ORACLE®

Academy

Database Programming with SQL

10-4
Correlated Subqueries

ORACLE
Academy

Objectives

This lesson covers the following objectives:

- Identify when correlated subqueries are needed.
- Construct and execute correlated subqueries.
- Create a query using the EXISTS and NOT EXISTS operators to test for returned rows from the subquery
- Construct and execute named subqueries using the WITH clause.

Purpose

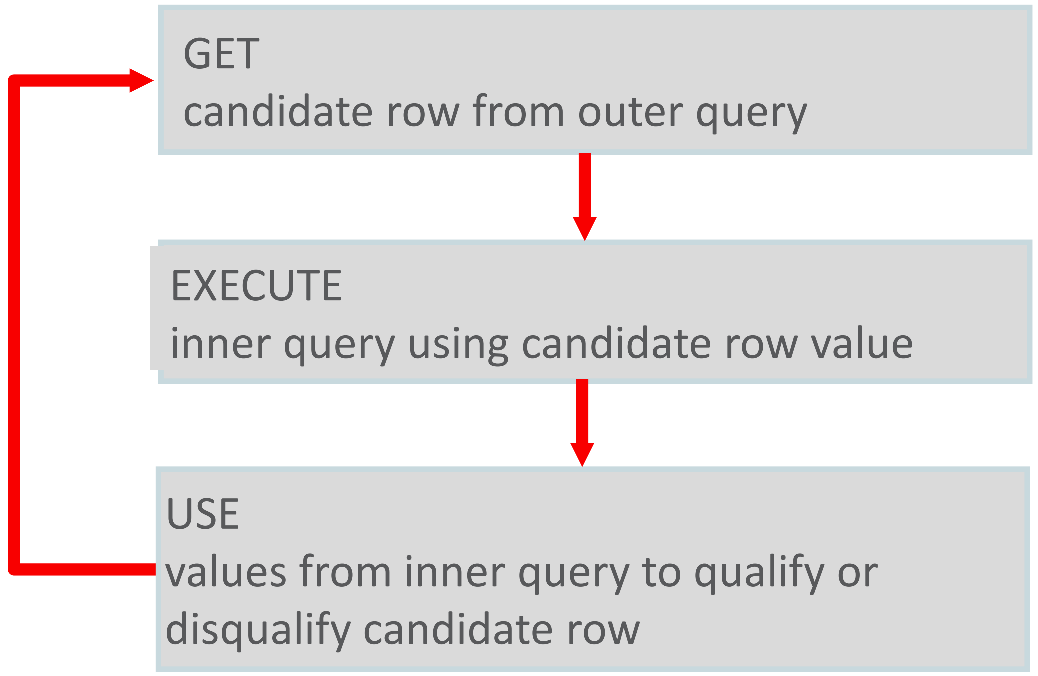
- Sometimes you have to answer more than one question in one sentence.
- Your friend might ask you if you have enough money for a cinema ticket, popcorn, and a drink.
- Before you can answer your friend, you need to know the prices of the ticket, the popcorn, and the drink.
- You also need to see how much money you have in your pocket.
- So actually, what seemed like an easy question, turns into four questions that you need answers to before you can say "Yes" or "No."

Purpose

- In business, you might get asked to produce a report of all employees earning more than the average salary for their departments.
- So here you first have to calculate the average salary per department, and then compare the salary for each employee to the average salary of that employee's department.

