

MySQL & Maria DB

조인 (JOIN)

목차

- ▶ 조인의 필요성
- ▶ 상호 조인(Cross Join)
- ▶ Equi Join
- ▶ Non-Equi Join
- ▶ Self Join
- ▶ Outer Join
- ▶ Union

조인의 필요성

▶ 조인의 필요성

- ▶ 특정 부서 번호에 대한 부서이름은 무엇 인지는 부서(DEPT) 테이블에 있습니다. 특정 사원에 대한 부서명을 알아내기 위해서는 부서 테이블에서 정보를 얻어 와야 합니다.

```
C:\Windows\system32\cmd.exe
SQL> SELECT ENAME, DEPTNO
2 FROM EMP
3 ORDER BY DEPTNO;
```

ENAME	DEPTNO
CLARK	10
KING	10
MILLER	10
JONES	20
FORD	20
ADAMS	20
SMITH	20
SCOTT	20
WARD	30
TURNER	30
ALLEN	30
JAMES	30
BLAKE	30
MARTIN	30

14 개의 행이 선택되었습니다.

```
SQL>
```

```
C:\Windows\system32\cmd.exe
SQL> SELECT DEPTNO, DNAME
2 FROM DEPT;
```

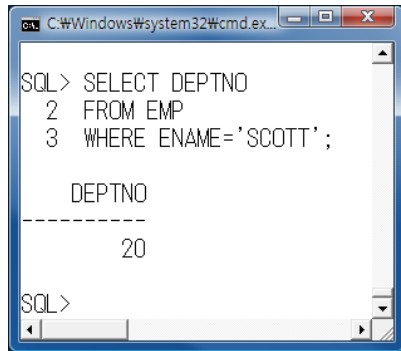
DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

```
SQL>
```

조인의 필요성

▶ 조인의 필요성

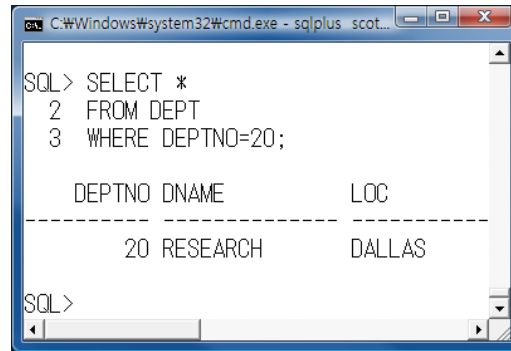
- ▶ **SCOTT**인 사원이 소속되어 있는 부서의 이름이 무엇인지 알아보려고 합니다.
- ▶ **SCOTT**이란 사원의 부서명을 알아내는 일 역시 사원 테이블에서 **SCOTT**이 소속된 부서 번호를 알아낸 후에 부서 테이블에서 해당 부서 번호에 대한 부서명을 얻어 와야 합니다.



```
C:\Windows\system32\cmd.exe - sqlplus scott...
SQL> SELECT DEPTNO
2 FROM EMP
3 WHERE ENAME='SCOTT';

DEPTNO
-----
      20

SQL>
```



```
C:\Windows\system32\cmd.exe - sqlplus scott...
SQL> SELECT *
2 FROM DEPT
3 WHERE DEPTNO=20;

DEPTNO DNAME          LOC
-----
      20 RESEARCH      DALLAS

SQL>
```

- ▶ 실습에서처럼 원하는 정보가 두 개 이상의 테이블에 나누어져 있다면 위와 같이 여러 번 질의를 해야 할까요?
- ▶ 다행히도 **SQL**에서는 두 개 이상의 테이블을 결합해야만 원하는 결과를 얻을 수 있을 때 한 번의 질의로 원하는 결과를 얻을 수 있는 조인 기능을 제공합니다.

상호 조인(Cross Join)

▶ 상호 조인(Cross Join)

- ▶ 다음은 Cross Join으로 특별한 키워드 없이 **SELECT** 문의 **FROM** 절에 사원(**EMP**) 테이블과 부서(**DEPT**) 테이블을 콤마로 연결하여 연속하여 기술하는 것입니다.

