



07장. 조인

이번 장에서는 한개 이상의 테이블에서 데이터를 조회하기 위해서 사용되는 조인에 대한 기본 개념과 다양한 조인 방법에 대해서 살펴보겠습니다.

이 장에서 다룰 내용



1 조인의 필요성

2 Cross Join

3 Equi Join

4 Non-Equi Join

5 Seif Join

이 장에서 다룰 내용



6 Outer Join

7 ANSI Join

01. 조인의 필요성



- ❖ 특정 부서 번호에 대한 부서이름은 무엇인지는 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>
```

01. 조인의 필요성



- ❖ SCOTT인 사원이 소속되어 있는 부서의 이름이 무엇인지 알아보려고 합니다.
- ❖ SCOTT이란 사원의 부서명을 알아내는 일 역시 사원 테이블에서 SCOTT이 소속된 부서 번호를 알아낸 후에 부서 테이블에서 해당 부서 번호에 대한 부서명을 얻어 와야 합니다.

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

DEPTNO
-----
20

SQL>
```

```
C:\Windows\system32\cmd.exe - sqlplus scot...
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 SQL에서는 두 개 이상의 테이블을 결합해야만 원하는 결과를 얻을 수 있을 때 한 번의 질의로 원하는 결과를 얻을 수 있는 조인 기능을 제공합니다.

02. Cross Join



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

예

