Database Programming

Multiple-Row Subqueries

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Correctly use the comparison operators IN, ANY, and ALL in multiple-row subqueries
- Construct and execute a multiple-row subquery in the WHERE clause or HAVING clause
- Describe what happens if a multiple-row subquery returns a null value
- Understand when multiple-row subqueries should be used, and when it is safe to use a single-row subquery
- Create a query using the EXISTS and NOT EXISTS operators to test for returned rows from the subquery

Purpose

A subquery is designed to find information you don't know so that you can find information you want to know.

However, single-row subqueries can return only one row. What if you need to find information based on several rows and several values? The subquery will need to return several rows.

We achieve this using multiple-row subqueries and the three comparison operators: IN, ANY, and ALL.

Query Comparison

Whose salary is equal to the salary of an employee in department 20 ?

Why does this example not work?

```
SELECT first_name, last_name
FROM employees
WHERE salary =
   (SELECT salary
    FROM employees
    WHERE department_id = 20);
```

LAST_NAME	DEPT_ID	SALARY
Hartstein	20	13000
Fay	20	6000

ORA-01427: single-row subquery returns more than one row

Query Comparison (cont.)

Because more than one employee exists in department 20, so the subquery returns multiple rows. We call this a multiple-row subquery.

```
SELECT first_name, last_name
FROM employees
WHERE salary =
   (SELECT salary
    FROM employees
    WHERE department_id = 20);
```

LAST_NAME	DEPT_ID	SALARY
Hartstein	20	13000
Fay	20	6000

ORA-01427: single-row subquery returns more than one row

Query Comparison (cont.)

The problem is the equal sign (=) in the WHERE clause of the outer query. How can one value be equal to (or not equal to) more than one value at a time? It's a silly question, isn't it?

```
SELECT first_name, last_name
FROM employees
WHERE salary =
   (SELECT salary
    FROM employees
    WHERE department_id = 20);
```

LAST_NAME	DEPT_ID	SALARY
Hartstein	20	13000
Fay	20	6000

ORA-01427: single-row subquery returns more than one row

IN, ANY, and ALL

Subqueries that return more than one value are called multiple-row subqueries.

Because we cannot use the single-row comparison operators (=, <, and so on), we need different comparison operators for multiple-row subqueries.

```
SELECT title, year
FROM d_cds
WHERE year IN
   (SELECT year
     FROM d_cds);
```

D_CDS

TITLE	YEAR
The Celebrants Live in Concert	1997
Songs from My Childhood	1999
Party Music for All Occasions	2000
Carpe Diem	2000

YEAR
1997
1999
2000
2002

IN, ANY, and ALL (cont.)

The multiple-row operators are: IN, ANY, and ALL. The NOT operator can be used with any of these three operators.

```
SELECT title, year
FROM d_cds
WHERE year IN
   (SELECT year
      FROM d_cds);
```

D_CDS

TITLE	YEAR
The Celebrants Live in Concert	1997
Songs from My Childhood	1999
Party Music for All Occasions	2000
Carpe Diem	2000

YEAR
1997
1999
2000
2002

IN

The IN operator is used within the outer query WHERE clause to select only those rows which are IN the list of values returned from the inner query.

For example, we are interested in all the CD titles that have the same year as the CD numbers less than 93.

```
SELECT title, year
FROM d_cds
WHERE year IN
   (SELECT year
       FROM d_cds
       WHERE cd_number < 93);
```

D_CDS

TITLE	YEAR
The Celebrants Live in Concert	1997
Party Music for All Occasions	2000
Back to the Shire	2002

YEAR
1997
2000
2002

IN (cont.)

Since we are not sure what years exist for the CDs numbered below 93, the inner query will return that list of years for us.

