# SQL (STRUCTURED QUERY LANGUAGE)

USING

MYSQL

RAJEEV SRIVASTAVA

# SQL: STRUCTURED QUERY LANGUAGE

- Stands for **"Structured Query Language"**
- Also pronounced as "**SEQUEL**" (Structured English QUEry Language)
- Originally developed at **IBM** in the 1970s by **Donald Chamberlin** and **Raymond Boyce**
- Standard access mechanism to every **RDBMS**.
- Case-Insensitive
- 4th Generation Language (Instructions for WHAT to do)
- Standard Based – ANSI / ISO
- First SQL Standard Published in 1986 (**SQL:86**) by ANSI
- Latest is **SQL:2016** OR **ISO/IEC 9075:2016**

## COMPONENTS (CATEGORIES) OF SQL STATEMENTS

- **DDL** : Data Definition Language(CREATE/ALTER/DROP/TRUNCATE)
- **DML**: Data Manipulation Language(INSERT/UPDATE/DELETE)
- **DCL** : Data Control Language (GRANT/REVOKE)
- **DTL** : Data Transaction Language OR
  **TCL** : Transaction Control Language (COMMIT/SAVEPOINT/ROLLBACK)
- **DRL** : Data Retrieval Language OR
  **DQL** : Data Query Language (SELECT)

## CREATING A TABLE

- Data in a Relational database is stored in the form of tables.
- The table is a collection of related data entries and it consists of columns and rows.

```
CREATE TABLE <tableName>  (
 <columnName>   <dataType>,
 <columnName>   <dataType>
);
```

- SQL statements ends with a Semicolon (;)

Find out Restrictions on Table and Column names in MySQL

Max Size of Table in MySQL

Max Number of Columns in a Table in MySQL

## CONSIDERATIONS FOR CREATING TABLE

- Points to be considered before creating a table -
  - What are the **Attributes** (columns/fields) of the tuples(records/rows) to be stored?
  - What are the **Data Types** of the attributes? Should varchar be used instead of char?
  - Which column(s) build the **Primary Key**?
  - What column(s) need to added as **Foreign Keys?**
  - Which column(s) do (not) allow **NULL** values?
  - Which column(s) will have **UNIQUE** values ie. do (not) allow duplicates?
  - Are there **DEFAULT** values for certain columns?

# MYSQL: DATA TYPES

Explore different Data Types available in MySQL with their uses.

- Data Type defines what kind of values can be stored in a column.
- Data Type also defines the way data will be stored in the system and the space required in disk.
- Data Type also impact database performance.
- Ex- Char, Varchar, Text, Integer, Float, Double, Date, Timestamp, Enum, Blob etc.
- More on SQL Datatypes : https://www.w3schools.com/sql/sql_datatypes.asp

# INSERT

- Used to insert data into a table
- Insert command always inserts values as new row –

```
INSERT INTO <tableName> VALUES (<val1>, <val2>);
```

- Insert data into only specific columns of a table -

```
INSERT INTO <tableName> (<col1>) VALUES (<val1>);
```

- Define an insertion order -

```
INSERT INTO <tableName> (<col2>, <col1>) VALUES (<val2>,
<val1>);
```

- Missing attribute → NULL.
- May drop attribute names if give them in order

## NULL VALUE

- When you do not insert data into a column of a table for a specific row, then by default a NULL value will be inserted into that column by the database.

```
INSERT INTO dept (deptno, deptname) VALUES (40, ‘BIOM’);
```

- NULL value does not occupy space in memory
- NULL value is independent of data type
- A NULL value is not a zero (0) OR an empty string (‘ ’), rather it represents an **Unknown** or **Not Applicable** value.

```
INSERT INTO dept (deptno) VALUES (40);
INSERT INTO dept (deptno, deptname) VALUES (40, NULL);
```

# SELECT

- Used to Retrieve/Fetch information from the database.

```
SELECT <colName> FROM <tableName> [WHERE <condition>];
SELECT <col1>, <col2>  FROM <tableName> [WHERE
<condition>];
```

- An asterisk symbol (*) Represents all columns/attributes.

```
SELECT *  FROM <tableName> [WHERE <condition>];
```

```
SELECT EMPNO, EMPNAME FROM EMPLOYEE;
SELECT * FROM EMPLOYEE WHERE EMPNAME = “AMIT”;
```

## COMBINING MORE THAT ONE CONDITIONS (AND/OR), NOT

- More than one conditions may be specified in WHERE clause to fetch the data matching multiple criteria -

```
    SELECT <colName> FROM <tableName> [WHERE <condition1>
    [AND|OR WHERE <condition2>]…];
```

- AND – will match both the conditions
- OR – will match either of the conditions
- NOT – will display not unmatching records

```
    SELECT <colName> FROM <tableName> WHERE NOT (<condition>);
```

```
SELECT * FROM EMPLOYEE WHERE EMPNAME = “AMIT” AND SALARY > 10000;

SELECT * FROM EMPLOYEE WHERE EMPNAME = “AMIT” OR SALARY > 10000;
SELECT * FROM EMPLOYEE WHERE NOT (EMPNAME = “AMIT”);
SELECT * FROM EMPLOYEE WHERE EMPNAME <> “AMIT”;
```

# SELECTION & PROJECTION

- **SELECTION** (σ) – limiting rows (by using WHERE clause)

```
SELECT * FROM <table name> WHERE <col1> = <val1> ;
```

- **PROJECTION** (π) – limiting columns (by using SELECT clause)

```
SELECT <col1>, <col2> FROM <table name>;
```

- SELECTION & PROJECTION – limiting rows and columns selection (by using SELECT and WHERE clauses together)

```
SELECT <col1>, <col2> FROM <table name> WHERE <col1> =
<val1> ;
```