B.TECH. COMPUTER SCIENCE AND ENGINEERING,
SEMESTER V
GNDU, AMRITSAR

SQL AND ITS BASIC QUERIES

Anhat Singh

Roll No: 17032000307

Section B, B.Tech. CSE

Subject: Relational Database Management System (CSL-332)

Submitted To: Dr. Hardeep Singh

2022

GURU NANAL DEV UNIVERSITY, AMRITSAR

CSL-332: RELATIONAL DATABASE MANAGEMENT SYSTEM

Anhat Singh

Acknowledgement

In successfully completing this file, many people have helped me. I would like to thank all those who are related to this project.

Firstly, I would thank Dr. Hardeep Singh for guiding me to complete this file. His suggestions and directions have helped in the completion of this file. I am highly indebted to him.

I thank Head of the Department, Dr. Sandeep Sharma, under whose inspiration I have learned a lot.

Then, I would like to thank my parents and my friends Akhil Aggarwal, Anu Sharma, Abhinoor Singh who have helped me with their valuable suggestions and guidance and have been very helpful in various stages of file completion.

(Anhat Singh)

Anhat Singh

Contents

Chapter 1

Introduction

1.1 What is SQL?

Structured Query Language or SQL is a common programming language for relational databases. Many IT professionals like data analysts, database administrators, SQL developers and data scientists often require using this coding language. Having knowledge and expertise in SQL is important for these roles. SQL, can be defined as a database coding language that extracts and manages data stored in a relational database. A relational database means you store and retrieve data in the form of relations or tables. For example, a table contains information about employees, such as employee id, name, contact number and department. This employee table is a relational database with only one relation called employee. You use SQL to communicate with a database. Using different commands, SQL tells the database what to do.

You can use the SQL commands to update, search, retrieve, add and delete data. Some of the standard and widely used commands are: Select, Update, Create, Delete, Group By and Insert. Though SQL is a standard of the American National Standards Institute (ANSI) and the International Organisation for Standardisation (ISO), many organisations use proprietary extensions based on the database used by the organisation. Furthermore, like other programming languages, SQL has a markup. This makes it essential for IT professionals to learn SQL markup.

1.2 Constraints in SQL

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Prevents actions that would destroy links between tables
- **CHECK** - Ensures that the values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column if no value is specified
- **CREATE INDEX** - Used to create and retrieve data from the database very quick.