```
SELECT *  
FROM EMP, DEPT;
```

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

SQL> SELECT *

2 FROM EMP, DEPT;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7369	SMITH	CLERK	7802	09/12/17	800		20	10	ACCOUNTING	NEW YORK
7489	ALLEN	SALESMAN	7898	01/02/20	1600	300	20	10	ACCOUNTING	NEW YORK
7521	WARD	SALESMAN	7898	01/02/22	1250	500	30	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	01/04/02	2975		20	10	ACCOUNTING	NEW YORK
7584	MARTIN	SALESMAN	7898	01/09/28	1250		30	10	ACCOUNTING	NEW YORK
7698	BLAKE	MANAGER	7839	01/05/01	2850		30	10	ACCOUNTING	NEW YORK
7782	CLARK	MANAGER	7839	01/06/09	2450		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	07/04/19	3000		20	10	ACCOUNTING	NEW YORK
7839	KING	PRESIDENT	7802	09/12/17	5000		10	10	ACCOUNTING	NEW YORK
7844	TURNER	SALESMAN	7898	01/09/08	1500	0	30	10	ACCOUNTING	NEW YORK
7876	ADAMS	CLERK	7788	07/05/23	1100		20	10	ACCOUNTING	NEW YORK

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7900	JAMES	CLERK	7898	01/12/03	950		30	10	ACCOUNTING	NEW YORK
7902	FORD	ANALYST	7802	01/12/03	3000		20	10	ACCOUNTING	NEW YORK
7938	MILLER	CLERK	7782	01/12/23	1300		10	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	08/12/17	800		20	20	RESEARCH	DALLAS
7489	ALLEN	SALESMAN	7898	01/02/20	1600	300	20	20	RESEARCH	DALLAS
7521	WARD	SALESMAN	7898	01/02/22	1250	500	30	20	RESEARCH	DALLAS
7566	JONES	MANAGER	7839	01/04/02	2975		20	20	RESEARCH	DALLAS
7584	MARTIN	SALESMAN	7898	01/09/28	1250	1400	30	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	01/05/01	2850		30	20	RESEARCH	DALLAS
7782	CLARK	MANAGER	7839	01/06/09	2450		10	20	RESEARCH	DALLAS
7788	SCOTT	ANALYST	7566	07/04/19	3000		20	20	RESEARCH	DALLAS

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7938	KING	PRESIDENT	81/11/17	5000		10	10	20	RESEARCH	DALLAS
7844	TURNER	SALESMAN	7898	01/09/08	1500	0	30	20	RESEARCH	DALLAS
7876	ADAMS	CLERK	7788	07/05/23	1100		20	20	RESEARCH	DALLAS
7900	JAMES	CLERK	7898	01/12/03	950		30	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	01/12/03	3000		20	20	RESEARCH	DALLAS
7938	MILLER	CLERK	7782	02/01/23	1300		10	20	RESEARCH	DALLAS
7489	ALLEN	SALESMAN	7898	01/02/20	1600	300	20	30	SALES	CHICAGO
7521	WARD	SALESMAN	7898	01/02/22	1250	500	30	30	SALES	CHICAGO
7566	JONES	MANAGER	7839	01/04/02	2975		20	30	SALES	CHICAGO
7584	MARTIN	SALESMAN	7898	01/09/28	1250	1400	30	30	SALES	CHICAGO

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7698	BLAKE	MANAGER	7839	81/05/01	2850		30	30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	81/06/09	2450		10	30	SALES	CHICAGO
7788	SCOTT	ANALYST	7566	87/04/19	3000		20	30	SALES	CHICAGO
7839	KING	PRESIDENT		81/11/17	5000		10	30	SALES	CHICAGO
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	87/05/23	1100		20	30	SALES	CHICAGO
7900	JAMES	CLERK	7698	81/12/03	950		30	30	SALES	CHICAGO
7902	FORD	ANALYST	7566	81/12/03	3000		20	30	SALES	CHICAGO
7934	MILLER	CLERK	7782	82/01/23	1300		10	30	SALES	CHICAGO
7869	SMITH	CLERK	7902	80/12/17	800		20	40	OPERATIONS	BOSTON
7439	ALLEN	SALESMAN	7698	81/02/20	1600	300	30	40	OPERATIONS	BOSTON
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7521	WARD	SALESMAN	7698	81/02/22	1250	500	30	40	OPERATIONS	BOSTON
7566	JONES	MANAGER	7839	81/04/02	2975		20	40	OPERATIONS	BOSTON
7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30	40	OPERATIONS	BOSTON
7698	BLAKE	MANAGER	7839	81/05/01	2850		30	40	OPERATIONS	BOSTON
7782	CLARK	MANAGER	7839	81/06/09	2450		10	40	OPERATIONS	BOSTON
7788	SCOTT	ANALYST	7566	87/04/19	3000		20	40	OPERATIONS	BOSTON
7839	KING	PRESIDENT		81/11/17	5000		10	40	OPERATIONS	BOSTON
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30	40	OPERATIONS	BOSTON
7876	ADAMS	CLERK	7788	87/05/23	1100		20	40	OPERATIONS	BOSTON
7900	JAMES	CLERK	7698	81/12/03	950		30	40	OPERATIONS	BOSTON
7902	FORD	ANALYST	7566	81/12/03	3000		20	40	OPERATIONS	BOSTON
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7934	MILLER	CLERK	7782	82/01/23	1300		10	40	OPERATIONS	BOSTON

02. Cross Join



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

02. 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.

03. Equi Join



- ❖ EQUI JOIN은 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, DEPT  
WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

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



03. Equi Join

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



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

```
SQL> SELECT *  
2 FROM EMP, DEPT  
3 WHERE EMP.DEPTNO=DEPT.DEPTNO;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7782	CLARK	MANAGER	7839	81/06/09	2450		10	10	ACCOUNTING	NEW YORK
7839	KING	PRESIDENT		81/11/17	5000		10	10	ACCOUNTING	NEW YORK
7934	MILLER	CLERK	7782	82/01/23	1300		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	81/04/02	2975		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	81/12/03	3000		20	20	RESEARCH	DALLAS
7876	ADAMS	CLERK	7788	87/05/23	1100		20	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	80/12/17	800		20	20	RESEARCH	DALLAS
7788	SCOTT	ANALYST	7566	87/04/19	3000		20	20	RESEARCH	DALLAS
7521	WARD	SALESMAN	7698	81/02/22	1250	500	30	30	SALES	CHICAGO
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30	30	SALES	CHICAGO
7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30	30	SALES	CHICAGO
7900	JAMES	CLERK	7698	81/12/03	950		30	30	SALES	CHICAGO
7698	BLAKE	MANAGER	7839	81/05/01	2850		30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30	30	SALES	CHICAGO

14 개의 행이 선택되었습니다.

3.1 Equi Join에 AND 연산하기

- ❖ 이름이 SCOTT인 사람의 부서명을 출력해봅시다.



