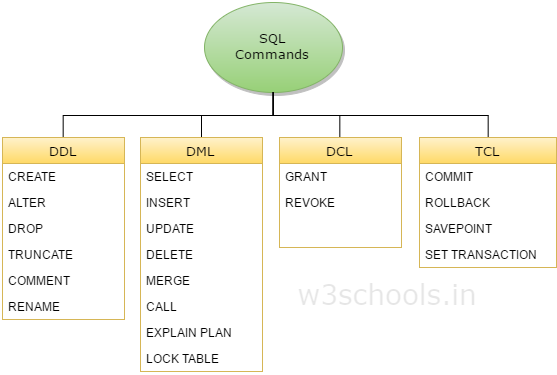
Online Editor:-

https://paiza.io/projects/7ppkVu-xLcLWGJtaNNzEQw?language=mysql

# SQL OVERVIEW



MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

MySQL is written in [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B). Its SQL parser is written in [yacc](https://en.wikipedia.org/wiki/Yacc), but it uses a home-brewed [lexical analyzer](https://en.wikipedia.org/wiki/Lexical_analysis).[[18]](#cite_note-18) MySQL works on many [system platforms](https://en.wikipedia.org/wiki/System_platform), including [AIX](https://en.wikipedia.org/wiki/AIX_operating_system), [BSDi](https://en.wikipedia.org/wiki/BSD/OS), [FreeBSD](https://en.wikipedia.org/wiki/FreeBSD), [HP-UX](https://en.wikipedia.org/wiki/HP-UX), [eComStation](https://en.wikipedia.org/wiki/EComStation), [i5/OS](https://en.wikipedia.org/wiki/IBM_i5/OS), [IRIX](https://en.wikipedia.org/wiki/IRIX), [Linux](https://en.wikipedia.org/wiki/Linux), [macOS](https://en.wikipedia.org/wiki/MacOS), [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [NetBSD](https://en.wikipedia.org/wiki/NetBSD), [Novell NetWare](https://en.wikipedia.org/wiki/Novell_NetWare), [OpenBSD](https://en.wikipedia.org/wiki/OpenBSD), [OpenSolaris](https://en.wikipedia.org/wiki/OpenSolaris), [OS/2](https://en.wikipedia.org/wiki/OS/2) Warp, [QNX](https://en.wikipedia.org/wiki/QNX), [Oracle Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)), [Symbian](https://en.wikipedia.org/wiki/Symbian), [SunOS](https://en.wikipedia.org/wiki/SunOS), [SCO OpenServer](https://en.wikipedia.org/wiki/SCO_OpenServer), SCO [UnixWare](https://en.wikipedia.org/wiki/UnixWare), Sanos and [Tru64](https://en.wikipedia.org/wiki/Tru64). A port of MySQL to [OpenVMS](https://en.wikipedia.org/wiki/OpenVMS) also exists

# What is Database

A database is an organized collection of data.

**Database handlers** create database in such a way that only one set of software program provide access of data to all the users.

The main purpose of database is to operate large amount of information by storing, retrieving and managing.

There are many dynamic websites on the world wide web now a days which are handled through databases. For example, a model to checks the availability of rooms in a hotel. It is an example of dynamic website that uses database.

There are many database available like MySQL, Sybase, Oracle, Mango DB, Informix, Postgre, SQL Server etc.

SQL or Structured Query Language is used to perform operation on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.

**What is DBMS?**

1) DBMS applications store data as file.

2) In DBMS, data is generally stored in either a hierarchical form or a navigational form.

3) Normalization is not present in DBMS.

4) DBMS does not apply any security with regards to data manipulation.

5) DBMS uses file system to store data, so there will be no relation between the tables.

6) DBMS has to provide some uniform methods to access the stored information.

7) DBMS does not support distributed database.

8) DBMS is meant to be for small organization and deal with small data. it supports single user.

9) Examples of DBMS are file systems, xml etc.

***What is RDBMS?***

1) RDBMS applications store data in a tabular form.

2) RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.

3) Normalization is present in RDBMS.

4) RDBMS defines the integrity constraint for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.

5)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.

6)RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.

7)RDBMS supports distributed database.

8)RDBMS is designed to handle large amount of data. it supports multiple users.

9)Examples of RDBMS are mysql, postgre, sql server, oracle etc.