1.3 SQL Commands

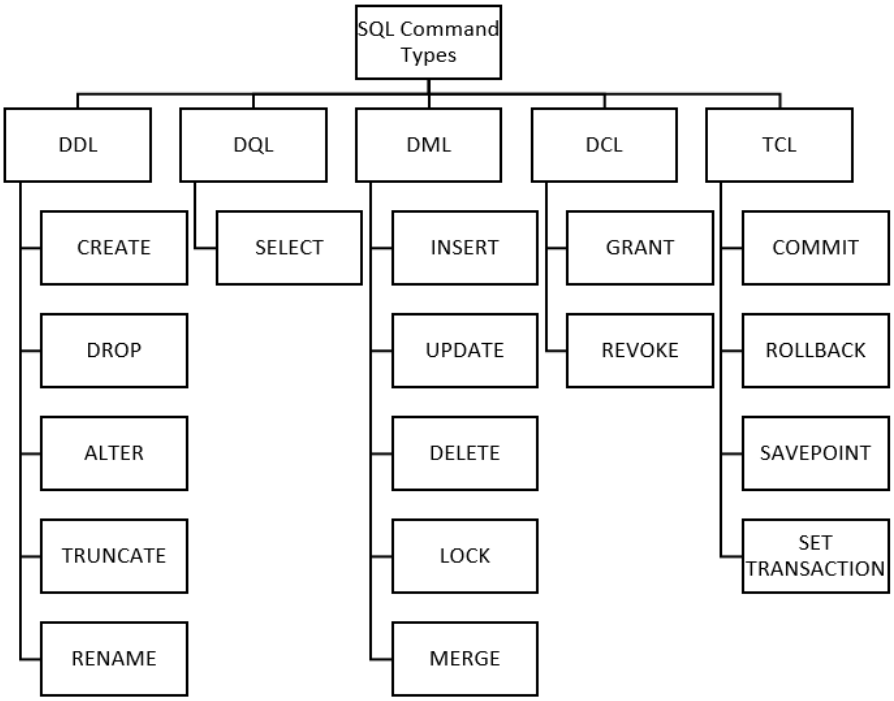

Figure 1.3.1: *Various SQL Commands*

Chapter 2

DDL in MySQL

2.1 Introduction

DDL is the short name for Data Definition Language, which deals with database schemas and descriptions of how the data should reside in the database. MySQL DDL is a subgroup of SQL commands among four: DDL, DML, DCL, and TCL. Structured Query Language(SQL) is the primary language of a database which performs different operations and queries in the available MySQL database, including creating a database or table to drop the same and others like updating, inserting, etc. The MySQL DDL gets involved with the schemas and explanations of the database to display how the database data should exist on the server. The DDL commands are significant for expressing and altering the structure of database tables, schemas or objects.

Various DDL Commands:

- **CREATE** - to create a database and its objects like (table)
- **ALTER** - alters the structure of the existing database
- **DROP** - delete objects from the database
- **TRUNCATE** - remove all records from a table, including all spaces allocated for the records that are removed
- **COMMENT** - add comments to the data dictionary
- **RENAME** - rename an object

2.2 CREATE Command

The CREATE statement is written using the following syntax:

```
CREATE DATABASE DatabaseName;
CREATE TABLE TableName
(Column1 Datatype1,
Column2 Datatype2,
…
ColumnNDatatypeN);
```

For Example, we will create a table named 'Emp' with some fields and similar data types that are valid in MySQL and, respectively a database named 'EmpDB' in MySQL server using the queries below:

```
CREATE DATABASE EmpDB;
CREATE TABLE Emp
(Emp_ID INT PRIMARY KEY,
 Emp_Name VARCHAR(255),
 Emp_City VARCHAR(255),
 Emp_AdmDate DATE NOT NULL);
```


2.3 ALTER Commande

ALTER DDL command is applied to modify the structure of the present database and related tables. With Alter query, we can add, alter or delete the present constraints on a table or columns on the table. The syntax is mentioned below:

```
ALTER TABLE TableName ADD ColumnNameData_Type;
ALTER TABLE TableName DROPColumnName;
ALTER TABLE TableName MODIFY COLUMNColumnNameData_Type;
```

Example:

```
ALTER TABLE Emp ADD Emp_Contact INT NOT NULL;
ALTER TABLE Emp DROP Emp_Contact;
ALTER TABLE Emp MODIFY COLUMN Emp_AdmDate Year;
```

2.4 DROP Command

This MySQL command is used to remove the database objects. In simple words, to delete the table existing in your database using the drop query syntax:

```
DROP TABLE TableName;
```

Example:

```
DROP TABLE Emp;
```

2.5 TRUNCATE Command

The truncate DDL command is implemented to delete all the data rows from the database table, which includes removing all spaces assigned for those table records.

The syntax of the TRUNCATE command is identical to the DROP statement as follows:

```
TRUNCATE TABLE TableName;
```

Example:

```
TRUNCATE TABLE Emp;
```


2.6 RENAME Command

The Rename DDL command query allows renaming any database objects in the server if needed for any admin works. Sometimes we want to modify the present table name and rename it. For this, let us apply the succeeding syntax with ALTER DDL command:

```
ALTER TABLE TableName_A RENAME TO TableName_B;
```

Example:

```
ALTER TABLE Emp RENAME TO Emp_Data;
```

Chapter 3

DML in MySQL

3.1 Introduction

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

Various DML Commands:

- **SELECT** - It is used to retrieve data from the database
- **INSERT** - It is used to insert data into a table
- **UPDATE** - It is used to update existing data within a table
- **DELETE** - It is used to delete records from a database table

3.2 SELECT Command

SELECT is used to display the contents of the tables. It is also used to select data from the database. Below is the syntax to select specified columns and records from the table.

