SQL Interview Question

Question : What is MySQL?

MySQL is an open-source, Relational Database Management System that stores data in a structured format using rows and columns. It's software that enables users to create, manage, and manipulate databases.

Database Commands:

1. **Create Database:**

```
CREATE DATABASE databaseName;
```

2. **Show Databases:**

```
SHOW DATABASES;
```

3. **Drop Database:**

```
DROP DATABASE database_Name;
```

4. **Select Database and Query:**

```
USE my_databaseName;
SELECT * FROM my_table;
```

Datatype in mysql

1.Numeric Data Types:

- INT: Integer, a whole number.
- FLOAT: Floating-point number.
- DOUBLE: Double-precision floating-point number.
- DECIMAL: Fixed-point number.

2.Date and Time Data Types:

- DATE: Date in 'YYYY-MM-DD' format.
- TIME: Time in 'HH:MM:SS' format.
- DATETIME: Date and time combination in 'YYYY-MM-DD HH:MM:SS' format.
- TIMESTAMP: Timestamp, typically 'YYYY-MM-DD HH:MM:SS' format, but also auto-updated to current timestamp.

2. **String Data Types:**

- CHAR: Fixed-length string, maximum length specified.
- VARCHAR: Variable-length string, maximum length specified.
- TEXT: Variable-length string, maximum length of 65,535 characters.
- ENUM: Enumeration, list of permitted values.

3. **Binary Data Types:**

- BINARY: Fixed-length binary string.
- BLOB: Binary Large Object, variable-length binary data

Question : How can create Table

```
CREATE TABLE tablename (
PersonID int,
FirstName varchar(255),
LastName varchar(255),
int age,
```

```
    Dob date,
   Address varchar(255),
   City varchar(255)
);
```

Commands: Database Insertion

```
 INSERT INTO table_name(customer_id,first_name,last_name,age,country)
VALUES(1,"priyanshu","Singh",20,"India"),
(2,"Anshu","Kumar",20,"India"),
(3,"piyush","Rajput",20,"India")          // multi value insertion
(4,"priya," Singh",20,"India");
or
insert into  table_name values(1,"Priyanshu",20);       // Single value insertion
insert into  table_name values(1,"Priyanshu",20);
```

Commands: Database Select/Show data

```
  SELECT * FROM table_name;
```

Commands: Drop Table

```
DROP TABLE table_name;
```

Commands: Delete Data in table

```
delete from table_name
where id=1;  // where clause condition can can anything using name ,id,other unique data
```

Commands : TRUNCATE TABLE:

Note: The TRUNCATE TABLE statement is used to delete all rows from a table, but it keeps the table structure intact.

```
TRUNCATE TABLE table_name;
```

Commands : Alter Add new column in table

```
ALTER TABLE table_name
 ADD COLUMN rollno INT; // rollno new coumn name Int dataype
```

Commands : Copy table

```
CREATE TABLE new_table_name AS
SELECT * FROM old_table_name;
```

Commands : Rename table name

RENAME TABLE current_table_name TO new_table_name;

Commands : Select

Q.SQL SELECT Statement:

This selects all columns from a table named users:

SELECT * FROM table_name;

Q.SQL SELECT TOP:

This selects the top 5 rows from the users table:

SELECT **TOP 5 *** FROM table_name;

Q.SQL SELECT Desecnding/reverse Order:

SELECT * FROM table_name ORDER BY id DESC LIMIT 1;

Q.SQL SELECT RANDOM:

SELECT * FROM **table_name ORDER** BY RAND() LIMIT 1;

Q.SQL SELECT IN:

SELECT * FROM users WHERE id IN (1, 2, 3);

Q.SQL SELECT Multiple:

SELECT name, email FROM users;

Command : Update

update database_name

Set amount=50000

where id=3; // condition according to you

Command : Rollback

Q.update salary of employee then rollback

update table_name

set Salary=10000000

where id=1;

RollBack;

Note: by mistake you update wrong data then using rollback you can correct it revese query make privious

Commands : Commit

CREATE TABLE ExampleTable (

```
    ID INT PRIMARY KEY,
    Name VARCHAR(255)
);
INSERT INTO ExampleTable (ID, Name) VALUES (1, 'John');
INSERT INTO ExampleTable (ID, Name) VALUES (2, 'Jane');
COMMIT;
```

Note: Commit use to pramanent save query if you applly rollback it can't ba revort

SQL Operators

•SQL AND Operator

•SQL OR Operator

•SQL LIKE Operator

•SQL IN Operator

•SQL NOT Operator

•SQL NOT EQUAL Operator

•SQL IS NULL Operator

•SQL UNION Operator

•SQL EXCEPT Operator

•SQL BETWEEN Operator

•SQL INTERSECT Operator

•SQL EXISTS Operator

Commnads : Operator

•SQL AND Operator

```
SELECT * FROM Customers(table_name)
WHERE age>=18 AND age<=28;
```

•SQL OR Operator

```
SELECT * FROM Customers(table_name)
WHERE country="UK" OR country="UAE";
```

