Cognizant | Academy

Business Focused. Learner Centric.

Oracle

Subqueries

Enabling Objectives

After completing this chapter, in the next 20 minutes you will be able to :

- Describe at least two subqueries with examples in Oracle.

Key Topics

- Using Subqueries.
- Using Correlated Subqueries.

Subqueries

Subquery

- Subquery allows you to break down a problem into individual components and solve it in a nested way.
- Subqueries are also referred as sub-select or nested select.
- Subqueries can be used in Select , Insert, Update, Delete and Set statements.

Base tables

- Employees:

phone_no	city	e_code	e_name	e_salary	m_id	join_date	dept_id
	Denver	12	Smith	13000	34	11-Jan-18	10
3456	Chicago	34	John	30000		01-Nov-17	10
	Denver	45	Samuel	20000	34	12-Dec-17	20
6543	Chicago	8	Sam	13000	45	10-Jan-18	

- Department:

Dept_id	Dept_name
10	Sales
20	Administration

Subqueries

- A Subquery is a select statement nested in various clauses of an SQL statement.
- It allows you to use the output from one query as the input of another query
- Syntax

```
SELECT column_list FROM table_name
WHERE test_expression operator( SELECT ……)
```

- Subqueries can be nested several level deep.
- Subqueries may returns a single value or multiple value

Example:

```
select e_name from employees where e_salary =
                (select max( e_salary) from employees  );
```

- Displays the names of the employee drawing the maximum salary.

```
select e_code, e_name from employees where dept_id =
                ( select dept_id from department where dept_name =
'Administration');
```

