# Unit-2
# The Fundamentals of SQL

The Structured Query Language is one of the fundamental building blocks of modern database architecture. SQL defines the methods used to create and manipulate relational databases on all major platforms. At first glance, the language may seem intimidating and complex, but it's not all that difficult.

About SQL

The correct pronunciation of SQL is a contentious issue within the database community. In its SQL standard, the American National Standards Institute declared that the official pronunciation is "es queue el." However, many database professionals have taken to the slang pronunciation "sequel." The choice is yours.

SQL comes in many flavors. Oracle databases use its proprietary PL/SQL. Microsoft SQL Server makes use of Transact-SQL. All of the variations are based upon the industry standard ANSI SQL. This introduction uses ANSI-compliant SQL commands that work on any modern relational database system.

DDL and DML

SQL commands can be divided into two main sub-languages. The Data Definition Language (DDL) contains the commands used to create and destroy databases and database objects. After the database structure is defined with DDL, database administrators and users can use the Data Manipulation Language (DML) to insert, retrieve and modify the data contained within it.

Data Definition Language Commands

The Data Definition Language is used to create and destroy databases and database objects. These commands are primarily used by database administrators during the setup and removal phases of a database project. Here's a look at the structure and usage of four basic DDL commands:

**CREATE.** Installing a database management system on a computer allows you to create and manage many independent databases. For example, you may want to maintain a database of customer contacts for your sales department and a personnel database for your HR department. The CREATE command is used to establish each of these databases on your platform. For example, the command:

```
CREATE DATABASE employees
```

creates an empty database named "employees" on your DBMS. After creating the database, the next step is to create tables that contain data. Another variant of the CREATE command can be used for this purpose. The command:

```
CREATE TABLE personal_info (first_name char(20) not null, last_name char(20) not null, employee_id int not null)
```

establishes a table titled "personal_info" in the current database. In the example, the table contains three attributes: first_name, last_name, and employee_id along with some additional information.

**USE.** The USE command allows you to specify the database you want to work with within your DBMS. For example, if you're currently working in the sales database and want to issue some commands that will affect the employee database, preface them with the following SQL command:

```
USE employees
```

It's important to always be conscious of the database you are working in before issuing SQL commands that manipulate data.

**ALTER.** Once you've created a table within a database, you may want to modify its definition. The ALTER command allows you to make changes to the structure of a table without deleting and recreating it. Take a look at the following command:

```
ALTER TABLE personal_info ADD salary money null
```

This example adds a new attribute to the personal_info table—an employee's salary. The "money" argument specifies that an employee's salary is stored using a dollars and cents format. Finally, the "null" keyword tells the database that it's OK for this field to contain no value for any given employee.

**DROP.** The final command of the Data Definition Language, DROP, allows us to remove entire database objects from our DBMS. For example, if we want to permanently remove the personal_info table that we created, we'd use the following command:

```
DROP TABLE personal_info
```

Similarly, the command below would be used to remove the entire employee database:

```
DROP DATABASE employees
```

Use this command with care. The DROP command removes entire data structures from your database. If you want to remove individual records, use the DELETE command of the Data Manipulation Language.

Data Manipulation Language Commands

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands are used by all database users during the routine operation of the database.

**INSERT.** The INSERT command in SQL is used to add records to an existing table. Returning to the personal_info example from the previous section, imagine that our HR department needs to add a new employee to its database. You could use a command similar to this one:

```
INSERT INTO personal_info
values('bart','simpson',12345,$45000)
```

Note that there are four values specified for the record. These correspond to the table attributes in the order they were defined: first_name, last_name, employee_id and salary.

**SELECT.** The SELECT command is the most commonly used command in SQL. It allows database users to retrieve the specific information they desire from an operational database. Take a look at a few examples, again using the personal_info table from the employee database.

The command shown below retrieves all the information contained within the personal_info table. Note that the asterisk is used as a wildcard in SQL. This literally means "Select everything from the personal_info table."

```
SELECT *
FROM personal_info
```

Alternatively, users may want to limit the attributes that are retrieved from the database. For example, the Human Resources department may require a list of the last names of all employees in the company. The following SQL command would retrieve only that information:

```
SELECT last_name
FROM personal_info
```

The WHERE clause can be used to limit the records that are retrieved to those that meet specified criteria. The CEO might be interested in reviewing the personnel records of all highly paid employees. The following command retrieves all of the data contained within personal_info for records that have a salary value greater than $50,000:

```
SELECT *
FROM personal_info
WHERE salary > $50000
```

**UPDATE.** The UPDATE command can be used to modify the information contained within a table, either in bulk or individually. Assume the company gives all employees a 3 percent cost-of-living increase in their salary annually. The following SQL command could be used to quickly apply this to all the employees stored in the database:

```
UPDATE personal_info
SET salary = salary * 1.03
```

When the new employee Bart Simpson demonstrates performance above and beyond the call of duty, management wishes to recognize his stellar accomplishments with a $5,000 raise. The WHERE clause could be used to single out Bart for this raise:

```
UPDATE personal_info
SET salary = salary + $5000
WHERE employee_id = 12345
```

**DELETE.** Finally, let's take a look at the DELETE command. You'll find that the syntax of this command is similar to that of the other DML commands. Unfortunately, our latest corporate earnings report didn't quite meet expectations and poor Bart has been laid off. The DELETE command with a WHERE clause can be used to remove his record from the personal_info table:

```
DELETE FROM personal_info
WHERE employee_id = 12345
```

JOINs

Now that you've learned the basics of SQL, it's time to move on to one of the most powerful concepts the language has to offer — the JOIN statement. A JOIN statement allows you to combine data in multiple tables to efficiently process large quantities of data. These statements are where the true power of a database resides.