•SQL NOT Operator

```
SELECT * FROM Customers
WHERE NOT country="UAE";
```

•SQL LIKE Operator

```
select * from Customers(table_name)
where first_name  like "J%";   // Select name stating J alphabet

select * from Customers
where first_name LIKE "B_tt_";  //select name starting name start B—tt--  alphabet
```

```
SELECT * FROM Customers
WHERE first_name LIKE 'J%' OR first_name LIKE 'D%';   // using operator

SELECT * FROM Customers
WHERE first_name LIKE '%n';
```

•SQL IN/ not in Operator

```
SELECT * FROM Customers
WHERE  country  in('USA','UAE');
SELECT * FROM Customers
WHERE  country not in('USA','UAE');
```

SQL NOT EQUAL Operator:

```
SELECT *  FROM products(table_name)
WHERE category != 'Electronics';
```

SQL UNION Operator:

The UNION operator is used to combine the result sets of two or more SELECT statements.

```
SELECT product_name FROM products
WHERE category = 'Electronics'
UNION
SELECT product_name FROM products
WHERE category = 'Appliances';
```

Q.SQL INTERSECT Operator:

The INTERSECT operator is used to return the common rows between two SELECT statements.

```
SELECT product_name FROM products(table name)
WHERE category = 'Electronics'
INTERSECT
SELECT product_name  FROM products (tbale name)
WHERE brand = 'Samsung';
```

Q.SQL EXISTS Operator:

```
SELECT * FROM employees
WHERE EXISTS (
   SELECT 1  FROM orders
   WHERE orders.employee_id = employees.employee_id
);
```

SQL Clauses

•SQL WHERE Clause

•SQL WITH Clause

•SQL HAVING Clause

•SQL ORDER By Clause

•SQL Group By Clause

•SQL LIMIT Clause

•SQL WHERE Clause

```
/* 1.  GREATER Than condition
SELECT * FROM Customers
WHERE age>=20 or age<=25;

/* 2. LIKE condition
SELECT * FROM Customers
WHERE country LIKE 'USA';
/* 3. IN  */
SELECT * FROM employe_data
WHERE DepartmentID IN ("1", "3");
/* 4. EUQAL OPERATION
SELECT * FROM Customers
WHERE customer_id=1;

 5. BETWEEN OPERATION
SELECT * FROM Customers
WHERE age BETWEEN 25 AND  30;
```

6.Q.SQL SELECT Statement:

This selects all columns from a table named users:

```
SELECT * FROM users;
```

•SQL HAVING Clause

count()

min()

max()

avg()

Q.min()

```
 select Min(Salary)
  from table_name
```

Q.max()

```
 SELECT MAX(Salary) FROM employe_data;
```

Q.count()

select count(EmployeeID)

FROM employe_data;

Q.avg

select AVG(Salary)

FROM employe_data;

Q.select max(Salary),min(Salary),avg(Salary) ,sum(Salary) from employe_data group by DepartmentID

having sum(Salary)>60000

•SQL LIMIT Clause

SELECT * FROM Customers where age>=25 limit 3;

Select from table_name limit 2;

•SQL ORDER By Clause

1. ORDER BY

SELECT * FROM Customers(tablename)

ORDER BY last_name;

2. ORDER BY DESC

SELECT * FROM Customers

ORDER BY last_name DESC ;

/* 3.ORDER BY ASC

SELECT * FROM Customers ORDER BY age ASC

4.ODER BY ASC AND DESC

SELECT * FROM Customers

ORDER BY age Asc , customer_id DESC;

•SQL Group By Clause

SELECT customer_id, SUM(total_amount) AS total_spent

FROM orders GROUP BY customer_id;

•SQL WITH Clause

SQL Operators

•SQL AND Operator

•SQL OR Operator

•SQL LIKE Operator

•SQL IN Operator

•SQL NOT Operator

•SQL NOT EQUAL Operator

•SQL IS NULL Operator

•SQL UNION Operator

•SQL EXCEPT Operator

•SQL BETWEEN Operator

•SQL INTERSECT Operator

•SQL EXISTS Operator

SQL AND Operator :

```
SELECT * FROM Customers
WHERE age>=18 AND age<=28;
```

•SQL OR Operator

```
SELECT * FROM Customers
WHERE country="UK" OR country="UAE";
```

•SQL NOT Operator

```
SELECT * FROM Customers
WHERE NOT country="UAE";
```

•SQL LIKE Operator

```
SELECT * FROM Customers
WHERE country LIKE 'USA';
```

•SQL Between Operator

```
SELECT * FROM Customers
WHERE age BETWEEN 25 AND  30;
```

•SQL IN Operator

```
SELECT * FROM employe_data
WHERE DepartmentID IN ("1", "3");
```

•SQL Alias Command :

```
select first_name as name
from  Customers;
```

Note : SQL aliases are used to give a table, or a column in a table, a temporary name.

