Software Engineering Assignment

MODULE: 5 (DataBase)

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. A database is usually controlled by a database management system (DBMS).

2) What is Normalization?

- Database normalization or database normalisation is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.

3) What is Difference between DBMS and RDBMS?

- The main difference between a DBMS and an RDBMS is that a DBMS is a software application used to store, retrieve, and manage data in a database, while an RDBMS is a type of DBMS that stores data in a relational database.

DBMS	RDBMS
DBMS stores data as file.	RDBMS stores data in tabular form.
Data elements need to access individually.	Multiple data elements can be accessed at the same time.
No relationship between data.	Data is stored in the form of tables which are related to each other.
Normalization is not present.	Normalization is present.
DBMS does not support distributed database.	RDBMS supports distributed database.

Software Engineering Assignment

DBMS	RDBMS
It stores data in either a navigational or hierarchical form.	It uses a tabular structure where the headers are the column names, and the rows contain corresponding values.
It deals with small quantity of data.	It deals with large amount of data.
Data redundancy is common in this model.	Keys and indexes do not allow Data redundancy.
It is used for small organization and deal with small data.	It is used to handle large amount of data.
Not all Codd rules are satisfied.	All 12 Codd rules are satisfied.
Security is less	More security measures provided.
It supports single user.	It supports multiple users.
Data fetching is slower for the large amount of data.	Data fetching is fast because of relational approach.
The data in a DBMS is subject to low security levels with regards to data manipulation.	There exists multiple levels of data security in a RDBMS.
Low software and hardware necessities.	Higher software and hardware necessities.
Examples: XML, Window Registry, Forxpro, dbaseIIIplus etc.	Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc.

Software Engineering Assignment

4) What is MF Cod Rule of RDBMS Systems?

- Codd's rule in DBMS also known as Codd's 12 rules/commandments is a set of thirteen rules (numbered 0 to 12) that define a database to be a correct Relational Database Management System (RDBMS). If a database follows Codd's 12 rules, it is called a True relational database management system

5) What do you understand By Data Redundancy?

- Data redundancy refers to the practice of keeping data in two or more places within a database or data storage system. Data redundancy ensures an organization can provide continued operations or services in the event something happens to its data -- for example, in the case of data corruption or data loss.

6) What is DDL Interpreter?

- DDL Interpreter DDL expands to Data Definition Language. DDL Interpreter as the name suggests interprets the DDL statements such as schema definition statements like create, delete, etc. The result of this interpretation is a set of a table that contains the meta-data which is stored in the data dictionary.

7) What is DML Compiler in SQL?

- DML Compiler DML expands to Data Manipulation Language in DBMS. DML Compiler again as the name suggests compiles(or translates) the DML statements such as select, update and delete statements into low-level instructions which is nothing but the machine-readable object code to make it executable.

Software Engineering Assignment

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

SQL Constraints

In a database table, we can add rules to a column known as **constraints**. These rules control the data that can be stored in a column.
For example, if a column has `NOT NULL` constraint, it means the column cannot store `NULL` values.
The constraints used in SQL are:

Constraint	Description
`NOT NULL`	values cannot be null
`UNIQUE`	values cannot match any older value
`PRIMARY KEY`	used to uniquely identify a row
`FOREIGN KEY`	references a row in another table
`CHECK`	validates condition for new value
`DEFAULT`	set default value if not passed
`CREATE INDEX`	used to speedup the read process

Note: These constraints are also called integrity constraints

---

NOT NULL Constraint

The `NOT NULL` constraint in a column means that the column cannot store `NULL` values.

For example,

```
CREATE TABLE Colleges (
  college_id INT NOT NULL,
  college_code VARCHAR(20) NOT NULL,
  college_name VARCHAR(50)
);
Run Code
```

Here, the `college_id` and the `college_code` columns of the `Colleges` table won't allow `NULL` values.

UNIQUE Constraint

The `UNIQUE` constraint in a column means that the column must have unique value. For example,

```
CREATE TABLE Colleges (
  college_id INT NOT NULL UNIQUE,
  college_code VARCHAR(20) UNIQUE,
  college_name VARCHAR(50)
);
Run Code
```

Here, the value of the `college_code` column must be unique. Similarly, the value of `college_id` must be unique as well as it cannot store `NULL` values.

PRIMARY KEY Constraint

The `PRIMARY KEY` constraint is simply a combination of `NOT NULL` and `UNIQUE` constraints. It means that the column value is used to uniquely identify the row. For example,

```
CREATE TABLE Colleges (
  college_id INT PRIMARY KEY,
  college_code VARCHAR(20) NOT NULL,
  college_name VARCHAR(50)
);
```

```
Run Code
```

Here, the value of the college_id column is a unique identifier for a row. Similarly, it cannot store NULL value and must be UNIQUE.