Key Difference between DBMS and RDBMS:

The key difference is that RDBMS (relational database management system) applications store data in a tabular form, while DBMS applications store data as files.

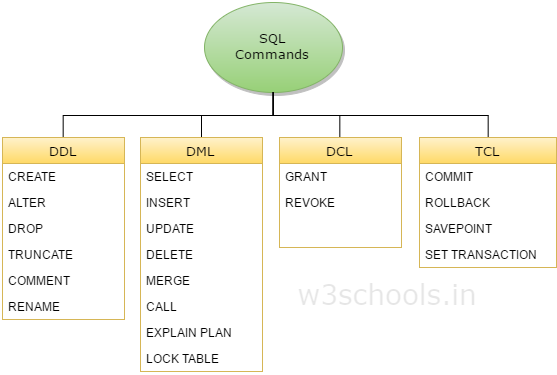
Does that mean there are no tables in a DBMS?

There can be, but there will be no “relation” between the tables, like in a RDBMS. In DBMS, data is generally stored in either a hierarchical form or a navigational form. This means that a single data unit will have one parent node and zero, one or more children nodes. It may even be stored in a graph form, which can be seen in the network model.

In a RDBMS, the tables will have an identifier called primary key. Data values will be stored in the form of tables. The relationships between these data values will be stored in the form of a table as well. Every value stored in the relational database is accessible. This value can be updated by the system. The data in this system is also physically and logically independent.

The four main categories of SQL statements are as follows:

1. DML (Data Manipulation Language)  
2. DDL (Data Definition Language)  
3. DCL (Data Control Language)  
4. TCL (Transaction Control Language)



### **DML (Data Manipulation Language)**

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

DML statements include the following:

SELECT – select records from a table  
INSERT – insert new records  
UPDATE – update/Modify existing records  
DELETE – delete existing records

### **DDL (Data Definition Language)**

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

CREATE – create a new Table, database, schema  
ALTER – alter existing table, column description  
DROP – delete existing objects from database

### **DCL (Data Control Language)**

DCL statements control the level of access that users have on database objects.

GRANT – allows users to read/write on certain database objects  
REVOKE – keeps users from read/write permission on database objects

### **TCL (Transaction Control Language)**

TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.

BEGIN Transaction – opens a transaction  
COMMIT Transaction – commits a transaction  
ROLLBACK Transaction – ROLLBACK a transaction in case of any error

# Comparison Functions and Operations

**Comparison Operators**

| **Name** | **Description** |
| --- | --- |
| [BETWEEN ... AND ...](#operator_between) | Check whether a value is within a range of values |
| [COALESCE()](#function_coalesce) | Return the first non-NULL argument |
| [=](#operator_equal) | Equal operator |
| [<=>](#operator_equal-to) | NULL-safe equal to operator |
| [>](#operator_greater-than) | Greater than operator |
| [>=](#operator_greater-than-or-equal) | Greater than or equal operator |
| [GREATEST()](#function_greatest) | Return the largest argument |
| [IN()](#function_in) | Check whether a value is within a set of values |
| [INTERVAL()](#function_interval) | Return the index of the argument that is less than the first argument |
| [IS](#operator_is) | Test a value against a boolean |
| [IS NOT](#operator_is-not) | Test a value against a boolean |
| [IS NOT NULL](#operator_is-not-null) | NOT NULL value test |
| [IS NULL](#operator_is-null) | NULL value test |
| [ISNULL()](#function_isnull) | Test whether the argument is NULL |
| [LEAST()](#function_least) | Return the smallest argument |
| [<](#operator_less-than) | Less than operator |
| [<=](#operator_less-than-or-equal) | Less than or equal operator |
| [LIKE](#operator_like) | Simple pattern matching |
| [NOT BETWEEN ... AND ...](#operator_not-between) | Check whether a value is not within a range of values |
| [!=, <>](#operator_not-equal) | Not equal operator |
| [NOT IN()](#function_not-in) | Check whether a value is not within a set of values |
| [NOT LIKE](#operator_not-like) | Negation of simple pattern matching |
| [STRCMP()](#function_strcmp) | Compare two strings |

MYSQL Commands:-

# DATABASE Commands

To create a database  Create database Sriram;

To show database  Show databases;

To use database  use Sriram;

To delete database  drop database Sriram;

To know current using database  select database();

