Database Programming with SQL

10-3
Multiple-Row Subqueries
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
Objectives

This lesson covers the following objectives:

• Distinguish between pair-wise and non-pair-wise subqueries

• 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?
- This example returns an error because more than one employee exists in department 20, 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);
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPT_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hartstein</td>
<td>20</td>
<td>13000</td>
</tr>
<tr>
<td>Fay</td>
<td>20</td>
<td>6000</td>
</tr>
</tbody>
</table>

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

• 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);
```

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.

• The multiple-row operators are:
  – IN,
  – ANY
  – ALL

• The NOT operator can be used with any of these three operators.
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 employees that were hired the same year as an employee in department 90.

```sql
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) IN
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whalen</td>
<td>17/Sep/1987</td>
</tr>
<tr>
<td>King</td>
<td>17/Jun/1987</td>
</tr>
<tr>
<td>Kochhar</td>
<td>21/Sep/1989</td>
</tr>
<tr>
<td>De Haan</td>
<td>13/Jan/1993</td>
</tr>
</tbody>
</table>
The inner query will return a list of the years that employees in department 90 were hired. The outer query will then return any employee that was hired the same year as any year in the inner query list.

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) IN
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whalen</td>
<td>17/Sep/1987</td>
</tr>
<tr>
<td>King</td>
<td>17/Jun/1987</td>
</tr>
<tr>
<td>Kochhar</td>
<td>21/Sep/1989</td>
</tr>
<tr>
<td>De Haan</td>
<td>13/Jan/1993</td>
</tr>
</tbody>
</table>
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.

• The example shown will return any employee whose year hired is less than at least one year hired of employees in department 90.

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ANY
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whalen</td>
<td>17/Sep/1987</td>
</tr>
<tr>
<td>King</td>
<td>17/Jun/1987</td>
</tr>
<tr>
<td>Kochhar</td>
<td>21/Sep/1989</td>
</tr>
<tr>
<td>Hunold</td>
<td>03/Jan/1990</td>
</tr>
<tr>
<td>Ernst</td>
<td>21/May/1991</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year Hired</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
</tr>
<tr>
<td>1989</td>
</tr>
<tr>
<td>1993</td>
</tr>
</tbody>
</table>
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 ALL operator compares a value to every value returned by the inner query.

• As no employee was hired before 1987, no rows are returned.

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ALL
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```

no data found
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);
```

<table>
<thead>
<tr>
<th>MANAGER_ID</th>
<th>LAST_NAME</th>
<th>EMPLOYEE_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>(null)</td>
<td>King</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>Kochhar</td>
<td>101</td>
</tr>
<tr>
<td>100</td>
<td>De Haan</td>
<td>102</td>
</tr>
<tr>
<td>102</td>
<td>Hunold</td>
<td>103</td>
</tr>
<tr>
<td>103</td>
<td>Mourgos</td>
<td>124</td>
</tr>
</tbody>
</table>

Result of subquery
NULL Values

• 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.

```sql
SELECT last_name, employee_id
FROM employees
WHERE employee_id <= ALL
    (SELECT manager_id
     FROM employees);
```

no data found
GROUP BY and HAVING

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

• 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?

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPT_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whalen</td>
<td>10</td>
<td>4400</td>
</tr>
<tr>
<td>Hartstein</td>
<td>20</td>
<td>13000</td>
</tr>
<tr>
<td>Fay</td>
<td>20</td>
<td>6000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>MIN(SALARY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4400</td>
</tr>
<tr>
<td>20</td>
<td>6000</td>
</tr>
<tr>
<td>50</td>
<td>2500</td>
</tr>
<tr>
<td>60</td>
<td>4200</td>
</tr>
<tr>
<td>80</td>
<td>8600</td>
</tr>
<tr>
<td>110</td>
<td>8300</td>
</tr>
<tr>
<td>(null)</td>
<td>7000</td>
</tr>
</tbody>
</table>
GROUP BY and HAVING

• 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.

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPT_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whalen</td>
<td>10</td>
<td>4400</td>
</tr>
<tr>
<td>Hartstein</td>
<td>20</td>
<td>13000</td>
</tr>
<tr>
<td>Fay</td>
<td>20</td>
<td>6000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>MIN(SALARY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4400</td>
</tr>
<tr>
<td>20</td>
<td>6000</td>
</tr>
<tr>
<td>50</td>
<td>2500</td>
</tr>
<tr>
<td>60</td>
<td>4200</td>
</tr>
<tr>
<td>80</td>
<td>8600</td>
</tr>
<tr>
<td>110</td>
<td>8300</td>
</tr>
<tr>
<td>(null)</td>
<td>7000</td>
</tr>
</tbody>
</table>
GROUP BY and HAVING

• Here is the 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))
ORDER BY department_id;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPT_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whalen</td>
<td>10</td>
<td>4400</td>
</tr>
<tr>
<td>Hartstein</td>
<td>20</td>
<td>13000</td>
</tr>
<tr>
<td>Fay</td>
<td>20</td>
<td>6000</td>
</tr>
</tbody>
</table>

Result of subquery

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>MIN(SALARY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4400</td>
</tr>
<tr>
<td>20</td>
<td>6000</td>
</tr>
<tr>
<td>50</td>
<td>2500</td>
</tr>
<tr>
<td>60</td>
<td>4200</td>
</tr>
<tr>
<td>80</td>
<td>8600</td>
</tr>
<tr>
<td>110</td>
<td>8300</td>
</tr>
<tr>
<td>(null)</td>
<td>7000</td>
</tr>
</tbody>
</table>
Multiple-Column Subqueries

- Subqueries can use one or more columns.
- If they use more than one column, they are called multiple-column subqueries.
- A multiple-column subquery can be either pair-wise comparisons or non-pair-wise comparisons.

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE (manager_id, department_id) IN
  (SELECT manager_id, department_id
   FROM employees
   WHERE employee_id IN (149, 174))
AND employee_id NOT IN (149, 174)
```

