ORACLE SQL

Lesson 01: SQL Basic Commands

Lesson Objectives

To understand the following topics:

- SQL Language
- DDL
- DML
- DRL
- TCL
- DCL

What is SQL?

SQL:

- SQL stands for Structured Query Language.
- SQL is used to communicate with a database.
- Statements are used to perform tasks such as update data on a database, or retrieve data from a database.
- Benefits of SQL are:
  - It is a Non-Procedural Language.
  - It is a language for all users.
  - It is a unified language.

What can SQL do?

SQL

- allows you to access a database.
- can execute queries against a database.
- can retrieve data from a database.
- can insert new records into a database.
- can delete records from a database.
- can update records in a database.

Rules for SQL statements

Rules for SQL statements:

- SQL keywords are not case sensitive. However, normally all commands (SELECT, UPDATE, etc) are upper-cased.
- “Variable” and “parameter” names are displayed as lower-case.
- New-line characters are ignored in SQL.
- Many DBMS systems terminate SQL statements with a semi-colon character.
- “Character strings” and “date values” are enclosed in single quotation marks while using them in WHERE clause or otherwise.

Standard SQL statement groups

Given below are the standard SQL statement groups:

Groups	Statements	Description
DQL	SELECT	DATA QUERY LANGUAGE – It is used to get data from the database and impose ordering upon it.

DML	DELETE
	INSERT
	UPDATE
	MERGE
		DATA MANIPULATION LANGUAGE – It is used to change database data.

DDL	DROP TRUNCATE CREATE ALTER	DATA DEFINITION LANGUAGE – It is used to manipulate database structures and definitions.
TCL	COMMIT ROLLBACK SAVEPOINT	TCL statements are used to manage the transactions.
DCL (Rights)	REVOKE GRANT	They are used to remove and provide access rights to database objects.

2.2: DDL

Table

Tables are objects, which store the user data.
Use the CREATE TABLE statement to create a table, which is the basic structure to hold data.
For example:

```
CREATE TABLE book_master
(book_code number,
   book_name varchar2(50),
   book_pub_year number,
   book_pub_author varchar2(50));
```

2.2: DDL

Create new table based on existing table

Constraints on an "old table" will not be applicable for a "new table".

```
     CREATE TABLE student_dept117 AS
SELECT student_code, student_name
FROM student_master WHERE  dept_code = 117
```

2.2: DDL
ALTER Table

Given below is an example of ALTER TABLE:

```
ALTER TABLE table_name
      [ADD (col_name col_datatype col_constraint
,...)]|
      [ADD (table_constraint)]|
      [DROP CONSTRAINT constraint_name]|
      [MODIFY existing_col_name
new_col_datatype
                                   new_constraint
new_default]
      [DROP COLUMN existing_col_name]
      [SET UNUSED COLUMN existing_col)name];
```

2.2: DDL

ALTER Table – Add clause

The “Add” keyword is used to add a column or constraint to an existing table.

- For adding three more columns to the emp table, refer the following example:

```
        ALTER TABLE Student_Master
    ADD (last_name varchar2(25) );
```

2.2: DDL

ALTER Table – Add clause

- For adding Referential Integrity on “mgr_code" column, refer the following example:

```
ALTER TABLE staff_master
   ADD CONSTRAINT FK FOREIGN KEY  (mgr_code)
REFERENCES staff_master(staff_code);
```

ALTER Table – MODIFY clause

MODIFY clause:

- The “Modify” keyword allows making modification to the existing columns of a table.
  - For Modifying the width of “sal” column, refer the following example:

```
ALTER TABLE staff_master
MODIFY (staff_sal number (12,2)  ) ;
```

ALTER Table – Enable | Disable clause

ENABLE | DISABLE Clause:

- The ENABLE | DISABLE clause allows constraints to be enabled or disabled according to the user choice without removing them from a table.
- Refer the following example:

```
ALTER TABLE staff_master DISABLE CONSTRAINT
SYS_C000934;
```

ALTER Table – DROP clause

The DROP clause is used to remove constraints from a table.

- For Dropping the FOREIGN KEY constraint on “department”, refer the following example:

```
ALTER TABLE  student_master
DROP CONSTRAINT  stu_dept_fk ;
```

2.2: DDL

Dropping Column

Given below are the ways for “Dropping” a column:

- 1a.Marking the columns as unused and then later dropping them.
- 1b. The following command can be used later to permanently drop the columns.

```
ALTER TABLE staff_master SET UNUSED COLUMN
staff_address;
ALTER TABLE staff_master SET UNUSED (staff_sal, hiredate);
```

```
ALTER TABLE emp DROP UNUSED COLUMNS;
```

2.2: DDL

Dropping Column

- Directly dropping the columns.

```
ALTER TABLE staff_master DROP COLUMN staff_sal;
```

Drop a Table

The DROP TABLE command is used to remove the definition of a table from the database.