Syntax:

```
Select column1,column2…,..column n from table;
```

Here, column 1, column 2....column n are the attributes of the table. Example to demonstrate the above syntax:

```
Select student_name from students;
Select * from students;
```

Select statement with a where clause:

```
Select column 1,column …2,.column n from table_name where [condition
];
```

Where the condition is the specified condition on which basis data is to be fetched. In the condition we can specify logical operators like >,<,=,LIKE,NOT and etc. Example:

```
select roll_no, student_name, course from students where roll_no=3;
```


3.3 INSERT Command

It is used to insert or add new rows or records in the existing table.

Syntax:

```
Insert into <table_name> values(<value1>,<value2>,<value3……>.,<
valuen>);
```

Where, *table_name* is the name of the table in which the data needs to be inserted and *values* are the values for each column of the table.

To insert values in the table we first need to create a table which is a DDL(Data definition language) statement. Example:

```
create table students (roll_no int,student_name varchar(150),course
varchar(150));
```

Once the table is created we can now insert values into it.

```
insert into students values(1,'ashish','java');
Insert into students values'(2,'rahul',C'++);
select * from students;
```

We have inserted 2 rows in the table. To view the entire table with records we use select statement with an asterisk (*)

3.4 UPDATE Command

It is used to change the existing values of the column i.e, changing the name of the student or changing the course of any student.

Syntax:

```
Update <table_name> set <column_name>=value where <condition>;
```

Wbere, *table_name* is the name of the table in which the value is to be changed and *condition* is the condition to get the specified row.

Below is the query of the update statement:

```
update students set roll_no=roll_no+10 where student_name='ashish';
update students set student_name='aman' where roll_no=2;
```


3.5 DELETE Command

Delete statement is used to delete rows of the table based on the specified conditions.

Syntax:

```
Delete from <table_name> where <condition>;
```

Where, *table_name* is the name of the table from which the data needs to be deleted and *condition* is the condition based on which the data is to be deleted.

Example:

```
DELETE FROM students WHERE roll_no=11;
DELETE FROM students WHERE student_name='divya';
```

The above example tells that, when delete command is performed on table students and wants to delete students_name='divya', then it deletes the entire details of 'divya' and gives the output of remaining students in the table. Here, one by one we have deleted all the rows of the table. In the end, the table is only left with the column's name and its schema. To delete the schema we have to use a DROP statement which is a DDL statement.

Chapter 4

DCL in MySQL

4.1 Introduction

DCL is short name of Data Control Language which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

Various DCL Commands:

- **GRANT** - allow users access privileges to the database
- **REVOKE** - withdraw users access privileges given by using the GRANT command

4.2 GRANT Command

This command is used to provide access or privileges on the database and its objects to the users.

Syntax:

```
GRANT PrivilegeName ON ObjectName TO {UserName |PUBLIC |RoleName} [
WITH GRANT OPTION];
```

Where,

- **PrivilegeName** – Is the privilege/right/access granted to the user.

- **ObjectName** – Name of a database object like TABLE/VIEW/STORED PROC.
- **UserName** – Name of the user who is given the access/rights/privileges.
- **PUBLIC** – To grant access rights to all users.
- **RoleName** – The name of a set of privileges grouped together.
- **WITH GRANT OPTION** – To give the user access to grant other users with rights.

Example to grant SELECT permission to Employee_Info table to user1:

```
GRANT SELECT ON Employee_Info TO user1;
```

4.3 REVOKE Command

This command is used to withdraw the user's access privileges given by using the REVOKE command.

