

Java Web Programming 입문 15

(Oracle PL/SQL #07)

오늘의 키워드

- ▶ 서브 쿼리 (Sub Query)
- ▶ DML (Data Manipulation Language)

서브 쿼리 (Sub Query)

- ▶ EMP Table내 Jones 보다 급여(SAL)가 높은 사원?

```
SQL> SELECT SAL  
      FROM EMP  
      WHERE ENAME = 'JONES';
```

The diagram illustrates the process of using a subquery. An arrow points from the first SQL query box to a table icon. Another arrow points from the table icon to the second SQL query box, indicating that the result of the first query is used as a value in the second query.

SAL
2975

```
SQL> SELECT ENAME, SAL  
      FROM EMP  
      WHERE SAL >= 2975;
```

서브 쿼리 (Sub Query)

- ▶ EMP Table내 Jones 보다 급여(SAL)가 높은 사원?

```
SQL> SELECT SAL  
      FROM EMP  
      WHERE ENAME = 'JONES';
```

```
SQL> SELECT ENAME, SAL  
      FROM EMP  
      WHERE SAL >= (SELECT SAL  
                    FROM EMP  
                    WHERE ENAME = 'JONES');
```

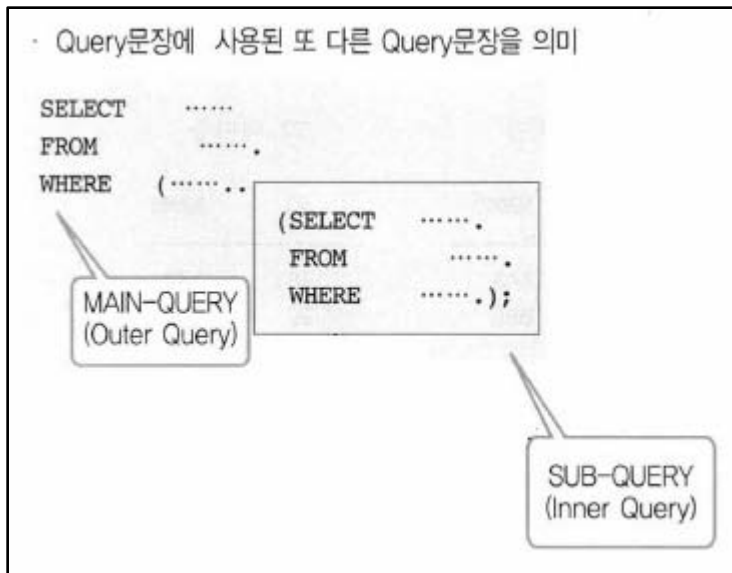
```
SQL> SELECT ENAME, SAL  
      FROM EMP  
      WHERE SAL >= 2975
```



SAL
2975

서브 쿼리 (Sub Query)

▶ What the...?



- ▶ 연산자의 오른쪽에 위치, 괄호로 묶어 사용
- ▶ 지정되지 않은 값을 근거로 할 때 유용
- ▶ Sub-Query는 Main-Query의 조건으로 사용됨
- ▶ Sub-Query의 결과는 Main-Query에 의해 사용
- ▶ 실행순서는 Sub-Query가 먼저 실행된 후 그 결과를 Main-Query에 전달후 실행
- ▶ Main-Query의 다양한 위치에서 사용가능
 - SELECT절
 - SELECT/DELETE/UPDATE문의 FROM 절
 - WHERE절
 - HAVING절
 - INSERT문의 INTO절/UPDATE문의 SET절
- ▶ Sub-Query에는 ORDER BY 절 사용 불가
 - (SELECT/DELETE/UPDATE 문장의 FROM절은 예외로 사용가능)

서브 쿼리 (Sub Query)

▶ 제작 순서?

- Sub-Query를 먼저 만든다
- Main-Query를 만든다
- 합친다



- ▶ 코끼리 냉장고에 넣기
 - ▶ 냉장고 문을 연다
 - ▶ 코끼리를 집어 넣는다
 - ▶ 냉장고 문을 닫는다
 - ▶ (쌤 = _ = ?)