예	<pre>SELECT * FROM EMP CROSS JOIN DEPT;</pre>
---	---

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	emp.DEPTNO	dept.DEPTNO	DNAME	LOC
1	7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800	<null>	20	10	ACCOUNTING	NEW YORK
2	7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800	<null>	20	20	RESEARCH	DALLAS
3	7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800	<null>	20	30	SALES	CHICAGO
4	7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800	<null>	20	40	OPERATIONS	BOSTON
5	7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30	10	ACCOUNTING	NEW YORK
6	7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30	20	RESEARCH	DALLAS
7	7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30	30	SALES	CHICAGO

상호 조인(Cross Join)

▶ 상호 조인(Cross Join)

- ▶ **Cross Join**의 결과 얻어지는 컬럼의 수는 사원 테이블의 컬럼의 수(8)와 부서 테이블의 컬럼의 수를 더한 것이므로 8이 됩니다. 로우 수는 사원 한 명에 대해서 **DEPT** 테이블의 4개의 로우와 결합되기에 56개(14*4)가 됩니다.
- ▶ **Cross Join**의 결과를 보면 사원 테이블에 부서에 대한 상세정보가 결합되긴 했지만, 조인 될 때 아무런 조건을 제시하지 않았기에 사원 한 명에 대해서 **DEPT** 테이블의 4개의 로우와 결합된 형태이기에 **Cross Join**의 결과는 아무런 의미를 갖지 못합니다.
- ▶ 조인 결과가 의미를 갖으려면 조인할 때 조건을 지정해야 합니다.

상호 조인(Cross Join)

▶ 상호 조인(Cross Join)

- ▶ 조인 조건에 따라 조인의 종류가 결정되는데 다음은 조인의 종류를 정리한 표입니다.

종 류	설 명
Equi Join	동일 칼럼을 기준으로 조인합니다.
Non-Equi Join	동일 칼럼이 없이 다른 조건을 사용하여 조인합니다.
Outer Join	조인 조건에 만족하지 않는 행도 나타낸다.
Seif Join	한 테이블 내에서 조인합니다.

Equi Join

▶ Equi Join

- ▶ **EQUI JOIN**은 가장 많이 사용하는 조인 방법으로서 조인 대상이 되는 두 테이블에서 공통적으로 존재하는 컬럼의 값이 일치되는 행을 연결하여 결과를 생성하는 조인 방법입니다.
- ▶ 다음은 사원 정보를 출력할 때 각 사원들이 소속된 부서의 상세 정보를 출력하기 위해서 두 개의 테이블을 조인한 예입니다.

예

```
SELECT *  
FROM EMP INNER JOIN DEPT  
ON EMP.DEPTNO = DEPT.DEPTNO;
```

- ▶ 사원(**EMP**) 테이블과 부서(**DEPT**) 테이블의 공통 컬럼인 **DEPTNO**의 값이 일치(=)되는 조건을 **ON** 절에 기술하여 사용하였습니다.
- ▶ 테이블을 조인하려면 일치되는 공통 컬럼을 사용해야 한다고 하였습니다. 컬럼의 이름이 같게 되면 혼동이 오기 때문에 컬럼 이름 앞에 테이블 이름을 기술합니다.

Equi Join

▶ Equi Join

- ▶ 다음은 두 테이블을 조인한 결과입니다. 결과를 살펴보면 다음과 같이 부서 번호를 기준으로 같은 값을 가진 학생 테이블의 컬럼과 부서 테이블의 컬럼이 결합됩니다.