- Displays the names of all the employees assigned to Administration department

Operators used

Operators used in single row Subquery

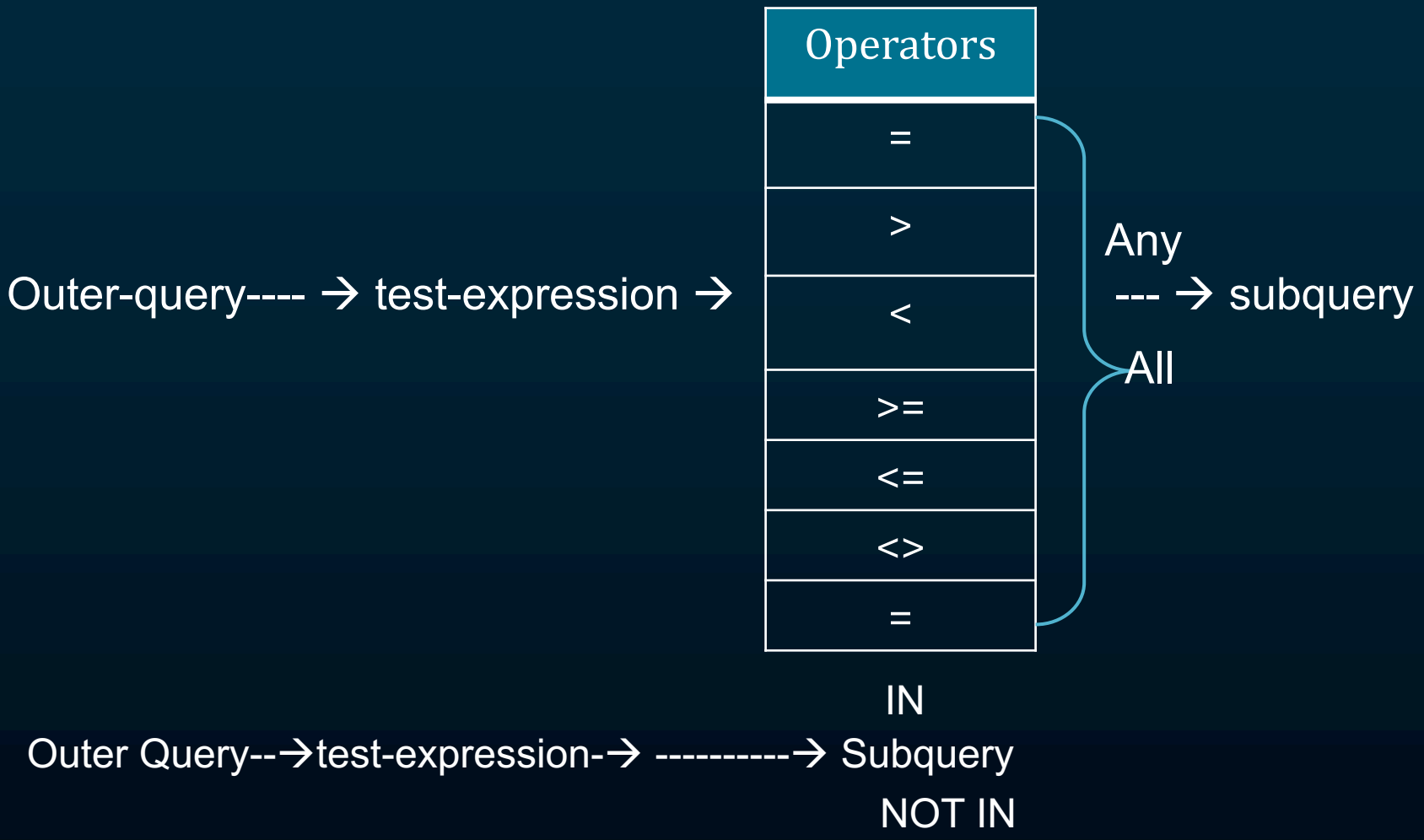
Outer-query---- → test-expression →

Operators
=
>
<
>=
<=
<>
=

--- → subquery

Operators used

Operators used in multiple row Subqueries



In Where Clause

- Multiple row Subqueries

```
Select e_name from employees where e_salary  > all
        ( select e_salary from employees where dept_id = 20);
```

Result : Displays the names of employees earning more than all the employees from department 20

```
Select e_name from employees where join_date in
        ( select join_date from employees where e_name like 'Sam%') ;
```

Result: Displays the names of the employees who have joined the organization on the same day as employee beginning with 'Sam'

Nested Subquery

- Subqueries with more than two levels

```
Select e_name from employees where e_salary  > all
        ( select e_salary from employees where dept_id =
                ( select dept_id from department where dept_name='Sales'));
```

Result: Displays the names of employees earning more than all the ` employees from Sales department

- Subqueries can also be placed in the having clause.

Multiple Column Subquery

- Subquery returning multiple columns

```
Select e_code, e_name from employee where (e_salary,
dept_id)  in  ( select e_salary, dept_id from employee where
e_code in( 12 , 8))  and e_code not in( 12,8);
```

- Subquery result will be used by outer query for final result

```
Select e_code, e_name from employee where (e_salary,
dept_id) in
```

e_salary	dept_id
13000	10
13000	

```
and e_code not in( 12,8);
```

Correlated - Subquery

Correlated Subquery

- Used when each row in subquery needs to match.
- Subquery references one or more columns in the outer query.

```
Select e_code, e_salary, dept_id from employee e_outer where
e_salary = ( select min (e_salary) from employee e_inner where
e_inner.dept_id = e_outer.dept_id);
```

Result

e_code	e_salary	dept_id
12	13000	10
45	20000	20

- The query retrieves details of employees who get the minimum salary in their department.

EXISTS & NOT EXISTS

- Exists: Conditionally derive a result only when a subquery returns at least one row.
- Not Exits: Other way round, only when no record found from subquery.
- Can be used in Select, Insert, Update and Delete statements.
- Syntax: WHERE [Not] EXISTS (subquery);
- Select **e_code**, **e_name** from **employee e_outer** where **exists** (select **e_code** from **employee e_inner** where **e_inner**. **m_id = e_outer.e_code**);

Result

e_code	e_name
34	John
45	Samuel

EXISTS & NOT EXISTS

```
Select e_code, e_name from employee e_outer where not exists
( select e_code from employee e_inner where e_inner. m_id =
        e_outer.e_code);
```

Result :

e_code	e_name
12	Smith
8	Sam

- Retrieves records of those employees who have no managers reporting to them

EXISTS & NOT EXISTS

- A EXISTS or NOT EXTSTS checks for existence of rows. The actual values returned by Subquery is not utilized. Thus constant values can be used in Subquery.

```
Select e_code, e_name from employee e_outer where not exists
        ( select 'X'  from employee e_inner where e_inner. m_id =
                e_outer.e_code);
```

Example

- Scenario to take a backup of the existing column values.
- Alter table **employee** add **d_name varchar(10);**
- Update **employee e_outer**
  set **d_name** = (select **dept_name** from **department** where **dept_id = e_outer.dept_id**);

Result : The newly added column, department name is updated by the single correlated update statement.

Correlated Delete

Example

- delete from employee e_outer where e_salary >

  (select avg (e_salary) from employee where employee.dept_id =

  e_outer.dept_id);

Result : Deletes records of employee having salary greater than average salary in their department.

Practice Check

Assuming employee tables contains the personnel details of all the employees. Explain the result of the following query:

```
SELECT first_name, last_name FROM employee WHERE salary =
(SELECT MIN( salary) FROM employee)
```

Practice Check

With the two table listed below Product & Orderitem where ID is primary key in Product table and productid is foreign key in Orderitem table.

Write a query to list products with order quantities greater than 50

PRODUCT
ID
Product Name
Supplier ID
UnitPrice
Package

ORDERITEM
ID
OrderId
ProductId
UnitPrice
Quantity

RECAP

In this chapter we have learnt how to:

- Use Subqueries.
- Use Correlated subqueries.

You have successfully completed -

Subqueries