ORACLE

16

Retrieving Data by Using Subqueries

Lesson Agenda

- Retrieving data by using a subquery as a source
- Writing a multiple-column subquery
- Using scalar subqueries in SQL
- Solving problems with correlated subqueries
- Using the `EXISTS` and `NOT EXISTS` operators
- Using the `WITH` clause

2

Retrieving Data by Using a Subquery as a Source

```
SELECT department_name, city
FROM   departments
NATURAL JOIN (SELECT l.location_id, l.city, l.country_id
              FROM   locations l
              JOIN   countries c
              ON(l.country_id = c.country_id)
              JOIN regions
              USING(region_id)
              WHERE region_name = 'Europe');
```

	DEPARTMENT_NAME	CITY
1	Human Resources	London
2	Sales	Oxford
3	Public Relations	Munich

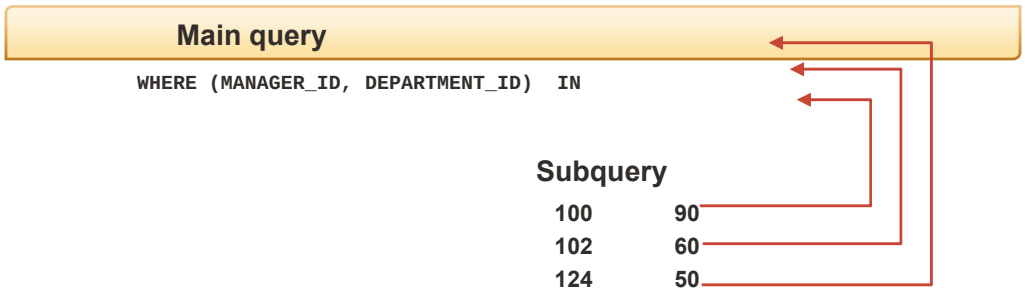
3

Lesson Agenda

- Retrieving data by using a subquery as a source
- Writing a multiple-column subquery
- Using scalar subqueries in SQL
- Solving problems with correlated subqueries
- Using the `EXISTS` and `NOT EXISTS` operators
- Using the `WITH` clause

4

Multiple-Column Subqueries



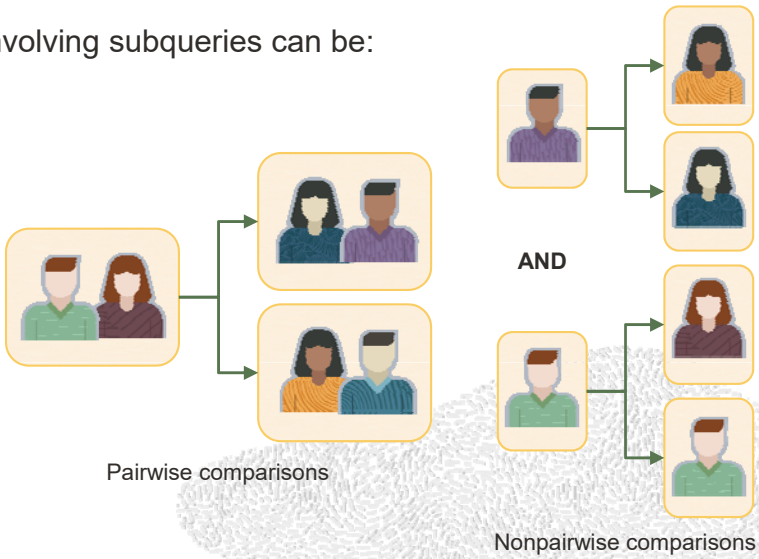
Each row of the main query is compared to values from a multiple-row and multiple-column subquery.

5

Column Comparisons

Multiple-column comparisons involving subqueries can be:

- Pairwise comparisons
- Nonpairwise comparisons



6

Pairwise Comparison Subquery

Display the details of the employees who are managed by the same manager and work in the same department as the employees with EMPLOYEE_ID 199 or 174.

```
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 (174, 199))
AND employee_id NOT IN (174,199);
```

	EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
1	141	124	50
2	142	124	50
3	143	124	50
4	144	124	50

...

