Database Programming

Correlated Subqueries

Objectives

This lesson covers the following objectives:

- Identify when correlated subqueries are needed.
- Construct and execute correlated subqueries.
- 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.

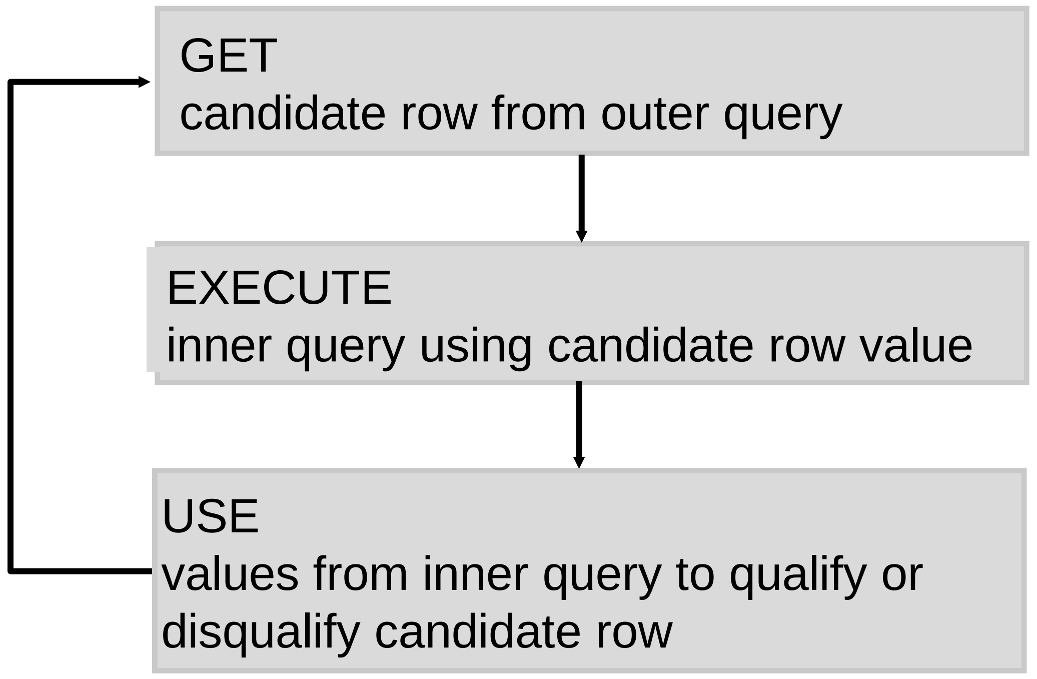
Before you can answer your friend, you need to know the prices of the ticket, the popcorn, and the drink. You also need to see how much money you have in your pocket. So actually, what seemed like an easy question, turns into four questions that you need answers to before you can say Yes or No.

Purpose (cont.)

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

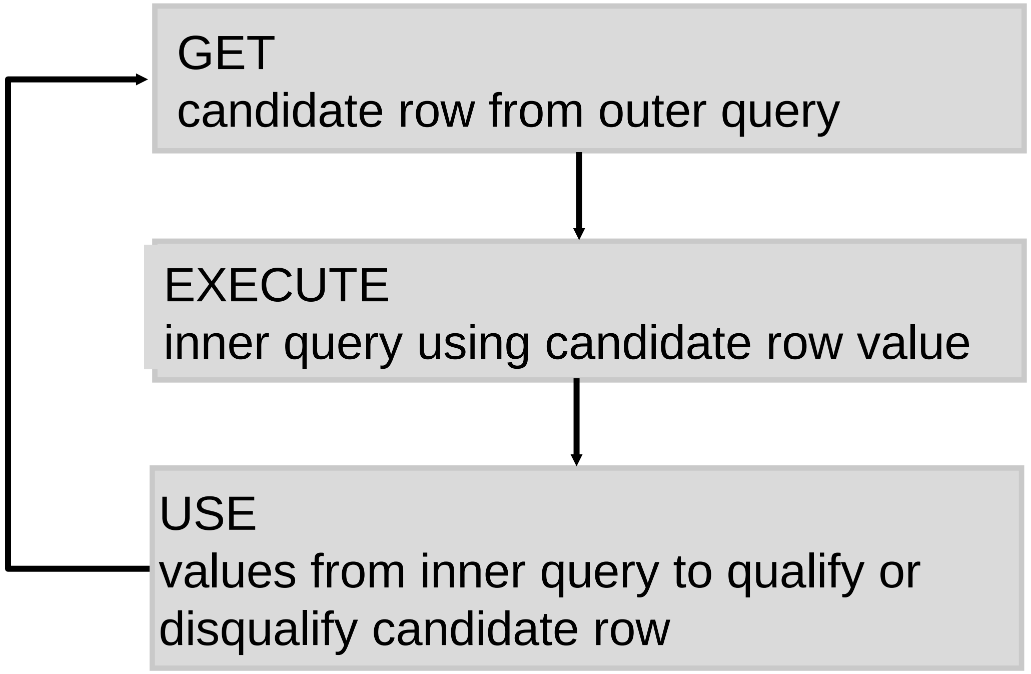
The Oracle server performs a correlated subquery when the subquery references a column from a table referred to in the parent statement.


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 (cont.)

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.


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
Alexander	Hunold	9000
Kevin	Mourgos	5800
Eleni	Zlotkey	10500
Ellen	Abel	11000
Michael	Hartstein	13000
Shelley	Higgins	12000

Correlated Subquery Example (cont.)

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
Alexander	Hunold	9000
Kevin	Mourgos	5800
Eleni	Zlotkey	10500
Ellen	Abel	11000
Michael	Hartstein	13000
Shelley	Higgins	12000

Correlated Subquery Example (cont.)

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
Alexander	Hunold	9000
Kevin	Mourgos	5800
Eleni	Zlotkey	10500
Ellen	Abel	11000
Michael	Hartstein	13000
Shelley	Higgins	12000

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.

WITH Clause (cont.)

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.

WITH Clause (cont.)

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;
```

WITH Clause (cont.)

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 construct this query, you will first need to get the total salary per department, then the average salary per department, and finally the departments with a total salary greater than the average of all departments.

WITH Clause (cont.)

Let’s examine an example of a WITH clause. Let’s start by creating two subqueries, one called dept_costs and a second called avg_cost. Avg_cost uses the result of dept_cost. Once these two subqueries have been run, the main 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.

WITH Clause (cont.)

```
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 should have learned how to:

- Identify when correlated subqueries are needed.
- Construct and execute correlated subqueries.
- Construct and execute named subqueries using the WITH clause.