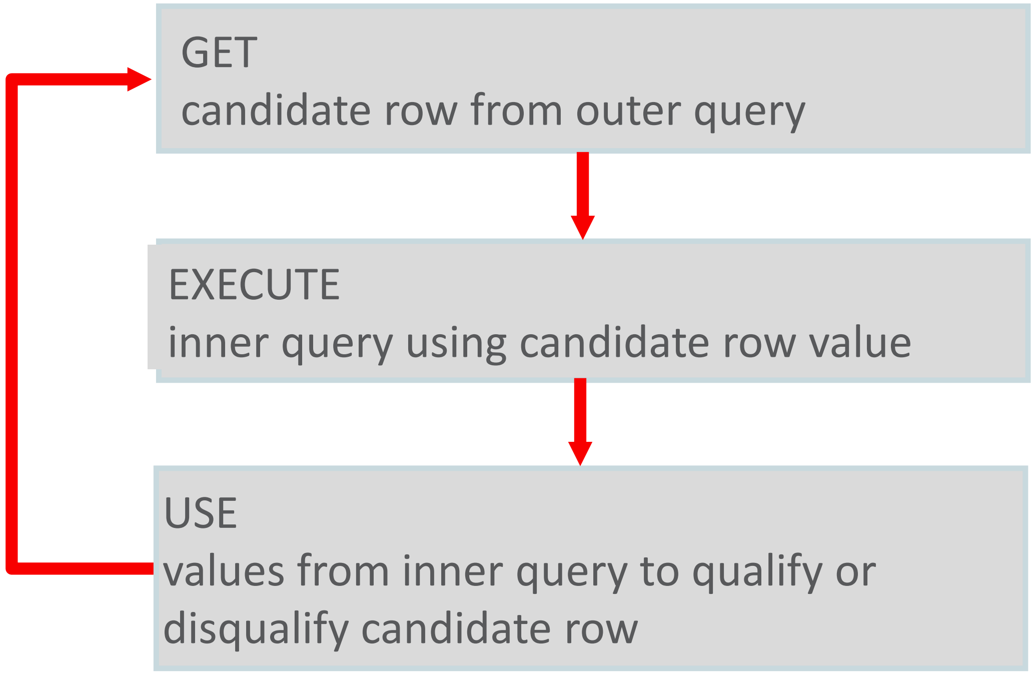
Correlated Subqueries

- The Oracle server performs a correlated subquery when the subquery references a column from a table referred to in the parent statement.


GET
candidate row from outer query
EXECUTE
inner query using candidate row value
USE
values from inner query to qualify or disqualify candidate row

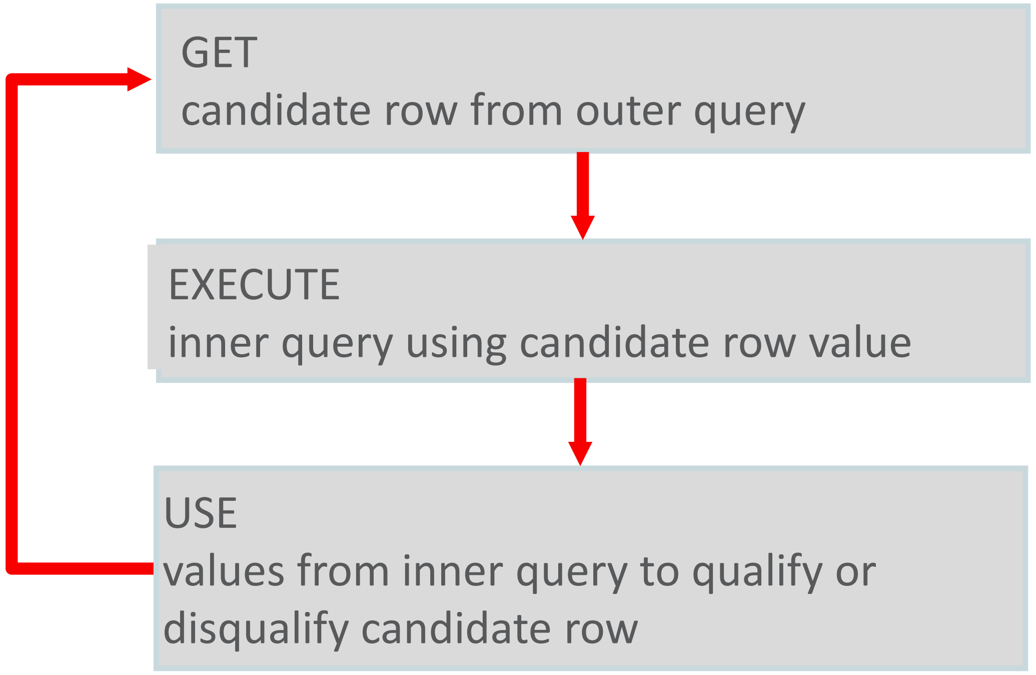
Correlated Subqueries

- A correlated subquery is evaluated once for each row processed by the parent statement.