서브 쿼리 (Sub Query)

▶ 제작 순서?

- Sub-Query를 먼저 만든다

```
SQL> SELECT HIREDATE  
      FROM EMP  
      WHERE ENAME = 'BLAKE'
```

- Main-Query를 만든다

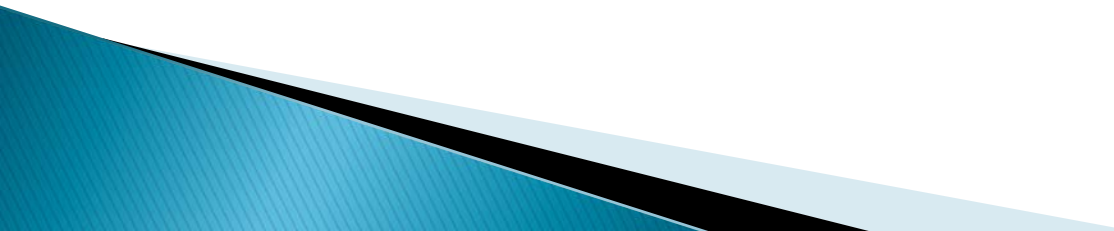
```
SQL> SELECT ENAME, HIREDATE  
      FROM EMP  
      WHERE HIREDATE > ???????
```

- 합친다

```
SQL> SELECT ENAME, HIREDATE  
      FROM EMP  
      WHERE HIREDATE > (SELECT HIREDATE  
                        FROM EMP  
                        WHERE ENAME = 'BLAKE');
```

서브 쿼리 (Sub Query)

▶ 종류

- 단일 행(Single Row) Sub-Query
 - 복수 행(Multiple Row) Sub-Query
 - 복수 컬럼(Multiple Column) Sub-Query
 - 상호관련(Interrelative) Sub-Query
- 

서브 쿼리 (Sub Query)

▶ 단일 행(Single Row) Sub-Query

- =, <, >, <=, >=, <>

```
SQL> SELECT *  
      FROM EMP  
      WHERE JOB = (SELECT JOB  
                  FROM EMP  
                  WHERE ENAME = 'FORD')
```

▶ 복수 행(Multiple Row) Sub-Query

- IN, ANY, ALL, EXISTS

```
SQL> SELECT *  
      FROM EMP  
      WHERE JOB IN (SELECT JOB  
                  FROM EMP  
                  WHERE ENAME = 'FORD'  
                  OR ENAME = 'JAMES')
```

서브 쿼리 (Sub Query)

▶ 복수 행(Multiple Row) Sub-Query

◦ IN

- Sub-Query에 의해 Return되는 값 중 조건에 해당하는 Row가 있으면 출력

```
SQL> SELECT ENAME, SAL
      FROM EMP
      WHERE SAL IN (SELECT MAX(SAL)
                    FROM EMP
                    GROUP BY DEPTNO)
```

◦ ANY, SOME

- Sub-Query에 의해 Return되는 값 중 하나 이상을 만족하는 행이 있으면 출력

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

```
SQL> SELECT *
      FROM EMP
      WHERE SAL < ANY (SELECT SAL
                      FROM EMP
                      WHERE DEPTNO = 30);
```

서브 쿼리 (Sub Query)

▶ 복수 행(Multiple Row) Sub-Query

◦ ALL

- Sub-Query에 의해 Return되는 값이 조건을 만족하면 행을 출력
- <ALL : 결과 중 가장 작은 값 보다 작은 조건의 행

```
SQL> SELECT EMPNO, ENAME, JOB, SAL
      FROM EMP
      WHERE SAL < ALL (SELECT SAL
                       FROM EMP
                       WHERE JOB = 'MANAGER')
```

- >ALL : 결과 중에 가장 큰 값보다 큰 조건의 행

```
SQL> SELECT EMPNO, ENAME, JOB, SAL
      FROM EMP
      WHERE SAL > ALL (SELECT SAL
                       FROM EMP
                       WHERE JOB = 'MNAGER')
```

