MODULE: 5(DATABASE)

- What do you understand By Database.

Ans: 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).

- What is Normalization?

Ans: **Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

- What is Difference between DBMS and RDBMS?

Ans:

No.	DBMS	RDBMS
1)	DBMS applications store **data as file**.	RDBMS applications store **data in a tabular form**.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	**Normalization is not** present in DBMS.	**Normalization is** present in RDBMS.
4)	DBMS does **not apply any security** with regards to data manipulation.	RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
5)	DBMS uses file system to store data, so there will be **no relation**	in RDBMS, data values are stored in the form of tables, so a **relationship** between

	between the tables.	these data values will be stored in the form of a table as well.
6)	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7)	DBMS **does not support distributed database**.	RDBMS **supports distributed database**.
8)	DBMS is meant to be for small organization and **deal with small data**. it supports **single user**.	RDBMS is designed to **handle large amount of data**. it supports **multiple users**.
9)	Examples of DBMS are file systems, **xml** etc.	Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc.

- What is E.F. Codd Rule of RDBMS Systems?

Ans: Codd's Twelve Rules of Relational Database

Codd rules were proposed by E.F. Codd which should be satisfied by the relational model. Codd's Rules are basically used to check whether DBMS has the quality to become Relational Database Management System (RDBMS). But, it is rare to find that any product has fulfilled all the rules of Codd. They generally follow the 8-9 rules of Codd. E.F. Codd has proposed 13 rules which are popularly known as Codd's 12 rules. These rules are stated as follows:

- **Rule 0: Foundation Rule**– For any system that is advertised as, or claimed to be, a relational database management system, that system must be able to manage databases entirely through its relational capabilities.
- **Rule 1: Information Rule**– Data stored in the Relational model must be a value of some cell of a table.
- **Rule 2: Guaranteed Access Rule**– Every data element must be accessible by the table name, its primary key, and the name of the attribute whose value is to be determined.
- **Rule 3: Systematic Treatment of NULL values**– NULL value in the database must only correspond to missing, unknown, or not applicable values.
- **Rule 4: Active Online Catalog**– The structure of the database must be stored in an online catalog that can be queried by authorized users.

- **Rule 5: Comprehensive Data Sub-language Rule-** A database should be accessible by a language supported for definition, manipulation, and transaction management operation.
- **Rule 6: View Updating Rule-** Different views created for various purposes should be automatically updatable by the system.
- **Rule 7: High-level insert, update and delete rule-** Relational Model should support insert, delete, update, etc. operations at each level of relations. Also, set operations like Union, Intersection, and minus should be supported.
- **Rule 8: Physical data independence-** Any modification in the physical location of a table should not enforce modification at the application level.
- **Rule 9: Logical data independence-** Any modification in the logical or conceptual schema of a table should not enforce modification at the application level. For example, merging two tables into one should not affect the application accessing it which is difficult to achieve.
- **Rule 10: Integrity Independence-** Integrity constraints modified at the database level should not enforce modification at the application level.
- **Rule 11: Distribution Independence-** Distribution of data over various locations should not be visible to end-users.
- **Rule 12: Non-Subversion Rule- Low-level** access to data should not be able to bypass the integrity rule to change data.

- What do you understand By Data Redundancy?

Ans: 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. The concept applies to areas such as databases, computer memory and file storage systems.

Data redundancy can occur within an organization intentionally or accidentally. If done intentionally, the same data is kept in different locations with the organization making a conscious effort to protect it and ensure its consistency. This data is often used for backups or disaster recovery.

If carried out by accident, duplicate data may cause data inconsistencies. Even though data redundancy can help minimize the chance of data loss, redundancy issues can affect larger data sets. For example, data that is stored in several places

takes up valuable storage space and makes it difficult for the organization to identify which data they should access or update.

The word *redundant* can also be used as an independent technical term to refer to the following:

1. Computer or network system components that are installed to back up primary resources in case they fail.
2. Redundant information that is unneeded or duplicated.
3. Redundant bits or extra binary digits that are generated and moved with a data transfer to ensure that no bits were lost during the data transfer.
4. Redundant data that protects a storage array against data loss in the event of a hard disk failure.

- What is DDL Interpreter?

Ans: 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.

- What is DML Compiler in SQL?

Ans: A DML (data manipulation language) refers to **a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database**. A DML is typically a sublanguage of a larger database language like SQL, with the DML containing some of the language's operators.

- What is SQL Key Constraints writing an Example of SQL Key Constraints.

Ans: Constraints in SQL means we are applying certain conditions or restrictions on the database. This further means that before inserting data into the database, we are checking for some conditions. If the condition we have applied to the database holds

true for the data which is to be inserted, then only the data will be inserted into the database tables.

Constraints available in SQL are:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT
7. CREATE INDEX

Now let us try to understand the different constraints available in SQL in more detail with the help of examples. We will use MySQL database for writing all the queries.

1. NOT NULL

- NULL means empty, i.e., the value is not available.
- Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.
- There must exist a value in the column to which the NOT NULL constraint is applied.