Syntax:

```
REVOKE PrivilegeName  ON ObjectName  FROM {UserName |PUBLIC |
RoleName};
```

Example to revoke the granted permission from user1:

```
REVOKE SELECT ON Employee_Info FROM user1;
```

Chapter 5

SQL Basic Queries

5.1 CREATE TABLE

```
1  CREATE TABLE Student
2     (studentID int,
3      FName varchar(25),
4      LName varchar(25),
5      Address varchar(50),
6      City varchar(15),
7      Marks int);
```

We have created a table with the name Student and added a few parameters into the table

	studentId	Fname	Lname	Address	city	Marks
▶	1	JHON	DOE	#21, MG ROAD	Bengaluru	550
	2	Luke	Warm	#2, Brigade ROAD	Bengaluru	439
	3	William	Ruffelo	#728, IG Road	Chennai	572
	4	Mark	Spencer	#33, Silk Board	Delhi	479
	5	Andrew	Kallis	#207, Marathalli	Bengaluru	239
	6	Shane	Ponting	#511, WhiteField	Kochi	361
*	NULL	NULL	NULL	NULL	NULL	NULL

Figure 5.1.1: *Table Instance*

5.2 SELECT

It is the most basic SQL query one can use for manipulating a database. The select command is used to select the data from the database and display it to the user.

5.2.1 DISTINCT Keyword

If we want to display certain field without any duplicates then we use the DISTINCT keyword along with the select command. Example:

```
Select DISTINCT FName From Student;
```


5.2.2 WHERE Clause

If we need only certain records from the table then we use the where clause. Where clause acts as a Filtering mechanism. Under the Where section we need to specify certain conditions, only if those conditions are met the records will be extracted. Syntax:

```
SELECT column1, column2, ...column N FROM table_name WHERE condition
;
```

Example:

```
SELECT FName FROM Students WHERE City='Delhi';
```

5.2.3 AND, OR, NOT Operators

If we need to add two or more conditions in the where clause then we can use the above-mentioned operators. These keywords will add more complexity to the query.

AND Operator

This operator displays a record if all the conditions separated by AND are TRUE. Syntax:

```
SELECT column1, column2, ...FROM table_name WHERE condition1 AND
condition2 AND condition3 ...;
```

Example:

```
SELECT * FROM Student WHERE FName='John' AND Lname='Doe';
```

OR Operator

This operator displays a record if any of the conditions separated by OR is TRUE.

Syntax:

```
SELECT column1, column2, ... FROM table_name WHERE condition1 OR
condition2 OR condition3 ...;
```

Example:

```
SELECT * FROM Student WHERE FName='John' OR Lname='Doe';
```

NOT Operator

This operator displays a record if the condition/conditions are NOT TRUE.

Syntax:

```
SELECT column1, column2, ...FROM table_name WHERE NOT condition;
```

Example:

```
SELECT * FROM Student WHERE NOT Lname='Doe';
```

Chapter 6

AGGREGATE Functions

6.1 Introduction

An aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria and a single value is returned. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

Some of the Aggregate functions are COUNT, SUM, AVG, MIN, MAX.

6.2 COUNT()

This function returns the number of rows that match specified criteria. Syntax:

```
SELECT COUNT(column_name) FROM table_name;
```

Example:

```
SELECT COUNT (studentID) FROM Student;
```

6.3 AVG()

This function returns the average value of a numeric column. Syntax:

```
SELECT AVG(column_name) FROM table_name;
```

Example:

```
SELECT AVG(Marks) FROM Student;
```

6.4 SUM()

This function returns the total sum of a numeric column. Syntax:

```
SELECT SUM(column_name) FROM table_name;
```

Example:

```
SELECT SUM(Marks)FROM Student;
```

6.5 MIN()

This function returns the smallest value of the selected column. Syntax:

```
SELECT MIN(column_name)FROM table_name;
```