◦ EXISTS

- Sub-Query의 결과를 만족하는 값이 존재하는지 여부를 확인하는 경우

```
SQL> SELECT DNAME, DEPTNO
      FROM DEPT
      WHERE EXISTS (SELECT *
                   FROM EMP
                   WHERE EMP.DEPTNO = 30)
```

서브 쿼리 (Sub Query)

▶ 복수 컬럼(Multiple Column) Sub-Query

- 실행 결과가 여러 개의 Column, 복수 여러 개의 Row를 Return

```
SQL> SELECT *  
      FROM EMP  
      WHERE (SAL, DEPTNO) IN (SELECT MIN (SAL), DEPTNO  
                               FROM EMP  
                               GROUP BY DEPTNO)
```

```
SQL> SELECT ENAME, DEPTNO, NVL (COMM, -1)  
      FROM EMP  
      WHERE (SAL, NVL (COMM, -1)) IN (SELECT SAL, NVL (COMM, -1)  
                                       FROM EMP  
                                       WHERE DEPTNO = 30);
```

▶ 상호관련(Interrelative) Sub-Query

- Main-Query절에 사용된 Table이 Sub-Query에서 재사용
- Sub-Query에 사용될 조건값을 Main-Query에서 참조해야할 경우 사용

```
SQL> SELECT E.EMPNO, E.ENAME, E.DEPTNO  
      FROM EMP E  
      WHERE SAL > (SELECT AVG (SAL)  
                   FROM EMP D  
                   WHERE D.DEPTNO = E.DEPTNO)
```

DML (Data Manipulation Language)

- ▶ 데이터 조작어
- ▶ 면접 단골 질문 (오버로딩, 오버라이딩)
 - Table에 Data를 추가, 변경, 삭제 수행
 - INSERT : Table에 Data를 저장
 - UPDATE : Table에 저장되어있는 Data를 변경
 - DELETE : Table에 저장되어 있는 Data를 삭제

DML (Data Manipulation Language)

▶ 준비!

- 실제 Data를 손대기 때문에 주의
- Dept Table의 복사본을 만들어 사용

• (1)

```
SQL> CREATE TABLE SCOTT.DEPT2  
  (  
    DEPTNO    NUMBER(2),  
    DNAME     VARCHAR2(14 BYTE),  
    LOC       VARCHAR2(13 BYTE)  
  )
```

```
SQL> INSERT INTO DEPT2 (SELECT * FROM DEPT)
```

• (2)

```
SQL> CREATE TABLE SCOTT.DEPT2  
      AS SELECT *  
      FROM DEPT
```

```
SQL> SELECT * FROM DEPT2
```

DML (Data Manipulation Language)

▶ INSERT

```
SQL> INSERT INTO DEPT2  
      VALUES (50, 'A', 'B');
```

```
SQL> SELECT *  
      FROM DEPT2;
```

- Data에 Null 을 넣을때
 - NULL의 명시적 삽입법

```
SQL> INSERT INTO DEPT2  
      VALUES (60, 'B', NULL);
```

- NULL 의 암시적 삽입법

```
SQL> INSERT INTO DEPT2 (DEPTNO, LOC)  
      VALUES (70, 'C');
```

DML (Data Manipulation Language)

▶ DML 기본형

◦ INSERT

```
INSERT INTO Table_Name [(Column_1, Column_2, Column_3, ... , Column_N)]  
VALUES [(Column_1_값, Column_2_값, Column_3_값, ... , Column_N_값)];
```

```
INSERT INTO DEPT (DEPTNO, DNAME, LOC)  
VALUES (99, '인사과', '서울');
```

◦ INSERT With Sub-Query

```
INSERT INTO Table_Name [(Column_1, Column_2, Column_3, ... , Column_N)]  
[SELECT Query]
```

```
INSERT INTO DEPT (DEPTNO, DNAME, LOC)  
SELECT DEPTNO, DNAME, LOC FROM DEPT2;
```


DML (Data Manipulation Language)

▶ INSERT

- 한번에 한 행만 입력 가능

```
SQL> INSERT INTO dept  
VALUES (99, '총무과', '서울', 98, '인사과', '대전');
```

