ORACLE STATEMENT

Oracle SQL statements are the core commands used to interact with an Oracle Database, allowing users to define, manipulate, and control data. These statements are categorized into several types, including Data Manipulation Language (DML) for managing data (INSERT, UPDATE, DELETE), Data Definition Language (DDL) for managing schema objects (CREATE, ALTER, DROP), and Transaction Control statements (COMMIT, ROLLBACK).
Here is a breakdown of key Oracle SQL statement types and examples:

1. Data Manipulation Language (DML) Statements

DML statements are used to query and modify data within existing schema objects.

- **INSERT:** Adds new rows to a table.

```
sql
INSERT INTO employees (employee_id, last_name) VALUES (100, 'Smith');
```

- **UPDATE:** Modifies existing data in a table.

```
sql
UPDATE employees SET salary = salary * 1.1 WHERE department_id = 10;
```

- **DELETE:** Removes rows from a table.

```
sql
DELETE FROM employees WHERE employee_id = 100;
```

- **SELECT:** Retrieves data from the database.

2. Data Definition Language (DDL) Statements

DDL statements create, alter, and drop schema objects. DDL statements require exclusive access to objects and implicitly commit the current transaction.

- **CREATE:** Creates new schema objects (e.g., `CREATE TABLE`, `CREATE INDEX`).
- **ALTER:** Modifies existing objects (e.g., `ALTER TABLE ... ADD ...`).
- **DROP:** Deletes objects (e.g., `DROP VIEW`).

3. Transaction Control Statements

These manage changes made by DML statements to ensure data integrity.

- **COMMIT:** Permanently saves changes to the database.
- **ROLLBACK:** Undoes changes made in the current transaction if errors occur.
- **SAVEPOINT:** Sets a point within a transaction to which you can roll back.

4. Other Key Statements

- **WITH Clause (Subquery Factoring):** Simplifies complex queries by defining a variable that can be used throughout a stored procedure.

- **ALTER SYSTEM:** A system control statement used to manage the properties of an Oracle Database instance.

Important Oracle Specifics

- **Implicit Transaction Management:** In Oracle, DML statements like `INSERT` or `UPDATE` do not automatically save changes. You must issue a `COMMIT` to save changes permanently.
- **PL/SQL Control Statements:** Within PL/SQL blocks, you can use `IF-THEN-ELSE` and `CASE` statements to handle procedural logic, which can run different statements based on data values.
- **Case Sensitivity:** In Oracle, the `=` operator is used for comparison in `IF` statements, and string literals are case-sensitive

Types of SQL Statements

The lists in the following sections provide a functional summary of SQL statements and are divided into these categories:

Data Definition Language (DDL) Statements

Data definition language (DDL) statements let you to perform these tasks:

- Create, alter, and drop schema objects
- Grant and revoke privileges and roles
- Analyze information on a table, index, or cluster
- Establish auditing options
- Add comments to the data dictionary

The `CREATE`, `ALTER`, and `DROP` commands require exclusive access to the specified object. For example, an `ALTER TABLE` statement fails if another user has an open transaction on the specified table.
The `GRANT`, `REVOKE`, `ANALYZE`, `AUDIT`, and `COMMENT` commands do not require exclusive access to the specified object. For example, you can analyze a table while other users are updating the table.

The database implicitly commits the current transaction before and after every DDL statement.

A DDL statement is either blocking or nonblocking, and both types of DDL statements require exclusive locks on internal structures.

Many DDL statements may cause the database to recompile or reauthorize schema objects. For information on how the database recompiles and reauthorizes schema objects and the circumstances under which a DDL statement would cause this, see *Oracle AI Database Concepts*.
DDL statements are supported by PL/SQL with the use of the `DBMS_SQL` package.

The DDL statements are:

- `ALTER` ... (All statements beginning with `ALTER`, except `ALTER SESSION` and `ALTER SYSTEM`—see "Session Control Statements" and "System Control Statements")
- `ANALYZE`
- `ASSOCIATE STATISTICS`
- `AUDIT`
- `COMMENT`
- `CREATE` ... (All statements beginning with `CREATE`)
- `DISASSOCIATE STATISTICS`
- `DROP` ... (All statements beginning with `DROP`)
- `FLASHBACK` ... (All statements beginning with `FLASHBACK`)
- `GRANT`
- `NOAUDIT`
- `PURGE`
- `RENAME`
- `REVOKE`
- `TRUNCATE`

Data Manipulation Language (DML) Statements

Data manipulation language (DML) statements access and manipulate data in existing schema objects. These statements do not implicitly commit the current transaction. The data manipulation language statements are:

- `CALL`
- `DELETE`
- `EXPLAIN PLAN`
- `INSERT`
- `LOCK TABLE`
- `MERGE`
- `SELECT`
- `UPDATE`

The `SELECT` statement is a limited form of DML statement in that it can only access data in the database. It cannot manipulate data stored in the database, although it can manipulate the accessed data before returning the results of the query.
The `SELECT` statement is supported in PL/SQL only when executed dynamically. However, you can use the similar PL/SQL statement `SELECT INTO` in PL/SQL code, and you do not have to execute it dynamically. The `CALL` and `EXPLAIN PLAN` statements are supported in PL/SQL only when executed dynamically. All other DML statements are fully supported in PL/SQL.

Transaction Control Statements

Transaction control statements manage changes made by DML statements. The transaction control statements are:

- `COMMIT`
- `ROLLBACK`
- `SAVEPOINT`
- `SET TRANSACTION`

- `SET CONSTRAINT`

All transaction control statements, except certain forms of the `COMMIT` and `ROLLBACK` commands, are supported in PL/SQL. For information on the restrictions, see COMMIT and ROLLBACK.

Session Control Statements

Session control statements dynamically manage the properties of a user session. These statements do not implicitly commit the current transaction.

PL/SQL does not support session control statements. The session control statements are:

- `ALTER SESSION`
- `SET ROLE`

System Control Statements

- Use `ADMINISTER KEY MANAGEMENT` to manage software and hardware keystores, encryption keys, and secrets.
- Use `ALTER SYSTEM` to dynamically manage the properties of an Oracle Database instance. This statement does not implicitly commit the current transaction and is not supported in PL/SQL.

Embedded SQL Statements

Embedded SQL statements place DDL, DML, and transaction control statements within a procedural language program. Embedded SQL is supported by the Oracle pre-compilers.

DDL COMMAND

Data Definition Language (DDL) commands in Oracle are used to define, alter, and manage the structure of database objects (such as tables, views, indexes, and users). A key characteristic of DDL in Oracle is that it is **auto-committed**, meaning any changes made cannot be rolled back.

Here are the primary DDL commands in Oracle:

1. CREATE

Used to create new database objects.

- **Table:** `CREATE TABLE employees (id NUMBER, name VARCHAR2(50));`
- **Index:** `CREATE INDEX idx_emp_name ON employees(name);`
- **View:** `CREATE VIEW emp_v AS SELECT * FROM employees;`
- **User:** `CREATE USER new_user IDENTIFIED BY password;`
- **Sequence:** `CREATE SEQUENCE emp_seq START WITH 1;`

2. ALTER

Modifies the structure of an existing object.

- **Add Column:** `ALTER TABLE employees ADD (email VARCHAR2(100));`
- **Modify Column:** `ALTER TABLE employees MODIFY (name VARCHAR2(100));`

- **Drop Column:** `ALTER TABLE employees DROP COLUMN email;`
- **Add Constraint:** `ALTER TABLE employees ADD CONSTRAINT pk_emp_id PRIMARY KEY (id);`

3. `DROP`

Removes an object from the database permanently.

- **Table:** `DROP TABLE employees;`
- **Drop with Purge (permanent delete):** `DROP TABLE employees PURGE;`
- **Drop Constraint:** `ALTER TABLE employees DROP CONSTRAINT pk_emp_id;`

4. `TRUNCATE`

Removes all rows from a table, freeing up space, but retains the table structure.

- **Syntax:** `TRUNCATE TABLE employees;`
- *Note: Truncate is faster than DELETE and cannot be rolled back.*

5. `RENAME`

Renames an existing database object.

- **Table:** `RENAME old_table_name TO new_table_name;`
- **Column:** `ALTER TABLE table_name RENAME COLUMN old_col TO new_col;`

6. `COMMENT`

Adds comments to the data dictionary regarding a table or column.

- **Table:** `COMMENT ON TABLE employees IS 'Table for employee details';`
- **Column:** `COMMENT ON COLUMN employees.id IS 'Primary Key';`

7. `FLASHBACK`

A specialized Oracle command to restore a dropped table from the recycle bin.

- **Syntax:** `FLASHBACK TABLE employees TO BEFORE DROP;`

Key DDL Behaviors in Oracle

- **Auto-Commit:** DDL commands implicitly commit the current transaction before and after the statement runs.
- **Exclusive Locks:** Commands like `CREATE`, `ALTER`, and `DROP` require exclusive access to the object, meaning they may fail if another user is updating the table.
- **Metadata Changes:** DDL affects the Data Dictionary (metadata) rather than the data itself.