For Example:

```
DROP TABLE staff_master;

DROP TABLE Department_master
     CASCADE CONSTRAINTS;
```

Rename a Table

The RENAME command is used to give a new name to the table.
Views can also be renamed using this command

For Example:

```
RENAME  staff_master TO new_staffmaster;
```

Concept of Data Manipulation Language

Data Manipulation Language (DML) is used to perform the following routines on database information:

- Retrieve
- Insert
- Modify

DML changes data in an object. If you insert a row into a table, that is DML.

All DML statements change data, and must be committed before the change becomes permanent.

INSERT

INSERT command:

- INSERT is a DML command. It is used to add rows to a table.
- In the simplest form of the command, the values for different columns in the row to be inserted have to be specified.
- Alternatively, the rows can be generated from some other tables by using a SQL query language command.

Inserting Rows into a Table

Inserting by specifying values:

Example: To insert a new record in the DEPT table

```
INSERT INTO table_name[(col_name1,col_name2,...)]
        {VALUES (value1,value2,....) | query};
```

```
INSERT INTO Department_master
VALUES (10, ‘Computer Science’);
```

Inserting Rows into a Table

Inserting rows in a table from another table using Subquery:

Example: The example given below assumes that a new_emp_table exists. You can use a subquery to insert rows from another table.

```
INSERT INTO new_staff_table
SELECT * FROM staff_master
WHERE staff_master.hiredate > '01-jan-82';
```

Inserting Rows into a Table

Inserting by using “substitution variables”:

Example: In the example given below, when the command is run, values are prompted every time.

```
INSERT INTO department_master
VALUES (&dept_code, ‘&dept_name’);
Enter a value for dept_code : 20
Enter a value for dept_name : Electricals
```

2.3: Data Manipulation Language

DELETE

The DELETE command is used to delete one or more rows from a table.

- The DELETE command removes all rows identified by the WHERE clause.

```
DELETE [FROM] {table_name | alias }
              [WHERE condition];
```

Deleting Rows from Table

Example 1: If the WHERE clause is omitted, all rows will be deleted from the table.
Example 2: If we want to delete all information about department 10 from the Emp table:

```
DELETE FROM staff_master;
```

```
DELETE FROM student_master WHERE  dept_code=10;
```

UPDATE

Use the UPDATE command to change single rows, groups of rows, or all rows in a table.

- In all data modification statements, you can change the data in only “one table at a time”.

```
UPDATE table_name
SET    col_name = value|
          col_name = SELECT_statement_returning_single_value|
          (col_name,...) = SELECT_statement
[WHERE condition];
```

Updating Rows from Table

Example 1: To UPDATE the column “dname” of a row, where deptno is 10, give the following command:

```
UPDATE department_master
SET dept_name= ‘Information Technology’
WHERE dept_code=10;
```

Updating Rows from Table

Example 2: To UPDATE the subject marks details of a particular student, give the following command:

```
UPDATE student_marks
SET subject1= 80 , subject2= 70
                WHERE student_code=1005;
```

Using a Subquery to do an Update

For making salary of “Anil” equal to that of staff member 100006, use the following command:

```
UPDATE staff_master
SET staff_sal = (SELECT staff_sal FROM staff_master
                                   WHERE staff_code = 100006 )
WHERE staff_name = ‘Anil';
```

2.4: DRL

The SELECT Statement

The SELECT command is used to retrieve rows from a single table or multiple Tables or Views.

- A query may retrieve information from specified columns or from all of the columns in the Table.
- It helps to select the required data from the table.

```
SELECT [ALL | DISTINCT] { * | col_name,...}
FROM table_name alias,...
        [ WHERE expr1 ]
        [ CONNECT BY expr2 [ START WITH expr3 ] ]
        [ GROUP BY expr4 ] [ HAVING expr5 ]
        [ UNION | INTERSECT | MINUS SELECT ... ]
        [ ORDER BY expr | ASC | DESC ];
```

2.4: DRL

Selecting Columns

Displays all the columns from the student_master table

```
SELECT  *  FROM student_master;
```

Displays selected columns from the student_master table

```
SELECT student_code,
              student_name
            FROM student_master;
```

2.4: DRL
The WHERE clause

The WHERE clause is used to specify the criteria for selection.

