## Review of the Oracle WHERE clause

The WHERE clause specifies a **search condition** for rows returned by the SELECT statement. The following illustrates the syntax of the *WHERE* clause:

*SELECT*

*column\_1,*

*column\_2,*

*...*

*FROM*

*table\_name*

*WHERE*

*search\_condition*

*ORDER BY*

*column\_1,*

*column\_2;*

The WHERE clause appears after the FROM clause but before the ORDER BY clause. Following the *WHERE* keyword is the **search\_condition** that defines a condition which returned rows must satisfy.

Besides the SELECT statement, you can use the WHERE clause in the DELETE or UPDATE statement to specify which rows to update or delete.

**Select certain rows using comparison operator in the WHERE clause**

Oracle provides you with many comparison operators illustrated in the following table:

| **Operator** | **Description** |
| --- | --- |
| = | Equality |
| !=, <>, ~=, ^= | Inequality |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| IN | Equal to any value in a list of values |
| ANY / SOME / ALL | Compare a value to a list or Subquery. It must be preceded by another operator such as =, >, <. |
| NOT IN | Not equal to any value in a list of values |
| [NOT] BETWEEN n and m | Equivalent to [Not] >= *n* and <= m. |
| [NOT] EXISTS | Return true if subquery returns at least one row |
| IS [NOT] NULL | NULL test |

### A) Selecting rows by using a simple equality operator

The following example returns only products whose names are 'Kingston':

SELECT

product\_name,

description,

list\_price,

FROM

products

WHERE

product\_name = 'Kingston';

|  |  |  |
| --- | --- | --- |
| **PRODUCT\_NAME** | **DESCRIPTION** | **LIST\_PRICE** |
| Kingston | Speed:DDR4-2133,Type:288-pin DIMM,CAS:15Module:4x16GBSize:64GB | 741.63 |
| Kingston | Speed:DDR3-1333,Type:240-pin DIMM,CAS:9Module:4x16GBSize:64GB | 671.38 |
| Kingston | Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x8GBSize:32GB | 653.5 |
| Kingston | Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x16GBSize:64GB | 644 |

In another example, to get products whose list price is equal to 799, you use the following statement.

SELECT

product\_name,

list\_price

FROM

products

WHERE

list\_price = 799;

|  |  |
| --- | --- |
| **PRODUCT\_NAME** | **LIST\_PRICE** |
| Intel Core i7-4770K | 799 |
| Crucial | 799 |

**B) Select rows that meet some Logical conditions**

To combine conditions you can use the AND, OR and NOT logical operators.

For example, to get all motherboards that belong to the category id *4* and have list prices greater than *500,* you use the following statement:

SELECT

product\_name,

list\_price

FROM

products

WHERE

category\_id = 4

AND

list\_price > 500;

|  |  |
| --- | --- |
| **PRODUCT\_NAME** | **LIST\_PRICE** |
| Supermicro X10SDV-8C-TLN4F | 948.99 |
| Intel DP35DPM | 789.79 |
| Asus X99-E-10G WS | 649 |
| Asus ROG MAXIMUS IX EXTREME | 573.99 |
| Asus RAMPAGE V EXTREME | 572.96 |
| Asus Z10PE-D8 WS | 561.59 |
| MSI X99A GODLIKE GAMING CARBON | 549.59 |
| Supermicro H8DG6-F | 525.99 |
| Asus Rampage V Edition 10 | 519.99 |
| Gigabyte GA-Z270X-Gaming 9 | 503.98 |

**C) Selecting rows that have a value between two values**

To find rows that have a value between two values, you use the BETWEEN AND operator in the WHERE clause.

For example, to get the products whose list prices are between *650* and *680*, you use the following statement.

SELECT

product\_name,

list\_price

FROM

products

WHERE

list\_price BETWEEN 650 AND 680

ORDER BY

list\_price;

|  |  |
| --- | --- |
| **PRODUCT\_NAME** | **LIST\_PRICE** |
| Kingston | 653.5 |
| Corsair Dominator Platinum | 659.99 |
| Intel Core i7-3930K | 660 |
| Kingston | 671.38 |
| G.Skill Ripjaws V Series | 677.99 |
| Intel Core i7-7820X | 678.75 |

Note that the following expressions are equivalent:

list\_price BETWEEN 650 AND 680

list\_price >= 650 AND list\_price <= 680

**D) Selecting rows that are in a list of values**

To query rows that are in a list of values, you use the IN operator as follows in the query that finds all products from category 1 or 4, that cost in the range from 900 to 1000.