Comparison Operator

1. Greater than

select * from employe_data where Salary>60000;

2. Less than

select * from employe_data where Salary<60000;

3. Greater than equal

select * from employe_data where Salary>=60000;

4. Less than equal

select * from employe_data where Salary<=60000;

5. Equal

select * from employe_data where Salary=60000;

SQL Data Constraints

•SQL NOT NULL Constraints

•SQL UNIQUE Constraints

•SQL Primary Key Constraints

•SQL Foreign Key Constraints

•SQL Composite Key

•SQL Unique Constraints

•SQL Alternate Key

•SQL CHECK Constraints

•SQL DEFAULT Constraints

unique : we can't insert duplicate vaue of id beacuase its unique

```
create table vivek(
ID INT unique,
name varchar(28),
age int
);
```

NOT NULL: we can't kept empty id nll bacause consytarint is not null

```
create table vivek(
ID INT NOT NULL,
name varchar(28),
age int
```

);

Primary key : unique key

```
create table vivek(
ID INT,
name varchar(28),
age int not null check(age<=18),
primary key(id)
);
```

•SQL Foreign Key Constraints : its refernce of another table link 2 table ech other

```
CREATE TABLE Orders (
    OrderID int PRIMARY KEY,
    CustomerID int,
    OrderDate date,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

SQL CHECK Constraints:

```
CREATE TABLE Employees (
    EmployeeID int PRIMARY KEY,
    Age int CHECK (Age >= 18)
);
```

SQL DEFAULT Constraints:

```
CREATE TABLE Orders (
    OrderID int PRIMARY KEY,
    OrderDate date DEFAULT CURRENT_DATE
);
```

String Function

1.Check Ascii value using sql

select ascii('a');

select ascii('A');

2.character to ascii aplpha numeric

SELECT CHAR(97);

3.Find character index

SELECT CHAR_LENGTH("Priyanshu");

4. Concat 2 sring

select concat("Priya","anshu");

Comment in sql : 1. single-line comment 2. multi-line comment

-- This is a single-line comment

/* This is a multi-line comment that spans across multiple lines */

Indexing

/* Search any data before indexing */

CREATE CLUSTERED INDEX idx_clustered

ON Empdatas(emp_id);

select * from Empdatas where emp_id=6

JOIN Operation

- **LEFT JOIN**
- **RIGHT JOIN**
- **CROSS JOIN**
- **NATURAL JOIN**
- **CROSS JOIN**
- **INNER JOIN**
- **SELF JOIN**

INNER JOIN

select * from student(tabale name)

inner join coursetabale name)

on student.id=couse.id;

LEFT JOIN

select name,age,address,Salary,Gender,City from Tempjoin

left join empdata

on Tempjoin.id=empdata.id;

RIGHT JOIN

select name,age,address,Salary,Gender,City from Tempjoin

right join empdata on Tempjoin.id=empdata.id;

FULL OUTER JOIN

SELECT *FROM table1

FULL OUTER JOIN table2 ON table1.common_column = table2.common_column

CROSS JOIN

SELECT Table1.Column1, Table1.Column2, Table2.Column3, Table2.Column4

FROM Table1

CROSS JOIN Table2;

Theory Question

Q. What is a database?

A Database is defined as a structured form of data storage in a computer or a collection of Related data in an organized manner and can be accessed in various ways. It is also the collection of schemas, tables, queries, views, etc. Databases help us with easily storing, accessing, and manipulating data held on a computer.

Application of database:

Ans: Company Information, Account information, manufacturing, banking, finance transactions, telecommunications.

Q.SQL:

Ans : SQL stands for Structured Query Language and is a computer language that we use to interact with a relational database. SQL is a tool for *organizing*, *managing*, and *retrieving* archived data from a computer database.

We use SQL for CRUD Operations:

- CREATE - To create databases, tables, insert tuples in tables etc
- READ - To read data present in the database.
- UPDATE - Modify already inserted data.
- DELETE - Delete database, table or specific data point/tuple/row or multiple row

What is DBMS: A Database Management System (DBMS) is software that enables users to create, manage, and organize databases. It provides an interface for users and applications to interact with the database by performing tasks such as storing, retrieving, updating, and deleting data

Types of SQL Commands:

1. **DQL** (Data Query Language): Used to retrieve data from databases. (**SELECT**)

2. **DDL** (Data Definition Language): Used to create, alter, and delete database objects (C**REATE, DROP, ALTER, RENAME, TRUNCATE**)

3. **DML** (Data Manipulation Language): Used to modify the database. (**INSERT, UPDATE, DELETE**)

4. **DCL** (Data Control Language): Used to grant & revoke permissions. (**GRANT, REVOKE**)

5**. TCL** (Transaction Control Language): Used to manage transactions (**COMMIT, ROLLBACK, START TRANSACTIONS**)

1. Data Definition Language (DDL)

Data Definition Language (DDL) is a subset of SQL (Structured Query Language) responsible for defining and managing the structure of databases DDL commands enable you to **create, modify, and delete, Alter, Drop, database** objects like tables,

2. DATA QUERY/RETRIEVAL LANGUAGE (DQL or DRL)

DQL (Data Query Language) is a subset of SQL focused on retrieving data from databases. It includes commands such as **SELECT,** which are used to query and retrieve data from database tables. DQL allows users to specify the data they want to retrieve, apply filtering and sorting

3.DATA MANIPULATION LANGUAGE

Data Manipulation Language (DML) is a subset of SQL (Structured Query Language) that allows users to manipulate data stored in a database. DML statements are used to perform operations such as inserting, updating, deleting, and querying data within database tables.

4.Data Control Language (DCL)

Data Control Language (DCL) is a part of SQL (Structured Query Language) that deals with controlling access to data within a database.

GRANT: This command allows users to give specific permissions to other users or roles. For example, a database administrator can grant SELECT permission on a table to a specific user, allowing them to retrieve data from that table.

REVOKE: This command is like taking back a permission slip. For example, if you let someone borrow your book but then decide you need it back, you revoke their borrowing permission. Similarly, in a database,

5.Transaction Control Language (TCL)

Transaction Control Language (TCL) deals with the management of transactions within a database. TCL commands are used to control the initiation, execution, and termination of transactions, which are sequences of one or more SQL statements that are executed as a single unit of work

COMMIT:

The COMMIT command is used to permanently save the changes made during a transaction

ROLLBACK:

The ROLLBACK command is used to undo changes made during a transaction.

It reverts all the changes applied to the database since the transaction began.

JOINS :join is a fundamental operation used to combine rows from two or more tables based on a related column between them. It allows you to retrieve data from multiple tables simultaneously,

Types of Joins:

1. Inner Join 2. Outer Join 3. Cross Join 4. Self-Join

