MySQL

Tushar B. Kute,
http://tusharkute.com

SQL

- SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.
- SQL is not a database system, but it is a query language.
- Suppose you want to perform the queries of SQL language on the stored data in the database.
- You are required to install any database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc

SQL

- SQL is a short-form of the structured query language, and it is pronounced as S-Q-L or sometimes as See-Quell.
- This database language is mainly designed for maintaining the data in relational database management systems.
- It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables).
- It is also designed for stream processing in RDBMS.

SQL

- You can easily create and manipulate the database, access and modify the table rows and columns, etc.
- This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.
- If you want to get a job in the field of data science, then it is the most important query language to learn.
- Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

SQL: Why?

- The basic use of SQL for data professionals and SQL users is to insert, update, and delete the data from the relational database.
- SQL allows the data professionals and users to retrieve the data from the relational database management systems.
- It also helps them to describe the structured data.
- It allows SQL users to create, drop, and manipulate the database and its tables.

SQL: Why?

- It also helps in creating the view, stored procedure, and functions in the relational database.
- It allows you to define the data and modify that stored data in the relational database.
- It also allows SQL users to set the permissions or constraints on table columns, views, and stored procedures.

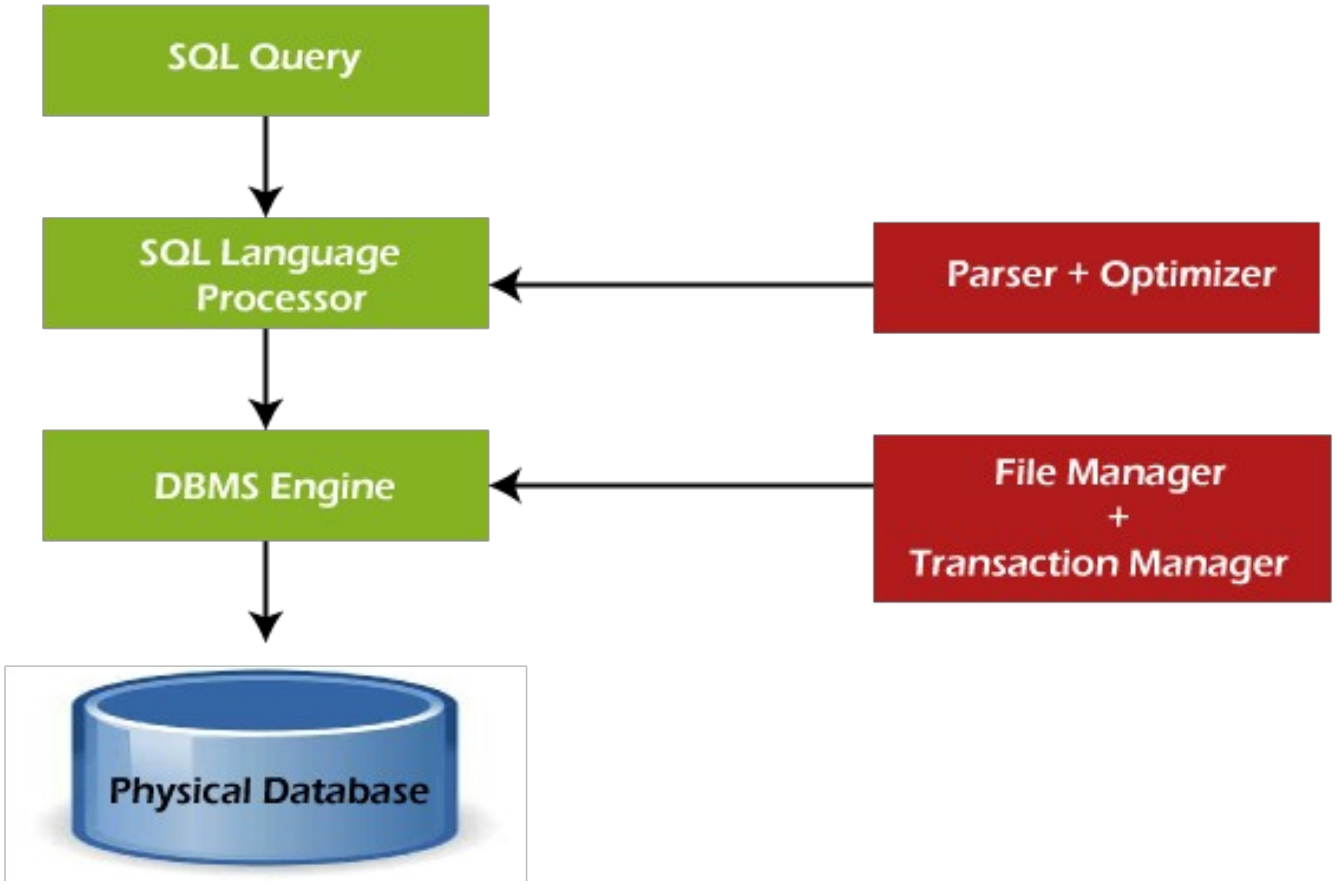
SQL: When?