7

Nonpairwise Comparison Subquery

Display the details of the employees who are managed by the same manager as the employees with EMPLOYEE_ID 174 or 141 and work in the same department as the employees with EMPLOYEE_ID 174 or 141.

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE manager_id IN
                  (SELECT manager_id
                   FROM employees
                   WHERE employee_id IN (174,141))
AND department_id IN
                  (SELECT department_id
                   FROM employees
                   WHERE employee_id IN (174,141))
AND employee_id NOT IN(174,141);
```

	EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
1	142	124	50
2	143	124	50

...

8

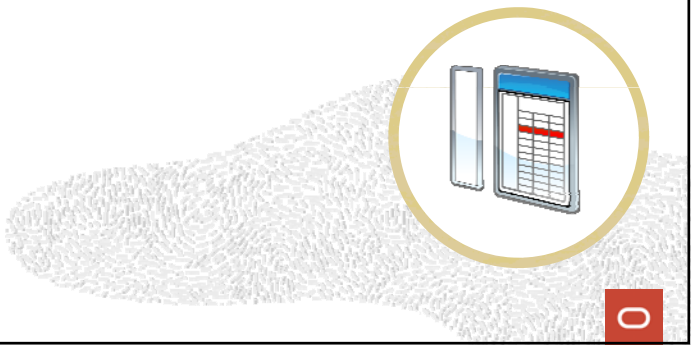
Lesson Agenda

- Retrieving data by using a subquery as a source
- Writing a multiple-column subquery
- Using scalar subqueries in SQL
- Solving problems with correlated subqueries
- Using the `EXISTS` and `NOT EXISTS` operators
- Using the `WITH` clause

9

Scalar Subquery Expressions

- A scalar subquery is a subquery that returns exactly one column value from one row.
- Scalar subqueries can be used in:
  - The condition and expression part of `DECODE` and `CASE`
  - All clauses of `SELECT` except `GROUP BY`
  - The `SET` clause and `WHERE` clause of an `UPDATE` statement



10

Scalar Subqueries: Examples

- Scalar subqueries in CASE expressions:

```
SELECT employee_id, last_name,
       (CASE
        WHEN department_id =
                (SELECT department_id
                 FROM departments
                 WHERE location_id = 1800)
         THEN 'Canada' ELSE 'USA' END) location
FROM   employees;
```

20

Scalar subqueries in the SELECT statement:

```
select department_id, department_name,
      (select count(*)
       from   employees e
       where  e.department_id = d.department_id) as emp_count
from   departments d;
```

11

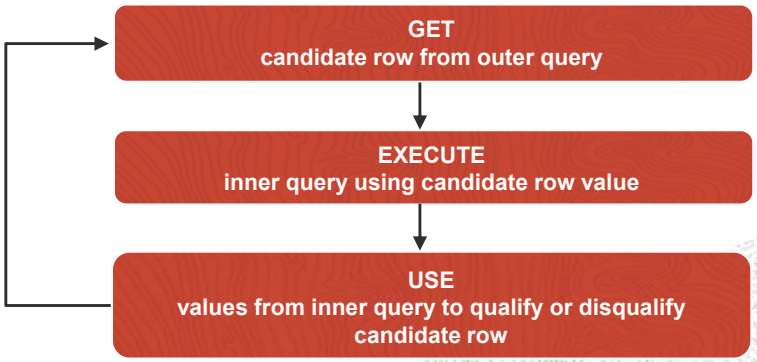
Lesson Agenda

- Retrieving data by using a subquery as a source
- Writing a multiple-column subquery
- Using scalar subqueries in SQL
- Solving problems with correlated subqueries
- Using the EXISTS and NOT EXISTS operators
- Using the WITH clause

12

Correlated Subqueries

Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query.



13

Correlated Subqueries

The subquery references a column from a table in the parent query.

Parent Query

```
SELECT column1, column2, ...
FROM   table1 Outer_table
WHERE  column1 operator
               (SELECT  column1, column2
                     FROM    table2
                     WHERE   expr1 =
   Outer_table.expr2);
```

Subquery

14

Using Correlated Subqueries: Example 1

Find all employees who earn more than the average salary in their department.

