

Oracle Database 12c: SQL Workshop I

Student Guide - Volume I

D80190GC10

Edition 1.0

August 2013

D83122

ORACLE®

Using Subqueries to Solve Queries

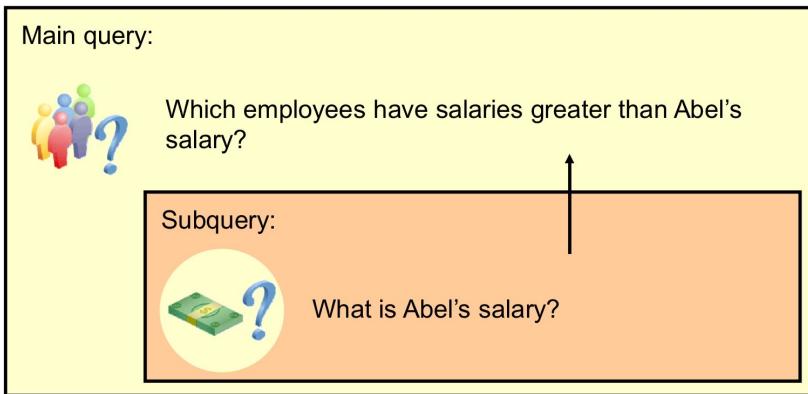


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Using a Subquery to Solve a Problem

Who has a salary greater than Abel's?



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suppose you want to write a query to find out who earns a salary greater than Abel's salary. To solve this problem, you need *two* queries: one to find how much Abel earns, and a second query to find who earns more than that amount.

You can solve this problem by combining the two queries, placing one query *inside* the other query.

The inner query (or *subquery*) returns a value that is used by the outer query (or *main query*). The execution plan of the query depends on the optimizer's decision on the structure of the subquery.

Subquery Syntax

- The subquery (inner query) executes *before* the main query (outer query).
- The result of the subquery is used by the main query.

```
SELECT      select_list
FROM        table
WHERE       expr operator
            (SELECT      select_list
             FROM       table);
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A subquery is a `SELECT` statement that is embedded in the clause of another `SELECT` statement. You can build powerful statements out of simple ones by using subqueries. They can be very useful when you need to select rows from a table with a condition that depends on the data in the table itself.

You can place the subquery in a number of SQL clauses, including the following:

- WHERE clause
- HAVING clause
- FROM clause

In the syntax:

`operator` includes a comparison condition such as `>`, `=`, or `IN`

Note: Comparison conditions fall into two classes: single-row operators (`>`, `=`, `>=`, `<`, `<>`, `<=`) and multiple-row operators (`IN`, `ANY`, `ALL`, `EXISTS`).

The subquery is often referred to as a nested `SELECT`, sub-`SELECT`, or inner `SELECT` statement. The subquery generally executes first, and its output is used to complete the query condition for the main (or outer) query.

Using a Subquery

```
SELECT last_name, salary  
FROM   employees  
WHERE  salary > 11000  
       (SELECT salary  
        FROM   employees  
        WHERE  last_name = 'Abel');
```

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12008

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the slide, the inner query determines the salary of employee Abel. The outer query takes the result of the inner query and uses this result to display all the employees who earn more than employee Abel.

Types of Subqueries

- Single-row subquery



- Multiple-row subquery



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- **Single-row subqueries:** Queries that return only one row from the inner SELECT statement
- **Multiple-row subqueries:** Queries that return more than one row from the inner SELECT statement

Note: There are also multiple-column subqueries, which are queries that return more than one column from the inner SELECT statement. These are covered in the *Oracle Database: SQL Workshop II* course.

Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A single-row subquery is one that returns one row from the inner SELECT statement. This type of subquery uses a single-row operator. The slide gives a list of single-row operators.

Example

Display the employees whose job ID is the same as that of employee 141:

```
SELECT last_name, job_id
  FROM employees
 WHERE job_id =
       (SELECT job_id
        FROM   employees
       WHERE  employee_id = 141);
```

#	LAST_NAME	#	JOB_ID
1	Rajs		ST_CLERK
2	Davies		ST_CLERK
3	Matos		ST_CLERK
4	Vargas		ST_CLERK

Executing Single-Row Subqueries

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = SA_REP
       (SELECT job_id
        FROM   employees
        WHERE  last_name = 'Taylor')
AND    salary > 8600
       (SELECT salary
        FROM   employees
        WHERE  last_name = 'Taylor');
```

LAST_NAME	JOB_ID	SALARY
Abel	SA_REP	11000

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A SELECT statement can be considered as a query block. The example in the slide displays employees who do the same job as “Taylor,” but earn more salary than him.