- "A Relational Model of Data for Large Shared Data Banks" was a paper which was published by the great computer scientist "E.F. Codd" in 1970.
- The IBM researchers Raymond Boyce and Donald Chamberlin originally developed the SEQUEL (Structured English Query Language) after learning from the paper given by E.F. Codd.
- They both developed the SQL at the San Jose Research laboratory of IBM Corporation in 1970.

SQL: Process

- When we are executing the command of SQL on any Relational database management system, then the system automatically finds the best routine to carry out our request, and the SQL engine determines how to interpret that particular command.
- Structured Query Language contains the following four components in its process:
  - Query Dispatcher
  - Optimization Engines
  - Classic Query Engine
  - SQL Query Engine, etc.

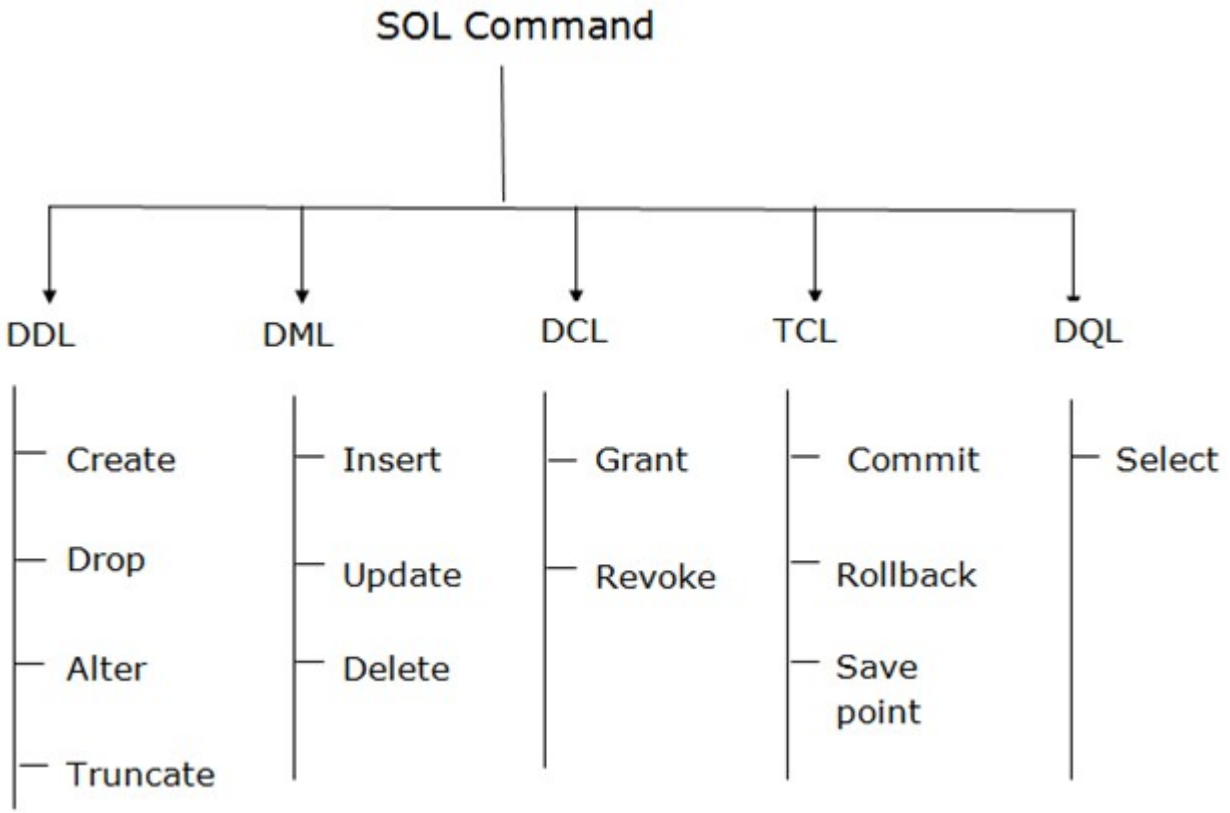
SQL: Process


SQL Query
SQL Language Processor
Parser + Optimizer
DBMS Engine
File Manager + Transaction Manager
Physical Database

SQL Commands

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

SQL Commands



Data Definition Language

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.
- Here are some commands that come under DDL:
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
- Here are some commands that come under DML:
  - INSERT
  - UPDATE
  - DELETE

Data Control Language

- DCL commands are used to grant and take back authority from any database user.
- Here are some commands that come under DCL:
  - Grant
  - Revoke

Transaction Control Language

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.
- Here are some commands that come under TCL:
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

Data Query Language

- DQL is used to fetch the data from the database.
- It uses only one command:
  - SELECT

Data Definition Language (DDL)

- CREATE
  - It is used to create a new table in the database.
- DROP:
  - It is used to delete both the structure and record stored in the table.
- ALTER:
  - It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.
- TRUNCATE:
  - It is used to delete all the rows from the table and free the space containing the table.

MySQL Data Types

- A Data Type specifies a particular type of data, like integer, floating points, Boolean, etc.
- It also identifies the possible values for that type, the operations that can be performed on that type, and the way the values of that type are stored.
- In MySQL, each database table has many columns and contains specific data types for each column.

MySQL Data Types

- We can determine the data type in MySQL with the following characteristics:
  - The type of values (fixed or variable) it represents.
  - The storage space it takes is based on whether the values are a fixed-length or variable length.
  - Its values can be indexed or not.
