Chapter 7. 서브 쿼리 I

서브 쿼리(Subquery)란 SQL 문장에 포함되어 있는 또 하나의 별도 SQL 문장이다. 이 장에서는 서브 쿼리의 정의와 서브 쿼리로 수행 할 수 있는 작업을 살펴보고, 서브쿼리의 종류와 단일행 및 복수행 서브쿼리를 작성하는 방법을 설명한다.

서브 쿼리의 정의

서브 쿼리는 SQL 문장내에 포함된 쿼리 문장으로서 여러 번의 쿼리 문장을 수행해야 얻을 수 있는 결과를 하나의 중첩된 SQL 문장으로 쉽게 얻을 수 있도록 해준다. 예를 들어, 사원 테이블에서 SCOTT보다 많은 급여를 받는 사원의 이름을 검색하기 위해서는 먼저 SCOTT가 받는 급여를 검색한 후, 검색된 급여보다 많은 급여를 받는 직원들을 검색하여야 한다.

```
SQL> SELECT SAL
  2  FROM EMP
  3  WHERE ENAME = 'SCOTT';

       SAL
----------
      3000

SQL> SELECT ENAME
  2  FROM EMP
  3  WHERE SAL > 3000;

ENAME
----------
KING
```

서브 쿼리를 사용하면 위와 같이 두 단계에 걸쳐서 수행되어야 할 작업을 한번에 수행할 수 있게 된다.

서브 쿼리의 사용방법

서브 쿼리의 사용방법은 다음과 같다.

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

위에서 바깥 쪽 쿼리를 메인 쿼리(Main query), 안쪽 쿼리를 서브 쿼리(Subquery)라고 부르며, 서브 쿼리가 먼저 실행되고 그 결과가 메인 쿼리에 전달되어 실행된다. 서브 쿼리는 SQL 문장의 WHERE절 뿐만 아니라 HAVING, FROM 절에도 포함될 수 있다. 위에서

*operator*는 단일행 연산자(>, =, >=, <, <>, <=)와 복수행 연산자(IN, ANY, ALL)로 분류할 수 있으며 SQL 문장내의 서브 쿼리가 단일행 서브 쿼리인 경우는 단일행 연산자를 사용하고 복수행 서브 쿼리인 경우는 복수행 연산자를 사용하여야만 한다.

서브 쿼리를 사용할 때는 다음과 같은 작성 지침을 준수하도록 한다.
- 서브 쿼리에는 괄호를 붙인다.
- 서브 쿼리가 WHERE 절에 포함 될 때는 비교 연산자의 오른쪽에 기술한다.
- Top-N 쿼리를 작성하는 경우 이외에는 ORDER BY 절에 서브 쿼리를 작성할 필요는 없다.
- 단일행 서브 쿼리에는 단일행 연산자를 사용하고 복수행 서브 쿼리인 경우는 복수행 연산자를 사용한다.

사원 테이블에서 SCOTT의 급여보다 많은 급여를 받는 사원의 이름을 검색하면 다음과 같다.

```
SQL> SELECT ENAME
  2  FROM EMP
  3  WHERE SAL > (SELECT SAL
  4               FROM EMP
  5               WHERE ENAME = 'SCOTT');

ENAME
----------
KING
```

서브 쿼리의 종류

서브 쿼리의 종류에는 단일행 서브 쿼리와 복수행 서브 쿼리가 있다. 이렇게 서브 쿼리를 구분하는 기준은 서브 쿼리가 리턴하는 행의 개수에 따라 다른데, 서브 쿼리가 한개의 행을 리턴하면 단일행 서브 쿼리라 하고 두개 이상의 행을 리턴하면 복수행 서브 쿼리라고 부른다.

단일행 서브 쿼리

단일행 서브 쿼리는 오직 하나의 행만을 리턴하며, 반드시 단일행 연산자(=, >, <=, <, >=, <>)를 사용해야만 한다.

사원 테이블에서 사번이 7844인 사원의 업무와 동일한 업무를 수행하는 사원의 사번, 이름, 업무를 검색하면 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, JOB
  2  FROM EMP
  3  WHERE JOB = (SELECT JOB
  4               FROM EMP
  5               WHERE EMPNO = 7844);

     EMPNO ENAME      JOB
---------- ---------- ---------
      7499 ALLEN      SALESMAN
      7521 WARD       SALESMAN
      7654 MARTIN     SALESMAN
      7844 TURNER     SALESMAN
```

사원 테이블에서 사번이 7521번인 사원과 같은 업무를 수행하고 있으며, 사번이 7900번인 사원의 급여보다 많은 급여를 받는 사원의 이름, 업무, 급여는 다음과 같다.

