

- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
IN	Equal to any member in the list
ANY	Must be preceded by =, !=, >, <, <=, >=. Compares a value to each value in a list or returned by a query. Evaluates to FALSE if the query returns no rows.
ALL	Must be preceded by =, !=, >, <, <=, >=. Compares a value to every value in a list or returned by a query. Evaluates to TRUE if the query returns no rows.

ORACLE

7-1

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Multiple-Row Subqueries

Subqueries that return more than one row are called multiple-row subqueries. You use a multiple-row operator, instead of a single-row operator, with a multiple-row subquery. The multiple-row operator expects one or more values:

```
SELECT last_name, salary, department_id
FROM   employees
WHERE  salary IN (SELECT   MIN(salary)
                  FROM     employees
                  GROUP BY department_id);
```

Example:

Find the employees who earn the same salary as the minimum salary for each department. The inner query is executed first, producing a query result. The main query block is then processed and uses the values that were returned by the inner query to complete its search condition. In fact, the main query appears to the Oracle server as follows:

```
SELECT last_name, salary, department_id
FROM   employees
WHERE  salary IN (2500, 4200, 4400, 6000, 7000, 8300,
                 8600, 17000);
```

Using the ANY Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

ORACLE

7-2

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Using the ANY Operator in Multiple-Row Subqueries

The ANY operator (and its synonym, the SOME operator) compares a value to *each* value returned by a subquery. The slide example displays employees who are not IT programmers and whose salary is less than that of any IT programmer. The maximum salary that a programmer earns is \$9,000.

- <ANY means less than the maximum.
- >ANY means more than the minimum.
- =ANY is equivalent to IN.

Using the ALL Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

ORACLE

7-3

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Using the ALL Operator in Multiple-Row Subqueries

The ALL operator compares a value to *every* value returned by a subquery. The example in the slide displays employees whose salary is less than the salary of all employees with a job ID of IT_PROG and whose job is not IT_PROG.

>ALL means more than the maximum and <ALL means less than the minimum.

The NOT operator can be used with IN, ANY, and ALL operators.

Using the EXISTS Operator

```
SELECT * from orders
WHERE NOT EXISTS
(Select *
FROM order_items
WHERE order_items.order_id = orders.order_id);
```

	ORDER_ID	ORDER_DATE	ORDER_MODE	CUSTOMER_ID	ORDER_STATUS	ORDER_TOTAL	SALES_REP_ID
1	2417	20-MAR-99 12.00.00.000000000	AM direct	105	5	1926.6	163
2	2448	18-JUN-99 12.00.00.000000000	AM direct	145	5	1388	158
3	2450	11-APR-99 12.00.00.000000000	AM direct	147	3	1636	159
4	2370	26-JUN-00 12.00.00.000000000	AM online	117	4	126	(null)
5	2402	02-JUL-99 12.00.00.000000000	AM direct	161	8	600	154
6	2456	07-NOV-98 12.00.00.000000000	AM direct	117	0	3878.4	163

ORACLE

7-4

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Using the EXISTS Operator

The EXISTS operator is used in queries where the query result depends on whether or not certain rows exist in a table. It evaluates to TRUE if the subquery returns at least one row.

The example in the slide displays orders that have no entries in order_items table. For each row in the ORDERS table, the condition is checked whether there exists a row in the ORDER_ITEMS table that has the same order ID. In case no such row exists, the condition is satisfied for the row under consideration and it is selected. If there exists a corresponding row in the ORDERS table, the row is not selected.

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```

0 rows selected

ORACLE

7-5

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Null Values in a Subquery

The SQL statement in the slide attempts to display all the employees who do not have any subordinates. Logically, this SQL statement should have returned 12 rows. However, the SQL statement does not return any rows. One of the values returned by the inner query is a null value and, therefore, the entire query returns no rows.

The reason is that all conditions that compare a null value result in a null. So whenever null values are likely to be part of the results set of a subquery, do not use the NOT IN operator. The NOT IN operator is equivalent to \neq ALL.

Notice that the null value as part of the results set of a subquery is not a problem if you use the IN operator. The IN operator is equivalent to $=$ ANY. For example, to display the employees who have subordinates, use the following SQL statement:

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```

Null Values in a Subquery (continued)

Alternatively, a WHERE clause can be included in the subquery to display all employees who do not have any subordinates:

```
SELECT last_name FROM employees
WHERE  employee_id NOT IN
        (SELECT manager_id
         FROM   employees
         WHERE  manager_id IS NOT NULL);
```

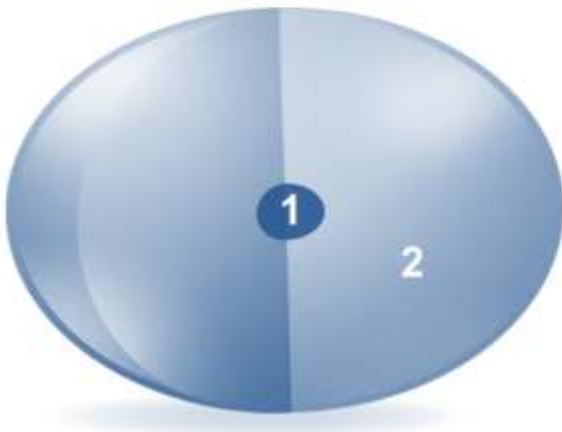
Using a subquery is equivalent to performing two sequential queries and using the result of the first query as the search values in the second query.

- 1.True
- 2.False

ORACLE

Answer: 1

Session Summary



Identify when a subquery can help solve a problem

Write subqueries when a query is based on unknown values

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT select_list
         FROM    table);
```

ORACLE

Session Summary

In this lesson, you should have learned how to use subqueries. A subquery is a `SELECT` statement that is embedded in the clause of another SQL statement. Subqueries are useful when a query is based on a search criterion with unknown intermediate values.

Subqueries have the following characteristics:

- Can pass one row of data to a main statement that contains a single-row operator, such as `=`, `<>`, `>`, `>=`, `<`, or `<=`
- Can pass multiple rows of data to a main statement that contains a multiple-row operator, such as `IN`
- Are processed first by the Oracle server, after which the `WHERE` or `HAVING` clause uses the results
- Can contain group functions



ORACLE

7-9

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 7: Overview

In this practice, you write complex queries using nested `SELECT` statements.

For practice questions, you may want to create the inner query first. Make sure that it runs and produces the data that you anticipate before you code the outer query.