What is DBMS & RDBMS and also Difference ?

* Database management system is a software which is used to manage the database. For example:**MySQL**, Oracle, etc are a very popular commercial database which is used in different applications.
* DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
* It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

Rdbms

**RDBMS** stands for *Relational Database Management Systems.*.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.

It is called Relational Database Management System (RDBMS) because it is based on relational model introduced by E.F. Codd.

DBMS + E.F. Codd 12 Rules =  RDBMS.

XAMPP <https://www.apachefriends.org/download.html>

X CROSS PLATFORM

A APATCH SERVER

M MYSQL

P PERL

P PHP

Xampp/xampp-control -> open

Apache / Mysql -> start

Google : localhost/phpmyadmin

SQL What is SQL & TYPES?

* SQL stands for **Structured Query Language**
* SQL lets you access and manipulate databases

**4 Types**

1. **DDL Data Definition Language** -----> **4 Commands  create / alter / drop / truncate**

* It is used to define the structure of **databases and tables**.
* We can **create**, **modify** or **delete** the structure of tables.

**Create :**

=>create database shop

=>create table customers(id int PRIMARY key AUTO\_INCREMENT, cust\_name varchar(100),user\_name varchar(100), password varchar(100),email varchar(100), mob bigint(11), address varchar(255), pincode bigint(11))

===================================================================

1 foreign key

=>create table feedbacks(id int PRIMARY key AUTO\_INCREMENT, fed\_comment varchar(100), fed\_date date,cus\_id int(11), FOREIGN key(cus\_id) REFERENCES customers(id));

2 foreign key

=>create table feedback\_product(id int PRIMARY key AUTO\_INCREMENT, fed\_comment varchar(100), fed\_date date, cus\_id int(11),FOREIGN key(cus\_id) REFERENCES customers(id), pro\_id int(11), FOREIGN key(pro\_id) REFERENCES product(id));

**alter:**

ALTER TABLE  customer add(gender varchar(100)) AFTER ‘password’;  // add column

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `customers` CHANGE `name` `username` VARCHAR(255)

ALTER TABLE `customer` CHANGE ‘phone’ ‘mobile’ BIGINT(11) //  change column name

ALTER TABLE `customer` DROP `gender`;   // column delete

===================================================================

**drop:**

drop database database\_name  // drop database delete

drop table tbl\_name   // drop table delete

ALTER TABLE `customer` DROP `gender`;   //  table column delete

**truncate:  / delete all data from table /empty table**

truncate table tabl\_name  // delete all table data not table

=====================================================================

1. DML  Data Manipulation Language    3 command   insert / delete / update

* insert:

insert into customer(cust\_name,user\_name,pass,email,mobile,address,pincode,gender) values("Akshay","akashay701","12 34","akashay@gmail.com","5646944","Ahmedabad","325874","Male")

INSERT INTO customers (cust\_name,user\_name,password,email,mob,gender,dob,address,pincode) VALUES ("Raj","raj@gmail.com","1234","raj@gmail.com","123467891","Male","1990-11-17","Chandlodia","382481");

* update:    data update as per id
* UPDATE customer set name="pavan Nagar" ,  password="abc" where id=3

* delete:  data delete as per id

delete from customer where id=3

============================================================

1. DQL  Data Query Language  :   Select

* Select Description: This will select  ‘n‘ columns from the table. Or To select all records from the database.

Select \* from customer      // get all data with all column

Select cus\_id,cust\_name from customer  // get all data with particular column

Select \* from customer where cus\_id=2

Select cus\_id,cust\_name from customer where cus\_id=2

1. TCL Transaction Control Language

=> rollback / commit

Rollback :  ctl+z  undo

commit : ctl+s  save as

Advance SQL

1) DISTINCT

SELECT DISTINCT COUNTRY FROM Customers;

2) ORDER BY

SELECT \* FROM Products ORDER BY ProductName ASC

SELECT \* FROM Products ORDER BY ProductName DESC

3) AND OR NOT

SELECT \* FROM Customers WHERE Country="Germany" AND City="Berlin"

SELECT \* FROM Customers WHERE Country="Germany" OR Country =" Mexico "

SELECT \* FROM Customers WHERE Not Country="Germany”

SELECT \* FROM Customers WHERE City NOT IN ('Paris', 'London');

SELECT \* FROM products WHERE NOT price > 50;

4) BETWEEN

SELECT \* FROM products WHERE price BETWEEN 10 AND 60;

SELECT \* FROM order WHERE ord\_date BETWEEN “01-06-2024” AND “31-06-2024”;

5)NUL / NOT NULL

SELECT \* FROM Customers WHERE Address IS NULL;

SELECT \* FROM Customers WHERE Address IS NOT NULL;

6) LIMIT

SELECT \* FROM Customers LIMIT 3;

SELECT \* FROM Customers FETCH FIRST 3 ROWS ONLY;

SELECT TOP 50 PERCENT \* FROM Customers;

SELECT TOP 3 \* FROM Customers WHERE Country='Germany';

SELECT TOP 3 \* FROM Customers ORDER BY CustomerName DESC;