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	emp.DEPTNO	dept.DEPTNO	DNAME	LOC
1	7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800	<null>	20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	1981-02-22 00:00:00	1250	500	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	1981-04-02 00:00:00	2975	<null>	20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981-09-28 00:00:00	1250	1400	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	1981-05-01 00:00:00	2850	<null>	30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	1981-06-09 00:00:00	2450	<null>	10	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	ANALYST	7566	1987-04-19 00:00:00	3000	<null>	20	20	RESEARCH	DALLAS
9	7839	KING	PRESIDENT	<null>	1981-11-17 00:00:00	5000	<null>	10	10	ACCOUNTING	NEW YORK
10	7844	TURNER	SALESMAN	7698	1981-09-08 00:00:00	1500	0	30	30	SALES	CHICAGO
11	7876	ADAMS	CLERK	7788	1987-05-23 00:00:00	1100	<null>	20	20	RESEARCH	DALLAS
12	7900	JAMES	CLERK	7698	1981-12-03 00:00:00	950	<null>	30	30	SALES	CHICAGO
13	7902	FORD	ANALYST	7566	1981-12-03 00:00:00	3000	<null>	20	20	RESEARCH	DALLAS
14	7934	MILLER	CLERK	7782	1982-01-23 00:00:00	1300	<null>	10	10	ACCOUNTING	NEW YORK

Equi Join

▶ Equi Join

- ▶ 이름이 SCOTT인 사람의 부서명을 출력해봅시다.

예

```
SELECT ENAME, DNAME  
FROM EMP INNER JOIN DEPT  
ON EMP.DEPTNO=DEPT.DEPTNO  
WHERE ENAME='SCOTT';
```

	ENAME	DNAME
1	SCOTT	RESEARCH

- ▶ 두 테이블에 동일한 이름의 칼럼을 사용하면 어느 테이블 소속인지 불분명하기에 애매모호한 상태라는 오류 메시지가 출력됩니다.

예

```
SELECT ENAME, DNAME, DEPTNO  
FROM EMP INNER JOIN DEPT  
ON EMP.DEPTNO=DEPT.DEPTNO  
WHERE ENAME='SCOTT';
```

[23000][1052] Column 'DEPTNO' in field list is ambiguous

Equi Join

▶ Equi Join

- ▶ 이러한 문제를 해결하기 위한 방법이 있어야 합니다. 이렇게 동일한 이름의 컬럼은 컬럼명 앞에 테이블 이름을 명시적으로 기술함으로써 컬럼이 어느 테이블 소속인지 구분할 수 있게 됩니다.

예

```
SELECT ENAME, DEPT.DNAME, EMP.DEPTNO  
FROM EMP INNER JOIN DEPT  
ON EMP.DEPTNO=DEPT.DEPTNO  
WHERE ENAME='SCOTT';
```

	ENAME	DNAME	DEPTNO
1	SCOTT	RESEARCH	20

- ▶ 테이블 이름에 별칭을 붙이는 방법은 FROM 절 다음에 테이블 이름을 명시하고 공백을 둔 다음에 별칭을 지정하면 됩니다.

예

```
SELECT E.ENAME, D.DNAME, E.DEPTNO  
FROM EMP E INNER JOIN DEPT D  
ON E.DEPTNO=D.DEPTNO  
WHERE E.ENAME='SCOTT';
```

	ENAME	DNAME	DEPTNO
1	SCOTT	RESEARCH	20

Equi Join

▶ Equi Join

- ▶ 돌발문제 1 : 뉴욕에서 근무하는 사원의 이름과 급여를 출력하시오.
- ▶ 돌발문제 2 : **ACCOUNTING** 부서 소속 사원의 이름과 입사일을 출력하시오.
- ▶ 돌발문제 3 : 직급이 **MANAGER**인 사원의 이름, 부서명을 출력하시오.

Non-Equi Join

▶ Non-Equi Join

- ▶ Non-Equi Join은 조인 조건에 특정 범위 내에 있는지를 조사하기 위해서 **WHERE** 절에 조인 조건을 = 연산자 이외의 비교 연산자를 사용합니다.
- ▶ Non-Equi Join을 학습하기 전에 급여 등급 테이블(**SALGRADE**)을 살펴보겠습니다.