```
SQL> SELECT ENAME, JOB, SAL
  2  FROM EMP
  3  WHERE JOB = (SELECT JOB
  4               FROM EMP
  5               WHERE EMPNO = 7521)
  6  AND SAL > (SELECT SAL
  7             FROM EMP
  8             WHERE EMPNO = 7900);

ENAME      JOB              SAL
---------- --------- ----------
ALLEN      SALESMAN         1600
WARD       SALESMAN         1250
MARTIN     SALESMAN         1250
TURNER     SALESMAN         1500
```

사원 테이블에서 가장 적은 급여를 받는 사원의 사번, 이름, 급여를 검색하면 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, SAL
  2  FROM EMP
  3  WHERE SAL = (SELECT MIN(SAL)
  4               FROM EMP);

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
```

서브 쿼리는 WHERE 절 뿐만 아니라 HAVING 절에도 사용할 수 있다.

```
SQL> SELECT DEPTNO, MIN(SAL)
  2  FROM EMP
  3  GROUP BY DEPTNO
  4  HAVING MIN(SAL) > (SELECT MIN(SAL)
  5                     FROM EMP
  6                     WHERE DEPTNO = 20);

    DEPTNO   MIN(SAL)
---------- ----------
        10       1300
        30        950
```

그러나, 다음의 예는 2개 이상의 행이 리턴되는 복수행 서브 쿼리에 단일행 연산자를 사용하였기 때문에 오류가 발생한 경우이다.

```
SQL> SELECT ENAME
  2  FROM EMP
  3  WHERE SAL = (SELECT MIN(SAL)
  4               FROM EMP
  5               GROUP BY DEPTNO);
WHERE SAL = (SELECT MIN(SAL)
             *
3행에 오류:
ORA-01427: 단일 행 부속 질의에 2개 이상의 행이 리턴되었습니다
```

또한, 서브 쿼리가 NULL을 리턴 하는 경우에는 상황에 따라 원치 않는 결과를 얻을 수도 있다. 아래 서브 쿼리는 NULL을 메인 쿼리의 WHERE 절로 리턴하기 때문에 EMP 테이블의 JOB 컬럼에 NULL이 있든 없든 상관 없이 WHERE 절의 비교 결과가 NULL이 되어 어떤 결과도 출력할 수가 없게 된다. 그러므로, 서브 쿼리의 결과로 혹시 NULL이 나올 가능성이 있는지도 사전에 확인 할 필요가 있다.

```
SQL> SELECT ENAME
  2  FROM EMP
  3  WHERE JOB = (SELECT JOB
  4               FROM EMP
  5               WHERE ENAME = 'KIM');

선택된 레코드가 없습니다.
```

복수행 서브 쿼리

복수행 서브 쿼리는 두개 이상의 행을 리턴하며, 반드시 복수행(IN, ANY, ALL) 연산자를 사용해야만 한다.

■ IN

IN은 WHERE 조건에서 사용하는 일반 비교 연산자와 동일하다. 예를 들어, 사원 테이블에서 부서별 최소 급여를 받는 사원의 사번, 이름, 급여, 부서코드를 검색하면 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, SAL, DEPTNO
  2  FROM EMP
  3  WHERE SAL IN (SELECT MIN(SAL)
  4                FROM EMP
  5                GROUP BY DEPTNO);

     EMPNO ENAME             SAL     DEPTNO
---------- ---------- ---------- ----------
      7369 SMITH             800         20
      7900 JAMES             950         30
      7934 MILLER           1300         10
```

위 문장은 다음 문장과 동일하다.

```
SQL> SELECT EMPNO, ENAME, SAL, DEPTNO
  2  FROM EMP
  3  WHERE SAL IN (800, 950, 1300);
```

■ ALL

ALL은 복수행 서브 쿼리의 결과가 메인 쿼리의 WHERE 절에서 부등호 조건으로 비교될 때 사용되며, 서브 쿼리에서 리턴된 모든 결과값이 WHERE 절에서 모두 비교된다. ALL 연산자를 이해하기 위해 다음과 같은 예를 살펴보도록 하자.

사원 테이블에서 업무가 MANAGER인 사원의 급여보다 적은 급여를 받는 사원들의 이름을 검색하고자 한다. 먼저, 업무가 MANAGER인 사원의 급여를 검색하면 다음과 같다.