How to Generate DDL for Existing Objects

You can generate the DDL script for existing objects using the `DBMS_METADATA` package:

sql

```sql
SELECT DBMS_METADATA.GET_DDL('TABLE', 'EMPLOYEES') FROM DUAL;
```

DML COMMANDS

Data Manipulation Language (DML) commands in Oracle are used to manage, manipulate, and modify data within existing table structures. Unlike Data Definition Language (DDL) commands, DML commands do not implicitly commit the current transaction, allowing changes to be rolled back.
Here is a breakdown of the primary DML commands in Oracle:

1. Primary DML Commands

- **`INSERT`**: Adds new rows to a table.
  - *Example:* `INSERT INTO employees (id, name) VALUES (101, 'Smith');`
- **`UPDATE`**: Modifies existing data within a table.
  - *Example:* `UPDATE employees SET salary = 5000 WHERE id = 101;`
- **`DELETE`**: Removes rows from a table.
  - *Example:* `DELETE FROM employees WHERE id = 101;`
- **`MERGE`**: Performs a "UPSERT" operation—conditionally updates existing rows or inserts new rows into a table based on a match condition.

2. Other DML-Related Commands

- **`SELECT`**: Although often categorized under Data Query Language (DQL), `SELECT` is functionally treated as DML because it retrieves data.
- **`LOCK TABLE`**: Explicitly locks a table to control concurrent access by other users.
- **`CALL`**: Calls a PL/SQL or Java subprogram.
- **`EXPLAIN PLAN`**: Displays the execution plan for a SQL statement.

3. Key Concepts

- **Transaction Control**: Because DML statements do not automatically save changes, they must be finalized using Transaction Control Language (TCL) commands:
  - **`COMMIT`**: Makes changes permanent.
  - **`ROLLBACK`**: Undoes changes since the last commit.
  - **`SAVEPOINT`**: Sets points within a transaction to which you can roll back.
- **Impact of DML**: DML affects data at the row level, whereas DDL affects the schema (structure) of the database.
- **Performance**: The `WHERE` clause is essential in `UPDATE` and `DELETE` to avoid modifying all rows in a table.

DCL COMMANDS

Data Control Language (DCL) commands in Oracle are used to manage security and access rights to database objects. These commands allow Database Administrators (DBAs) to control which users can access data and what operations they can perform (SELECT, INSERT, UPDATE, etc.).
The two primary DCL commands in Oracle are **GRANT** and **REVOKE**.

1. GRANT

The `GRANT` command gives users or roles specific privileges on database objects, such as tables, views, or procedures.

- **Syntax:**

```sql
GRANT privilege_name ON object_name TO user_name [WITH GRANT OPTION];
```

- **Example: Grant Object Privilege**
  Allow `user1` to select and update data in the `employees` table:

```sql
GRANT SELECT, UPDATE ON employees TO user1;
```

- **Example: Grant System Privilege**
  Allow a user to create a session (log in):

```sql
GRANT CREATE SESSION TO user1;
```

- **Example: With Grant Option**
  Allows `user1` to grant the same privilege to others:

```sql
GRANT SELECT ON employees TO user1 WITH GRANT OPTION;
```

2. REVOKE

The `REVOKE` command removes previously granted privileges from a user or role.

- **Syntax:**

```sql
REVOKE privilege_name ON object_name FROM user_name;
```

- **Example: Revoke Privilege**
  Withdraw the update permission from `user1` on the `employees` table:

```sql
REVOKE UPDATE ON employees FROM user1;
```

Key DCL Concepts in Oracle

- **System Privileges:** Permits users to perform specific actions in the database, such as `CREATE TABLE`, `ALTER TABLE`, or `CREATE USER`.

- **Object Privileges:** Permits users to perform actions on specific schema objects, such as `SELECT`, `INSERT`, `UPDATE`, `DELETE`, or `EXECUTE`.
- **Roles:** DCL is often used to grant privileges to a *role* (a bundle of privileges) rather than individual users to simplify management.
  - *Example:* `GRANT CONNECT, RESOURCE TO my_role;`
- **Roles vs. Users:** You can grant privileges to users or to roles.
  - *Example:* `GRANT SELECT ON employees TO manager_role;`

Difference between GRANT/REVOKE and DDL

While GRANT and REVOKE act as DCL, Oracle sometimes classifies them under Data Definition Language (DDL) because they affect the security definition of an object, though they are fundamentally for control, not structure modification