1. Inner Join: only the rows from the participating tables where the join condition is satisfied for both tables are included in the result set. This means that if there is no matching row in one of the tables based on the specified join condition, that row will not appear in the final result.

2) Outer Join:. An outer join is a method of combining two or more tables where the result includes unmatched rows from one or both tables, depending on the type of outer join used.

Types:

1. Left Outer Join (Left Join): A left outer join returns all the rows from the left table and the matching rows from the right table If there is no match in the right table, the result will still include the left table's row with NULL values in the right table's columns

2. Right Outer Join (Right Join):

A right outer join is similar to a left outer join, but it returns all rows from the right table and the matching rows from the left table. If there is no match in the left table, the result will still include the right table's row with NULL values in the left table's columns.

3. Full Outer Join (Full Join):

A full outer join returns all rows from both the left and right tables, including matches and non_matche If there's no match, NULL values appear in columns from the table where there's no corresponding value.

3.Cross Join

A cross join, also known as a Cartesian product, is a type of join operation in a Database Management System (DBMS) that combines every row from one table with every row from another table.

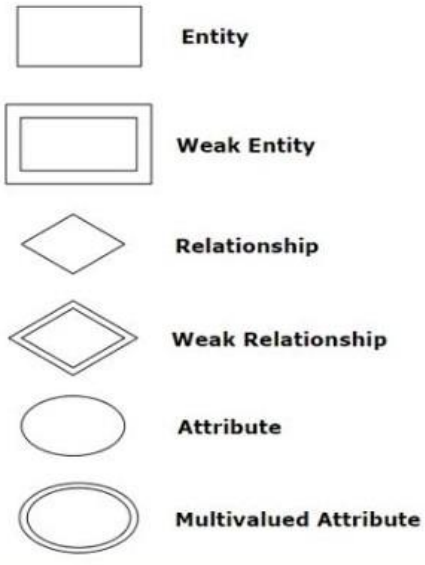
4) Self Join:

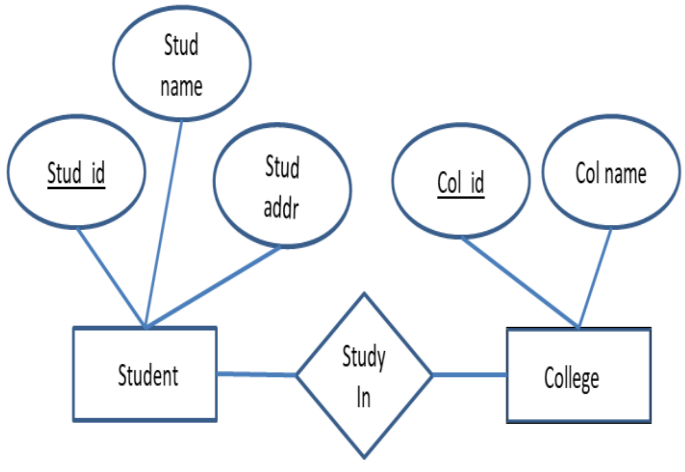
A self-join is a type of join operation where a table is joined by itself. This can be useful when you have a table that contains hierarchical or recursive data,

Need of Joining table : Joining tables is necessary in databases when one table doesn't have all the information needed for a query. By merging data from different tables using shared columns, joints help users get a complete dataset, revealing how different things in the database are related.

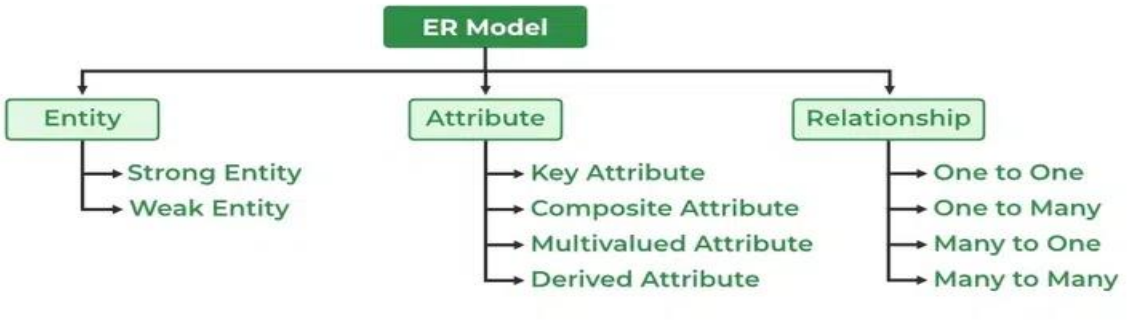
ER diagram:

ER diagram or Entity Relationship diagram is a conceptual model that gives the graphical representation of the logical structure of the database. It shows all the constraints and relationships that exist among the different components
ER diagrams are used during the database design phase to visualize and understand the database schema.





Components of ER Diagram:



Why Use ER Diagrams In DBMS?

ER diagrams are used to represent the E-R model in a database, which makes them easy to convert into relations (tables).

ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.

ER diagrams require no technical knowledge and no hardware support.

These diagrams are very easy to understand and easy to create even for a naive user.

It gives a standard solution for visualizing the data logically.

Entity: An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

Entity Set :

1. Strong Entity : A **Strong Entity** s a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has **a primary key,** that helps in identifying it uniquely,

2. Week Entity: A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities. in other words, a primary key does not exist for a weak entity set.

Attributes:

- **Simple Attributes -** Simple attributes are those attributes which cannot be divided further. Ex. Age,Dod
- **Composite Attribute**

An attribute composed of many other attributes is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country.

- **Multi Val**ued Attributes - Multi valued attributes are those attributes which can take

more than one value for a given entity from an entity set. Ex. Mobile No, Email ID

●**Derived Attributes -** Derived attributes are those attributes which can be derived from other attribute(s). Ex. Age can be derived from DOB.

● **Key Attributes :** The attribute which uniquely identifies each entity in the entity set is called the key attribute. For example, Roll_No will be unique for each student.

Keys: A key is a set of attributes that can identify each tuple uniquely in the given relation

Type of key: 1. Candidate Key 2. Composite Key 3. Foreign Key 4. Super Key 5. Primary Key

1.Primary Key: The Primary Key uniquely identifies each record in a table. It must contain unique values and cannot have NULL values. There can be only one primary key in a table. **Example: Student'd,** Aadhar number

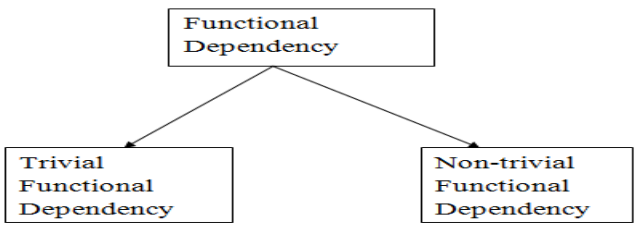
2.Candidate Key: A candidate key is a set of one or more columns in a table that can uniquely identify each record within that table. **Example:** Email, **Student'd**

3.Foreign Key: A Foreign Key is a column or a set of columns on a table that establishes a relationship with a Primary Key or a Unique Key in another table. It enforces referential integrity between the two related tables.

4.Super Key: A Super Key is a set of one or more columns in a table whose combined values can uniquely identify each row in the table. It's a superset of a Candidate Key. **Example:** Student'd Phone

5.Composite Key: A Composite Key, also known as a composite primary key or concatenated key, is a combination of two or more columns in a table that together uniquely identify each row in the table.

Functional Dependency: The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table. $\alpha \rightarrow \beta$



● **Trivial Functional Dependencies –**

o A functional dependency X → Y is said to be trivial if and only if Y ⊆ X. o

Thus, if RHS of a functional dependency is a subset of LHS, then it is called a

trivial functional dependency.

● **Non-Trivial Functional Dependencies –**

o A functional dependency X → Y is said to be non-trivial if and only if Y ⊄ X. o

