

RESTRICTING AND SORTING DATA

http://www.tutorialspoint.com/sql_certificate/restricting_and_sorting_data.htm

Copyright © tutorialspoint.com

The essential capabilities of SELECT statement are Selection, Projection and Joining. Displaying specific columns from a table is known as a project operation. We will now focus on displaying specific rows of output. This is known as a select operation. Specific rows can be selected by adding a WHERE clause to a SELECT query. As a matter of fact, the WHERE clause appears just after the FROM clause in SELECT query hierarchy. The sequence has to be maintained in all scenarios. If violated, Oracle raises an exception.

Syntax:

```
SELECT *|[[DISTINCT] column| expression [alias],...]  
FROM table  
[WHERE condition(s)]
```

In the syntax,

- WHERE clause is the keyword
- [condition] contains column names, expressions, constants, literals and a comparison operator.

Suppose that your manager is working on the quarterly budget for your organization. As part of this activity, it is necessary to produce a listing of each employee's essential details, but only for employees that are paid at least \$25,000 annually. The SQL query below accomplishes this task. Note the use of the WHERE clause shown in bold text.

```
SELECT Employee_ID, Last_Name, First_Name, Salary  
FROM employees  
WHERE Salary >= 25000;
```

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	SALARY
88303	Jones	Quincey	\$30,550.00
88404	Barlow	William	\$27,500.00
88505	Smith	Susan	\$32,500.00

3 rows selected

Points to be noted -

- A SELECT clause can contain only one WHERE clause. However, multiple filter conditions can be appended to WHERE clause using AND or OR operator.
- The columns, literals or expressions in a predicate clause must be of similar or interconvertible data types.
- Column alias cannot be used in the WHERE clause.
- Character literals must be enclosed within single quotation marks and are case sensitive.
- Date literals must be enclosed within single quotation marks and are format sensitive. Default format is **DD-MON-RR**.

Comparison Operators

Comparison operators are used in predicates to compare one term or operand with another term. SQL offers comprehensive set of equality, inequality and miscellaneous operators. They can be used depending on the data and filter condition logic in the SELECT query. When you use comparison operators in a WHERE clause, the arguments *objects or values you are comparing* on both sides of the operator must be either a column name, or a specific value. If a specific value is used, then the value must be either a numeric value or a literal string. If the value is a character string or

date, you must enter the value within single quotation marks " .

Oracle has nine comparison operators to be used in equality or inequality conditions.

Operator	Meaning
=	equal to
<	less than
>	greater than
>=	greater than or equal to
<=	less than or equal to
!=	not equal to
<>	not equal to
!>	not greater than
!<	not less than

Other Oracle operators are BETWEEN..AND, IN, LIKE, and IS NULL.

The BETWEEN Operator

The BETWEEN operator can be used to compare a column value within a definite range. The specified range must have a lower and upper limit where both are inclusive during comparison. Its use is similar to composite inequality operator `<= and >=` . It can be used with numeric, character and date type values.



For example, the WHERE condition **SALARY BETWEEN 1500 AND 2500** in a SELECT query will list those employees whose salary is between 1500 and 2500.

The IN Operator

The IN operator is used to test a column value in a given set of value. If the column can be equated to any of the values from the given set, the condition is validated. The condition defined using the IN operator is also known as the membership condition.

For example, the WHERE condition **SALARY IN 1500, 3000, 2500** in a SELECT query will restrict the rows where salary is either of 1500, 3000 or 2500.

The LIKE Operator

The LIKE operator is used for pattern matching and wildcard searches in a SELECT query. If a portion of the column value is unknown, wildcard can be used to substitute the unknown part. It uses wildcard operators to build up the search string, thus search is known as Wildcard search. These two operators are Percentile ' and Underscore . Underscore  substitutes a single character while percentile ' replaces more than one characters. They can be used in combination as well.

For example, the below SELECT query lists the first names of those employees whose last name starts with 'SA'.

```
SELECT first_name
FROM employees
WHERE last_name LIKE 'SA%';
```

IS NOT NULL Conditions

To be noted, NULL values cannot be tested using equality operator. It is because NULL values are unknown and unassigned while equality operator tests for a definite value. The IS NULL operator serves as equality operator to check NULL values of a column.

For example, the WHERE condition **COMMISSION_PCT IS NULL** in a SELECT query will list employees who don't have commission percentage.

Logical Operators

Multiple filter conditions can be added to the WHERE clause predicate. More than one condition can be combined together using logical operators AND, OR and NOT.

- AND: joins two or more conditions, and returns results only when all of the conditions are true.
- OR: joins two or more conditions, and it returns results when any of the conditions are true.
- NOT: negates the expression that follows it.

