Theory for SQL commands – Whys and What's

Contents

Condition Precedence

Precedence is the order in which Oracle evaluates different conditions in the same expression. When evaluating an expression containing multiple conditions, Oracle evaluates conditions with higher precedence before evaluating those with lower precedence. Oracle evaluates conditions with equal precedence from left to right within an expression.

In Table 1 it lists the levels of precedence among SQL condition from high to low. The conditions listed on the same line have the same precedence. As the table indicates, Oracle evaluates operators before conditions.

Table 1 SQL Condition Precedence

Type of Condition	Purpose
=, !=, <, >, <=, >=,	comparison
IS [NOT] NULL, LIKE, [NOT] BETWEEN, [NOT] IN, EXISTS, IS OF *type*	comparison
NOT	exponentiation, logical negation
AND	conjunction
OR	disjunction

Comparison Conditions

Comparison conditions compare one expression with another. The result of such a comparison can be TRUE, FALSE, or NULL.

Table 2 Comparison Conditions

Type of Condition	Purpose	Example
=	Equality test.	select ename, job from emp where job = 'CLERK'

Type of Condition	Purpose	Example
		and deptno = 20; ORDER BY empno;
!=	Inequality test	select ename, job from emp where job! = 'CLERK';
>= <=	Greater-than-or-equal-to and less-than-or-equal-to tests.	SELECT ename, job FROM emp WHERE where sal>= comm;

Join - Selecting data from more than one table

To select the data from more than one table, you list the columns required in the **SELECT** statement, list the tables required in the **FROM** clause, specify the join between the tables (via the Primary and Foreign Keys) in the **WHERE / AND** clauses and set the selection criteria in an additional **AND** statement.

The JOIN operations are:

- **INNER JOIN operation**: Specifies a join between two tables with an explicit join clause.
- **LEFT OUTER JOIN operation**: Specifies a join between two tables with an explicit join clause, preserving unmatched rows from the first table.
- **RIGHT OUTER JOIN operation:** Specifies a join between two tables with an explicit join clause, preserving unmatched rows from the second table.
- **CROSS JOIN operation**: Specifies a join that produces the Cartesian product of two tables. It has no explicit join clause.
- **NATURAL JOIN operation**: Specifies an inner or outer join between two tables. It has no explicit join clause. Instead, one is created implicitly using the common columns from the two tables.

In all cases, you can specify additional restrictions on one or both of the tables being joined in outer join clauses or in the WHERE clause.

The number of join conditions required is the number of tables to be joined minus 1. Therefore, if you wanted to join 3 tables, the number of join conditions would be 2.

The syntax is as follows:

SELECT tablename.columnname [,tablename.columnname, …]
FROM tablename1, tablename2 [,tablename, …]
WHERE tablename1.primarykey = tablename2.foreignkey
[AND tablename2.primarykey = tablename3.foreignkey, …]
AND columnname OPERATOR value
[AND columnname OPERATOR value..];

The LIKE Condition

The LIKE conditions specify a test involving pattern matching. Whereas the equality operator (=) exactly matches one character value to another, the LIKE conditions match a portion of one character string value to another by searching the first value for the pattern specified by the second. LIKE matches strings using characters as defined by the input character set.

The pattern can contain special pattern-matching characters:

- An underscore (_) in the pattern matches exactly one character (as opposed to one byte in a multibyte character set) in the value.
- A percent sign (%) in the pattern can match zero or more characters (as opposed to bytes in a multibyte character set) in the value. The pattern '%' cannot match a null.

You can include the actual characters % or _ in the pattern by using the ESCAPE clause, which identifies the escape character. If the escape character precedes the character % or _ in the pattern, then Oracle interprets this character literally in the pattern rather than as a special pattern-matching character. You can also search for the escape character itself by repeating it. For example, if @ is the escape character, then you can use @@ to search for @.

Table 3 LIKE Conditions

Type of Condition	Operation	Example
x [NOT] LIKE y [ESCAPE 'z']	TRUE if *x* does [not] match the pattern *y*. Within *y*, the character % matches any string of zero or more characters except null. The character _ matches any single character. Any character can follow ESCAPE except percent (%) and underbar (_). A wildcard character is treated as a literal if preceded by the escape character.	SELECT* FROM emp WHERE ename **LIKE** 'KI%'; SELECT* FROM emp WHERE ename **LIKE** '__N%';

To process the LIKE conditions, Oracle divides the pattern into subpatterns consisting of one or two characters each. The two-character subpatterns begin with the escape character and the other character is %, or _, or the escape character.

LIKE Condition: General Examples

This condition is true for all last_name values beginning with Ma:

- last_name LIKE 'Ma%'

All of these last_name values make the condition true:

- Mallin, Markle, Marlow, Marvins, Marvis, Matos

Case is significant, so last_name values beginning with MA, ma, and mA make the condition false.

Consider this condition:

- last_name LIKE 'SMITH_'

This condition is true for these last_name values:

- SMITHE, SMITHY, SMITHS

This condition is false for SMITH because the special underscore character (_) must match exactly one character of the last_name value.Range Conditions

Table 4 Range Conditions

Type of Condition	Operation	Example
[NOT] **BETWEEN** x AND y	[Not] greater than or equal to *x* and less than or equal to *y*.	SELECT * FROM emp WHERE sal **BETWEEN** 1000 and 1500 ORDER BY empno;

Null Conditions

A NULL condition tests for nulls. This is the only condition that you should use to test for nulls.

Table 5 Null Conditions

Type of Condition	Operation	Example
IS [NOT] NULL	Tests for nulls.	SELECT ename FROM emp WHERE sal **IS NOT NULL** AND job = 'CLARK';