# SQL OVERVIEW

 

SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS.

SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database.

## Data Definition Language

SQL uses the following set of commands to define database schema −

## CREATE

Creates new databases, tables and views from RDBMS.

**For example** −

```
Create database tutorialspoint;
Create table article;
Create view for_students;
```

## DROP

Drops commands, views, tables, and databases from RDBMS.

**For example**−

```
Drop object_type object_name;
Drop database tutorialspoint;
Drop table article;
Drop view for_students;
```

## ALTER

Modifies database schema.

```
Alter object_type object_name parameters;
```

**For example**−

```
Alter table article add subject varchar;
```

This command adds an attribute in the relation **article** with the name **subject** of string type.

## Data Manipulation Language

SQL is equipped with data manipulation language *DML*. DML modifies the database instance by inserting, updating and deleting its data. DML is responsible for all froms data modification in a database. SQL contains the following set of commands in its DML section −

- SELECT/FROM/WHERE
- INSERT INTO/VALUES
- UPDATE/SET/WHERE
- DELETE FROM/WHERE

These basic constructs allow database programmers and users to enter data and information into

the database and retrieve efficiently using a number of filter options.

## SELECT/FROM/WHERE

- **SELECT** − This is one of the fundamental query command of SQL. It is similar to the projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.
- **FROM** − This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.
- **WHERE** − This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.

**For example** −

```
Select author_name
From book_author
Where age > 50;
```

This command will yield the names of authors from the relation **book_author** whose age is greater than 50.

## INSERT INTO/VALUES

This command is used for inserting values into the rows of a table *relation*.

**Syntax**−

```
INSERT INTO table (column1 [, column2, column3 ... ]) VALUES (value1 [, value2, value3
... ])
```

Or

```
INSERT INTO table VALUES (value1, [value2, ... ])
```

**For example** −

```
INSERT INTO tutorialspoint (Author, Subject) VALUES ("anonymous", "computers");
```

## UPDATE/SET/WHERE

This command is used for updating or modifying the values of columns in a table *relation*.

**Syntax** −

```
UPDATE table_name SET column_name = value [, column_name = value ...] [WHERE condition]
```

**For example** −

```
UPDATE tutorialspoint SET Author="webmaster" WHERE Author="anonymous";
```

## DELETE/FROM/WHERE

This command is used for removing one or more rows from a table *relation*.

**Syntax** −

```
DELETE FROM table_name [WHERE condition];
```

**For example** −

```
DELETE FROM tutorialspoints
   WHERE Author="unknown";
```