Example:

```
SELECT MIN(Marks) AS LeastMarks FROM Student;
```

6.6 MAX()

This function returns the largest value of the selected column. Syntax:

```
SELECT MAX(column_name) FROM table_name;
```

Example:

```
SELECT MAX(Marks) AS HighestMarks FROM Student;
```

Chapter 7

Group By, Having, Order By

7.1 GROUP BY

This functionality is used to arrange a similar type of data into a group. For instance, if the column in a table consists of similar data or value in different rows then we can use GROUP BY function to group the data. Syntax:

```
SELECT column_name(s) FROM table_name WHERE condition GROUP BY
column_name(s);
```

Example:

```
SELECT COUNT(StudentID), Fname FROM Student GROUP BY Fname;
```

7.2 HAVING

This clause is used to place conditions where we need to decide which group will be the part of final result-set. Also, we can not use the aggregate functions like SUM(), COUNT() etc. with WHERE clause. At such situation, we have to use HAVING condition. Syntax:

```
SELECT column_name(s) FROM table_name WHERE condition GROUP BY
column_name(s) HAVING condition;
```

Example:

```
SELECT Fname, SUM(Marks) FROM Student GROUP BY Fname HAVING SUM(
Marks)>500;
```

7.3 ORDER BY

This keyword is used to sort the result-set in ascending or descending order. The ORDER BY keyword shall sort the records in ascending order by default. If we want to sort the records in descending order, use the DESC keyword. Syntax:

```
SELECT column1, column2, ... FROM table_name ORDER BY column1,
column2, ... ASC|DESC;
```

Example:

```
SELECT COUNT(StudentID), City FROM Student GROUP BY City ORDER BY
COUNT(StudentID) DESC;
```

Chapter 8

NULL Values and Some more Operators

8.1 NULL Values

In SQL we use the NULL term to represent a missing value. A NULL value in a table is a value appears to be blank. A field with a NULL value is a field with no value in SQL. Keep note that a NULL value is different than a zero value or a field that contains spaces. We have special keywords i.e, IS NULL and IS NOT NULL.

8.1.1 IS NULL

Syntax:

```
SELECT column_names FROM table_name WHERE column_name IS NULL;
```

Example:

```
Select Fname, Lname From Student Where Marks IS NULL;
```

8.1.2 IS NOT NULL

Syntax:

```
SELECT column_names FROM table_name WHERE column_name IS NOT NULL;
```

Example:

```
Select Fname, Lname From Student Where Marks IS NOT NULL;
```

8.2 IN Operator

IN operator is used to specify multiple values inside the WHERE clause. It acts as a short for multiple OR. Syntax:

```
SELECT column_name(s) FROM table_name WHERE column_name IN (value1,
value2, ...);
```

Example:

```
SELECT StudentID, Fname, Lname FROM Student WHERE City IN ('Delhi',
'Goa', 'Pune','Bengaluru');
```


8.3 BETWEEN Operator

BETWEEN operator will select a particular value within the specified range. It is compulsory to add the beginning and the end value (Range). Syntax:

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

Example:

```
SELECT StudentID, Fname, Lname FROM Student WHERE Marks BETWEEN 400
AND 500;
```

8.4 UNION Operator

UNION Syntax:

```
SELECT Column1, Column2, Column3, ..., ColumnN FROM Table1
UNION
SELECT Column1, Column2, Column3, ..., ColumnN FROM Table2;
```

UNION ALL Syntax:

```
SELECT Column1, Column2, Column3, ..., ColumnN FROM Table1
UNION ALL
SELECT Column1, Column2, Column3, ..., ColumnN FROM Table2;
```

8.5 INTERSECT Operator

This clause used to combine two SELECT statements and return the intersection of the data-sets of both the SELECT statements. Syntax:

```
SELECT Column1 , Column2 .... FROM TableName WHERE Condition
INTERSECT
SELECT Column1 , Column2 .... FROM TableName WHERE Condition;
```

Anhat Singh