View Updating Rule

## All views that are theoretically updatable must also be updatable by the system.

## Rule 7: High-level Insert, Update, and Delete

## 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.

## Rule 8: Physical Data Independence

## Application programs and activities should remain unaffected when changes are made to the physical storage structures or methods.

## Rule 9: Logical Data Independence

## Application programs and activities should remain unaffected when changes are made to the logical structure of the data, such as adding or modifying tables.

## Rule 10: Integrity Independence

## Integrity constraints should be specified separately from application programs and stored in the catalog. They should be automatically enforced by the database system.

## Rule 11: Distribution Independence

## The distribution of data across multiple locations should be invisible to users, and the database system should handle the distribution transparently.

## Rule 12: Non-Subversion Rule

## 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.

## 5). What do you understand By Data Redundancy?

## Ans.

## Data redundancy refers to the situation in which the same data is stored in multiple places within a database or across multiple databases in a system.

## It occurs when there is unnecessary duplication of data, and it can have several negative consequences. Data redundancy is typically considered undesirable in database design and management for several reasons:

## Wasted Storage:

## Storing the same data multiple times consumes additional storage space. This can lead to increased storage costs and inefficient use of resources.

## Data Inconsistency:

## When the same data is stored in multiple locations, it becomes challenging to ensure that all copies of the data remain consistent. - Changes made to one copy of the data may not be reflected in others, leading to data inconsistencies and inaccuracies.

## Data Update Anomalies:

## Data redundancy can result in update anomalies. For example, if the same customer's address is stored in multiple records and needs to be updated, it must be changed in multiple places.

## Failure to update all occurrences can lead to discrepancies in the data.

## Increased Maintenance Complexity:

## Managing and maintaining redundant data is more complex and time-consuming.

## Database administrators must ensure that all copies of the data remain synchronized.

## Data Retrieval Challenges:

## Redundant data can make data retrieval more complex and slower.

## It can require more processing and potentially result in inconsistent query results.

## Security Risks:

## Redundant data can pose security risks.

## Unauthorized access to one copy of the data may lead to unauthorized access to other copies.

## Ensuring consistent security across all copies of the data is challenging.

## 6). What is DDL Interpreter?

## Ans.

## **DDL** is **Data Definition Language**.

## **DDL Command are used to change the structure of a database and database objects**.

## Here are some common DDL commands and operations that a DDL interpreter handles:

## CREATE TABLE:

## DDL interpreters process the CREATE TABLE statement to define the structure of database tables, including column names, data types, and constraints such as primary keys and foreign keys.

## ALTER TABLE:

## They handle ALTER TABLE statements for making changes to an existing table, such as adding or dropping columns, modifying column definitions, or adding constraints.

## DROP TABLE:

## DDL interpreters execute the DROP TABLE statement to remove a table and its associated data from the database.

## CREATE INDEX:

## They process CREATE INDEX statements to create indexes on specific columns for improving query performance.

## DROP INDEX:

## DDL interpreters execute DROP INDEX statements to remove indexes from the database.

## CREATE VIEW:

## DDL interpreters manage CREATE VIEW statements, which define virtual tables based on the data from one or more physical tables.

## CREATE DATABASE:

## They handle CREATE DATABASE statements to create new databases within the DBMS.

## ALTER DATABASE:

## DDL interpreters may also process ALTER DATABASE statements to modify database-level settings.

## DROP DATABASE:

## They execute DROP DATABASE statements to delete entire databases.

## Other Schema-Related Operations:

## DDL interpreters are responsible for enforcing various constraints (e.g., UNIQUE, CHECK), defining stored procedures and triggers, and managing user permissions and access control.

## 7). What is DML Compiler in SQL?

## Ans.

## DML is Data Manipulation Language.

## DML Compiler in Structured Query Language change the data present in the SQL database.

## Here are the key components involved in processing DML operations in a SQL database system:

## Query Processor:

## The query processor is responsible for interpreting and executing SQL queries and DML statements. It includes various subcomponents, such as the query optimizer, which determines the most efficient way to retrieve or manipulate data.

## Execution Engine:

## The execution engine is responsible for actually executing the DML statements. It interacts with the storage engine to read and write data from and to the underlying storage.

