SQL Learning Notes

# Oracle DB Architecture

Oracle Database architecture consists of physical and logical structures. Physical structures include data files, redo log files, and control files. Logical structures include tablespaces, schemas, tables, and indexes. The architecture also includes memory structures (SGA and PGA) and background processes (DBWR, LGWR, SMON, etc.).

# DDL (Data Definition Language)

DDL is used to define and manage database objects such as tables, indexes, and views. Common DDL commands include CREATE, ALTER, DROP, and TRUNCATE.

# DML (Data Manipulation Language)

DML is used to retrieve and manipulate data in the database. Common DML commands include SELECT, INSERT, UPDATE, and DELETE.

# DCL (Data Control Language)

DCL is used to control access to data in the database. Common DCL commands include GRANT and REVOKE.COMMIT and SAVEPOINT.

# SELECT

The SELECT statement is used to query data from a database. It specifies the columns to retrieve and can include various clauses to filter and sort the data.

# FROM

The FROM clause specifies the tables or views from which to retrieve the data. It can include joins to combine data from multiple tables.

# WHERE

The WHERE clause filters the results returned by the SELECT statement based on specified conditions.

# IN

The IN operator is used to filter results based on a list of values. It allows you to specify multiple values in a WHERE clause.

# ANY

The ANY operator is used to compare a value to any value in a list or subquery. It returns true if the comparison is true for any value.

# EXISTS and NOT EXISTS

The EXISTS operator is used to check for the existence of rows in a subquery. NOT EXISTS checks for the non-existence of rows. These operators are often used in correlated subqueries. They work in background as Boolean values.

# ORDER BY

The ORDER BY clause is used to sort the results of a query by one or more columns, either in ascending (ASC) or descending (DESC) order.

# HAVING

The HAVING clause is used to filter results after an aggregation is performed, often used with the GROUP BY clause.

# Subquery and its usage in SELECT, FROM, WHERE with one example

Subquery in SELECT: SELECT (SELECT COUNT(\*) FROM employees) AS total\_employees  
Subquery in FROM: SELECT \* FROM (SELECT \* FROM employees WHERE department\_id = 10)  
Subquery in WHERE: SELECT \* FROM employees WHERE department\_id IN (SELECT department\_id FROM departments WHERE location\_id = 1700)

Usage wise: Where is mostly used then from and then select.

# Oracle-specific joins

Nested Loop Join: Iterates over one table and for each row, it searches for matching rows in another table.  
Sort Merge Join: Sorts both tables and merges them based on the join condition.  
Hash Join: Uses a hash table to join large datasets efficiently.

# INNER Join, OUTER Join, LEFT Join, RIGHT Join, CROSS Join

INNER Join: Returns rows that have matching values in both tables.  
LEFT Join (LEFT OUTER Join): Returns all rows from the left table and the matched rows from the right table. Non-matching rows in the right table result in NULLs.  
RIGHT Join (RIGHT OUTER Join): Returns all rows from the right table and the matched rows from the left table. Non-matching rows in the left table result in NULLs.  
FULL OUTER Join: Returns rows when there is a match in one of the tables. Non-matching rows result in NULLs.  
CROSS Join: Returns the Cartesian product of the two tables, i.e., all possible combinations of rows.

# SET operations (UNION, INTERSECT, UNION ALL, MINUS)

UNION: Combines the results of two queries and removes duplicates.  
UNION ALL: Combines the results of two queries and includes duplicates.  
INTERSECT: Returns only the rows that are common to both queries.  
MINUS: Returns rows from the first query that are not in the second query.

# VIEWS

A view is a virtual table based on the result set of a SELECT query. It can simplify complex queries and enhance security by restricting access to specific data.

# Simple View, Complex View, Materialized View

Simple View: Based on a single table, no group functions, and can be updated.  
Complex View: Based on multiple tables, includes group functions or joins, and may not be updatable.

Indexing is not possible in simple and complex views.

Materialized View: A view that stores the result set physically, can be refreshed periodically to improve performance on complex queries. It can be indexed. Data retrieval is fast here.