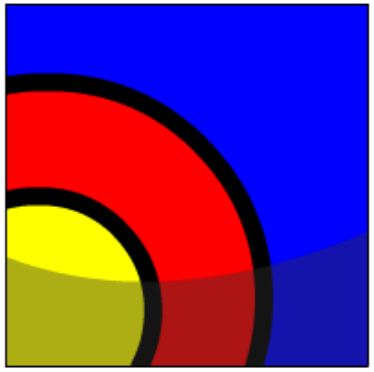
Correlated Subqueries

What Will I Learn?

In this lesson, you will learn:



- To identify when correlated subqueries are needed.
- How to write correlated subqueries.
- When and how to use the WITH clause.

Why Learn It?

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 Coca Cola. Before you can answer your friend, you need to know what the prices are of the ticket, the popcorn, and the Coca Cola. You also need to see how much money you have in your pocket. So actually, what seemed like an easy question, turns into 4 questions you need answers to, before you can say Yes or No.

In business, you might get asked to produce a report of all employees earning more than the average salary for their department. 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.

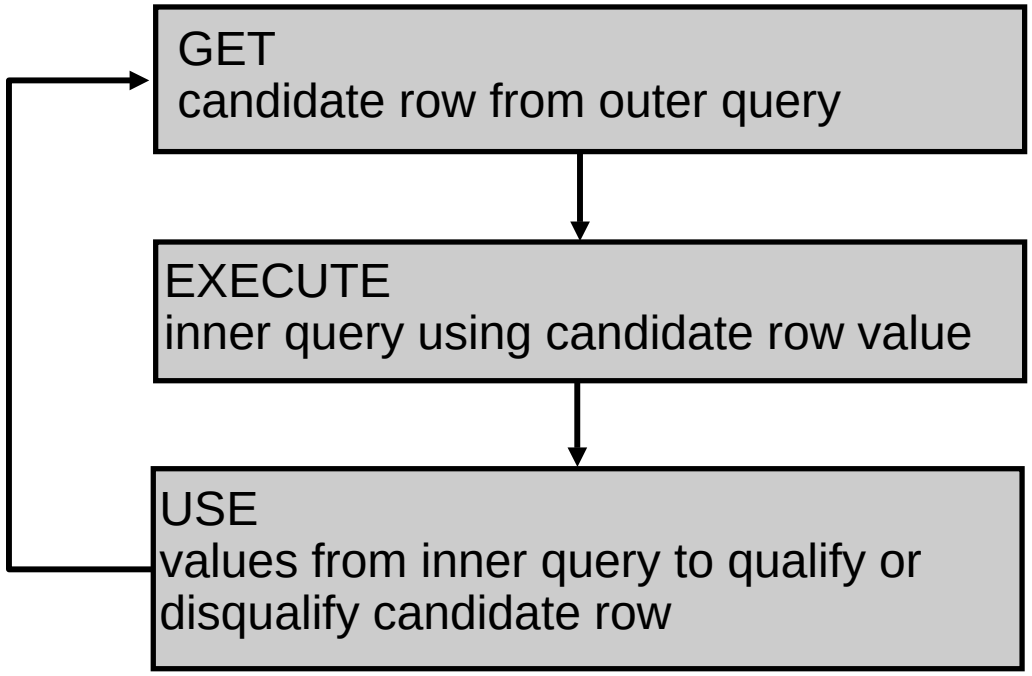
Tell Me / Show Me

CORRELATED SUBQUERIES

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

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

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



Tell Me / Show Me

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.

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 main query. A correlated subquery, however, executes once for each candidate row considered by the outer query. In other words, the inner query is driven by the outer query. The part that makes this example a correlated subquery 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
Alexander	Hunold	9000
Kevin	Mourgos	5800
Eleni	Zlotkey	10500
Ellen	Abel	11000
Michael	Hartstein	13000
Shelley	Higgins	12000

Tell Me / Show Me

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 block 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.
- The WITH clause improves performance.
- The WITH clause makes the query easier to read.

Tell Me / Show Me

WITH clause

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;
```

Tell Me / Show Me

WITH clause

Try to write the query for the following requirement:

Display the department name and total salaries for those departments whose total salary is greater than the average salary across departments.

To solve this query you will need to first get the total salaries per department, then the average salary per department and then you can list just the ones with total salary greater than the average of all departments.

Tell Me / Show Me

WITH clause

On the next slide you will se an example of a WITH clause. It starts by creating two subqueries, one called dept_costs and a second called avg_cost. Avg_cost uses the result of dept_cost and once these two subqueries have been run the actual query itself is executed.

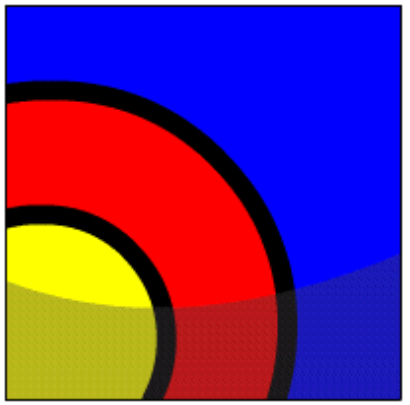
The query itself selects from both dept_cost and avg_cost. By creating the subqueries first, you do not need to create two temporary tables to hold the results of the SUM of salary per department and the AVG of department salaries.

Tell Me / Show Me

WITH clause

```
WITH
 dept_costs  AS (
   SELECT d.department_name, SUM(e.salary) AS dept_total
   FROM   employees e JOIN departments d
   ON     e.department_id = d.department_id
   GROUP BY d.department_name),
avg_cost    AS (
   SELECT SUM(dept_total)/COUNT(*) AS dept_avg
   FROM   dept_costs)
SELECT *
FROM   dept_costs
WHERE  dept_total >
       (SELECT dept_avg
        FROM avg_cost)
ORDER BY department_name;
```

Summary



In this lesson you have learned:

- To Identify when correlated subqueries are needed.
- How to write correlated subqueries.
- When and how to use the WITH clause.

Summary

Practice Guide

The link for the lesson practice guide can be found in the course resources in Section 0.