

Java Web Programming 입문 13

(Oracle PL/SQL #05)

오늘의 키워드

▶ 1. 조인 (Join)

- Equi-Join, NonEqui-Join
- Self-Join
- Outer-Join
- Join시 조건의 개수
- SQL92 VS SQL99
 - Cross Join
 - Natural Join
 - Join ~ Using
 - Join ~ On
 - Outer Join

Equi-Join, NonEqui-Join

▶ Equi-Join

- '=' 을 사용하여 Join
- EMP.DEPTNO = DEPT.DEPTNO

▶ NonEqui-Join

- 'NOT' EQUI ? : '='을 사용하는 JOIN 뺀 나머지

Equi-Join, NonEqui-Join

▶ NonEqui-Join 사용예

```
SQL> DESC SALGRADE
```

```
SQL> SELECT E.ENAME, S.GRADE  
       FROM EMP E, SALGRADE S  
       WHERE E.SAL <= S.HISAL AND E.SAL >= S.LOSAL;
```

```
SQL> SELECT E.ENAME, S.GRADE  
       FROM EMP E, SALGRADE S  
       WHERE E.SAL BETWEEN S.LOSAL AND S.HISAL
```

Self-Join

- ▶ EMP Table을 사용하여 하나의 Row에 사원과 그 사원의 상급자를 함께 나타내고 싶다면?

EMPNO	ENAME	MGR
7788	SCOTT	7799
7799	SMITH	7800
7800	KING	



```
SQL> SELECT *  
      FROM EMP  
      WHERE MGR = EMPNO;
```

- ▶ 단일 Table 단순 조회만으로는 불가 (능해 보인다)

Self-Join

- ▶ Table을 복사해버리면 된다.

EMPNO	ENAME	MGR	EMPNO	ENAME	MGR
7788	SCOTT	7799	7788	SCOTT	7799
7799	SMITH	7800	7799	SMITH	7800
7800	KING		7800	KING	

(COPY TABLE)
'두개 TABLE을 JOIN'

- ▶ 그리고 두 개 Table을 Join

```
SQL> SELECT E.ENAME, C.ENAME  
       FROM EMP E, COPY_EMP C  
       WHERE E.MGR = C.EMPNO;
```

Self-Join

- ▶ 물리적 저장공간을 할애하여 복사 Table을 만들어 두는건 낭비

```
SQL> SELECT E.ENAME, C.ENAME
      FROM EMP E, EMP C
      WHERE E.MGR = C.EMPNO; -- 별칭을 다르게 하면 논리적 복사
                             -- 효과를 얻을 수 있다.
```

EMPNO	ENAME	MGR	EMPNO	ENAME	MGR
7788	SCOTT	7799	7788	SCOTT	7799
7799	SMITH	7800	7799	SMITH	7800
7800	KING		7800	KING	

- ▶ 이렇게 논리적으로 하나의 Table을 복사한듯이 사용하면 Self-Join

Outer-Join

- ▶ 앞의 Self-Join 예제를 실행했을 때...

```
SQL> SELECT E.EMPNO, E.ENAME, E.MGR, C.EMPNO, C.ENAME  
       FROM EMP E, EMP C  
       WHERE E.MGR = C.EMPNO
```

EMPNO	ENAME	MGR	EMPNO_1	ENAME_1
7902	FORD	7566	7566	JONES
7788	SCOTT	7566	7566	JONES
7844	TURNER	7698	7698	BLAKE
7499	ALLEN	7698	7698	BLAKE
7521	WARD	7698	7698	BLAKE
7900	JAMES	7698	7698	BLAKE
7654	MARTIN	7698	7698	BLAKE
7934	MILLER	7782	7782	CLARK
7876	ADAMS	7788	7788	SCOTT
7698	BLAKE	7839	7839	KING
7566	JONES	7839	7839	KING
7782	CLARK	7839	7839	KING
7369	SMITH	7902	7902	FORD

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-06-09	2450		10
7788	SCOTT	ANALYST	7566	1987-04-19	3000		20
7839	KING	PRESIDENT		1981-11-17	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
7876	ADAMS	CLERK	7788	1987-05-23	1100		20
7900	JAMES	CLERK	7698	1981-12-03	950		30
7902	FORD	ANALYST	7566	1981-12-03	3000		20
7934	MILLER	CLERK	7782	1982-01-23	1300		10

Outer-Join

- ▶ Join 조건으로 걸었던 Column의 값이 Null
- ▶ Null이 포함되어있는 녀석들도 강제로 조회
 - **Outer-Join**

```
SQL> SELECT E.ENAME, C.ENAME  
       FROM EMP E, EMP C  
       WHERE E.MGR(+) = C.EMPNO;
```

```
SQL> SELECT E.ENAME, C.ENAME  
       FROM EMP E, EMP C  
       WHERE E.MGR = C.EMPNO(+);
```

- ▶ NULL 값을 가지고 있어도 출력하게 할 Table 한쪽에 ‘(+)’ 를 붙여준다.

```
SQL> SELECT E.ENAME, C.ENAME  
       FROM EMP E, EMP C  
       WHERE E.MGR(+) = C.EMPNO(+);
```



Join시 조건의 개수

- ▶ 여러 개의 Table을 Join할 때, 정확한 자료의 조회 (데카르트 곱을 피하는)를 만족하기 위해 필요한 조건 개수

<ul style="list-style-type: none">- Table 2개 일때 : 조건은 최소 1개- Table 3개 일때 : 조건은 최소 2개...- Table N개 일때 : 필요한 최소 조건은 $n-1$개
--

- ▶ 왜 최소 $n-1$ 개?