  - How MySQL performs a comparison of values of a particular data type.

MySQL Data Types: Numeric

- MySQL has all essential SQL numeric data types. These data types can include the exact numeric data types (For example, integer, decimal, numeric, etc.), as well as the approximate numeric data types (For example, float, real, and double precision).
- It also supports BIT datatype to store bit values. In MySQL, numeric data types are categories into two types, either signed or unsigned except for bit data type.

MySQL Data Types: Numeric

- TINYINT
  - It is a very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. We can specify a width of up to 4 digits. It takes 1 byte for storage.
- SMALLINT
  - It is a small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. We can specify a width of up to 5 digits. It requires 2 bytes for storage.

MySQL Data Types: Numeric

- MEDIUMINT
  - It is a medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. We can specify a width of up to 9 digits. It requires 3 bytes for storage.
- INT
  - It is a normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. We can specify a width of up to 11 digits. It requires 4 bytes for storage.

MySQL Data Types: Numeric

- BIGINT
  - It is a large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. We can specify a width of up to 20 digits. It requires 8 bytes for storage.
- FLOAT(m,d)
  - It is a floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is the number of decimals, and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float type. It requires 2 bytes for storage.

MySQL Data Types: Numeric

- DOUBLE(m,d)
  - It is a double-precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d).
  - This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double. It requires 8 bytes for storage.

MySQL Data Types: Numeric

- DECIMAL(m,d)
  - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.
- BIT(m)
  - It is used for storing bit values into the table column. Here, M determines the number of bit per value that has a range of 1 to 64.
- BOOL
  - It is used only for the true and false condition. It considered numeric value 1 as true and 0 as false.
- BOOLEAN
  - It is Similar to the BOOL.

MySQL Data Types: Date and Time

- This data type is used to represent temporal values such as date, time, datetime, timestamp, and year.
- Each temporal type contains values, including zero. When we insert the invalid value, MySQL cannot represent it, and then zero value is used.

MySQL Data Types: Date and Time

- YEAR[(2|4)]
  - Year value as 2 digits or 4 digits. The default is 4 digits. It takes 1 byte for storage.
- DATE
  - Values range from '1000-01-01' to '9999-12-31'. Displayed as 'yyyy-mm-dd'. It takes 3 bytes for storage.
- TIME
  - Values range from '-838:59:59' to '838:59:59'. Displayed as 'HH:MM:SS'. It takes 3 bytes plus fractional seconds for storage.