```
SQL> SELECT SAL
  2  FROM EMP
  3  WHERE JOB = 'MANAGER';

       SAL
----------
      2975
      2850
      2450
```

여기서, 업무가 MANAGER인 사원들의 최소 급여보다 급여가 적은 사원을 검색하려면, 다음과 같이 검색하여야 한다.

```
SQL> SELECT EMPNO, ENAME, SAL
  2  FROM EMP
  3  WHERE SAL < 2450;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7521 WARD             1250
      7654 MARTIN           1250
      7844 TURNER           1500
      7876 ADAMS            1100
      7900 JAMES             950
      7934 MILLER           1300

8 개의 행이 선택되었습니다.
```

위와 같은 절차를 서브 쿼리로 작성하면 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, SAL
  2  FROM EMP
  3  WHERE SAL < ALL (SELECT SAL
  4                   FROM EMP
  5                   WHERE JOB='MANAGER');

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7521 WARD             1250
      7654 MARTIN           1250
      7844 TURNER           1500
      7876 ADAMS            1100
      7900 JAMES             950
      7934 MILLER           1300

8 개의 행이 선택되었습니다.
```

■ ANY

ANY는 복수행 서브 쿼리의 결과가 메인 쿼리의 WHERE 절에서 부등호 조건으로 비교될 때 사용되며, 서브 쿼리에서 리턴된 각각의 결과값이 WHERE 절에서 비교된다. 마찬가지로 ANY 연산자를 이해하기 위해 다음과 같은 예를 살펴본다.

사원 테이블에서 업무가 MANAGER인 사원의 급여보다 많은 급여를 받는 사원들의 이름을 검색하고자 한다. 먼저, 업무가 MANAGER인 사원의 급여를 검색하면 다음과 같다.

```
SQL> SELECT SAL
  2  FROM EMP
  3  WHERE JOB = 'MANAGER';

       SAL
----------
      2975
      2850
      2450
```

여기서, 업무가 MANAGER인 사원들의 최소 급여보다 급여가 많은 사원을 검색하면 다음과 같이 검색하여야 한다.

```
SQL> SELECT EMPNO, ENAME, SAL
  2  FROM EMP
  3  WHERE SAL > 2450;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7566 JONES            2975
      7698 BLAKE            2850
      7788 SCOTT            3000
      7839 KING             5000
      7902 FORD             3000
```

위와 같은 절차를 서브 쿼리로 작성하면 다음과 같다.

```
SQL>  SELECT EMPNO, ENAME, SAL
  2   FROM EMP
  3   WHERE SAL > ANY (SELECT SAL
  4                    FROM EMP
  5                    WHERE JOB='MANAGER');

     EMPNO ENAME             SAL
---------- ---------- ----------
      7566 JONES            2975
      7698 BLAKE            2850
      7788 SCOTT            3000
      7839 KING             5000
      7902 FORD             3000
```

ANY와 ALL 연산자를 정리하면 다음과 같다.

· < ANY : < 서브쿼리의 리턴값 중 최대값

· > ANY : > 서브쿼리의 리턴값 중 최소값

· < ALL : < 서브쿼리의 리턴값 중 최소값

· > ALL : > 서브쿼리의 리턴값 중 최대값

복습

1. 사원 테이블에서 BLAKE 보다 급여가 많은 사원들의 사번, 이름, 급여를 검색하시오.

2. 사원 테이블에서 MILLER보다 늦게 입사한 사원의 사번, 이름, 입사일을 검색하시오.

3. 사원 테이블에서 사원 전체 평균 급여보다 급여가 많은 사원들의 사번, 이름, 급여를 검색하시오.

4. 사원 테이블에서 CLARK와 같은 부서이며, 사번이 7698인 직원의 급여보다 많은 급여를 받는 사원들의 사번, 이름, 급여를 검색하시오.

5. 사원 테이블에서 부서별 최대 급여를 받는 사원들의 사번, 이름, 부서코드, 급여를 검색하시오.