- For example: displays the selected columns from the student_master table based on the condition being satisfied

```
SELECT  student_code,  student_name, student_dob
                   FROM student_master
 WHERE dept_code = 10;
```

Character Strings and Dates

Are enclosed in single quotation marks
Character values are case sensitive
Date values are format sensitive

```
SELECT  student_code, student_dob
          FROM  student_master
         WHERE student_name = ‘Sunil’ ;
```

Using AND or OR Clause

Use of OR operator:

```
SELECT  book_code   FROM  book_master
         WHERE  book_pub_author LIKE '%Kanetkar%'
        OR  book_name LIKE '%Pointers%';
```

Using NOT Clause

The NOT operator finds rows that do not satisfy a condition.

- For example: List staff members working in depts other than 10 & 20.

```
SELECT staff_code,staff_name  FROM  staff_master
     WHERE  dept_code NOT IN ( 10,20 );
```

2.4: DRL

Treatment of NULL Values

NULL is the absence of data.

Treatment of this scenario requires use of IS NULL operator.

```
SQL>    SELECT student_code
        2        FROM student_master
        3        WHERE dept_code IS NULL;
```

2.4: DRL

The DISTINCT clause

The SQL DISTINCT clause is used to eliminate duplicate rows.

- For example: Displays student codes from student_marks tables. the student codes are displayed without duplication

```
SELECT DISTINCT student_code
        FROM student_marks;
```

2.4: DRL
The ORDER BY clause

The ORDER BY clause presents data in a sorted order.

- It uses an "ascending order" by default.
- You can use the DESC keyword to change the default sort order.
- It can process a maximum of 255 columns.

In an ascending order, the values will be listed in the following sequence:

- Numeric values
- Character values
- NULL values

In a descending order, the sequence is reversed.

2.4: DRL

Sorting Data

The output of the SELECT statement can be sorted using ORDER BY clause

- ASC : Ascending order, default
- DESC : Descending order

Display student details from student_master table sorted on student_code in descending order.

```
SELECT Student_Code,Student_Name,Dept_Code,
        Student_dob
    FROM Student_Master
    ORDER BY Student_Code DESC ;
```

Quick Guidelines

It is necessary to always include a WHERE clause in your SELECT statement to narrow the number of rows returned.

- If you do not use a WHERE clause, then Oracle will perform a table scan of your table, and return all the rows.
- By returning data you do not need, you cause the SQL engine to perform I/O it does not need to perform, thus wasting SQL engine resources.

Quick Guidelines

- In addition, the above scenario increases network traffic, which can also lead to reduced performance.
- And if the table is very large, a table scan will lock the table during the time-consuming scan, preventing other users from accessing it, and will hurt concurrency.

In your queries, do not return column data that is not required.

- For example:
  - You should not use SELECT * to return all the columns from a table if all the data from each column is not required.
  - In addition, using SELECT * prevents the use of covered indexes, further potentially decreasing the query performance.

Quick Guidelines

Carefully evaluate whether the SELECT query requires the DISTINCT clause or not.

- The DISTINCT clause should only be used in SELECT statements.
  - This is mandatory if you know that "duplicate" returned rows are a possibility, and that having duplicate rows in the result set would cause problems with your application.
- The DISTINCT clause creates a lot of extra work for SQL Server.
  - The extra load reduces the "physical resources" that other SQL statements have at their disposal.
- Hence, use the DISTINCT clause only if it is necessary.

Quick Guidelines

If you use LIKE in your WHERE clause, try to use one or more leading character in the clause, if at all possible.

- **For example**: Use LIKE 'm%' not LIKE '%m'

Certain operators in the WHERE clause prevents the query optimizer from using an Index to perform a search.

- **For example**: "IS NULL", "<>", "!=", "!>", "!<", "NOT", "NOT EXISTS", "NOT IN", "NOT LIKE", and "LIKE '%500'"

Quick Guidelines

Do not use ORDER BY in your SELECT statements unless you really need to use it.

- Whenever SQL engine has to perform a sorting operation, additional resources have to be used to perform this task.

Defining Transaction

A “transaction” is a logical unit of work that contains one or more SQL statements.

- “Transaction” is an atomic unit.
- The effects of all the SQL statements in a transaction can be either:
  - all committed (applied to the database), or
  - all rolled back (undone from the database)
- A “transaction” begins with the first executable SQL statement.

2.5: TC L

Defining Transaction

- A “transaction” ends when any of the following occurs:
  - A user issues a COMMIT or ROLLBACK statement without a SAVEPOINT clause.
  - A user runs a DDL statement such as CREATE, DROP, RENAME, or ALTER.
    - If the current transaction contains any DML statements, Oracle first commits the transaction, and then runs and commits the DDL statement as a new, single statement transaction.
  - A user disconnects from Oracle. The current transaction is committed.
  - A user process terminates abnormally. The current transaction is rolled back.