MySQL Data Types: Date and Time

- DATETIME
  - Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Displayed as 'yyyy-mm-dd hh:mm:ss'. It takes 5 bytes plus fractional seconds for storage.
- TIMESTAMP(m)
  - Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC. Displayed as 'YYYY-MM-DD HH:MM:SS'. It takes 4 bytes plus fractional seconds for storage.

MySQL Data Types: String

- The string data type is used to hold plain text and binary data, for example, files, images, etc.
- MySQL can perform searching and comparison of string value based on the pattern matching such as LIKE operator, Regular Expressions, etc.

MySQL Data Types: String

- CHAR(size)
  - It can have a maximum size of 255 characters. Here size is the number of characters to store. Fixed-length strings. Space padded on the right to equal size characters.
- VARCHAR(size)
  - It can have a maximum size of 255 characters. Here size is the number of characters to store. Variable-length string.

MySQL Data Types: String

- TINYTEXT(size)
  - It can have a maximum size of 255 characters. Here size is the number of characters to store.
- TEXT(size)
  - Maximum size of 65,535 characters. Here size is the number of characters to store.

MySQL Data Types: String

- MEDIUMTEXT(size)
  - It can have a maximum size of 16,777,215 characters. Here size is the number of characters to store.
- LONGTEXT(size)
  - It can have a maximum size of 4GB or 4,294,967,295 characters. Here size is the number of characters to store.
- BINARY(size)
  - It can have a maximum size of 255 characters. Here size is the number of binary characters to store. Fixed-length strings. Space padded on the right to equal size characters.

MySQL Data Types: String

- VARBINARY(size)
  - It can have a maximum size of 255 characters. Here size is the number of characters to store. Variable-length string. (introduced in MySQL 4.1.2)
- ENUM
  - It takes 1 or 2 bytes that depend on the number of enumeration values. An ENUM can have a maximum of 65,535 values.
  - It is short for enumeration, which means that each column may have one of the specified possible values. It uses numeric indexes (1, 2, 3…) to represent string values.

MySQL Data Types: Binary

- BLOB in MySQL is a data type that can hold a variable amount of data.
- They are categories into four different types based on the maximum length of values can hold.

MySQL Data Types: Binary

- TINYBLOB
  - It can hold a maximum size of 255 bytes.
- BLOB(size)
  - It can hold the maximum size of 65,535 bytes.
- MEDIUMBLOB
  - It can hold the maximum size of 16,777,215 bytes.
- LONGBLOB
  - It can hold the maximum size of 4gb or 4,294,967,295 bytes.

Creating a table

- MySQL allows us to create a table into the database by using the CREATE TABLE command. Following is a generic syntax for creating a MySQL table in the database.

```
CREATE TABLE [IF NOT EXISTS] table_name(
   column_definition1,
   column_definition2,
   ........,
   table_constraints
);
```

Creating a table

- database_name
  - It is the name of a new table. It should be unique in the MySQL database that we have selected. The IF NOT EXIST clause avoids an error when we create a table into the selected database that already exists.
- column_definition
  - It specifies the name of the column along with data types for each column. The columns in table definition are separated by the comma operator. The syntax of column definition is as follows:
  - column_name1 data_type(size) [NULL | NOT NULL]
- table_constraints
  - It specifies the table constraints such as PRIMARY KEY, UNIQUE KEY, FOREIGN KEY, CHECK, etc.

Creating a table

```
CREATE TABLE employee_table(
    id int NOT NULL AUTO_INCREMENT,
    name varchar(45) NOT NULL,
    occupation varchar(35) NOT NULL,
    age int NOT NULL,
    PRIMARY KEY (id)
);
```

Options

- NOT NULL is a field attribute, and it is used because we don't want this field to be NULL. If we try to create a record with a NULL value, then MySQL will raise an error.
- The field attribute AUTO_INCREMENT specifies MySQL to go ahead and add the next available number to the id field. PRIMARY KEY is used to define a column's uniqueness. We can use multiple columns separated by a comma to define a primary key.