The outer query will then return any title that has the same year as any year in the inner query list.

```
SELECT title, year
FROM d_cds
WHERE year IN
   (SELECT year
       FROM d_cds
       WHERE cd_number < 93);
```

D_CDS

TITLE	YEAR
The Celebrants Live in Concert	1997
Party Music for All Occasions	2000
Back to the Shire	2002

YEAR
1997
2000
2002

ANY

The ANY operator is used when we want the outer-query WHERE clause to select the rows which match the criteria (<, >, =, etc.) of at least one value in the subquery result set.

```
SELECT title, producer
FROM d_cds
WHERE year < ANY
 (SELECT year
  FROM d_cds
  WHERE producer = 'The Music Man');
```

D_CDS

TITLE	PRODUCER	YEAR
The Celebrants Live in Concert	Old Town Records	1997
Graduation Songbook	Tunes are Us	1998
Songs from my Childhood	Old Town Records	1999
Party Music for all Occasions	The Music Man	2000
Carpe Diem	R&B Inc.	2000

D_CDS

YEAR
2000
2001

ANY (cont.)

The example shown will return any CD title whose year is less than at least one CD title year produced by “The Music Man.”

```
SELECT title, producer
FROM d_cds
WHERE year < ANY
 (SELECT year
  FROM d_cds
  WHERE producer = 'The Music Man');
```

D_CDS

TITLE	PRODUCER	YEAR
The Celebrants Live in Concert	Old Town Records	1997
Graduation Songbook	Tunes are Us	1998
Songs from my Childhood	Old Town Records	1999
Party Music for all Occasions	The Music Man	2000
Carpe Diem	R&B Inc.	2000

D_CDS

YEAR
2000
2001

ALL

The ALL operator is used when we want the outer-query WHERE clause to select the rows which match the criteria (<, >, =, etc.) of all of the values in the subquery result set.

The example shown will return any CD title whose year is greater than all the CD title years produced by "The Music Man."

```
SELECT title, producer,year
FROM d_cds
WHERE year > ALL
 (SELECT year
  FROM d_cds
  WHERE producer = 'The Music Man');
```

D_CDS

TITLE	PRODUCER	YEAR
Back to the Shire	Middle Earth Records	2002
Whirled Peas	Old Town Records	2004

D_CDS

YEAR
2000
2001

ALL (cont.)

The ALL operator compares a value to every value returned by the inner query.

```
SELECT title, producer,year
FROM d_cds
WHERE year > ALL
 (SELECT year
  FROM d_cds
  WHERE producer = ‘The Music Man’);
```

D_CDS

TITLE	PRODUCER	YEAR
Back to the Shire	Middle Earth Records	2002
Whirled Peas	Old Town Records	2004

D_CDS

YEAR
2000
2001

NULL Values

Suppose that one of the values returned by a multiple-row subquery is null, but other values are not.

- If IN or ANY are used, the outer query will return rows which match the non-null values

```
SELECT last_name, employee_id
FROM employees
WHERE employee_id IN
    (SELECT manager_id
     FROM employees);
```

LAST_NAME	EMPLOYEE_ID
King	100
Kochhar	101
De Haan	102
Hunold	103
Mourgos	124

MANAGER_ID
(null)
100
100
102
103
103
100
124

NULL Values (cont.)

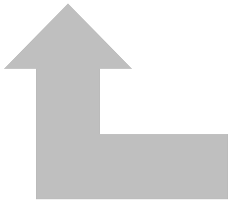
- If ALL is used, the outer query returns no rows because ALL compares the outer query row with every value returned by the subquery, including the null. And comparing anything with null results in null.

The example lists those employees who are managers.

```
SELECT last_name, employee_id
FROM employees
WHERE employee_id IN
    (SELECT manager_id
     FROM employees);
```

LAST_NAME	EMPLOYEE_ID
King	100
Kochhar	101
De Haan	102
Hunold	103
Mourgos	124

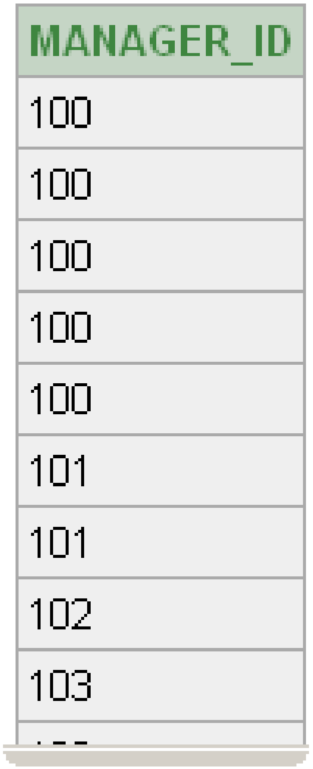
MANAGER_ID
(null)
100
100
102
103
103
100
124



NULL Values in Subqueries

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

Now, none of the values returned by the inner query is a null value, thus it works.