예

```
SELECT * FROM SALGRADE;
```

	GRADE ↕	LOSAL ↕	HISAL ↕
1	1	700	1200
2	2	1201	1400
3	3	1401	2000
4	4	2001	3000
5	5	3001	9999

Non-Equi Join

▶ Non-Equi Join

- ▶ 급여 등급 테이블(**salgrade**)에는 급여에 대한 등급을 다음과 같이 나누어 놓았습니다.
- ▶ 급여의 등급은 총 5등급으로 나누어져 있으며, 1등급은 급여가 **700**부터 **1200** 사이이고, 2등급은 **1201**부터 **1400** 사이이고, 3등급은 **1401**부터 **2000** 사이이고, 4등급은 **2001**부터 **3000**사이이고, 5등급이면 **3001**부터 **9999**사이입니다.
- ▶ 급여 등급을 5개로 나누어 놓은 **salgrade**에서 정보를 얻어 와서 각 사원의 급여 등급을 지정해보도록 합시다. 이를 위해서 사원(**emp**) 테이블과 급여 등급(**salgrade**) 테이블을 조인하도록 합시다. 다음은 사원의 급여가 몇 등급인지 살펴보는 예제입니다.

Non-Equi Join

▶ Non-Equi Join

예

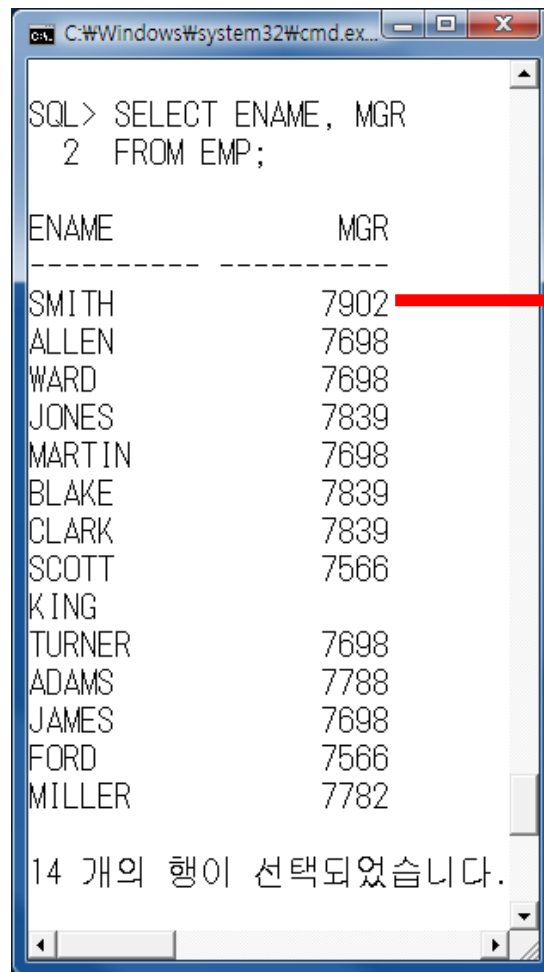
```
SELECT ENAME, SAL, GRADE  
FROM EMP INNER JOIN SALGRADE  
ON SAL BETWEEN LOSAL AND HISAL;
```

	ENAME	SAL	GRADE
1	SMITH	800	1
2	ALLEN	1600	3
3	WARD	1250	2
4	JONES	2975	4
5	MARTIN	1250	2
6	BLAKE	2850	4
7	CLARK	2450	4
8	SCOTT	3000	4
9	KING	5000	5
10	TURNER	1500	3
11	ADAMS	1100	1
12	JAMES	950	1
13	FORD	3000	4
14	MILLER	1300	2

