Unit 5 – Structured Query Language

Prof. Sumit P. Makwana

Table of Contents

What is SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What SQL can do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Categories of SQL Command

Data Definition Language (DDL)

- Commands which are related to defining structure of a database/tables/views etc. are known as Data Definition Language (DDL) Commands
- e.g., Create, Drop, Alter, Truncate

Data Manipulation Language (DML)

- Commands which are related to manipulation of data of tables are known as Data Manipulation Commands.
- e.g., Insert, Update, Delete

Data Control Language (DCL)

- Commands which are related to control the permission of database/tables/view etc are known as Data Control Commands
- e.g., Grant, Revoke

Transaction Control Language (TCL)

- Commands which are related to control the series of queries (Transaction) are known as Transaction Control Language
- Commit, Rollback, Save Point

Data Query Language (DQL)

- Commands which are used to querying (reading/fetching) data from database/tables/view etc are known as Data Query Language
- e.g., Select

Data Types in Oracle

- CHARACTER/STRING Datatype
  - CHAR
  - VARCHAR2 and VARCHAR and NVARCHAR2
  - LONG
- NUMERIC Datatypes
  - NUMBER
- DATE Datatypes
  - DATE
- Binary Datatypes
  - BLOB
  - BFILE
  - RAW
  - LONG RAW

Create Tables

Tables are the fundamental structure in database to store the data. Table compromises of rows and columns. Each column a data value which is to be stored, and each row represents one record of that table.

Tables can be created using Create command

Syntax

```
CREATE TABLE table_name
(
    column1_name DATATYPE(SIZE),
    column2_name DATATYPE(SIZE),
    ...
)
```

Example

```
CREATE TABLE product
(
    name VARCHAR2(50),
    price NUMBER(10,2),
    description VARCHAR2(150)
 )
```

Modify Table Structure

After the table is created, if we wanted to make changes with table structure like adding new column, removing existing columns, changing data types of columns etc. we can use Alter command for modification of table structure.

Add Column to Table

Below syntax can be used to add new column to an existing table.

Syntax

```
ALTER TABLE table_name

ADD column_name DATATYPE(SIZE)
```

Example

```
ALTER TABLE student

ADD mobile_no VARCHAR2(10)
```

Above example will add new column with name mobile_no with datatype varchar2(10) to table student.

Remove Column from Table

Below syntax can be used to remove existing column from a table. When we delete any column and if the column has some data, then data of that column will be permanently lost and can not be undone.

Syntax

```
ALTER TABLE table_name

DROP COLUMN column_name
```

Example

```
ALTER TABLE student

DROP COLUMN mobile_no
```

Above example will remove column mobile_no from the table student.

Modify Column Datatype

If we want to modify the datatype or size of the column, below syntax can be used.

Syntax

```
-- Syntax for older version of Database

ALTER TABLE table_name

MODIFY COLUMN column_name DATATYPE(SIZE)
```

```
-- Syntax for newer version of Database

ALTER TABLE table_name

MODIFY column_name DATATYPE(SIZE)
```

Example

```
ALTER TABLE student

MODIFY mobile_no VARCHAR2(12)
```

Above example will modify the datatype of column mobile_no to varchar2 with the size to 12 in student table.

Rename Column

If we want to rename an existing table column, below syntax can be used.

Syntax

```
ALTER TABLE table_name

RENAME COLUMN old_column_name TO new_column_name
```

Example

```
ALTER TABLE student

RENAME COLUMN mobile_no to phone_no
```

Above example will rename column mobile_no to phone_no in the student table.

Rename Table

If we want to rename an table, below syntax can be used.

Syntax

```
ALTER TABLE table_name

RENAME to new_table_name
```

Example

```
ALTER TABLE student

RENAME to student_detail
```

Above example will rename table student to student_detail.

Insert Data

We can insert the data into table using insert query.

Syntax

```
INSERT INTO table_name (column_name, column_name,
column_name, ...) VALUES (value, value, value)
```

Note: Text, character, and string literals are always surrounded by single quotation marks.

Example

```
INSERT INTO product (name, price, description)
VALUES ('Laptop', '59999', "Laptop with i5
Processor and 4GB RAM')
```

If we do not want to enter data into all columns, we can do it by skipping the column name in the query.