SELECT

product\_name, list\_price,

category\_id

FROM

products

WHERE

category\_id IN (1, 4)

AND

list\_price between 900 and 1000

ORDER BY

product\_name;

The previous statement can be rewritten like:

SELECT

product\_name, list\_price,

category\_id

FROM

products

WHERE

(category\_id = 1 OR category\_id = 4)

AND

list\_price between 900 and 1000

ORDER BY

product\_name;

|  |  |  |
| --- | --- | --- |
| **PRODUCT\_NAME** | **LIST\_PRICE** | **CATEGORY\_ID** |
| Intel Core i7-5960X (OEM/Tray) | 977.99 | 1 |
| Intel Xeon E5-2650 V3 (OEM/Tray) | 939.49 | 1 |
| Supermicro X10SDV-8C-TLN4F | 948.99 | 4 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

To query rows that are NOT in a list of values, you use the NOT IN operator as follows in the query that finds all products NOT coming from category 1 and 4, that cost in the range from 900 to 1000.

SELECT

product\_name, list\_price,

category\_id

FROM

products

WHERE

category\_id NOT IN (1, 4)

and list\_price between 900 and 1000

ORDER BY

product\_name;

|  |  |  |
| --- | --- | --- |
| **PRODUCT\_NAME** | **LIST\_PRICE** | **CATEGORY\_ID** |
| AMD 100-506061 | 999.99 | 2 |
| ATI FirePro R5000 | 999.99 | 2 |
| PNY VCQK4200-PB | 949.89 | 2 |

**E) Selecting rows which contain value as a part of a string**

The following statement uses LIKE operator with a Wild Card % to retrieve products whose name starts with Asus,and cost less than 300.

SELECT

product\_name,

list\_price

FROM

products

WHERE

product\_name LIKE 'Asus%'

AND

list\_price < 300

ORDER BY

list\_price;

|  |  |
| --- | --- |
| **PRODUCT\_NAME** | **LIST\_PRICE** |
| Asus VANGUARD B85 | 287 |
| Asus Sabertooth 990FX | 295.72 |
| Asus MAXIMUS IX CODE | 298.98 |

In this example, we used the LIKE operator to match rows based on the **specified pattern.**

Now, if we want to see all products with name ending on *F* and second letter is *u*, then:

SELECT product\_name, list\_price

FROM products

WHERE product\_name LIKE '\_u%F'

ORDER BY list\_price;

|  |  |
| --- | --- |
| **PRODUCT\_NAME** | **LIST\_PRICE** |
| Supermicro X11SSL-CF | 419.99 |
| Supermicro X9SRH-7TF | 479.99 |
| Supermicro H8DG6-F | 525.99 |
| Supermicro X10SDV-8C-TLN4F | 948.99 |

**F) Selecting rows which have a column with a NULL value**

The IS NULL operator returns true if the expression or column is NULL. Otherwise, it returns false.

The following query returns all sales orders that *do not have a responsible salesman* and their status is *Pending*. Sorting is by the most recent order dates first (descending).

SELECT \*

FROM orders

WHERE salesman\_id IS NULL

AND status = 'Pending'

ORDER BY order\_date DESC;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ORDER\_ID** | **CUSTOMER\_ID** | **STATUS** | **SALESMAN\_ID** | **ORDER\_DATE** |
| 21 | 21 | Pending |  | 27-MAY-17 |
| 10 | 44 | Pending |  | 24-JAN-17 |
| 16 | 16 | Pending |  | 27-SEP-16 |
| 68 | 9 | Pending |  | 13-JUN-16 |

To negate the IS NULL operator, you use the IS NOT NULL operator as follows:

The operator IS NOT NULL returns true if the expression or value in the column is not null. Otherwise, it returns false.

For example, the following example returns all sales orders which *have a responsible salesman* and are related to customers *1 or 4*.

SELECT \*

FROM orders

WHERE salesman\_id IS NOT NULL

AND customer\_id IN (1,4)

ORDER BY order\_date DESC;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ORDER\_ID** | **CUSTOMER\_ID** | **STATUS** | **SALESMAN\_ID** | **ORDER\_DATE** |
| 94 | 1 | Shipped | 62 | 27-OCT-17 |
| 1 | 4 | Pending | 56 | 15-OCT-17 |
| 60 | 1 | Shipped | 62 | 30-JUN-17 |
| 77 | 1 | Shipped | 60 | 02-JAN-17 |
| 85 | 4 | Pending | 55 | 01-DEC-16 |
| 105 | 1 | Pending | 54 | 17-NOV-16 |
| 63 | 4 | Shipped | 61 | 30-JUN-16 |