To explore the use of a basic JOIN operation to combine data from two tables, continue with the example using the PERSONAL_INFO table and add an additional table to the mix. Assume you have a table called DISCIPLINARY_ACTION that was created with the following statement:

```
CREATE TABLE disciplinary_action (action_id int not null, employee_id int not null, comments char(500))
```

This table contains the results of disciplinary actions on company employees. You'll notice that it doesn't contain any information about the employee other than the employee number. It's easy to imagine many scenarios where you might want to combine information from the DISCIPLINARY_ACTION and PERSONAL_INFO tables.

Assume you've been tasked with creating a report that lists the disciplinary actions taken against all employees with a salary greater than $40,000. The use of a JOIN operation, in this case, is straightforward. We can retrieve this information using the following command:

```
SELECT personal_info.first_name, personal_info.last_name, disciplinary_action.comments
FROM personal_info, disciplinary_action
WHERE personal_info.employee_id = disciplinary_action.employee_id
AND personal_info.salary > 40000
```

The code specifies the two tables that we want to join in the FROM clause and then includes a statement in the WHERE clause to limit the results to records that had matching employee IDs and met our criteria of a salary greater than $40,000.

SQL commands are divided into four subgroups, DDL, DML, DCL, and TCL.

Figure - SQL Commands:

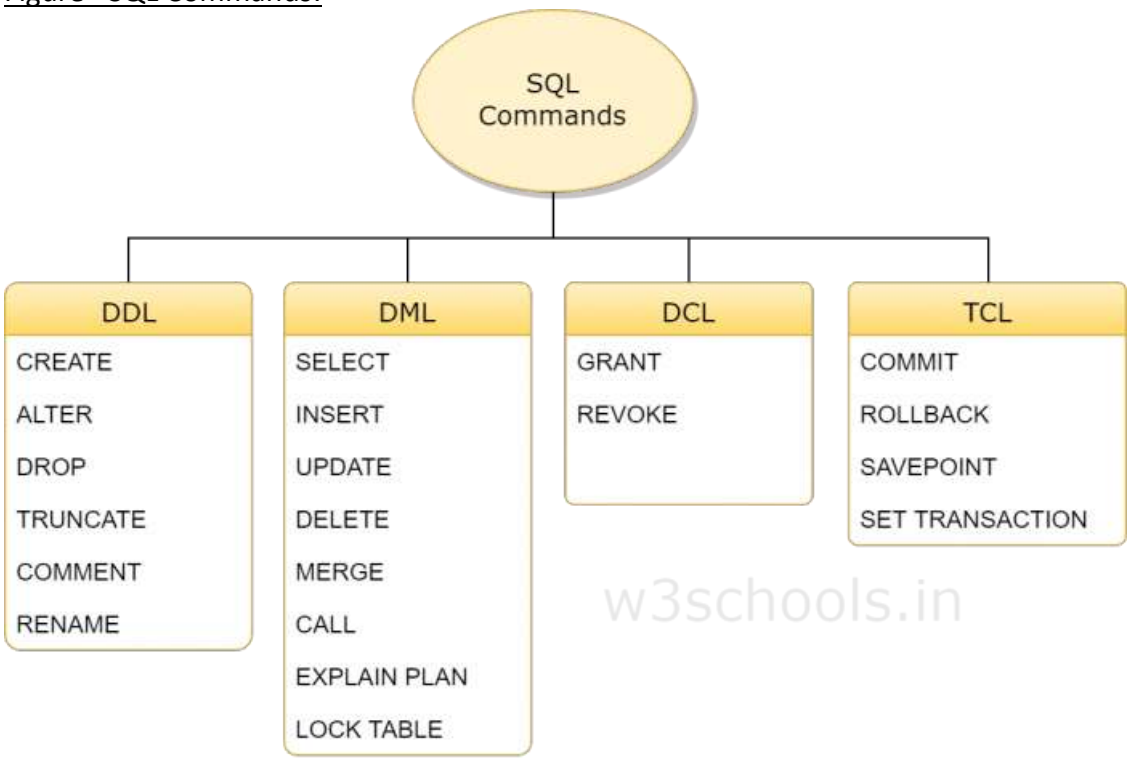


Table of Contents



**DDL**

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- 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 are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

**DML**

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.

- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table
- MERGE - UPSERT operation (insert or update)
- CALL - call a PL/SQL or Java subprogram
- EXPLAIN PLAN - interpretation of the data access path
- LOCK TABLE - concurrency Control

**DCL**

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.

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

**TCL**

TCL is short name of Transaction Control Language which deals with a transaction within a database.

- COMMIT - commits a Transaction
- ROLLBACK - rollback a transaction in case of any error occurs
- SAVEPOINT - to rollback the transaction making points within groups
- SET TRANSACTION - specify characteristics of the transaction