Thus, if there exists at least one attribute in the RHS of a functional dependency

that is not a part of LHS, then it is called a non-trivial functional dependency.

Normalization:

It is a process of analyzing the given relation schemas based on their functional dependencies and primary keys to achieve the following desirable properties:

1. Minimizing Redundancy
2. Minimizing the Insertion, Deletion, And Update Anomalies Relation

Normal Forms:

● **First Normal Form (1NF) - First** Normal Form (1NF) if each cell of the table contains only an atomic value i.e. if the attribute of every tuple is either single valued or a null value.

● **Second Normal Form (2NF) -** The database must already be in First Normal Form (1NF).

Every non-prime attribute (attributes that are not part of any candidate key) must be fully functionally dependent on the entire primary key, not just part of it.

- **Third Normal Form (3NF) -** The database must already be in Second Normal Form (2NF).

- Every non-prime attribute (attribute that are not part of any candidate key) must be transitively dependent on the primary key.
- Every non-prime attribute must be dependent on the primary key directly, only if o Relation already exists in 2NF. o No transitive dependency exists for non-prime attribute
-
- **Boyce-Codd Normal Form -** A given relation is called in BCNF if and only if
- o Relation already exists in 3NF For every non-trivial functional dependency (X -> Y) in the table, X must be a super key.

Transaction: A Database Transaction is a set of database operations that must be treated as a whole, which means either all operations are executed or none of them

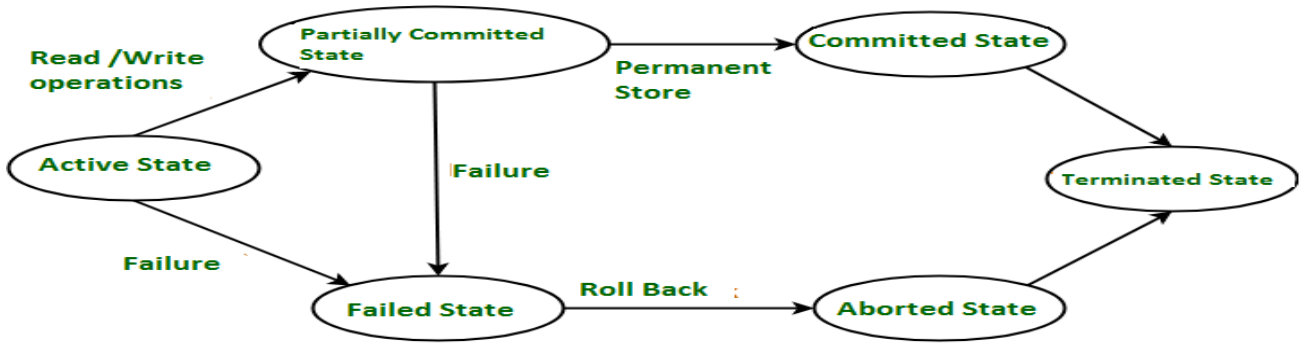
Example. An example can be a bank transaction from one account to another account. Either both debit and credit operations must be executed or neither of them.

Operations in Transaction:

● **Read Operation** - Read(A) instruction will read the value of 'A' from the database and

will store it in the buffer in main memory.

● **Write Operation** – Write(A) will write the updated value of 'A' from the buffer to the database



Transaction States in DBMS

● **Active State –**

o This is the first state in the life cycle of a transaction.

o A transaction is called in an active state as long as its instructions are getting executed.

o All the changes made by the transaction now are stored in the buffer in main memory

.

● **Partially Committed State –**

o After the last instruction of the transaction has been executed, it enter into a partially committed state.

o After entering this state, the transaction is considered to be partially committed.

o It is not considered fully committed because all the changes made by the transaction are still stored in the buffer in main memory.

● **Committed State –**

oAfter all the changes made by the transaction have been successfully store into the database it enters into a committed state

o Now, the transaction is considered to be fully committed.

● **Failed State –**

o When a transaction is getting executed in the active state or partially committed

state and some failure occur due to which it becomes impossible to continue

the execution, it enters into a failed state**.**

● **Aborted State –**

o After the transaction has failed and entered into a failed state, all the changes made by it have to be undone.

o to undo the changes made by the transaction, it becomes necessary to roll back the transaction.

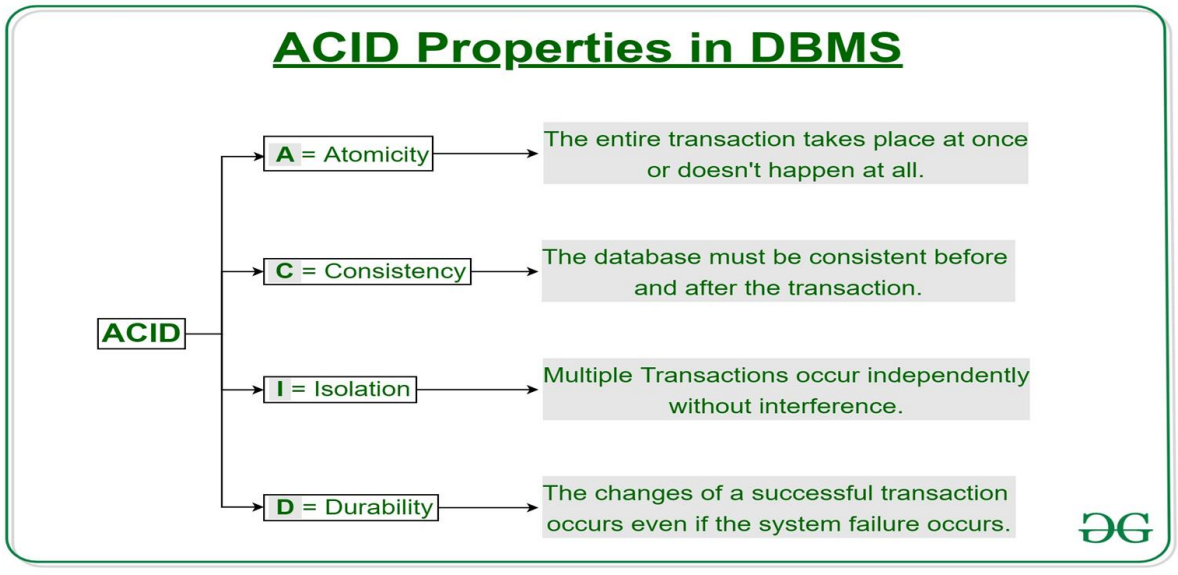
o After the transaction has rolled back completely, it enters into an aborted sta**te.**

● **Terminated State –**

o This is the last state in the life cycle of a transaction.

o After entering the committed state or aborted state, the transaction finally enters into a terminated state where its life cycle finally comes to an end

ACID Properties: ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability. These are the four key properties that ensure reliability and consistency of transactions in a database management system (DBMS)



● **Atomicity –**

o This property ensures that either the transaction occurs completely, or it does not occur at all.

o in other words, it ensures that no transaction occurs partially.

● **Consistency –** This property ensures that a transaction transforms the database from one consistent state to another consistent state.

● **Isolation –** This property ensures that the execution of concurrent transactions does not interfere with each other. Each transaction operates independently and as if it is the only transaction executing on the database.

● **Durability –** Durability means that once you've confirmed a transaction, its changes are permanent. Even if there's a system crash, your data won't disappear. So, durability ensures that your data stays safe and can be recovered after any mishap.

Trigger in SQL

A **Trigger** in Structured Query Language is a set of procedural statements which are executed automatically when there is any response to certain events on the particular table in the database. Triggers are used to protect the data integrity in the database.

Data Mining: is the process of investigating hidden patterns of information to various perspectives for categorization into useful data, which is collected and assembled in particular areas such as data warehouses, efficient analysis, data mining algorithm, helping decision making

Data Warehousing: A data warehouse is a separate database system from the operational databases (DBMS) used by an organization. It is designed to store and manage large volumes of data collected from various sources such as operational databases, files, external data sources, etc. The primary goal of a data warehouse is to provide a consolidated, integrated, and historical view of an organization's data to support decision-making process**es.**

