→What do you understand By Database

What is Data?

In simple words, data can be facts related to any object in consideration. For example, your name, age, height, weight, etc. are some data related to you. A picture, image, file, pdf, etc. can also be considered data.

What is Database?

A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy.

Let us discuss a database example: An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc.

Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

What is Normalization?

Database Normalization is the most important factor in Database design or Data modeling. Database Normalization is the process to eliminate data redundancies and store the data logically to make data management easier. Database relationships and keys are useful in the Database Normalization process. The Database Normalization was developed by E.F.Codd. In the database normalization process, there are series of rules called Normal Forms.

What is Difference between DBMS and RDBMS?

DBMS →

DBMS applications store **data as file**.
In DBMS, data is generally stored in either a hierarchical form or a navigational form.
Normalization is not present in DBMS.
DBMS does **not apply any security** with regards to data manipulation.
DBMS uses file system to store data, so there will be **no relation between the tables**.
DBMS has to provide some uniform methods to access the stored information
DBMS **does not support distributed database**.
DBMS is meant to be for small organization and **deal with small data**. it supports **single user**.
Examples of DBMS are file systems, **xml** etc.

RDBMS →

RDBMS applications store **data in a tabular form**.
In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
Normalization is present in RDBMS.

RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.
RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
RDBMS **supports distributed database**.
RDBMS is designed to **handle large amount of data**. it supports **multiple users**.
Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc.

What is MF Cod Rule of RDBMS Systems?

Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

What do you understand By Data Redundancy?

In database management, data redundancy refers to the duplication of data in a database. This means that the same piece of data is stored in multiple places within the database.

Data redundancy can have both advantages and disadvantages.

The main advantage of data redundancy is that it can improve the reliability and availability of the data, since multiple copies of the data are stored. If one copy of the data is lost or becomes unavailable, the other copies can be used to access the data.

However, data redundancy can also have some disadvantages.

One of the main disadvantages is that it can lead to data inconsistencies, where different copies of the data are not the same.

This can be a problem if the data is updated, as the different copies of the data will not be in sync with each other. This can make it difficult to know which copy of the data is correct.

Overall, while data redundancy can be useful in some situations, it is important to carefully consider the trade-offs before implementing it.

What is DDL Interpreter?

DDL is defined as **Data Definition Language** : DDL is used to define data **structures** which deals with database descriptions.

For example,

CREATE : It helps to create database
ALTER : it helps to alter the pre-existing data
DROP : It helps to delete objects or data from the database
TRUNCATE : It remove all records from a table
COMMENT : It helps to add comments to the data dictionary.
RENAME : It helps to rename an object in the database

It interprets DDL statements and record them in tables containing metadata.
It is a language in database through which you can make the logical design of the schemas .

What is DML Compiler in SQL?

DML **is defined by** Data Manipulation Language **to manipulate** data itself **with the following commands:**

SELECT : it helps to retrieve data from one or more tables in the database
INSERT : it helps to insert data into a existing table
UPDATE : it helps to update existing data within a existing table
DELETE : it helps to delete all records from a table

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

What is SQL Key Constraints writing an Example of SQL Key Constraints?

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

NOT NULL : Ensures that a column cannot have a NULL value
UNIQUE : Ensures that all values in a column are different

PRIMARY KEY : A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
FOREIGN KEY : Prevents actions that would destroy links between tables
CHECK : Ensures that the values in a column satisfies a specific condition
DEFAULT : Sets a default value for a column if no value is specified
CREATE INDEX : Used to create and retrieve data from the database very quickly

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

Save point 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 save point.

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.

Save point is helpful when we want to roll back only a small part of a table and not the whole table.

In simple words, we can say save point is a bookmark in SQL.

```
start transaction;
savepoint a;
insert into student(sname,age,mobile,courseid) values("Viraj",48,"8789859654",3);
insert into student(sname,age,mobile,courseid) values("virat",38,"6005460004",3);
savepoint b;
insert into student(sname,age,mobile,courseid) values("yuvarj",45,"6546000489",3);
insert into student(sname,age,mobile,courseid) values("",48,"4546454",3);
rollback to b;
```

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

A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

How to create MySQL triggers?

A trigger is a named database object that is associated with a table, and it activates when a particular event (e.g. an insert, update or delete) occurs for the table. The statement CREATE TRIGGER creates a new trigger in MySQL. Here is the syntax is below.

Syntax:

Here is the basic syntax of the CREATE TRIGGER statement:

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
trigger_body;
```