7) Aggregate Function

* MIN() - returns the smallest value within the selected column
* MAX() - returns the largest value within the selected column
* COUNT() - returns the number of rows in a set
* SUM() - returns the total sum of a numerical column
* AVG() - returns the average value of a numerical column

SELECT MIN(Price) FROM Products;

SELECT MIN(Price) AS MIN\_PRICE FROM Products;

SELECT MAX(Price) FROM Products;

SELECT COUNT(**ProductID**) FROM Products;

8) Like

SELECT \* FROM Customers WHERE CustomerName LIKE 'raj%';

SELECT \* FROM Customers WHERE CustomerName LIKE '%aa';

9) Aliases / AS

SELECT CustomerID FROM Customers;

SELECT CustomerID AS ID FROM Customers;

10) Join

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Types Of Join : 3 Types   **1) inner join  2) outer join 3) cross join**

1) Inner Join /Join

**customer           Feedabck**

cust\_id            PK fed\_id    PK

cust\_name        cust\_id   FK

pass   msg

* select \* from customer join feedback   on   customer.cust\_id = feedback.cust\_id
* select feedback.\*,customers.name from customer join feedback   on   customer.cust\_id = feedback.cust\_id

customer           order           product

cust\_id            order\_id         prod\_id

cust\_name        cust\_id         pro\_name

pass   prod\_id   pro\_price

select \* from order join customer on order.cust\_id=customer.cust\_id

join product on order.prod\_id=product.prod\_id

2) Outer Join

* Left Outer Join

select \* from **user\_tbl**    left outer join     **feedback** on user\_tbl.uid=feedback.uid

* Right Outer Join

select \* from user\_tbl right outer join feedback on user\_tbl.uid=feedback.uid

* Full join

select \* from user\_tbl full join feedback

3) Cross Join

select \* from user\_tbl cross join feedback

==================================================================================

11)  **index** & **Views** in SQL

INDEX

The CREATE INDEX statement is used to create indexes in tables.Indexes are used to retrieve data from the database more quickly than otherwise.

The users cannot see the indexes, they are just used to speed up searches/queries 100 times faster.  (SBI BANK   find Account Number)

Type : 2 type

**Simple**:   on only one table column

**Composite** : on more than 1 column in table

Syntex :CREATE INDEX custindex ON customer(cus\_id,mobile))

**Views  (Security Concept/ sub table virtual table)  Exa: (BANK DUPLICATE TABLE)**

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

**CREATE VIEW customer\_view AS SELECT cus\_id,cust\_name,mobile,address,pincode FROM customer**

12) Cursor /Procedure/Trigger

**Cursor**

A cursor is a temporary work area created in the system memory when a SQL

statement is executed

Two Type :

**Implicit** : DML statement **insert/update/delete/              Select** only one row

**Explicit** : **Select** more than one row data

function abc(aid,aname,aemail)

{

“insert in to customer (id,name,email) value (aid, aname, aemail)”

}  
  
abc(“1”,”nagar”,”nagar@gmail.com”)

abc(“2”,”raj”,”raj@gmail.com”)

**Procedure :**

A stored procedure is a prepared SQL code that you can save, so the code

can be reused over and over again.So if you have an SQL query that you write over

and over again, save it as a stored procedure, and then just call it to execute it.

**Create procedure insert\_cust(**

**In cust\_name varchar(255),**

**In user\_name varchar(255),**

**In password varchar(255),**

**In email varchar(255),**

**In mobile bigint(11),**

**in address varchar(255),**

**in pincode bigint(11)**

**)**

**Begin**

**insert into customer(cust\_name,user\_name,password,email,mobile,address,pincode) values(cust\_name,user\_name,password,email,mobile,address,pincode);**

**End**

**//**

**than**

**=>call insert\_cust('janak','janak@123','1234','**[**janak@gmail.com**](mailto:janak@gmail.com)**','21231541','Maninagar','123456')**

**=>call insert\_cust('raj','raj@123','1234','**[**janak@gmail.com**](mailto:janak@gmail.com)**','21231541','Maninagar','123456')**

**Trigger :**

A MySQL trigger is a stored program (with queries) which is executed automatically to respond to a specific event such as insertion, updation or deletion occurring in a table.

**BEFORE INSERT – activated before data is inserted into the table.**

**AFTER INSERT- activated after data is inserted into the table.**

**BEFORE UPDATE – activated before data in the table is updated.**

**AFTER UPDATE - activated after data in the table is updated.**

**BEFORE DELETE – activated before data is removed from the table.**

**AFTER DELETE – activated after data is removed from the table**

=>create table reg\_log( uid int(100),unm varchar(100),pass varchar(100),gen varchar(100),lag varchar(100),cid varchar(100),img varchar(100),status varchar(100),entry\_date\_time datetime);

=>CREATE TRIGGER insert\_trigger\_reg BEFORE INSERT ON reg FOR EACH ROW

BEGIN

insert into reg\_log(uid,unm,pass,gen,lag,cid,img,status,Entry\_date\_time) values (new.uid,new.unm,new.pass,new.gen,new.lag,new.cid,new.img,new.status,now());

END//