예

```
SELECT ENAME, DNAME
FROM EMP, DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO
AND ENAME='SCOTT';
```

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

SQL> -- 이름이 SCOTT인 사원의 부서명을 알아봅시다.
SQL> SELECT ENAME, DNAME
      2  FROM EMP, DEPT
      3  WHERE EMP.DEPTNO=DEPT.DEPTNO
      4  AND ENAME='SCOTT';

ENAME      DNAME
-----
SCOTT      RESEARCH
```

3.2 컬럼명의 모호성 해결



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

예

```
SELECT ENAME, DNAME, DEPTNO  
FROM EMP, DEPT  
WHERE EMP.DEPTNO = DEPT.DEPTNO  
AND ENAME='SCOTT';
```

```
C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger  
  
SQL> -- 이름이 SCOTT인 사원의 부서명을 알아내어 사원이름 부서명 부서번호를 출력하시오.  
SQL> SELECT ENAME, DNAME, DEPTNO  
2 FROM EMP, DEPT  
3 WHERE EMP.DEPTNO=DEPT.DEPTNO  
4 AND ENAME='SCOTT';  
SELECT ENAME, DNAME, DEPTNO  
*  
1행에 오류:  
ORA-00918: 열의 정의가 애매합니다
```

3.2 컬럼명의 모호성 해결

- ❖ 이러한 문제를 해결하기 위한 방법이 있어야 합니다. 이렇게 동일한 이름의 컬럼은 컬럼 명 앞에 테이블 명을 명시적으로 기술함으로써 컬럼이 어느 테이블 소속인지 구분할 수 있게 됩니다.



예

```
SELECT EMP.ENAME, DEPT.DNAME, EMP.DEPTNO  
FROM EMP, DEPT  
WHERE EMP.DEPTNO=DEPT.DEPTNO  
AND ENAME='SCOTT';
```

```
C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger  
  
SQL> -- 컬럼명 앞에 테이블 명을 명시하기  
SQL> SELECT EMP.ENAME, DEPT.DNAME, EMP.DEPTNO  
2 FROM EMP, DEPT  
3 WHERE EMP.DEPTNO=DEPT.DEPTNO  
4 AND ENAME='SCOTT';  
  