Why is the use of DBMS recommended? Explain by listing some of its major advantages?

- **Controlled Redundancy:** DBMS Controlled redundancy in a database management system (DBMS) ensures that data is stored in a single place, preventing duplication. This integration of data into one central location eliminates redundancies,
- **Data Sharing:** Sharing of data among multiple users simultaneously can also be done in DBMS as the same database will be shared among all the users and by different application programs.
- **Backup and Recovery Facility:**DBMS minimizes the pain of creating the backup of data again and again by providing a feature of ‘backup and recovery’ which automatically creates the data backup and restores the data whenever required.

- **Enforcement of Integrity Constraints:** Integrity Constraints are very important to be enforced on the data so that the refined data after putting some constraints are stored in the database and this is followed by DBMS.
- **Independence of Data:** It simply means that you can change the structure of the data without affecting the structure of any of the application programs.

. What are indexes? A database index is a specialized data structure that enhances the speed of data retrieval operations on a database table. It achieves this by creating a sorted copy of selected columns or fields from the table

What is CLAUSE in SQL? A clause in SQL is a part of a query that lets you filter or customize how you want your data to be queried to you.

Database Schema

. A database schema is a **logical representation of data** that shows how the data in a database should be stored logically. It shows how the data is organized and the relationship between the tables.

- Database schema contains table, field, views and relation between different keys like primary key, foreign key.

There are 3 levels of data abstraction in the DBMS.

View Level: a view schema refers to the structure or definition of a view. A view is a virtual table created by a query that selects rows and columns from one or more tables. It presents the data in a customized way, often simplifying complex data structures

Physical Level The physical database schema describes how data is physically stored on the storage system. It includes details such as file organization and index creation. Essentially, it's the actual code or syntax used to create the database structure at the storage level.

Logical Level: The logical database schema defines the structure of the database at a conceptual level,ncluding tables, columns, relationships, and constraints,

RDBMS: RDBMS stands for Relational Database Management System. It's a software application used to manage relational databases, which store data in tables consisting of rows and columns.

MySQL: MySQL is indeed a widely used Relational Database Management System (RDBMS), known for its reliability, performance, and ease of use. One of its key advantages is that it's free and open source, meaning users can access and modify its source code as needed without any licensing fees.

Advantage of database

1. Quick Data Sharing: DBMS facilitates fast and efficient data sharing among users, enhancing collaboration and responsiveness.
2. Informed Decision Making: With well-managed data access, DBMS enables better quality information for making informed decisions.
3. Improved Privacy: DBMS implements privacy rules to restrict unauthorized access, enhancing data privacy and confidentiality.
4. User-Friendly Interface: DBMS presents data in a logical manner, making it easy for users to perform various tasks.
5. Data Abstraction: DBMS uses abstraction to simplify complex data structures, enabling users to interact with the system effectively.

MySQL, Advantage:

Open Source: MySQL is freely available under the GNU General Public License (GPL). This makes it accessible to a wide range of users without any licensing costs.

High Performance: MySQL is optimized for fast data retrieval and processing. It offers high-performance indexing, caching mechanisms, and efficient query execution, making it suitable for demanding applications.

Cross-Platform Compatibility: MySQL is available for multiple platforms, including Windows, Linux, macOS, and Unix-like operating systems, ensuring compatibility with a wide range of environments.

Ease of Use: MySQL comes with user-friendly tools and utilities for database administration, management, and monitoring. Its simple installation process and intuitive interfaces make it easy for developers and administrators to work with.

What is Serializability in DBMS?

A system is considered serializable if its outcome remains consistent regardless of how the operations are sequenced, as long as there is no overlap in their execution.
Type of Serializability : Conflict Equivalent Schedules: View Serializability:

DBMS Integrity Constraints

Integrity constraints are the set of predefined rules that are used to maintain the quality of information. Integrity constraints ensure that the data insertion, data updating, data deleting, and other processes have to be performed in such a way that the data integrity is not affected.

Entity Integrity Constraint: This constraint ensures that each row (or entity) in a table has a unique and non-null primary key value, meaning that primary key attributes cannot contain null values and must be unique within the table.