E.g., in below query we have not written description column and its value, as we do not want to enter the value of description.

```
INSERT INTO product VALUES (name, price) values
('Laptop', '59999')
```

Note if we want to insert data into all columns of the table, we can skip the column names.

Syntax

```
INSERT INTO table_name VALUES (value1, value2,
value3, ...)
```

Above syntax only work when we want to enter data in all columns of the table. Also, the sequence of the value should be same as column sequence.

Example

```
INSERT INTO product VALUES ('Laptop', '59999',
"Laptop with i5 Processor and 4GB RAM')
```

Select (Fetch) Data

We can select (fetch) the data from table using select query.

Select all attributes (columns) and records (rows)

Syntax

```
SELECT * FROM table_name
```

Example

```
SELECT * FROM product
```

Above example will fetch all the records and all the attributes (columns) from the table product.

Select specific attributes (columns)

Instead of writing * sign in select query, we can specify the list of attributes separated by comma to fetch only specific attributes from the table.

Syntax

```
SELECT column_name, column_name, ... FROM
table_name
```

Example

```
SELECT name, price FROM product
```

Above example will fetch only name, and price attributes from the table product.

Select specific records (rows)

Syntax

```
SELECT * FROM table_name WHERE condition
```

Instead of writing * sign we can specify the list of attributes separated by comma to fetch only specific attributes from the table. We can use any conditional operator (=, <, <=, >, >=, <>) to make any condition and also use any logical operator (and, or, not) to combine multiple conditions.

Example 1

```
SELECT * FROM product WHERE price > 5000
```

Above example will fetch those records whose price is greater than 5000.

Example 2

```
SELECT * FROM product WHERE price < 50000 AND
name='Laptop'
```

Above example will fetch those records whose price is greater than 5000 and product name is Laptop.

Select specific attributes (columns) and specific records (rows)

We can combine above two syntax to select specific attributes and specific records.

Syntax

```
SELECT column_name, column_name, ... FROM
table_name WHERE condition
```

Instead of writing * sign in select query we can specify the list of attributes separated by comma to fetch only specific attributes from the table.

Example

```
SELECT name FROM product WHERE price > 10000
```

Above example will records whose price is greater than 10000 and will only display name attribute.

Update Data

To update (modify) the existing data of table we can use update query.

Syntax

```
-- To update only single attribute (column)

UPDATE table_name SET column_name = 'new_value'
WHERE condition


-- To update multiple attribute (columns)

UPDATE tablename SET column_name = 'new_value',
column_name = 'new_value' WHERE condition
```

Example

```
update product set price = '5000'
```

In above example we have updated the price attribute of the product table. Above query will set the price value to 5000 in all records. Because by default update query affect all the records if no condition is specified.

But most of the times we don't want to update all the records but we only want to update the selective records. To update selective records, we can append the condition in update query so that, only those records which satisfy the query will be updated.

Example

```
UPDATE product SET price = '50000' WHERE name =
'Laptop'
```

Above example will set the price attribute to 50000 only in those records whose name attribute is 'Laptop'.

Update multiple attributes (columns)

We can also update the multiple attributes in a single query by specifying multiple attribute-value pair separated by comma.

Example

```
UPDATE product SET price = '50000', description =
'Laptop with 16GB RAM' WHERE name = 'Laptop'
```

Above example will set the new value to price and description attribute in those records whose name attribute is 'Laptop'

Delete Data

To delete the data (records) from the table we can use delete query.

Syntax

```
-- Delete all records

DELETE FROM table_name

-- Delete only specific records

DELETE FROM table_name WHERE condition
```

Example

```
DELETE FROM product
```

Above example will delete all the records from the table product.

If we want to delete only some records, we can specify the condition. When condition is specified only those records which satisfy the condition will be deleted.

Example

```
DELETE FROM product where name = 'Laptop'
```

Above example will delete the records from product table where the value of name attribute is 'Laptop'

Delete Table

To delete entire table with data, drop command is used.

Syntax

```
DROP table table_name
```

E**xample**

```
DROP table product
```

Above example will delete entre product table.

Difference between DELETE and DROP command is, delete command only deletes the records and table structure remains. But drop command delete entire table (all records and structure of table is deleted)