Self Join

▶ Self Join

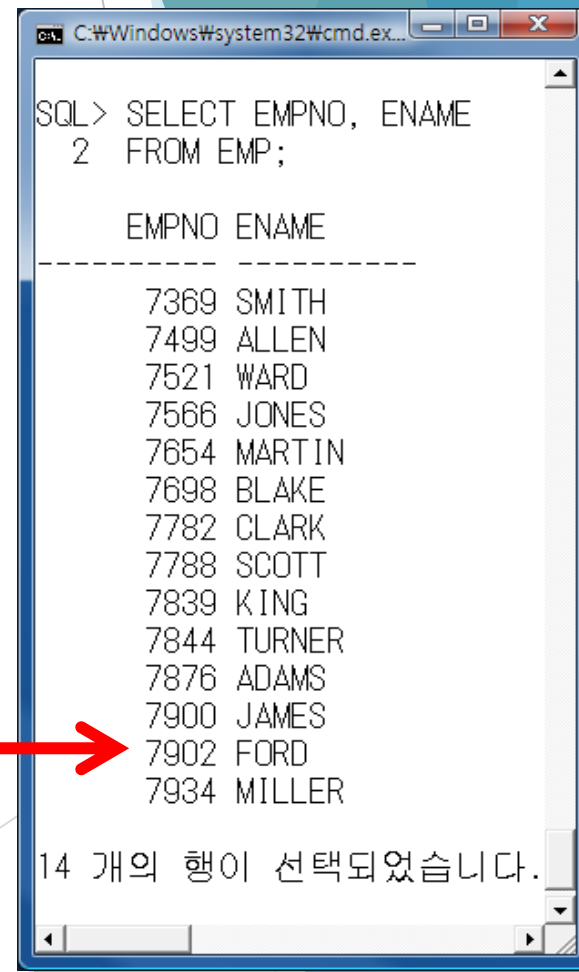
- ▶ 조인은 두 개 이상의 서로 다른 테이블을 서로 연결하는 것뿐만 아니라, 하나의 테이블 내에서 조인을 해야만 원하는 자료를 얻는 경우가 생깁니다.
- ▶ **Self Join**이란 말 그대로 자기 자신과 조인을 맺는 것을 말합니다.
- ▶ **Self Join**을 보다 구체적인 예를 통해서 알아보도록 합시다.
- ▶ **SMITH**의 매니저 이름이 무엇인지 알아내려면 어떻게 구해야 할까요?



```
SQL> SELECT ENAME, MGR
2 FROM EMP;
```

ENAME	MGR
SMITH	7902
ALLEN	7698
WARD	7698
JONES	7839
MARTIN	7698
BLAKE	7839
CLARK	7839
SCOTT	7566
KING	
TURNER	7698
ADAMS	7788
JAMES	7698
FORD	7566
MILLER	7782

14 개의 행이 선택되었습니다.



```
SQL> SELECT EMPNO, ENAME
2 FROM EMP;
```

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

14 개의 행이 선택되었습니다.

Self Join

▶ Self Join

- ▶ 아래와 같은 쿼리를 실행해 봅시다. 같은 테이블에 다른 **Alias**를 참조해서 같은 테이블을 연속해서 참조할 수 있습니다.

예

```
SELECT E1.ENAME employee, E2.ENAME manager  
FROM EMP E1 INNER JOIN EMP E2  
ON E1.MGR=E2.EMPNO;
```

	employee	manager
1	SMITH	FORD

Self Join

▶ Self Join

- ▶ 돌발문제 1 : 매니저가 **KING**인 직원들의 이름과 직급을 출력하시오.
- ▶ 돌발문제 2 : **SCOTT**과 동일한 근무지에서 근무하는 직원의 이름을 출력하시오.

Outer Join

▶ Outer Join

- ▶ Seif Join을 학습하면 특정 사원의 매니저 이름을 구했습니다.
- ▶ 결과를 꼼꼼히 살펴보면 이름이 **KING**인 사원 한사람의 정보가 빠져 있음을 확인할 수 있습니다.
- ▶ **KING**은 이 회사의 사장(**PRESIDENT**)으로 매니저가 존재하지 않으므로 **MGR** 컬럼 값이 **NULL**입니다. 사원 번호(**EMPNO**)가 **NULL**인 사원은 없으므로 조인 조건에 만족하지 않아서 **KING**은 Seif Join의 결과에서 배제되었습니다.
- ▶ 조인 조건에 만족하지 못하였더라도 해당 로우를 나타내고 싶을 때에 사용하는 것이 외부 조인 (**Outer Join**)입니다.