GET
candidate row from outer query
EXECUTE
inner query using candidate row value
USE
values from inner query to qualify or disqualify candidate row

Correlated Subqueries

- The parent statement can be a SELECT, UPDATE, or DELETE statement.


GET
candidate row from outer query
EXECUTE
inner query using candidate row value
USE
values from inner query to qualify or disqualify candidate row

Correlated Subquery Example

- Whose salary is higher than the average salary of their department?
- To answer that question, we need to write a correlated subquery.
- Correlated subqueries are used for row-by-row processing.

```
SELECT o.first_name,
       o.last_name,
       o.salary
FROM employees o
WHERE o.salary >
   (SELECT AVG(i.salary)
    FROM employees i
    WHERE i.department_id =
          o.department_id);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Shelley	Higgins	12000
Eleni	Zlotkey	10500
Ellen	Abel	11000
Kevin	Mourgos	5800
Alexander	Hunold	9000
Michael	Hartstein	13000

Correlated Subquery Example

- Each subquery is executed once for every row of the outer query.
- With a normal subquery, the inner SELECT query runs first and executes once, returning values to be used by the outer query.

```
SELECT o.first_name,
       o.last_name,
       o.salary
FROM employees o
WHERE o.salary >
   (SELECT AVG(i.salary)
    FROM employees i
    WHERE i.department_id =
          o.department_id);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Shelley	Higgins	12000
Eleni	Zlotkey	10500
Ellen	Abel	11000
Kevin	Mourgos	5800
Alexander	Hunold	9000
Michael	Hartstein	13000

Correlated Subquery Example

- A correlated subquery, however, executes once for each row considered by the outer query.
- In other words, the inner query is driven by the outer query.
- The correlated subquery in this example is marked in red.

```
SELECT o.first_name,
       o.last_name,
       o.salary
FROM employees o
WHERE o.salary >
   (SELECT AVG(i.salary)
    FROM employees i
    WHERE i.department_id =
          o.department_id);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Shelley	Higgins	12000
Eleni	Zlotkey	10500
Ellen	Abel	11000
Kevin	Mourgos	5800
Alexander	Hunold	9000
Michael	Hartstein	13000

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.

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

Not a Manager
Whalen
Gietz
Abel
Taylor
Grant
Rajs
Davies
Matos
Vargas
Ernst
…

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.

```
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.

```
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.

WITH Clause

- If you have to write a very complex query with joins and aggregations used many times, you can write the different parts of the statement as query blocks and then use those same query blocks in a SELECT statement.
- Oracle allows you to write named subqueries in one single statement, as long as you start your statement with the keyword WITH.
- The WITH clause retrieves the results of one or more query blocks and stores those results for the user who runs the query.

WITH Clause

- The WITH clause improves performance.
- The WITH clause makes the query easier to read.
- The syntax for the WITH clause is as follows:

```
WITH subquery-name AS (subquery),
     subquery-name AS (subquery)
     SELECT  column-list
     FROM     {table | subquery-name | view}
     WHERE  condition is true;
```

ORACLE
Academy

WITH Clause

- Write the query for the following requirement:
  - Display a list of employee last names that are not managers.
- To construct this query, you will first need to get a list of manager_ids from the employee table, then return the names of the employees whose employee id is not on the managers list.
- We can create a named subquery using the WITH clause to retrieve the manager_id from the employees table, then the outer query will return the employees that do not appear on that list.

WITH Clause

```
WITH managers AS
   (SELECT DISTINCT manager_id
    FROM employees
    WHERE manager_id IS NOT NULL)

SELECT last_name AS "Not a manager"
FROM employees
WHERE employee_id NOT IN
   (SELECT *
    FROM managers);
```

Not a manager
Whalen
Gietz
Abel
Taylor
Grant
Rajs
Davies
Vargas
Ernst
…

Summary

In this lesson, you should have learned how to:

- Identify when correlated subqueries are needed.
- Construct and execute correlated subqueries.
- Create a query using the EXISTS and NOT EXISTS operators to test for returned rows from the subquery
- Construct and execute named subqueries using the WITH clause.

ORACLE®
Academy