The AND operator links two or more conditions in a WHERE clause and returns TRUE only if all the conditions are true. Suppose that a manager needs a list of female employees. Further, the list should only include employees with last names that begin with the letter "E" or that come later in the alphabet. Additionally, the result table should be sorted by employee last name. There are two simple conditions to be met. The WHERE clause may be written as: WHERE Gender = 'F' AND last_name > 'E'.

```
SELECT last_name "Last Name", first_name "First Name", Gender "Gender"
FROM employees
WHERE Gender = 'F' AND last_name > 'E'
ORDER BY last_name;
```

The OR operator links more than one condition in a WHERE clause and returns TRUE if either of the condition returns true. Suppose that your organizational manager's requirements change a bit. Another employee listing is needed, but in this listing the employees should: 1 be female or, 2 have a last name that begins with the letter "T" or a letter that comes later in the alphabet. The result table should be sorted by employee last name. In this situation either of the two conditions can be met in order to satisfy the query. Female employees should be listed along with employees having a name that satisfies the second condition.

The NOT operator is used to negate an expression or condition.

The ORDER BY Clause

When you display only a few rows of data, it may be unnecessary to sort the output; however, when you display numerous rows, managers may be aided in decision making by having the information sorted. Output from a SELECT statement can be sorted by using the optional ORDER BY clause. When you use the ORDER BY clause, the column name on which you are ordering must also be a column name that is specified in the SELECT clause.

The below SQL query uses an ORDER BY clause to sort the result table by the last_name column in ascending order. Ascending order is the default sort order.

```
SELECT last_name, first_name
FROM employees
WHERE last_name >= 'J'
ORDER BY last_name;
```

last_name	first_name
Jones	Quincey
Klepper	Robert
Quattromani	Toni
Schultheis	Robert

Sorting can be based on numeric and date values also. Sorting can also be done based on multiple columns.

By default, the ORDER BY clause will sort output rows in the result table in ascending order. We can use the keyword DESC *short for descending* to enable descending sort. The alternative default is ASC which sorts in ascending order, but the ASC keyword is rarely used since it is the default. When the ASC or DESC optional keyword is used, it must follow the column name on which you are sorting in the WHERE clause.

Positional Sorting - Numeric position of the column in the selected column list can be given in ORDER BY clause, instead of column name. It is mainly used in UNION queries *discussed later*. The Query orders the result set by salary since it appears 2nd in the column list.

```
SELECT first_name, salary
FROM employees
ORDER BY 2;
```

Substitution Variables

When a SQL query has to be executed more than once for the different set of inputs, substitution variables can be used. Substitution variables can be used to prompt for user inputs before the query execution. They are widely used in query based report generation which takes data range from the users as input for the conditional filtering and data display. Substitution variables are prefixed by a single-ampersand **&** symbol to temporarily store values. For example,

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY
FROM employees
WHERE LAST_NAME = &last_name
OR EMPLOYEE_ID = &EMPNO;
```

When the above SELECT query is executed, oracle identifies the '&' as substitution variable. It prompts user to enter value for 'last_name' and 'EMPNO' as below.

```
Enter value for last_name:
Enter value for empno:
```

Once the user provides inputs to both the variables, values are substituted, query is verified and executed.

Points to be noted -

- If the variable is meant to substitute a character or date value, the literal needs to be enclosed in single quotes. A useful technique is to enclose the ampersand substitution variable in single quotes when dealing with character and date values.
- Both SQL Developer and SQL* Plus support the substitution variables and the DEFINE/UNDEFINE commands. Though SQL Developer or SQL* Plus does not support validation checks *exceptfordatatype* on user input.
- You can use the substitution variables not only in the WHERE clause of a SQL statement, but also as substitution for column names, expressions, or text.

Using the Double-Ampersand Substitution Variable

When the same substitution variable is used at more than one place, then to avoid re-entering the same data again, we use double ampersand substitution. In such cases, value of the substitution variable, once entered, would be substituted at all instances of usage.

```
SELECT first_name, HIRE_DATE, SEPARATION_DATE
FROM employees
WHERE HIRE_DATE LIKE '&DT%' AND SEPARATION_DATE '&&DT%'
```

Note that the same value of &DT is substituted twice in the above query. So, its value once given by the user will be substituted at two places.

The DEFINE and VERIFY Commands

Setting the definition of variables in a session is set by DEFINE feature of SQL* Plus. The variables can be defined in the session, so as to avoid halt during query execution. Oracle reads the same variable whenever encountered in an SQL query. It is in ON state by default. With the help of DEFINE clause, one can declare a variable in command line before query execution as **DEFINE variable=value;**

Verify command verifies the above substitution showing as OLD and NEW statement. It is OFF by default and can be set to ON using SET command.

```
SQL> SET DEFINE ON
```

```
SQL> SET VERIFY ON
SQL> DEFINE NAME = MARTIN'
SQL> SELECT first_name, SALARY
FROM employees
WHERE first_name = '&NAME';
OLD 1: select first_name, sal from employee where first_name = '&first_name'
new 1: select first_name, sal from employee where first_name = 'MARTIN'

first_name      SALARY
-----
MARTIN          5000
```

Processing math: 100%