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>176</td>
<td>149</td>
<td>80</td>
</tr>
</tbody>
</table>
Multiple-Column Subqueries

- The example below shows a multiple-column pair-wise subquery with the subquery highlighted in red and the result in the table below.

- The query lists the employees whose manager and departments are the same as the manager and department of employees 149 or 174.

```sql
SELECT employee_id, manager_id, department_id
FROM employees
WHERE (manager_id, department_id) IN
    (SELECT manager_id, department_id
     FROM employees
     WHERE employee_id IN (149, 174))
AND employee_id NOT IN (149, 174)
```

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>176</td>
<td>149</td>
<td>80</td>
</tr>
</tbody>
</table>
Multiple-Column Subqueries

- A non-pair-wise multiple-column subquery also uses more than one column in the subquery, but it compares them one at a time, so the comparisons take place in different subqueries.

```
SELECT  employee_id, 
        manager_id, 
        department_id 
FROM    employees 
WHERE  manager_id IN 
    (SELECT  manager_id 
         FROM    employees 
         WHERE   employee_id 
                 IN 
                 (149,174)) 
AND    department_id IN 
    (SELECT  department_id 
         FROM    employees 
         WHERE   employee_id 
                 IN 
                 (149,174)) 
AND  employee_id NOT IN(149,174);
```
Multiple-Column Subqueries

- You will need to write one subquery per column you want to compare against when performing non-pair-wise multiple column subqueries.

- The example on the right shows a multiple-column non-pair-wise subquery with the subqueries highlighted in red.

```sql
SELECT employee_id, 
       manager_id, 
       department_id 
FROM employees 
WHERE manager_id IN 
  (SELECT manager_id 
   FROM employees 
   WHERE employee_id IN 
       (149,174)) 
AND department_id IN 
  (SELECT department_id 
   FROM employees 
   WHERE employee_id IN 
       (149,174)) 
AND employee_id NOT IN(149,174);
```

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>176</td>
<td>149</td>
<td>80</td>
</tr>
</tbody>
</table>
Multiple-Column Subqueries

- This query is listing the employees who have the same manager_id and department_id as employees 149 or 174.

```sql
SELECT employee_id,
       manager_id,
       department_id
FROM employees
WHERE manager_id IN
    (SELECT manager_id
     FROM employees
     WHERE employee_id IN (149, 174))
AND department_id IN
    (SELECT department_id
     FROM employees
     WHERE employee_id IN (149, 174))
AND employee_id NOT IN (149, 174);
```

<table>
<thead>
<tr>
<th>Result of 1st subquery</th>
<th>Result of 2nd subquery</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANAGER_ID</td>
<td>DEPARTMENT_ID</td>
</tr>
<tr>
<td>100</td>
<td>80</td>
</tr>
<tr>
<td>149</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>176</td>
<td>149</td>
<td>80</td>
</tr>
</tbody>
</table>

Result of 1st subquery

Result of 2nd subquery
EXISTS & NOT EXISTS in Subqueries

• EXISTS, and its opposite NOT EXISTS, are two clauses that can be used when testing for matches in subqueries.

• EXISTS tests for a TRUE, or a matching result in the subquery.