The example consists of three query blocks: the outer query and two inner queries. The inner query blocks are executed first, producing the query results `SA_REP` and `8600`, respectively. The outer query block is then processed and uses the values that were returned by the inner queries to complete its search conditions.

Both inner queries return single values (`SA_REP` and `8600`, respectively), so this SQL statement is called a single-row subquery.

Note: The outer and inner queries can get data from different tables.

Using Group Functions in a Subquery

```
SELECT last_name, job_id, salary  
FROM   employees  
WHERE  salary = 2500  
       (SELECT MIN(salary)  
        FROM   employees);
```

#	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can display data from a main query by using a group function in a subquery to return a single row. The subquery is in parentheses and is placed after the comparison condition.

The example in the slide displays the employee last name, job ID, and salary of all employees whose salary is equal to the minimum salary. The MIN group function returns a single value (2500) to the outer query.

HAVING Clause with Subqueries

- The Oracle server executes the subqueries first.
- The Oracle server returns results into the HAVING clause of the main query.

```
SELECT      department_id, MIN(salary)
FROM        employees
GROUP BY   department_id
HAVING     MIN(salary) > 2500
          (SELECT MIN(salary)
           FROM   employees
           WHERE  department_id = 50);
```

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	90	17000
3	20	6000
4	110	8300
5	80	8600
6	60	4200
7	10	4400

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can use subqueries not only in the WHERE clause, but also in the HAVING clause. The Oracle server executes the subquery and the results are returned into the HAVING clause of the main query.

The SQL statement in the slide displays all the departments that have a minimum salary greater than that of department 50.

Example

Find the job with the lowest average salary.

```
SELECT      job_id, AVG(salary)
FROM        employees
GROUP BY   job_id
HAVING     AVG(salary) = (SELECT      MIN(AVG(salary))
                           FROM        employees
                           GROUP BY   job_id);
```

	JOB_ID	AVG(SALARY)
1	ST_CLERK	2925

What Is Wrong with This Statement?

```
SELECT employee_id, last_name  
FROM employees  
WHERE salary =  
      (SELECT MIN(salary)  
       FROM employees  
       GROUP BY department_id);
```

ORA-01427: single-row subquery returns more than one row
01427. 00000 - "single-row subquery returns more than one row"
*Cause:
*Action:

Single-row operator with
multiple-row subquery

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A common error with subqueries occurs when more than one row is returned for a single-row subquery.

In the SQL statement in the slide, the subquery contains a GROUP BY clause, which implies that the subquery will return multiple rows, one for each group that it finds. In this case, the results of the subquery are 4400, 6000, 2500, 4200, 7000, 17000, and 8300.

The outer query takes those results and uses them in its WHERE clause. The WHERE clause contains an equal (=) operator, a single-row comparison operator that expects only one value. The = operator cannot accept more than one value from the subquery and, therefore, generates the error.

To correct this error, change the = operator to IN.

No Rows Returned by the Inner Query

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



Subquery returns no rows because there is no employee named "Haas."

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Another common problem with subqueries occurs when no rows are returned by the inner query.

In the SQL statement in the slide, the subquery contains a `WHERE` clause. Presumably, the intention is to find the employee whose name is Haas. The statement is correct, but selects no rows when executed because there is no employee named Haas. Therefore, the subquery returns no rows.

The outer query takes the results of the subquery (null) and uses these results in its `WHERE` clause. The outer query finds no employee with a job ID equal to `NULL`, and so returns no rows. If a job existed with a value of `null`, the row is not returned because comparison of two `null` values yields a `null`; therefore, the `WHERE` condition is not true.