MANAGER_ID
100
100
100
100
100
101
101
102
103

GROUP BY and HAVING

As you might suspect, the GROUP BY clause and the HAVING clause can also be used with multiple-row subqueries.

LAST_NAME	DEPT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000

What if you wanted to find the departments whose minimum salary is less than the salary of any employee who works in department 10 or 20?

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
50	2500
60	4200
80	8600
110	8300
(null)	7000

GROUP BY and HAVING (cont.)

We need a multiple-row subquery which returns the salaries of employees in departments 10 and 20. The outer query will use a group function (MIN) so we need to GROUP the outer query BY department_id.

LAST_NAME	DEPT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
50	2500
60	4200
80	8600
110	8300
(null)	7000

GROUP BY and HAVING (cont.)

Here is the needed SQL statement:

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) <ANY
  (SELECT salary
   FROM employees
   WHERE department_id IN (10,20));
```

LAST_NAME	DEPT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
50	2500
60	4200
80	8600
110	8300
(null)	7000

GROUP BY and HAVING (cont.)

You can even have a GROUP BY clause in the subquery !

Which departments have a minimum salary that is greater than all of the minimum salaries of those departments whose department_ids are less than 50? Here is the SQL statement:

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) >ALL
   (SELECT MIN(salary)
    FROM employees
    WHERE department_id < 50
    GROUP BY department_id);
```

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000

DEPARTMENT_ID	MIN(SALARY)
80	8600
90	17000
110	8300
(null)	7000

One Last Point About Subqueries

Some subqueries may return a single row or multiple rows, depending on the data values in the rows.

If even the slightest possibility exists of returning multiple rows, make sure you write a multiple-row subquery.

```
SELECT first_name, last_name, job_id
FROM employees
WHERE job_id =
   (SELECT job_id
    FROM employees
    WHERE last_name = ‘Ernst’);
```

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG
Alexander	Hunold	IT_PROG
Diana	Lorentz	IT_PROG

One Last Point About Subqueries (cont.)

For example: who has the same job_id as Ernst? This single-row subquery works correctly because there is only one Ernst in the table.

```
SELECT first_name, last_name, job_id
FROM employees
WHERE job_id =
   (SELECT job_id
    FROM employees
    WHERE last_name = ‘Ernst’);
```

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG

But what if later, the business hires a new employee named Susan Ernst?

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG
Alexander	Hunold	IT_PROG
Diana	Lorentz	IT_PROG

One Last Point About Subqueries (cont.)

It would be better to write a multiple-row subquery.

The multiple-row subquery syntax will still work even if the subquery returns a single row.

If in doubt, write a multiple-row subquery !

```
SELECT first_name, last_name, job_id
FROM employees
WHERE job_id IN
      (SELECT job_id
        FROM employees
        WHERE last_name = 'Ernst');
```

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG
Susan	Ernst	SA_MAN

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG
Alexander	Hunold	IT_PROG
Diana	Lorentz	IT_PROG
Susan	Ernst	SA_MAN
Eleni	Zlotkey	SA_MAN

Summary

In this lesson, you should have learned how to:

- Correctly use the comparison operators IN, ANY, and ALL in multiple-row subqueries
- Construct and execute a multiple-row subquery in the WHERE clause or HAVING clause
- Describe what happens if a multiple-row subquery returns a null value
- Understand when multiple-row subqueries should be used, and when it is safe to use a single-row subquery
- Create a query using the EXISTS and NOT EXISTS operators to test for returned rows from the subquery