

5. SQL SELECT Statement with subselect

A SUBSELECT (also called subquery) – a SELECT statement that is nested within another SQL statement (often within another SELECT). Subselect is enclosed with parentheses. Basically, a subquery can be used anywhere an expression can be used.

- In the **SELECT** statement, a subselect can be placed in the following clauses:
 - **WHERE, HAVING** (as a part of the logic condition)
 - **SELECT** (such a subselect must always return a single value for any row of the external SELECT statement)
 - **FROM** (as a data source)
 - **ORDER BY** (such a subselect must return a single value)

5.1 Non-correlated subselect

- a) A non-correlated subselect is executed independently of the SELECT statement, in which it is nested (called the outer SELECT); the result of the subselect is then returned to the outer SELECT (there can be several levels of nesting of subselects); such a subselect is a standalone query.
- b) Usually, a non-correlated subselect returns a table with one column (the exception is when the subselect is used with the operator EXISTS, or is put in the FROM clause – in such cases the resulting table may have many columns).

c) **non-correlated subselect in the clauses WHERE and HAVING**

SELECT list of expressions **FROM** list of tables

WHERE expression1 **operator** (**SELECT** expression2 **FROM** ...)

SELECT list of expressions **FROM** list of tables

GROUP BY ...

HAVING expression1 **operator** (**SELECT** expression2 **FROM** ...)

▪ **operators:**

- **operators of comparison** =, <, >, <>, <=, >=, !=

can be used, if the subselect returns a single value

- **operators of comparison** =, <, >, <>, <=, >=, !=

together with one of the keywords **ANY** or **ALL** can be used, if the subselect returns (or may return) more than one value

ANY – the condition returns true, if the value of expression1 is in the given relations with **at least one** value returned by the subselect

ALL – the condition returns true, if the value of expression1 is in the given relations with **all** values returned by the subselect

- **IN / NOT IN** may be used when the subselect can return many values
- **BETWEEN / NOT BETWEEN** for subselects returning single value

```
SELECT full_name, salary, dept_no  
FROM employee  
WHERE salary=(SELECT MAX(salary)  
                FROM employee)
```

```
SELECT *  
FROM employee  
WHERE job_country ='USA' AND  
       salary<=ALL(SELECT salary  
                   FROM employee  
                   WHERE job_country ='USA')
```

```
SELECT *  
FROM employee  
WHERE job_country IN(SELECT country FROM country  
                     WHERE currency  
                     LIKE '%dollar%')  
  
SELECT COUNT(*), dept_no  
FROM employee  
GROUP BY dept_no  
HAVING COUNT(*)>=ALL(SELECT COUNT(*)  
                     FROM employee  
                     GROUP BY dept_no)
```

5.2 Correlated subselect is not a standalone query; both the inner query and the outer query are **interdependent**; the inner SELECT needs a value – a parameter - from the outer SELECT statement in order to be processed; the inner query depends on the outer query before it can be processed;

a) for every row processed by the inner query, the outer query is processed as well; there is a condition, called the condition of correlating, in the inner query, to determine how the inner query and the outer query are correlated; this condition is usually put in the WHERE clause of the subselect;

b) Often aliases are needed, to distinguish between the tables used in the outer and inner query;

c) Correlated subselect in clauses WHERE and HAVING:

- The result of subselect is used to determine the final set of rows returned by the outer SELECT;
- The following operator is often used with correlated subselects:
 - **EXISTS** – returns true, if subselect returns **at least one row**
 - the subselect used together with EXISTS may return the whole row of data

```
SELECT full_name, salary, dept_no
FROM employee e1
WHERE salary=
      (SELECT MAX(salary) FROM employee e2
       WHERE e1.dept_no=e2.dept_no)
```

```
SELECT COUNT(*) FROM employee e
WHERE EXISTS(SELECT * FROM department d
             WHERE mngr_no IS NULL
             AND d.dept_no=e.dept_no)
```

5.3 subselect in the SELECT clause

such a subselect must return a single value; it is often a correlated subselect

```
SELECT full_name, salary,  
       (SELECT AVG(salary)  
        FROM employee)  
FROM employee  
WHERE dept_no='600'
```

```
SELECT full_name as employer,  
       (SELECT full_name FROM employee  
        WHERE emp_no=(SELECT mngr_no  
                        FROM department d  
                        WHERE d.dept_no=e.dept_no)) as boss  
FROM employee e  
WHERE e.full_name='Nelson, Robert'
```

5.4 subselect in the FROM clause

such a subselect returns a set of rows, which is used as a data source for the outer query; in order to refer to columns or expressions from the inner query, one should give aliases to them, or give an alias to the „table” returned by the subselect

```
SELECT amount, dept_no  
FROM (SELECT COUNT(*) as amount, dept_no  
      FROM employee GROUP BY dept_no)  
WHERE amount <=3
```

```
SELECT MAX(amount)  
FROM (SELECT COUNT(*) as amount  
      FROM employee  
      GROUP BY dept_no)
```

5.5 subselect in the ORDER BY clause

Such a subselect let us to sort the result set of the outer query according to the value returned by the subselect; must return a single value for each row of the outer SELECT

```
SELECT dept_no, full_name, (SELECT count(*) FROM employee e2 WHERE e1.dept_no=e2.dept_no)
FROM employee e1
```

```
ORDER BY (SELECT count(*) FROM employee e2 WHERE e1.dept_no=e2.dept_no)
```

the same query as above can be rewritten as follows, using column position in ORDER BY

```
SELECT dept_no, full_name, (SELECT count(*) FROM employee e2 WHERE e1.dept_no=e2.dept_no)
FROM employee e1
```

```
ORDER BY 3
```

or as below

```
SELECT * FROM employee e1
```

```
ORDER BY (SELECT COUNT(*) FROM employee e2
          WHERE e1.dept_no=e2.dept_no)
```