FOREIGN KEY Constraint

The FOREIGN KEY (REFERENCES in some databases) constraint in a column is used to reference a record that exists in another table. For example,

```
CREATE TABLE Orders (
  order_id INT PRIMARY KEY,
  customer_id int REFERENCES Customers(id)
);
Run Code
```

Here, the value of the college_code column references the row in another table named Customers.

It means that the value of customer_id in the Orders table must be a value from the id column of the Customers table.

CHECK Constraint

The CHECK constraint checks the condition before allowing values in a table. For example,

```
CREATE TABLE Orders (
  order_id INT PRIMARY KEY,
  amount int CHECK (amount >= 100)
);
Run Code
```

Here, the value of the amount column must be **greater than or equal to 100**. If not, the SQL statement results in an error.

DEFAULT Constraint

The `DEFAULT` constraint is used to set the default value if we try to store `NULL` in a column. For example,

```
CREATE TABLE College (
  college_id INT PRIMARY KEY,
  college_code VARCHAR(20),
  college_country VARCHAR(20) DEFAULT 'US'
);
Run Code
```

Here, the default value of the `college_country` column is **US**.

If we try to store the `NULL` value in the `college_country` column, its value will be **US**.

CREATE INDEX Constraint

If a column has `CREATE INDEX` constraint, it's faster to retrieve data if we use that column for data retrieval. For example,

```
-- create table
CREATE TABLE Colleges (
  college_id INT PRIMARY KEY,
  college_code VARCHAR(20) NOT NULL,
  college_name VARCHAR(50)
);

-- create index
CREATE INDEX college_index
ON Colleges(college_code);
Run Code
```

Here, the SQL command creates an index named `customers_index` on the `Customers` table using `customer_id` column.

> **Note:** We cannot see the speed difference with less records in a table. However, we can easily notice the speed difference between using indexes and not using indexes.

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

- A SAVEPOINT is a logical rollback point in a transaction. Usually, when you execute the ROLLBACK command, it undoes the changes until the last COMMIT. But, if you create save points you can partially roll the transaction back to these points. You can create multiple save points between two commits.

The SAVEPOINT Command

A SAVEPOINT is a logical rollback point in a transaction.

Usually, when you execute the ROLLBACK command, it undoes the changes until the last COMMIT. But, if you create save points you can partially roll the transaction back to these points. You can create multiple save points between two commits.

The syntax to create a SAVEPOINT among the transactions is as shown below.

```
SAVEPOINT savepoint_name;
```

Then, to roll back to the SAVEPOINT created, you can use the following syntax –

```
ROLLBACK TO savepoint_name;
```

Example

Following is an example where you plan to delete the three different records from the CUSTOMERS table. You want to create a SAVEPOINT before each delete, so that you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state.

Consider the CUSTOMERS table having the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00

The following code block contains the series of operations.

```
SAVEPOINT SP1;
Query OK, 0 rows affected (0.00 sec)

DELETE FROM CUSTOMERS WHERE ID=1;
Query OK, 1 row affected (0.01 sec)

SAVEPOINT SP2;
Query OK, 0 rows affected (0.00 sec)

DELETE FROM CUSTOMERS WHERE ID=2;
Query OK, 0 rows affected (0.00 sec)

SAVEPOINT SP3;
Query OK, 0 rows affected (0.00 sec)

DELETE FROM CUSTOMERS WHERE ID=3;
Query OK, 1 row affected (0.01 sec)
```

Now that the three deletions have taken place, let us assume that you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone –

```
ROLLBACK TO SP2;
```

Verification

If you display the CUSTOMERS table, you can notice that only the first deletion took place since you rolled back to SP2.

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00

The RELEASE SAVEPOINT Command

The RELEASE SAVEPOINT command is used to remove an existing SAVEPOINT.

The syntax for a RELEASE SAVEPOINT command is as follows.

```
RELEASE SAVEPOINT SAVEPOINT_NAME;
```

Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the last SAVEPOINT.

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

- A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when specific table columns are updated in simple words a trigger is a collection of SQL statements with particular names that are stored in system memory. It belongs to a specific class of stored procedures that are automatically invoked in response to database server events. Every trigger has a table attached to it.
  Because a trigger cannot be called directly, unlike a stored procedure, it is referred to as a special procedure. A trigger is automatically called whenever a data modification event against a table takes place, which is the main distinction between a trigger and a procedure. On the other hand, a stored procedure must be called directly.

  The following are the key differences between triggers and stored procedures:

  1. Triggers cannot be manually invoked or executed.
  2. There is no chance that triggers will receive parameters.
  3. A transaction cannot be committed or rolled back inside a trigger.

- **Syntax:**
  create trigger [trigger_name]

  [before | after]

  {insert | update | delete}

  on [table_name]

  [for each row]

  [trigger_body]