NOTE: NULL does not mean zero. NULL means empty column, not even zero.

Syntax to apply the NOT NULL constraint during table creation:

1. CREATE TABLE TableName (ColumnName1 datatype NOT NULL, ColumnName2 datatype,…., ColumnNameN datatype);

Syntax to apply the NOT NULL constraint on an existing table's column:

1. ALTER TABLE TableName CHANGE Old_ColumnName New_ColumnName Datatype NOT NULL;

2. UNIQUE

- Duplicate values are not allowed in the columns to which the UNIQUE constraint is applied.
- The column with the unique constraint will always contain a unique value.
- This constraint can be applied to one or more than one column of a table, which means more than one unique constraint can exist on a single table.
- Using the UNIQUE constraint, you can also modify the already created tables.

Syntax to apply the UNIQUE constraint on a single column:

1. CREATE TABLE TableName (ColumnName1 datatype UNIQUE, ColumnName 2 datatype,…., ColumnNameN datatype);

Syntax to apply the UNIQUE constraint on more than one column:

1. CREATE TABLE TableName (ColumnName1 datatype, ColumnName2 dat atype,…., ColumnNameN datatype, UNIQUE (ColumnName1, ColumnNam e 2));

Syntax to apply the UNIQUE constraint on an existing table's column:

1. ALTER TABLE TableName ADD UNIQUE (ColumnName);

3. PRIMARY KEY

- PRIMARY KEY Constraint is a combination of NOT NULL and Unique constraints.
- NOT NULL constraint and a UNIQUE constraint together forms a PRIMARY constraint.
- The column to which we have applied the primary constraint will always contain a unique value and will not allow null values.

Syntax of primary key constraint during table creation:

1. CREATE TABLE TableName (ColumnName1 datatype PRIMARY KEY, Column Name2 datatype,…., ColumnNameN datatype);

Syntax to apply the primary key constraint on an existing table's column:

1. ALTER TABLE TableName ADD PRIMARY KEY (ColumnName);

4. FOREIGN KEY

- A foreign key is used for referential integrity.
- When we have two tables, and one table takes reference from another table, i.e., the same column is present in both the tables and that column acts as a primary key in one table. That particular column will act as a foreign key in another table.

Syntax to apply a foreign key constraint during table creation:

1. CREATE TABLE tablename(ColumnName1 Datatype(SIZE) PRIMARY KEY, Col umnNameN Datatype(SIZE), FOREIGN KEY(ColumnName) REFERENCES PA RENT_TABLE_NAME(Primary_Key_ColumnName));

Syntax to apply the foreign key constraint with constraint name:

1. CREATE TABLE tablename(ColumnName1 Datatype PRIMARY KEY, Co lumnNameN Datatype(SIZE), CONSTRAINT ConstraintName FOREIGN KEY(ColumnName) REFERENCES PARENT_TABLE_NAME(Primary_ Key_ColumnName));

Syntax to apply the foreign key constraint on an existing table's column:

1. ALTER TABLE Parent_TableName ADD FOREIGN KEY (ColumnName) REFERENCES Child_TableName (ColumnName);

- What is save Point? How to create a save Point write a Query?

Ans: A savepoint is **a special mark inside a transaction that allows all commands that are executed after it was established to be rolled back**, restoring the transaction state to what it was at the time of the savepoint.

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.

The syntax for rolling back to a SAVEPOINT is as shown below.

ROLLBACK TO SAVEPOINT_NAME;

Query:-

```
START TRANSACTION;

SAVEPOINT ini;

INSERT INTO student VALUES (10, "Saurabh Singh", 54, "Kashmir", "1989-01-06");

ROLLBACK TO ini ;
```

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

Ans: A SQL trigger is a database object which fires when an event occurs in a database. For example, we can execute a SQL query that will "do something" in a database when a change occurs on a database table, such as when a record is inserted, updated, or deleted. For example, a trigger can be set on a record insert in a database table. For example, if you want to increase the blogs count in the Reports table when a new record is inserted in the Blogs table, we can create a trigger on the Blogs table on INSERT and update the Reports table by increasing the blog count to 1.

Types of Triggers in SQL Server

There are two types of triggers:

1. DDL Trigger
2. DML Trigger

1. DDL Triggers in SQL Server

The DDL triggers are fired in response to DDL (Data Definition Language) command events that start with Create, Alter, and Drop, such as Create_table, Create_view, drop_table, Drop_view, and Alter_table.

Code of a DDL Trigger

```
create trigger saftey
on database
for
create_table,alter_table,drop_table
as
print'you can not create ,drop and alter table in this database'
rollback;
```

2. DML Triggers in SQL Server

The DML triggers are fired in response to DML (Data Manipulation Language) command events that start with Insert, Update, and Delete. Like insert_table, Update_view and Delete_table.

```
create trigger deep
on emp
for
insert,update,delete
as
print'you can not insert,update and delete this table i'
rollback;
```