- INSERT에 명시된 COLUMN 개수와 VALUES 절의 개수는 같게!

```
SQL> INSERT INTO dept (deptno, dname)  
VALUES (99, '총무과', '서울')
```

- 입력될 값의 DATA TYPE과 COLUMN의 DATA TYPE은 동일하게!

```
SQL> INSERT INTO DPET (DEPTNO, DNAME)  
VALUES ('99', 총무과)
```

DML (Data Manipulation Language)

▶ INSERT

- 입력될 값의 크기는 정의된 Column의 크기보다 클 수 없다.

```
SQL> INSERT INTO DEPT (DEPTNO, DNAME)  
VALUES (999, '총무과');
```

- NULL 값에 주의

```
SQL> INSERT INTO DEPT (DEPTNO, DNAME)  
VALUES (99, '총무과');
```

```
SQL> INSERT INTO DEPT (DEPTNO, DNAME, LOC)  
VALUES (97, '총무과', '');
```

```
SQL> INSERT INTO DEPT (DEPTNO, DNAME, LOC)  
VALUES (98, '총무과', NULL);
```

DML (Data Manipulation Language)

▶ INSERT

- Sub-Query 사용시 VALUES 절 사용하지 못함

```
SQL> INSERT INTO EMP  
VALUES SELECT *  
FROM EMP;
```

- INSERT절에 명시된 Column수, Data Type과 Sub-Query에 사용된 Column수, Data Type 일치

```
SQL> INSERT INTO EMP (EMPNO, ENAME, DEPTNO)  
SELECT EMPNO, ENAME  
FROM EMP
```

- Sub-Query가 Return 하는 Row만큼 INSERT

```
SQL> INSERT INTO EMP  
SELECT *  
FROM EMP
```

DML (Data Manipulation Language)

▶ UPDATE

```
SQL> UPDATE DEPT2  
      SET DNAME = '서울';
```



```
SQL> SELECT *  
      FROM DEPT2;
```

```
SQL> UPDATE DEPT2  
      SET DNAME = '부산'  
      WHERE DEPTNO = 50;
```



```
SQL> SELECT *  
      FROM DEPT2;
```

DML (Data Manipulation Language)

▶ DELETE

```
SQL> DELETE FROM DEPT2  
      WHERE DEPTNO IN (50, 60, 70);
```



```
SQL> SELECT *  
      FROM DEPT2;
```

```
SQL> DELETE FROM DEPT2;
```



```
SQL> SELECT *  
      FROM DEPT2;
```



```
SQL> ROLLBACK;
```

DML (Data Manipulation Language)

▶ DML 기본형

◦ UPDATE

```
UPDATE Table_Name  
    SET [Column_1 = Column_1_값, Column_2 = Column_2_값 , ... , Column_N_값]  
[WHERE Condition]
```

```
UPDATE DEPT  
    SET LOC = '서울'  
WHERE DEPTNO = 20;
```

◦ DELETE

```
DELETE [FROM] Table_Name  
[WHERE Condition]
```

```
DELETE DEPT  
WHERE DEPTNO = 20;
```

DML (Data Manipulation Language)

▶ UPDATE

- WHERE Condition이 없을 경우 모든 Row의 Column값을 변경

```
SQL> UPDATE DEPT  
      SET LOC = '부산';
```

```
SQL> UPDATE EMP  
      SET SAL = SAL * 1.1;
```

- Sub-Query 사용

```
SQL> UPDATE EMP  
      SET (JOB, DEPTNO) = (SELECT JOB, DEPTNO  
                          FROM EMP  
                          WHERE EMPNO = 7934)  
      WHERE EMPNO = 7782;
```

▶ DELETE

```
SQL> DELETE FROM EMP  
      WHERE EMPNO = 7782;
```

```
SQL> DELETE FROM EMP;
```

```
SQL> DELETE FROM EMP  
      WHERE DEPTNO = (SELECT DEPTNO  
                     FROM DEPT  
                     WHERE DNAME = 'ACCOUNTING');
```