Statement Execution and Transaction Control

A “SQL statement” that runs successfully is different from a committed transaction.

However, until the “transaction” that contains the “statement” is committed, the “transaction” can be rolled back. As a result, all the changes in the statement can be undone.

Hence we can say, “a statement, rather than a transaction, runs successfully”.

2.5: TC L
Commit Transactions

Committing a transaction means making “permanent” all the changes performed by the SQL statements within the transaction.

- This can be done either explicitly or implicitly.

2.5: TC L

Commit Transactions

COMMIT statement makes “permanent” all the changes that are performed in the current transaction.

Syntax:

```
COMMIT [WORK];
```

Commit Transactions

COMMIT types:

- Implicit: Database issues an implicit COMMIT before and after any data definition language (DDL) statement
- Explicit

Example of COMMIT command:

```
DELETE FROM student_master
WHERE student_name = ‘Amit';
COMMIT ;
```

Rollback Transactions

Rolling back a transaction means “undoing changes” to data that have been performed by SQL statements within an “uncommitted transaction”.

- Oracle uses “undo tablespaces” (or rollback segments) to store old values.
- Oracle also uses the “redo log” that contains a record of changes.

Rollback Transactions

- Oracle lets you roll back an entire “uncommitted transaction”.
  - Alternatively, you can roll back the trailing portion of an “uncommitted transaction” to a marker called a “savepoint”.

Save points in Transactions

In a transaction, you can declare intermediate markers called “savepoints” within the context of a transaction.

- By using “savepoints”, you can arbitrarily mark your work at any point within a long transaction.
- In this manner, you can keep an option that is available later to roll back the work performed, however:
  - before the current point in the transaction, and
  - after a declared savepoint within the transaction

Save points in Transactions

- For example: You can use savepoints throughout a long complex series of updates. So if you make an error, you do not need to resubmit every statement.

Examples of Rollback and Save points

Example 1:

```
INSERT INTO department_master
VALUES (70, 'PERSONNEL'') ;
SAVEPOINT A ;
INSERT INTO department_master
VALUES (80, 'MARKETING') ;
SAVEPOINT B ;
```

```
ROLLBACK TO A ;
```

2.5:Data Control Language

User Access Control

With Oracle server you can maintain database security, by:

- Controlling database access
- Giving access to specific objects in the database
- Confirming given and received privileges with the Oracle data dictionary
- Creating synonyms for database objects

Object Privileges

Object Privileges are required to manipulate the content of the database objects
Owner of the object has all the object privileges on that object.
Owner can give or take out privileges on a particular object
Object privileges can differ from object to object

Object Privileges

Object Privilege	Table	View	Sequence	Procedure
SELECT	☑	☑	☑	
UPDATE	☑	☑		
DELETE	☑	☑		
INSERT	☑	☑		
ALTER	☑		☑	
INDEX	☑			
REFERENCES	☑	☑		
EXECUTE				☑

Data Control Language - Syntax

Granting Object Privileges

```
GRANT object_privileges|ALL [(columnname)] ON object
TO {user|role|public} WITH GRANT OPTION
```

Revoking Object Privileges

```
REVOKE {privilege,[privilege…]|ALL} ON object FROM
{user,[user,….]|role|public} [CASCADE CONSTRAINTS
```

Data Control Language - Example

Grant Query and Update privileges

```
GRANT SELECT ON student_master TO user1,user2;
```

```
GRANT UPDATE (subject1,subject2,subject3
ON student_marks TO user1,user2;
```

Grant privileges and allow to pass it on

```
GRANT SELECT, INSERT ON student_master
TO user1
WITH GRANT OPTION;
```

Data Control Language -Example

Revoking UPDATE privileges from user2

```
REVOKE UPDATE on student_marks FROM user2;
```

SUMMARY

- In this lesson, you have learnt:
  - What is SQL?
    - Rules for SQL statements
    - Standard SQL statement groups
  - What is SELECT statement?
  - The concept of DDL,DML,DRL,TCL,DCL
  - Transactions

Review Question

❖Question 1: SQL categories are ___.
- Option 1: DDL
- Option 2: DML
- Option 3: DSL
- Option 4: DQL
- Option 5: TCL
- Option 6: TDL

❖Question 2: A transaction is committed when the user issues a DDL statement.
- True/False

❖Question 3: All DML statements are auto committed
- True/False