Unique Key

- A unique key in MySQL is a single field or combination of fields that ensure all values going to store into the column will be unique. It means a column cannot stores duplicate values.
- For example, the email addresses and roll numbers of students in the "student_info" table or contact number of employees in the "Employee" table should be unique.
- MySQL allows us to use more than one column with UNIQUE constraint in a table. It can accept a null value, but MySQL allowed only one null value per column.
- It ensures the integrity of the column or group of columns to store different values into a table.

Unique Key: Examples

```
CREATE TABLE table_name(
    col1 datatype,
    col2 datatype UNIQUE,
    ...
);
```

Unique Key: Examples

```
CREATE TABLE Student2 (
    Stud_ID int NOT NULL UNIQUE,
    Name varchar(45),
    Email varchar(45),
    Age int,
    City varchar(25)
);
```

Primary Key

- MySQL primary key is a single or combination of the field, which is used to identify each record in a table uniquely.
- If the column contains primary key constraints, then it cannot be null or empty. A table may have duplicate columns, but it can contain only one primary key.
- It always contains unique value into a column.

Primary Key

- When you insert a new row into the table, the primary key column can also use the AUTO_INCREMENT attribute to generate a sequential number for that row automatically.
- MySQL automatically creates an index named "Primary" after defining a primary key into the table. Since it has an associated index, we can say that the primary key makes the query performance fast.

Primary Key

- Following are the rules for the primary key:
  - The primary key column value must be unique.
  - Each table can contain only one primary key.
  - The primary key column cannot be null or empty.
  - MySQL does not allow us to insert a new row with the existing primary key.
  - It is recommended to use INT or BIGINT data type for the primary key column.

Primary Key : Way-1

```
Mysql> CREATE TABLE Login(
   login_id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(40),
   password VARCHAR(55),
   email VARCHAR(55)
);
```

Primary Key : Way-2

```
mysql> CREATE TABLE Students (
        Student_ID int,
        Roll_No int,
        Name varchar(45) NOT NULL,
        Age int,
        City varchar(25),
        Primary Key(Student_ID, Roll_No)
    );
```

Check

- The check constraint is an integrity constraint that controls the value in a particular column.
- It ensures the inserted or updated value in a column must be matched with the given condition.
- In other words, it determines whether the value associated with the column is valid or not with the given condition.

Check

- Before version 8.0.16, MySQL uses the limited version of this constraint.
- In the previous versions, we can create this constraint, but it does not work. It means its syntax is supported but not works with the database.
- With an earlier version, the CREATE TABLE statement can include a CHECK constraint, but they are parsed and ignored by MySQL.
- We can use it with the below syntax in the previous versions:

```
CHECK (expr)
```

Check

- The following statement creates a new table called vehicle where we specify the check constraint on a column:

```
CREATE TABLE vehicle (
    vehicle_no VARCHAR(18) PRIMARY KEY,
    model_name VARCHAR(45),
    cost_price DECIMAL(10,2 ) NOT NULL CHECK (cost_price >= 0),
    sell_price DECIMAL(10,2) NOT NULL CHECK (sell_price >= 0)
);
```

Select

- The SELECT statement in MySQL is used to fetch data from one or more tables. We can retrieve records of all fields or specified fields that match specified criteria using this statement.

```
SELECT field_name1, field_name 2,... field_nameN
FROM table_name1, table_name2...
[WHERE condition]
[GROUP BY field_name(s)]
[HAVING condition]
[ORDER BY field_name(s)]
[OFFSET M ][LIMIT N];
```

Aggregate Functions

- MySQL's aggregate function is used to perform calculations on multiple values and return the result in a single value like the average of all values, the sum of all values, and maximum & minimum value among certain groups of values.
- We mostly use the aggregate functions with SELECT statements in the data query languages.

Aggregate Functions

- count()
  - It returns the number of rows, including rows with NULL values in a group.
- sum()
  - It returns the total summed values (Non-NULL) in a set.
- avg()
  - It returns the average value of an expression.
- Min()
  - It returns the minimum (lowest) value in a set.

Aggregate Functions

- max( )
  - It returns the maximum (highest) value in a set.
- groutp_concat()
  - It returns a concatenated string.

Where clause

- WHERE Clause in MySQL is a keyword used to specify the exact criteria of data or rows that will be affected by the specified SQL statement.
- The WHERE clause can be used with SQL statements like INSERT, UPDATE, SELECT, and DELETE to filter records and perform various operations on the data.