Multiple-Row Subqueries

- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
IN	Equal to any member in the list
ANY	Must be preceded by =, !=, >, <, <=, >=. Returns TRUE if at least one element exists in the result-set of the Subquery for which the relation is TRUE.
ALL	Must be preceded by =, !=, >, <, <=, >=. Returns TRUE if the relation is TRUE for all elements in the result set of the Subquery.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Subqueries that return more than one row are called multiple-row subqueries. You use a multiple-row operator, instead of a single-row operator, with a multiple-row subquery. The multiple-row operator expects one or more values:

```
SELECT last_name, salary, department_id
  FROM employees
 WHERE salary IN (SELECT MIN(salary)
                   FROM employees
                  GROUP BY department_id);
```

Example

Find the employees who earn the same salary as the minimum salary for each department. The inner query is executed first, producing a query result. The main query block is then processed and uses the values that were returned by the inner query to complete its search condition. In fact, the main query appears to the Oracle server as follows:

```
SELECT last_name, salary, department_id
  FROM employees
 WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300,
                  8600, 17000);
```

Using the ANY Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
  FROM employees      9000, 6000, 4200
 WHERE salary < ANY
          (SELECT salary
             FROM employees
            WHERE job_id = 'IT_PROG')
AND     job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The ANY operator (and its synonym, the SOME operator) compares a value to *each* value returned by a subquery. The slide example displays employees who are not IT programmers and whose salary is less than that of any IT programmer. The maximum salary that a programmer earns is \$9,000.

- <ANY means less than the maximum.
- >ANY means more than the minimum.
- =ANY is equivalent to IN.

Using the ALL Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees      9000, 6000, 4200
WHERE  salary < ALL
       (SELECT salary
        FROM   employees
        WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

#	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141 Rajs	ST_CLERK	3500	
2	142 Davies	ST_CLERK	3100	
3	143 Matos	ST_CLERK	2600	
4	144 Vargas	ST_CLERK	2500	

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The ALL operator compares a value to every value returned by a subquery. The example in the slide displays employees whose salary is less than the salary of all employees with a job ID of IT_PROG and whose job is not IT_PROG.

>ALL means more than the maximum and <ALL means less than the minimum.

The NOT operator can be used with IN, ANY, and ALL operators.

Using the EXISTS Operator

```
SELECT employee_id,salary,last_name FROM employees M
WHERE EXISTS
(SELECT employee_id FROM employees W
 WHERE (W.manager_id=M.employee_id) AND W.salary > 10000);
```

	EMPLOYEE_ID	SALARY	LAST_NAME
1	100	24000	King
2	149	10500	Zlotkey
3	101	17000	Kochhar

```
SELECT * FROM departments
WHERE NOT EXISTS
(SELECT * FROM employees
 WHERE employees.department_id=departments.department_id);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	190	Contracting	(null)	1700

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The EXISTS operator is used in queries where the query result depends on whether or not certain rows exist in a table. It evaluates to TRUE if the subquery returns at least one row.

The first example in the slide displays managers in the EMPLOYEES table who earns a salary more than 10000. For each row in EMPLOYEES table, the condition is checked whether there exists a manager_id who earns a salary more than 10000.

The second example in the slide displays departments that have no employees. For each row in the DEPARTMENTS table, the condition is checked whether there exists a row in the EMPLOYEES table that has the same department ID. In case no such row exists, the condition is satisfied for the row under consideration and it is selected. If there exists a corresponding row in the EMPLOYEES table, the row is not selected.

Null Values in a Subquery

```
SELECT emp.last_name
  FROM employees emp
 WHERE emp.employee_id NOT IN
       (SELECT mgr.manager_id
        FROM   employees mgr);
```



Subquery returns no rows because one of the values returned by a subquery is Null.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SQL statement in the slide attempts to display all the employees who do not have any subordinates. Logically, this SQL statement should have returned 12 rows. However, the SQL statement does not return any rows. One of the values returned by the inner query is a null value and, therefore, the entire query returns no rows.

The reason is that all conditions that compare a null value result in a null. So whenever null values are likely to be part of the results set of a subquery, do not use the NOT IN operator. The NOT IN operator is equivalent to <> ALL.

Notice that the null value as part of the results set of a subquery is not a problem if you use the IN operator. The IN operator is equivalent to =ANY. For example, to display the employees who have subordinates, use the following SQL statement:

```
SELECT emp.last_name
  FROM employees emp
 WHERE emp.employee_id IN
       (SELECT mgr.manager_id
        FROM   employees mgr);
```