1. **Referential Integrity Constraint**: Also known as foreign key constraints, these constraints maintain the relationships between tables by ensuring that values in a foreign key column (referencing another table's primary key) always correspond to existing values in the referenced table's primary key column. This constraint prevents orphaned rows and maintains data consistency.
2. **Domain Integrity Constraint**: Domain integrity constraints define the allowable values for columns in a table. These constraints specify the data type, format, and range of values that can be stored in a column, ensuring that only valid data is inserted into the database.
3. **Check Constraint**: Check constraints define conditions that data must meet for insertion or updating operations to be successful. These conditions can involve comparisons, logical expressions, or predefined functions to validate data integrity.
4. **Unique Constraint**: Unique constraints ensure that values in a specified column (or combination of columns) are unique across all rows in a table, except for null values. This constraint prevents duplicate entries in the table.
5. **Assertion Constraint**: Assertion constraints define conditions that must be true for the entire database, rather than for individual tables or columns. These constraints are typically used to enforce complex business rules that involve multiple tables or rows.

Difference between where and having clause

Point	WHERE Clause	HAVING Clause
1. Filters rows before groups are aggregated	Yes, filters rows before aggregation process	No, filters groups after aggregation process
2. Usage with GROUP BY Clause	Can be used without GROUP BY Clause	Can be used with GROUP BY Clause
3. Scope of implementation	Implements in row operations	Implements in column operation
4. Usage with SQL statements	Can be used with SELECT, UPDATE, DELETE statement	Can only be used with SELECT statement
5. Usage with functions	Used with single row functions like UPPER, LOWER, etc.	Used with multiple row functions like SUM, COUNT, etc.

Difference Between SQL vs NoSQL

SQL	NoSQL
SQL database Called Relational Database	NoSQL Called Non-Relational database
SQL Database is table-based data store	NoSQL Database is Document based data Stored
Ex- Oracle, Postgres, SQL Server	Ex – MongoDB, Cassandara
These databases have fixed or static or predefined schema	They have a dynamic schema

Type of Database

These databases have fixed or static or predefined schema

Relational Databases (RDBMS):

Relational databases store and manage data in tables with rows and columns.
They use Structured Query Language (SQL) for querying and manipulating data.
Examples include MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and SQLite.

NoSQL Databases:
NoSQL databases are non-relational databases that provide flexible schema design and are suitable for handling unstructured or semi-structured data.
They can be document-oriented, key-value stores, column-family stores, or graph databases.
Examples include MongoDB, Couchbase, Cassandra, Redis, and Neo4j.

Object-Oriented Databases (OODBMS):
object-oriented databases store data as objects, allowing for complex data structures and relationships.
They are suitable for applications with complex data models and object-oriented programming languages.
Examples include db4o and Object DB

Graph Databases:
Graph databases are optimized for storing and querying graph data structures, consisting of nodes, edges, and properties.
They are suitable for applications that require traversing relationships between entities.
Examples include Neo4j, Amazon Neptune, and Janus Graph.

Columnar Databases:

In-Memory Databases:

Time-Series Databases:
Time-series databases specialize in storing and analyzing time-stamped data, such as sensor data, IoT data, and financial data.
They optimize storage and retrieval for time-series data patterns.
Examples include Influx DB, Prometheus, and Timescale DB.

Aggregation: Aggregation in the context of databases refers to the process of applying a function (or operation) to multiple rows of data and returning a single summary value. This summary value represents some aspect of the dataset, such as a total, average, count, maximum, or minimum, Sum value.

What is the difference between primary key and unique constraints?
The primary key cannot have NULL value, the unique constraints can have NULL values. There is only one primary key in a table, but there can be multiple unique constraints.

Clause In Dbms :
.A clause in SQL is a part of a query that specifies certain conditions, actions, or constraints to customize the retrieval or manipulation of data from a database.

Example: Select, Where Order by, Group by, Having Clause

What is Denormalization?
Denormalization is a database optimization technique in which we add redundant data to one or more tables.

Relationship:
A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course.

1. Unary Relationship: When there is only ONE entity set participating in a relation, the relationship is called a unary relationship
2. Binary Relationship: When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.
3. Ternary Relationship: When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

Cardinality : The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:
1. One-to-One: When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one.

2. One-to-Many: In one-to-many mapping as well where each entity can be related to more than one entity and the total number of tables that can be used in this is

3. Many-to-One:

When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one.

4. Many-to-Many: When entities in all entity sets can take part more than once in the relationship cardinality is many to many.