Where clause

- The basic syntax for the WHERE clause when used in a MySQL SELECT WHERE statement is as follows.

  SELECT * FROM tableName WHERE condition;

Where clause with AND

- The WHERE condition in MySQL when used together with the AND logical operator, is only executed if ALL filter criteria specified are met.
- Let's now look at a practical example – Suppose we want to get a list of all the movies in category 2 that were released in 2008, we would use the script shown below is achieve that.

  SELECT * FROM 'movies' WHERE 'category_id' = 2 AND 'year_released' = 2008;

Where clause with OR

- The WHERE clause when used together with the OR operator, is only executed if any or the entire specified filter criteria is met.
- The following script gets all the movies in either category 1 or category 2

```
SELECT * FROM 'movies' WHERE 'category_id' = 1 OR 'category_id' = 2;
```

Where clause with IN

- The WHERE in MySQL clause, when used together with the IN keyword only affects the rows whose values matches the list of values provided in the IN keyword.
- The MySQL IN statement helps to reduce number of OR clauses you may have to use.
- The following MySQL WHERE IN query gives rows where membership_number is either 1 , 2 or 3

```
SELECT * FROM 'members' WHERE
'membership_number' IN (1,2,3);
```

Where clause with NOT IN

- The WHERE clause when used together with the NOT IN keyword DOES NOT affects the rows whose values matches the list of values provided in the NOT IN keyword.
- The following query gives rows where membership_number is NOT 1 , 2 or 3

```
SELECT * FROM 'members' WHERE
'membership_number' NOT IN (1,2,3);
```

Where clause with NOT IN

- SELECT * FROM 'members' WHERE 'gender' = 'Female';
- SELECT * FROM 'payments' WHERE 'amount_paid' > 2000;
- SELECT * FROM 'movies' WHERE 'category_id'<> 1;

The Between operator

- The BETWEEN operator selects values within a given range.
- The values can be numbers, text, or dates.
- The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

The Between operator

- SELECT * FROM Products

  WHERE Price NOT BETWEEN 10 AND 20;

- SELECT * FROM student WHERE name BETWEEN 'Chintoo' AND 'Jenny'

Is NULL

- MySQL IS NULL condition is used to check if there is a NULL value in the expression.
- It is used with SELECT, INSERT, UPDATE and DELETE statement.
- Syntax:
  - expression IS NULL
- Parameter
  - expression: It specifies a value to test if it is NULL value.

Is NULL and Is not NULL

```
SELECT *
FROM officers
WHERE officer_name IS NULL;
```

```
SELECT *
FROM officers
WHERE officer_name IS NOT NULL;
```

Insert into

- INSERT INTO is used to store data in the tables. The INSERT command creates a new row in the table to store data.
- The data is usually supplied by application programs that run on top of the database.
- Basic syntax:
  - INSERT INTO 'table_name' (column_1, column_2,...) VALUES (value_1,value_2,...);

Insert into

- INSERT INTO 'members' ('full_names','gender','physical_address','contact_number') VALUES ('Leonard Hofstadter','Male','Woodcrest',0845738767);

Insert from another table

- The INSERT command can also be used to insert data into a table from another table.
- The basic syntax is as shown below.

```
INSERT INTO table_1 SELECT * FROM table_2;
```

Delete query

- MySQL Delete command is used to delete rows that are no longer required from the database tables.
- It deletes the whole row from the table and returns count of deleted rows.
- Delete command comes in handy to delete temporary or obsolete data from your database.
- The Delete query in MySQL can delete more than one row from a table in a single query.
- This proves to be advantages when removing large numbers of rows from a database table.

Delete query

- To delete a row in MySQL, the DELETE FROM statement is used:

  DELETE FROM 'table_name' [WHERE condition];

- Example:

  DELETE FROM 'movies' WHERE 'movie_id' = 18;

  DELETE FROM 'movies' WHERE 'movie_id' IN (20,21);

Update Query

- UPDATE MySQL command is used to modify rows in a table. The update command can be used to update a single field or multiple fields at the same time.
- It can also be used to update a MySQL table with values from another table.
- MySQL Update Command Syntax

```
UPDATE 'table_name' SET 'column_name' =
'new_value' [WHERE condition];
```

Update Query

- UPDATE 'members' SET 'contact_number' = '0759 253 542' WHERE 'membership_number' = 1;

- UPDATE 'members' SET 'full_names' = 'Janet Smith Jones', 'physical_address' = 'Melrose 123' WHERE 'membership_number' = 2;

Transaction Control Language

- A single unit of work in a database is formed after the consecutive execution of commands is known as a transaction.
- There are certain commands present in SQL known as TCL commands that help the user manage the transactions that take place in a database.
- COMMIT. ROLLBACK and SAVEPOINT are the most commonly used TCL commands in SQL.