• To answer the question: "Which employees are not managers?"
  – You first have to ask, "Who are the managers?"
  – And then ask, "Who does NOT EXIST on the managers list?"
EXISTS & NOT EXISTS in Subqueries

• In this example, the subquery is selecting the employees that are managers.

• The outer query then returns the rows from the employee table that do NOT EXIST in the subquery.

```sql
SELECT last_name AS "Not a Manager"
FROM employees emp
WHERE NOT EXISTS
  (SELECT *
   FROM employees mgr
   WHERE mgr.manager_id = emp.employee_id);
```

<table>
<thead>
<tr>
<th>Not a Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abel</td>
</tr>
<tr>
<td>Davies</td>
</tr>
<tr>
<td>Ernst</td>
</tr>
<tr>
<td>Fay</td>
</tr>
<tr>
<td>Gietz</td>
</tr>
<tr>
<td>Grant</td>
</tr>
<tr>
<td>Lorentz</td>
</tr>
<tr>
<td>Matos</td>
</tr>
<tr>
<td>Rajs</td>
</tr>
<tr>
<td>Taylor</td>
</tr>
<tr>
<td>Vargas</td>
</tr>
<tr>
<td>Whalen</td>
</tr>
</tbody>
</table>
EXISTS & NOT EXISTS in Subqueries

- If the same query is executed with a NOT IN instead of NOT EXISTS, the result is very different.

- The result of this query suggests there are no employees who are also not managers, so all employees are managers, which we already know is not true.

```sql
SELECT last_name AS "Not a Manager"
FROM employees emp
WHERE emp.employee_id NOT IN
  (SELECT mgr.manager_id
   FROM employees mgr);
```

no data found
EXISTS & NOT EXISTS in Subqueries

• The cause of the strange result is due to the NULL value returned by the subquery.

• One of the rows in the employees table does not have a manager, and this makes the entire result wrong.

• Subqueries can return three values: TRUE, FALSE, and UNKNOWN.

• A NULL in the subquery result set will return an UNKNOWN, which Oracle cannot evaluate, so it doesn't.

```sql
SELECT last_name AS "Not a Manager"
FROM   employees emp
WHERE emp.employee_id NOT IN
  (SELECT mgr.manager_id
   FROM employees mgr);
```

no data found
EXISTS & NOT EXISTS in Subqueries

• BEWARE of NULLS in subqueries when using IN or NOT IN.
• If you are unsure whether or not a subquery will include a null value, either eliminate the null by using IS NOT NULL in a WHERE clause.
• For example: WHERE emp.manager_id IS NOT NULL or use NOT EXISTS to be safe.
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.

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

<table>
<thead>
<tr>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>JOB_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bruce</td>
<td>Ernst</td>
<td>IT_PROG</td>
</tr>
<tr>
<td>Alexander</td>
<td>Hunold</td>
<td>IT_PROG</td>
</tr>
<tr>
<td>Diana</td>
<td>Lorentz</td>
<td>IT_PROG</td>
</tr>
</tbody>
</table>

Result of subquery
One Last Point About Subqueries

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

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

<table>
<thead>
<tr>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>JOB_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bruce</td>
<td>Ernst</td>
<td>IT_PROG</td>
</tr>
<tr>
<td>Alexander</td>
<td>Hunold</td>
<td>IT_PROG</td>
</tr>
<tr>
<td>Diana</td>
<td>Lorentz</td>
<td>IT_PROG</td>
</tr>
</tbody>
</table>

Result of subquery
One Last Point About Subqueries

- 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!

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

<table>
<thead>
<tr>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>JOB_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bruce</td>
<td>Ernst</td>
<td>IT_PROG</td>
</tr>
<tr>
<td>Susan</td>
<td>Ernst</td>
<td>SA_MAN</td>
</tr>
<tr>
<td>Alexander</td>
<td>Hunold</td>
<td>IT_PROG</td>
</tr>
<tr>
<td>Diana</td>
<td>Lorentz</td>
<td>IT_PROG</td>
</tr>
<tr>
<td>Susan</td>
<td>Ernst</td>
<td>SA_MAN</td>
</tr>
<tr>
<td>Eleni</td>
<td>Zlotkey</td>
<td>SA_MAN</td>
</tr>
</tbody>
</table>

Result of subquery
There are 2 people with last name 'Ernst'
Terminology

Key terms used in this lesson included:

• EXIST and NOT EXIST
• Non-pair-wise multiple column subquery
• Pair-wise multiple column subquery
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
Summary

In this lesson, you should have learned how to:

- Distinguish between pair-wise and non-pair-wise subqueries
- Create a query using the EXISTS and NOT EXISTS operators to test for returned rows from the subquery