# TABLE Commands

## DML COMMANDS

To show tables  Show tables;

To describe table desc emp;

To insert values into table  insert into emp values(100,’sriram’,’IT’,50000);

Insert into emp(name,dept) values(‘naresh’,’sales’);

Insert into emp(name,dept) values(‘naresh’,’sales’),(‘Balu’,’IT’),(‘BALA’,’sales’);

Insert into emp(name,dept) values(‘naresh’,’sales’) where dept=’sales’;

Insert into emp(name,dept) select name,dept from emp where salary=30000;

To select table values 

Select \* from emp;

Select \* from Sriram.emp;

Select name,salary from emp;

Select salary from emp where salary >= 30000;

Select salary from emp where salary >= 30000 and dept=’sales’;

Select \* from emp where salary = 30000 or dept =’sales’;

To update values  update emp set dept=’IT’ , it will change all dept values as IT

Update emp set dept=’sales’ where name=’srirm’;

Update emp set dept=’sales’,salary=2000 where name=’srirm’;

To delete Values  delete from emp where salary=2000;

Delete from emp where name=’Balu’ and salary=2000;

Delete from emp; -> to delete all table values

Update emp set name=’’ where salary=0;  To delete single value

## DDL COMMANDS

To create a table  create table emp(id int auto\_increment primary\_key,

Name varchar(20),

Dept varchar(10),

Salary int(10) default ‘3000’ not null );

To delete a table  drop table emp;

To alter a table  alter table emp drop dept  to delete entire column from emp table

Alter table emp modify salary int(10)  to change column data type

Alter table emp add column sal int not null;  to add a column to table

Alter table emp add column sal int not null after name ;  to add a column to table after name column

Alter table emp add column sal int not null after name,add column village varchar(20) ;  to add multiple columns to table.

Alter table emp change name NAME varchar(20); To change column name

To rename table  rename table emp to employee;

Rename table emp to employee,dept to department;

To truncate table  TRUNCATE TABLE emp;

## D/B TRUNCATE, DELETE and DROP

DELETE

The DELETE command is used to remove rows from a table. A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed. After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it. Note that this operation will cause all DELETE triggers on the table to fire.

* DELETE is a DML command.
* DELETE is executed using a row lock, each row in the table is locked for deletion.
* We can use where clause with DELETE to filter & delete specific records.
* The DELETE command is used to remove rows from a table based on WHERE condition.
* It maintain the log, so it slower than TRUNCATE.

### TRUNCATE

TRUNCATE removes **all rows** from a table. The operation cannot be rolled back and no triggers will be fired. As such, TRUCATE is faster and doesn't use as much undo space as a DELETE.

* TRUNCATE is a DDL command
* TRUNCATE is executed using a table lock and whole table is locked for remove all records.
* We cannot use Where clause with TRUNCATE.
* TRUNCATE removes all rows from a table.

### DROP

* The DROP command removes a table from the database.
* All the tables' rows, indexes and privileges will also be removed.
* No DML triggers will be fired.
* The operation cannot be rolled back.
* DROP and TRUNCATE are DDL commands, whereas DELETE is a DML command.
* DELETE operations can be rolled back (undone), while DROP and TRUNCATE operations cannot be rolled back.

# Comment a line

## **Comment Syntax**

MySQL Server supports three comment styles:

* From a # character to the end of the line.
* From a --  sequence to the end of the line. In MySQL, the --  (double-dash) comment style requires the second dash to be followed by at least one whitespace or control character (such as a space, tab, newline, and so on). This syntax differs slightly from standard SQL comment syntax, as discussed in [Section 1.8.2.4, “'--' as the Start of a Comment”](http://dev.mysql.com/doc/refman/5.7/en/ansi-diff-comments.html).
* From a /\* sequence to the following \*/ sequence, as in the C programming language. This syntax enables a comment to extend over multiple lines because the beginning and closing sequences need not be on the same line.

The following example demonstrates all three comment styles:

mysql> SELECT 1+1; # This comment continues to the end of line

mysql> SELECT 1+1; -- This comment continues to the end of line

mysql> SELECT 1 /\* this is an in-line comment \*/ + 1;

mysql> SELECT 1+

/\*

this is a

multiple-line comment

\*/

1;

# Primary Key Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only one primary key, which may consist of single or multiple fields.