Outer Join

▶ Outer Join

- ▶ 아래와 같이 해 봅시다.

예

```
SELECT E1.ENAME employee, E2.ENAME manager  
FROM EMP E1 LEFT OUTER JOIN EMP E2  
ON E1.MGR=E2.EMPNO;
```

	employee	manager
1	SMITH	FORD
2	ALLEN	BLAKE
3	WARD	BLAKE
4	JONES	KING
5	MARTIN	BLAKE
6	BLAKE	KING
7	CLARK	KING
8	SCOTT	JONES
9	KING	<null>
10	TURNER	BLAKE
11	ADAMS	SCOTT
12	JAMES	BLAKE
13	FORD	JONES
14	MILLER	CLARK

Outer Join

▶ Outer Join

- ▶ DEPT 테이블과 닮은 DEPT01 테이블을 만들어 보겠습니다. 혹시 DEPT01 테이블이 존재한다면 DEPT01 테이블이 생성되지 않으므로 DROP 명령어로 삭제 후 생성하도록 합니다.

```
DROP TABLE DEPT01;
```

- ▶ 부서번호와 부서명을 컬럼으로 갖는 DEPT01 테이블을 생성합니다.

```
CREATE TABLE DEPT01(  
  DEPTNO INT(2),  
  DNAME VARCHAR(14));
```

- ▶ 데이터를 추가합니다

```
INSERT INTO DEPT01 VALUES(10, 'ACCOUNTING');  
INSERT INTO DEPT01 VALUES (20, 'RESEARCH');
```

Outer Join

▶ Outer Join

- ▶ 동일한 방법으로 DEPT02 테이블을 생성합니다.

```
DROP TABLE DEPT02;
```

```
CREATE TABLE DEPT02(  
  DEPTNO INT(2),  
  DNAME VARCHAR(14));
```

```
INSERT INTO DEPT02 VALUES(10, 'ACCOUNTING');  
INSERT INTO DEPT02 VALUES (30, 'SALES');
```

```
SELECT * FROM DEPT02;
```

Outer Join

▶ Outer Join

- ▶ DEPT01 테이블의 20번 부서와 조인할 부서번호가 DEPT02에는 없지만, 20번 부서도 출력되도록 하기 위해서 DEPT01 테이블이 왼쪽에 존재하기에 LEFT OUTER JOIN을 사용합시다

예

```
SELECT *  
FROM DEPT01 LEFT OUTER JOIN DEPT02  
ON DEPT01.DEPTNO = DEPT02.DEPTNO;
```

dept01.DEPTNO	dept01.DNAME	dept02.DEPTNO	dept02.DNAME
10	ACCOUNTING	10	ACCOUNTING
20	RESEARCH	<null>	<null>

Outer Join

▶ Outer Join

- ▶ DEPT02 테이블에만 있는 30번 부서까지 출력되도록 하기 위해서 **RIGHT OUTER JOIN**을 사용합니다.