## Storage Engine:

## The storage engine manages the physical storage of data on the storage media. It handles tasks like data retrieval and storage and ensures data consistency.

## Transaction Manager:

## The transaction manager ensures that DML operations follow the principles of ACID (Atomicity, Consistency, Isolation, Durability). It manages transactions and handles issues like concurrent access and data integrity.

## Query Cache:

## In some database systems, there is a query cache that stores the results of frequently executed DML statements to improve query performance.

## Lock Manager:

## The lock manager is responsible for managing locks on data to prevent conflicts between concurrent DML operations.

## Buffer Manager:

## The buffer manager caches frequently accessed data in memory to reduce the need for disk I/O during DML operations.

## Log Manager:

## The log manager records changes made by DML operations in a transaction log, which is used for recovery and rollback in case of failures.

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

## Ans.

## SQL key constraints are rules or conditions applied to database columns to ensure the integrity and uniqueness of data.

## There are three primary types of key constraints in SQL: Primary Key, Unique Key, and Foreign Key.

## Primary Key (PK):

## A primary key is a column or set of columns that uniquely identifies each row in a table.

## It enforces the uniqueness and integrity of data in that table.

## The primary key constraint also implies that the column(s) cannot contain NULL values.

## Ex :-

## Creating a table with a primary key constraint:

## CREATE TABLE Student (

## StudentId int primary key,

## FirstName varchar(50),

## LastName varchar(50)

## );

## In this example, the "StudentID" column is specified as the primary key for the "Students" table.

## It will ensure that each student has a unique ID, and NULL values are not allowed.

## Unique Key (UNIQUE Constraint):

## A unique key constraint enforces the uniqueness of values in a column or a set of columns, but it allows for NULL values.

## It's used to ensure that a column or a combination of columns has no duplicate values.

## Ex :-

## Creating a table with a unique key constraint:

## CREATE TABLE Product (

## ProductId int unique,

## ProductName varchar(100)

## );

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

## Ans.

## To create a savepoint and use it within a SQL query, you typically need to use the SQL statements provided by your database management system (DBMS).

## The specific syntax can vary between different database systems, but I'll provide a general example using SQL :

## Create a Savepoint:

## You can create a savepoint within a transaction using the SAVEPOINT statement.

## For example:

## SAVEPOINT my\_savepoint;

## This statement creates a savepoint named "my\_savepoint" within the current transaction.

## Perform Some Actions:

## Execute various SQL statements as part of your transaction.

## Rollback to the Savepoint:

## If you want to roll back to the savepoint due to an error or any other reason, you can use the ROLLBACK TO statement.

## For example:

## ROLLBACK TO my\_savepoint;

## This will undo all changes made since the savepoint was created.

## Commit or Rollback the Transaction :

## After you've made all the necessary changes, you can either commit the transaction to make the changes permanent or roll back the entire transaction to discard all changes.

## Here are the commands for both:

## To commit the transaction:

## COMMIT;

## To roll back the entire transaction:

## ROLLBACK;

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

## Ans.

## In SQL, a trigger is a database object that is associated with a table and is automatically executed in response to certain events or actions, such as INSERT, UPDATE, DELETE, or other data manipulation operations performed on the table.

## Triggers are used to enforce referential integrity, perform logging, and automate tasks in response to data changes.

## Ex :-

## CREATE OR REPLACE TRIGGER log changes

## AFTER INSERT OR UPDATE OR DELETE

## ON employees

## FOR EACH ROW

## BEGIN

## IF INSERTING THEN

## INSERT INTO employee\_audit (employee\_id, action, action\_date)

## VALUES (:NEW.employee\_id, 'INSERT', SYSDATE);

## ELSIF UPDATING THEN

## INSERT INTO employee\_audit (employee\_id, action, action\_date)

## VALUES (:NEW.employee\_id, 'UPDATE', SYSDATE);

## ELSIF DELETING THEN

## INSERT INTO employee\_audit (employee\_id, action, action\_date)

## VALUES (:OLD.employee\_id, 'DELETE', SYSDATE);

## END IF;

## END;

## In this example, the trigger log\_changes is created to log changes made to the employees table.

## The trigger is set to execute AFTER INSERT, UPDATE, or DELETE operations on the table. It logs the changes into an employee\_audit table.