ENAME      DNAME      DEPTNO  
-----  
SCOTT      RESEARCH      20
```

3.3 테이블에 별칭 부여하기

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

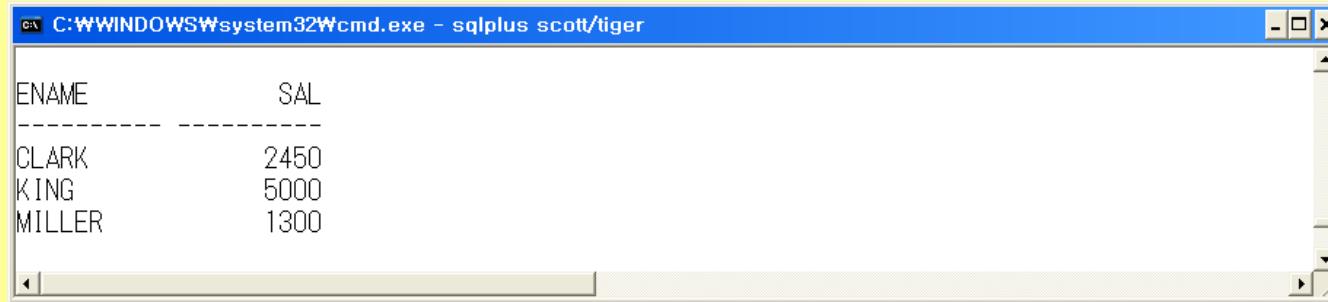


예

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

탄탄히 다지기

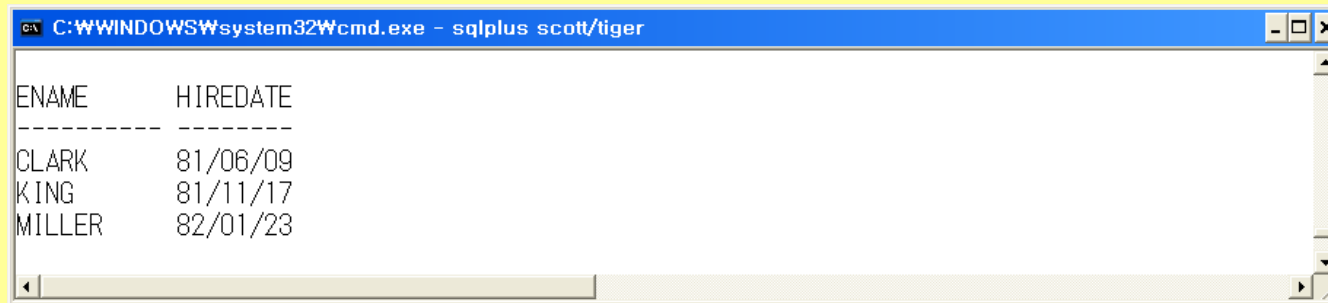
1. 뉴욕에서 근무하는 사원의 이름과 급여를 출력하시오.



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The window displays the output of a SQL query. The output is a table with two columns: "ENAME" and "SAL". The data rows are: CLARK (2450), KING (5000), and MILLER (1300). The table is separated from the header by a dashed line.

ENAME	SAL
CLARK	2450
KING	5000
MILLER	1300

2. ACCOUNTING 부서 소속 사원의 이름과 입사일을 출력하시오.

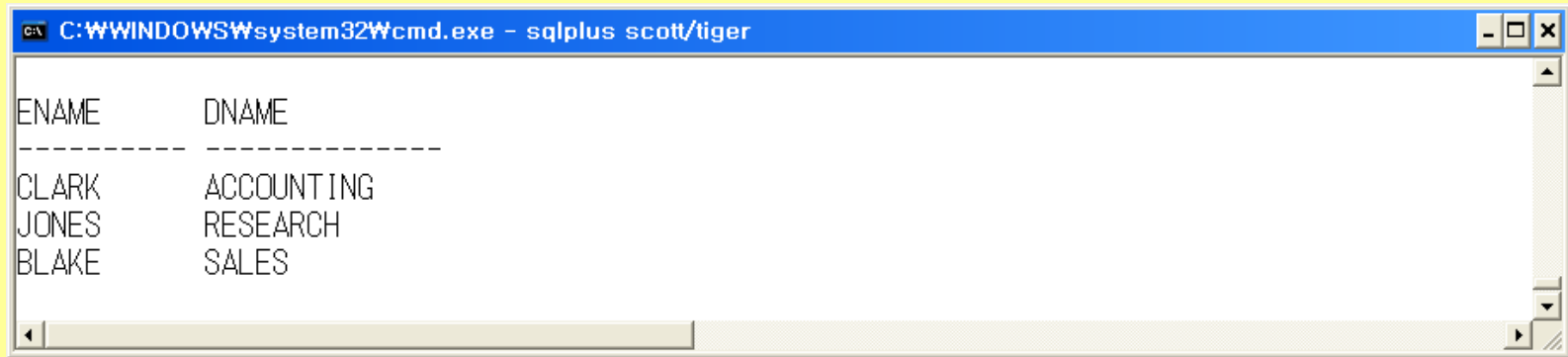


A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The window displays the output of a SQL query. The output is a table with two columns: "ENAME" and "HIREDATE". The data rows are: CLARK (81/06/09), KING (81/11/17), and MILLER (82/01/23). The table is separated from the header by a dashed line.

ENAME	HIREDATE
CLARK	81/06/09
KING	81/11/17
MILLER	82/01/23

탄탄히 다지기

3. 직급이 MANAGER인 사원의 이름, 부서명을 출력하시오.



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The window displays the output of an SQL query, which is a table with two columns: ENAME and DNAME. The table contains three rows of data: CLARK in the ACCOUNTING department, JONES in the RESEARCH department, and BLAKE in the SALES department. The output is formatted with a dashed line separating the header from the data rows.

ENAME	DNAME
CLARK	ACCOUNTING
JONES	RESEARCH
BLAKE	SALES

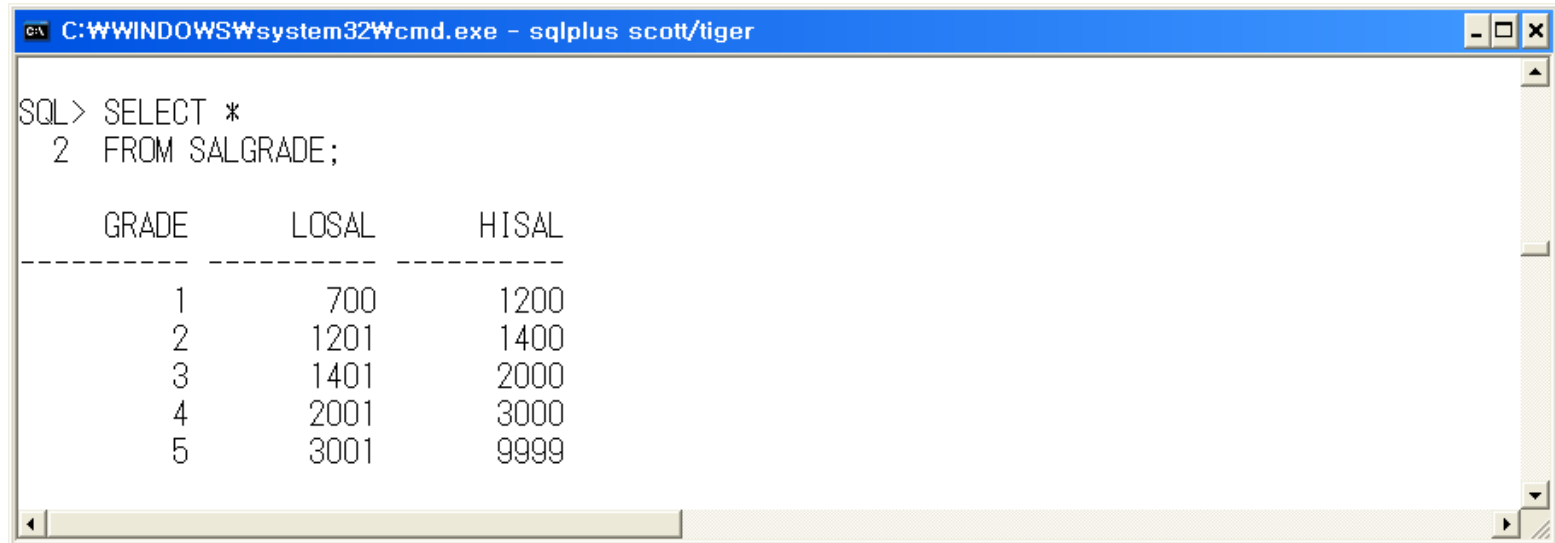
04. Non-Equi Join



- ❖ 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;
```



```
SQL> SELECT *
2 FROM SALGRADE;
```