Commit

- COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk.
- Whenever DDL commands such as INSERT, UPDATE and DELETE are used, the changes made by these commands are permanent only after closing the current session.
- So before closing the session, one can easily roll back the changes made by the DDL commands. Hence, if we want the changes to be saved permanently to the disk without closing the session, we will use the commit command.
- Syntax:
  - COMMIT;

Commit

- mysql> USE school;
- mysql> CREATE TABLE t_school….
- mysql> START TRANSACTION;
- mysql> INSERT INTO t_school…….
- mysql> SELECT *FROM t_school;
- mysql> COMMIT;

Commit

- Autocommit is by default enabled in MySQL. To turn it off, we will set the value of autocommit as 0.

  mysql> SET autocommit = 0;

```
mysql> SET autocommit = 0;
Query OK, 0 rows affected (0.08 sec)
```

- MySQL, by default, commits every query the user executes. But if the user wishes to commit only the specific queries instead of committing every query, then turning off the autocommit is useful.

Savepoint

- We can divide the database operations into parts.
- For example, we can consider all the insert related queries that we will execute consecutively as one part of the transaction and the delete command as the other part of the transaction.
- Using the SAVEPOINT command in SQL, we can save these different parts of the same transaction using different names.

Savepoint

- For example, we can save all the insert related queries with the savepoint named INS.
- To save all the insert related queries in one savepoint, we have to execute the SAVEPOINT query followed by the savepoint name after finishing the insert command execution.
- Syntax:
  - SAVEPOINT savepoint_name;

Rollback

- While carrying a transaction, we must create savepoints to save different parts of the transaction.
- According to the user's changing requirements, he/she can roll back the transaction to different savepoints.
- Consider a scenario: We have initiated a transaction followed by the table creation and record insertion into the table.
- After inserting records, we have created a savepoint INS. Then we executed a delete query, but later we thought that mistakenly we had removed the useful record.

Rollback

- Therefore in such situations, we have an option of rolling back our transaction.
- In this case, we have to roll back our transaction using the ROLLBACK command to the savepoint INS, which we have created before executing the DELETE query.
- Syntax:

```
ROLLBACK TO savepoint_name;
```

Rollback

- mysql> USE school;
- mysql> CREATE TABLE t_school….
- mysql> INSERT INTO t_school
- mysql> SELECT *FROM t_school;
- mysql> START TRANSACTION;
- mysql> SAVEPOINT Insertion;
- mysql> UPDATE t_school SET….
- mysql> SELECT *FROM t_school;
- mysql> SAVEPOINT Updation;
- mysql> ROLLBACK TO Insertion;
- mysql> SELECT *FROM t_school;

Rollback

- mysql> USE bank;
- mysql> CREATE TABLE customer….
- mysql> INSERT INTO customer….
- mysql> SELECT *FROM customer;
- mysql> START TRANSACTION;
- mysql> SAVEPOINT Insertion;
- mysql> DELETE FROM customer WHERE….
- mysql> SELECT *FROM customer;
- mysql> SAVEPOINT Deletion;
- mysql> ROLLBACK TO Insertion;
- mysql> SELECT *FROM customer;

Data Control Language

- Data control language (DCL) is used to access the stored data. It is mainly used for revoke and to grant the user the required access to a database.
- It helps in controlling access to information stored in a database. It complements the data manipulation language (DML) and the data definition language (DDL).
- It provides the administrators, to remove and set database permissions to desired users as needed.
- These commands are employed to grant, remove and deny permissions to users for retrieving and manipulating a database.