Join시 조건의 개수

- ▶ A, B, C 라는 세개의 Table 을 Join 해야 할 경우
 - 최소한 $A=B$, $B=C$ 로 조건을 나타낼 수 있어야 A에 있는 Row 하나에 정확한 B, C Row들을 매칭할 수 있다.
- ▶ A : EMP , B : DEPT, C : SALGRADE

```
SQL> SELECT E.EMPNO, D.DEPTNO, S.GRADE  
       FROM EMP E, DEPT D, SALGRADE S  
       WHERE E.DEPTNO = D.DEPTNO  
             AND E.SAL BETWEEN S.LOSAL AND S.HISAL;
```

SQL92 VS SQL99

▶ 문법사용 방식의 차이

- 앞의 Join들은 모두 SQL92 방식 (Old Ver.?)
- SQL99 방식은 SQL92때부터 존재하던 Join 방식을 한눈에 무슨 Join 기법을 사용하는지 알아보기 위해 만든 방식
- SQL92, SQL99 모두 사용된다.

SQL92 VS SQL99

► Cross Join

- Cartesian Product

- SQL92

```
SQL> SELECT E.ENAME, D.DNAME  
       FROM EMP E, DEPT D;
```

- SQL99

```
SQL> SELECT E.ENAME, D.DNAME  
       FROM EMP E CROSS JOIN DEPT D;
```

SQL92 VS SQL99

► Natural Join

- Equi-Join

- SQL92


```
SQL> SELECT E.ENAME, D.DNAME  
       FROM EMP E, DEPT D  
       WHERE E.DEPTNO = D.DEPTNO;
```

- SQL99

```
SQL> SELECT E.ENAME, D.DNAME, E.DEPTNO  
       FROM EMP E NATURAL JOIN DEPT D;
```



```
SQL> SELECT E.ENAME, D.DNAME, DEPTNO  
       FROM EMP E NATURAL JOIN DEPT D;
```



SQL92 VS SQL99

▶ Join ~ Using


- Join 조건에 사용될 Column을 지정후 Natural Join

- SQL92

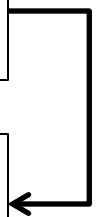
```
SQL> SELECT E.ENAME, D.DNAME  
       FROM EMP E, DEPT D  
       WHERE E.DEPTNO = D.DEPTNO;
```

- SQL99

```
SQL> SELECT E.ENAME, D.DNAME, E.DEPTNO  
       FROM EMP E JOIN DEPT D USING (DEPTNO);
```



```
SQL> SELECT E.ENAME, D.DNAME, DEPTNO  
       FROM EMP E JOIN DEPT D USING (DEPTNO);
```



SQL92 VS SQL99

▶ Join ~ On

- Self-Join 또는 각 Table에 같은 값을 가지지만 Column명은 다른 Column들로 Join 조건을 줄 때

- SQL92

```
SQL> SELECT E.ENAME, C.ENAME  
       FROM EMP E, EMP C  
       WHERE E.MGR = C.EMPNO;
```

- SQL99

```
SQL> SELECT E.ENAME, D.DNAME  
       FROM EMP E JOIN DEPT D ON ( E.DEPTNO = D.DEPTNO );
```

```
SQL> SELECT E.ENAME, D.DNAME, E.DEPTNO, D.DEPTNO2  
       FROM EMP E JOIN  
       (SELECT DEPTNO AS DEPTNO2, DNAME, LOC FROM DEPT) D  
       ON (E.DEPTNO = D.DEPTNO2);
```


SQL92 VS SQL99

▶ Outer Join

- Self-Join 또는 각 Table에 같은 값을 가지지만 Column명은 다른 Column들로 Join 조건을 줄 때

```
SELECT E.EMPNO, E.ENAME, E.MGR, D.DNAME  
FROM EMP E, DEPT D  
WHERE E.DEPTNO = D.DEPTNO (+)
```

```
SELECT E.EMPNO, E.ENAME, E.MGR, D.DNAME  
FROM EMP E, DEPT D  
WHERE E.DEPTNO (+) = D.DEPTNO
```

```
SELECT E.EMPNO, E.ENAME, E.MGR, D.DNAME  
FROM EMP E LEFT OUTER JOIN DEPT D ON E.DEPTNO = D.DEPTNO
```

```
SELECT E.EMPNO, E.ENAME, E.MGR, D.DNAME  
FROM EMP E RIGHT OUTER JOIN DEPT D ON E.DEPTNO = D.DEPTNO
```

```
SELECT E.EMPNO, E.ENAME, E.MGR, D.DNAME  
FROM EMP E FULL OUTER JOIN DEPT D ON E.DEPTNO = D.DEPTNO
```