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

**MySQL:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);

# Foreign Key Constraint

A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table

The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

**MySQL:**

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

# Check Key constraint

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that you can not have any person below 18 years:

**MySQL:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CHECK (Age>=18)  
);

# Default constraint

The DEFAULT constraint is used to provide a default value for a column.

The default value will be added to all new records IF no other value is specified.

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

**My SQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    City varchar(255) DEFAULT 'Sandnes'  
);

# JOINS

### What are JOINS?

Joins help retrieving data from two or more database tables.

The tables are mutually related using primary and foreign keys.

Note: JOIN is the most misunderstood topic amongst SQL leaners. For sake of simplicity and ease of understanding , we will be using a new Database to practice sample.  As shown below

### Summary

* JOINS allow us to combine data from more than one table into a single result set.
* JOINS have better performance compared to sub queries
* INNER JOINS only return rows that meet the given criteria.
* OUTER JOINS can also return rows where no matches have been found. The unmatched rows are returned with the NULL keyword.
* The major JOIN types include Inner, Left Outer, Right Outer, Cross JOINS etc.
* The frequently used clause in JOIN operations is "ON". "USING" clause requires that matching columns be of the same name.
* JOINS can also be used in other clauses such as GROUP BY, WHERE, SUB QUERIES, AGGREGATE FUNCTIONS etc.

### Cross JOIN

Cross JOIN is a simplest form of JOINs which matches each row from one database table to all rows of another.

In other words it gives us combinations of each row of first table with all records in second table.

Suppose we want to get all member records against all the movie records, we can use the script shown below to get our desired results.

SELECT \* FROM `movies` CROSS JOIN `members`

### INNER JOIN

The inner JOIN is used to return rows from both tables that satisfy the given condition.

Suppose , you want to get list of members who have rented movies together with titles of movies rented by them. You can simply use an INNER JOIN for that, which returns rows from both tables that satisfy with given conditions.

SELECT members.`first\_name` , members.`last\_name` , movies.`title`

FROM members ,movies

WHERE movies.`id` = members.`movie\_id`

### Outer JOINs

MySQL Outer JOINs return all records matching from both tables .

It can detect records having no match in joined table. It returns NULL values for records of joined table if no match is found.

Sounds Confusing ? Let's look into an example -

### LEFT JOIN

Assume now you want to get titles of all movies together with names of members who have rented them. It is clear that some movies have not being rented by any one. We can simply use LEFT JOIN for the purpose.

The LEFT JOIN returns all the rows from the table on the left even if no matching rows have been found in the table on the right. Where no matches have been found in the table on the right, NULL is returned.

SELECT A.`title` , B.`first\_name` , B.`last\_name`

FROM `movies` AS A

LEFT JOIN `members` AS B

ON B.`movie\_id` = A.`id`

### RIGHT JOIN

RIGHT JOIN is obviously the opposite of LEFT JOIN. The RIGHT JOIN returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL is returned.

In our example,  let's assume that you need to get names of members and movies rented by them. Now we have a new member who has not rented any movie yet

SELECT A.`first\_name` , A.`last\_name` , B.`title`

FROM `members` AS A

RIGHT JOIN `movies` AS B

ON B.`id` = A.`movie\_id`

# Subquery’s

### A MySQL subquery is a query that is nested inside another query such as [SELECT](http://www.mysqltutorial.org/mysql-select-statement-query-data.aspx), [INSERT](http://www.mysqltutorial.org/mysql-insert-statement.aspx), [UPDATE](http://www.mysqltutorial.org/mysql-update-data.aspx)or [DELETE](http://www.mysqltutorial.org/mysql-delete-statement.aspx). In addition, a MySQL subquery can be nested inside another subquery.

A MySQL subquery is also called an inner query while the query that contains the subquery is called an outer query.

Let’s take a look at the following subquery that returns employees who locate in the offices in the USA.

* The subquery returns all offices codes of the offices that locate in the USA.
* The outer query selects the last name and first name of employees whose office code is in the result set returned by the subquery.

A subquery is a [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html) statement within another statement.

SELECT first\_name,last\_name, salary FROM emp\_details WHERE salary >(SELECT salary FROM emp\_details WHERE first\_name='Alexander');

SELECT \* FROM t1 WHERE column1 = (SELECT column1 FROM t2);