예

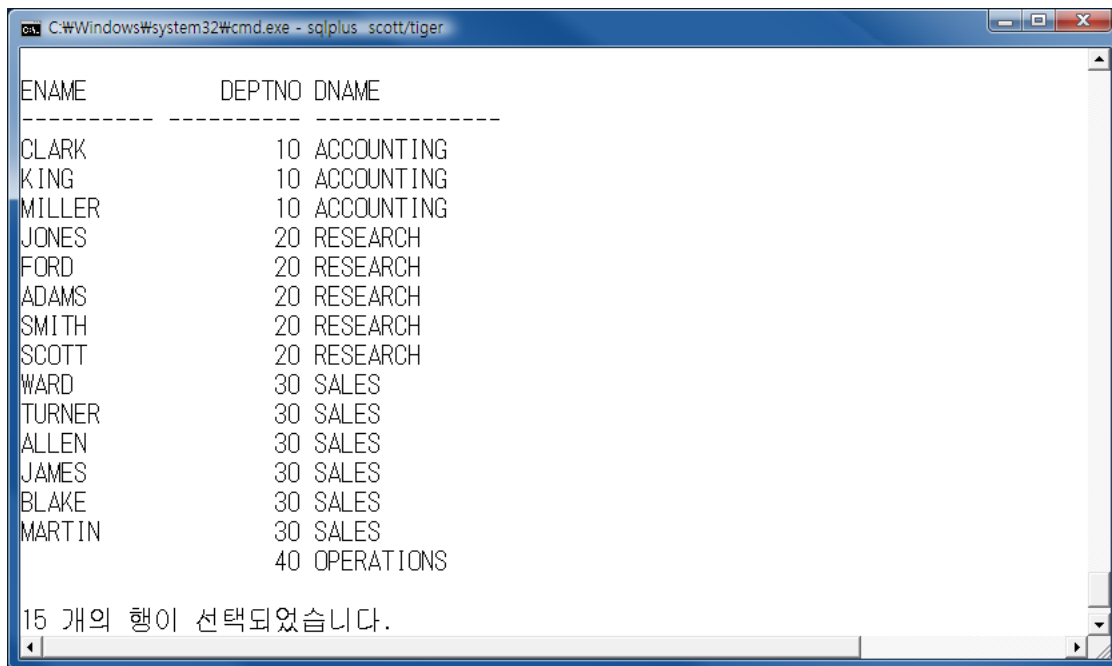
```
SELECT *  
FROM DEPT01 RIGHT OUTER JOIN DEPT02  
USING(DEPTNO);
```

	DEPTNO	DNAME	DNAME
1	10	ACCOUNTING	ACCOUNTING
2	30	SALES	<null>

Outer Join

▶ Outer Join

- ▶ **돌발문제 1** : 사원 테이블과 부서 테이블을 조인하여 사원이름과 부서번호와 부서명을 출력하도록 합시다. 부서 테이블의 **40번** 부서와 조인할 사원 테이블의 부서번호가 없지만, 아래 그림과 같이 **40번** 부서의 부서 이름도 출력되도록 쿼리문을 작성해보시오.



C:\Windows\system32\cmd.exe - sqlplus scott/tiger

ENAME	DEPTNO	DNAME
CLARK	10	ACCOUNTING
KING	10	ACCOUNTING
MILLER	10	ACCOUNTING
JONES	20	RESEARCH
FORD	20	RESEARCH
ADAMS	20	RESEARCH
SMITH	20	RESEARCH
SCOTT	20	RESEARCH
WARD	30	SALES
TURNER	30	SALES
ALLEN	30	SALES
JAMES	30	SALES
BLAKE	30	SALES
MARTIN	30	SALES
	40	OPERATIONS

15 개의 행이 선택되었습니다.

Union

▶ UNION

- ▶ LEFT OUTER JOIN과 RIGHT OUTER JOIN 을 합치려면 Union을 사용할 수 있습니다.
- ▶ Union은 중복된 열을 제거하며 사용이 가능합니다. 아래의 쿼리를 날려 실행해 보세요

예

```
SELECT *  
FROM DEPT01 LEFT OUTER JOIN DEPT02  
USING(DEPTNO)  
UNION  
SELECT *  
FROM DEPT01 RIGHT OUTER JOIN DEPT02  
USING(DEPTNO);
```

	DEPTNO	DNAME	DNAME
1	10	ACCOUNTING	ACCOUNTING
2	20	RESEARCH	<null>
3	30	SALES	<null>

Union

▶ UNION ALL

- ▶ 중복된 열을 전부 출력하고 싶다면 UNION ALL을 사용할 수 있습니다.

예

```
SELECT *  
FROM DEPT01 LEFT OUTER JOIN DEPT02  
USING(DEPTNO)  
UNION ALL  
SELECT *  
FROM DEPT01 RIGHT OUTER JOIN DEPT02  
USING(DEPTNO);
```

	DEPTNO	DNAME	DNAME
1	10	ACCOUNTING	ACCOUNTING
2	20	RESEARCH	<null>
3	10	ACCOUNTING	ACCOUNTING
4	30	SALES	<null>