Grant

- The grant statement enables system administrators to assign privileges and roles to the MySQL user accounts so that they can use the assigned permission on the database whenever required.
- Syntax

```
GRANT privilege_name(s)
ON object
TO user_account_name;
```

Grant

- privilege_name(s)
  - It specifies the access rights or grant privilege to user accounts. If we want to give multiple privileges, then use a comma operator to separate them.
- object
  - It determines the privilege level on which the access rights are being granted. It means granting privilege to the table; then the object should be the name of the table.
- user_account_name
  - It determines the account name of the user to whom the access rights would be granted.

Grant : Privilege Levels

- Global

```
GRANT ALL
ON *.*
TO john@localhost;
```

  - It applies to all databases on MySQL server. We need to use *.* syntax for applying global privileges. Here, the user can query data from all databases and tables of the current server.

Grant : Privilege Levels

- Database

```
GRANT ALL
ON mydb.*
TO john@localhost;
```

  - It applies to all objects in the current database. We need to use the db_name.* syntax for applying this privilege.
  - Here, a user can query data from all tables in the given database.

Grant : Privilege Levels

- Table

```
GRANT DELETE
ON mydb.employees
TO john@localhost;
```

  - It applies on all columns in a specified table. We need to use db_name.table_name syntax for assigning this privilege.
  - Here, a user can query data from the given table of the specified database.

Grant : Privilege Levels

- Column

```
GRANT SELECT (col1), INSERT (col1,
col2), UPDATE (col2)
ON mydb.mytable
TO john@localhost;
```

  - It applies on a single column of a table. Here, we must have to specify the column(s) name enclosed with parenthesis for each privilege.
  - The user can select one column, insert values in two columns, and update only one column in the given table.

Grant : Privilege Levels

- Stored Routine

```
GRANT EXECUTE
ON PROCEDURE mydb.myprocedure
TO john@localhost;
```

  - It applies to stored routines (procedure and functions).
  - It contains CREATE ROUTINE, ALTER ROUTINE, EXECUTE, and GRANT OPTION privileges.
  - Here, a user can execute the stored procedure in the current database.

Grant : Privilege Levels

- Proxy

```
GRANT PROXY
ON root
TO tushar@localhost;
```

  - It enables one user to be a proxy for other users.

Grant : Example

- Let us understand the GRANT privileges through the example. First, we need to create a new user named "john@localhost" using the following statement:

```
mysql> CREATE USER john@localhost
IDENTIFIED BY 'john12345';
```

- Next, execute the SHOW GRANT statement to check the privileges assigned to john@localhost using the following query:

```
mysql> SHOW GRANTS FOR john@localhost;
```

```
mysql> GRANT ALL ON mystudentdb.* TO
john@localhost;
```

Revoke Statement

- The revoke statement enables system administrators to revoke privileges and roles to the MySQL user accounts so that they cannot use the assigned permission on the database in the past.
- Syntax:

```
REVOKE privilege_name(s)
ON object
FROM user_account_name;
```

Revoke: Privileges

- Global

```
REVOKE ALL, GRANT OPTION FROM
john@localhost;
```

  - It applies to remove all access rights from the user on MySQL server.

Revoke: Privileges

- Database

```
REVOKE ALL ON mydb.*
FROM john@localhost;
```

  - It applies to revoke all privileges from objects in the current database.

Revoke: Privileges

- Table

```
REVOKE DELETE
ON mydb.employees
FROM john@localhost;
```

  - It applies to revoke privileges from all columns in a specified table.

Revoke: Privileges

- Column

```
REVOKE SELECT (col1), INSERT (col1,
col2), UPDATE (col2) ON mydb.mytable
FROM john@localhost;
```

  - It applies to revoke privileges from a single column of a table.

Revoke: Privileges

- Stored Routine

```
REVOKE EXECUTE ON PROCEDURE/FUNCTION
mydb.myprocedure
FROM john@localhost;
```

  - It applies to revoke all privileges from stored routines (procedure and functions).

Revoke: Privileges

- Proxy

```
REVOKE PROXY ON root
FROM peter@localhost;
```

  - It enables us to revoke the proxy user.

Revoke: Example

- `mysql> REVOKE ALL, GRANT OPTION FROM john@localhost;`
- `mysql> GRANT SELECT, UPDATE, INSERT ON mystudentdb.* TO john@localhost;`
- `mysql> SHOW GRANTS FOR john@localhost;`
- `mysql> REVOKE UPDATE, INSERT ON mystudentdb.* FROM john@localhost;`

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com