```
SELECT last_name, salary, department_id
FROM   employees outer_table
WHERE  salary >
       (SELECT AVG(salary)
            FROM   employees inner_table
            WHERE inner_table.department_id =
            outer_table.department_id);
```

	LAST_NAME	SALARY	DEPARTMENT_ID
1	King	24000	90
2	Hunold	9000	60
3	Ernst	6000	60
4	Greenberg	12008	100
5	Faviet	9000	100
6	Raphaely	11000	30

....

Each time a row from the outer query is processed, the inner query is evaluated.

15

Using Correlated Subqueries: Example 2

Display the details of the highest earning employees in each department.

```
SELECT department_id, employee_id, salary
FROM EMPLOYEES e
WHERE salary =
      (SELECT  MAX(DISTINCT salary)
       FROM EMPLOYEES
       WHERE e.department_id = department_id);
```

	DEPARTMENT_ID	EMPLOYEE_ID	SALARY
1	90	100	24000
2	60	103	9000
3	100	108	12008
4	30	114	11000

....

16

Lesson Agenda

- Retrieving data by using a subquery as a source
- Writing a multiple-column subquery
- Using scalar subqueries in SQL
- Solving problems with correlated subqueries
- Using the `EXISTS` and `NOT EXISTS` operators
- Using the `WITH` clause

17

Using the EXISTS Operator

- The `EXISTS` operator tests for existence of rows in the results set of the subquery.

18

Using the EXISTS Operator

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees outer
WHERE  EXISTS ( SELECT NULL
                FROM   employees
                WHERE  manager_id =
                       outer.employee_id);
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90
4	103	Hunold	IT_PROG	60
5	108	Greenberg	FI_MGR	100
6	114	Raphaely	PU_MAN	30
7	120	Weiss	ST_MAN	50
8	121	Fripp	ST_MAN	50

...

Find All Departments That Do Not Have Any Employees

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT NULL
                  FROM   employees
                  WHERE  department_id = d.department_id);
```

	DEPARTMENT_ID	DEPARTMENT_NAME
1	120	Treasury
2	130	Corporate Tax
3	140	Control And Credit
4	150	Shareholder Services
5	160	Benefits
6	170	Manufacturing
7	180	Construction
8	190	Contracting
9	200	Operations
10	210	IT Support

...

All Rows Fetched: 16

Lesson Agenda

- Retrieving data by using a subquery as a source
- Writing a multiple-column subquery
- Using scalar subqueries in SQL
- Solving problems with correlated subqueries
- Using the `EXISTS` and `NOT EXISTS` operators
- Using the `WITH` clause

21

WITH Clause

- Using the `WITH` clause, you can use the same query block in a `SELECT` statement when it occurs more than once within a complex query.
- The `WITH` clause retrieves the results of a query block and stores them in the user's temporary tablespace.
- The `WITH` clause can improve performance.

22

WITH Clause: Example

```
WITH CNT_DEPT AS
(
SELECT department_id,
 COUNT(*) NUM_EMP
FROM EMPLOYEES
GROUP BY department_id
)
SELECT employee_id,
 SALARY/NUM_EMP
FROM EMPLOYEES E
JOIN CNT_DEPT C
ON (e.department_id = c.department_id);
```

	EMPLOYEE_ID	SALARY/NUM_EMP
1	100	8000
2	103	1800
3	108	2001.3333333333333333333333333333333333
4	110	1366.6666666666666666666666666666666667
5	114	1833.3333333333333333333333333333333333
6	137	80
7	139	60

...

23

Summary

In this lesson, you should have learned how to:

- Write a multiple-column subquery
- Use scalar subqueries in SQL
- Solve problems with correlated subqueries
- Use the `EXISTS` and `NOT EXISTS` operators
- Use the `WITH` clause

24