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

04. Non-Equi Join



- ❖ 급여 등급 테이블(salgrade)에는 급여에 대한 등급을 다음과 같이 나누어 놓았습니다.
- ❖ 급여의 등급은 총 5등급으로 나누어져 있으며, 1등급은 급여가 700부터 1200 사이이고, 2등급은 1201부터 1400 사이이고, 3등급은 1401부터 2000 사이이고, 4등급은 2001부터 3000사이이고, 5등급이면 3001부터 9999사이입니다.
- ❖ 급여 등급을 5개로 나누어 놓은 salgrade에서 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.

04. Non-Equi Join



- ❖ 급여 등급 테이블(salgrade)에는 급여에 대한 등급을 다음과 같이 나누어 놓았습니다.
- ❖ 급여의 등급은 총 5등급으로 나누어져 있으며, 1등급은 급여가 700부터 1200 사이이고, 2등급은 1201부터 1400 사이이고, 3등급은 1401부터 2000 사이이고, 4등급은 2001부터 3000사이이고, 5등급이면 3001부터 9999사이입니다.

04. Non-Equi Join



예

```
SELECT ENAME, SAL, GRADE
FROM EMP, SALGRADE
WHERE SAL BETWEEN LOSAL AND HISAL;
```

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

SQL> SELECT ENAME, SAL, GRADE
2  FROM EMP, SALGRADE
3  WHERE SAL BETWEEN LOSAL AND HISAL;
```

ENAME	SAL	GRADE
SMITH	800	1
JAMES	950	1
ADAMS	1100	1
WARD	1250	2
MARTIN	1250	2
MILLER	1300	2
TURNER	1500	3
ALLEN	1600	3
CLARK	2450	4
BLAKE	2850	4
JONES	2975	4

ENAME	SAL	GRADE
SCOTT	3000	4
FORD	3000	4
KING	5000	5

14 개의 행이 선택되었습니다.

05. Seif Join



- ❖ 조인은 두 개 이상의 서로 다른 테이블을 서로 연결하는 것뿐만 아니라, 하나의 테이블 내에서 조인을 해야만 원하는 자료를 얻는 경우가 생깁니다.
- ❖ Seif Join이란 말 그대로 자기 자신과 조인을 맺는 것을 말합니다.
- ❖ Seif Join을 보다 구체적인 예를 통해서 알아보도록 합시다.
- ❖ SMITH의 매니저 이름이 무엇인지 알아내려면 어떻게 구해야 할까요?

05. Self Join



```
C:\Windows\system32\cmd.exe
SQL> SELECT ENAME, MGR
2 FROM EMP;
```

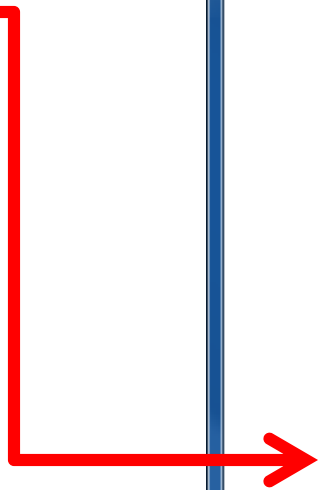
ENAME	MGR
SMITH	7902
ALLEN	7698
WARD	7698
JONES	7839
MARTIN	7698
BLAKE	7839
CLARK	7839
SCOTT	7566
KING	
TURNER	7698
ADAMS	7788
JAMES	7698
FORD	7566
MILLER	7782

14 개의 행이 선택되었습니다.

```
C:\Windows\system32\cmd.exe
SQL> SELECT EMPNO, ENAME
2 FROM EMP;
```

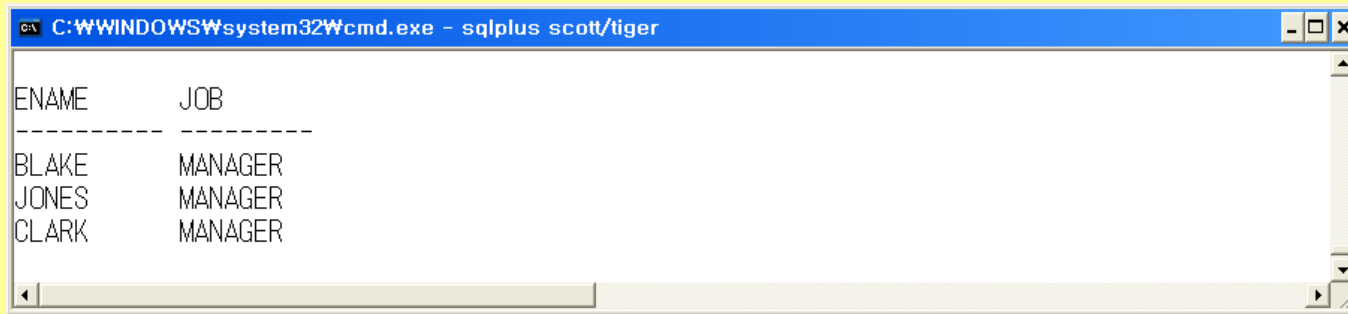
EMPNO	ENAME
7369	SMITH
7499	ALLEN
7521	WARD
7566	JONES
7654	MARTIN
7698	BLAKE
7782	CLARK
7788	SCOTT
7839	KING
7844	TURNER
7876	ADAMS
7900	JAMES
7902	FORD
7934	MILLER

14 개의 행이 선택되었습니다.



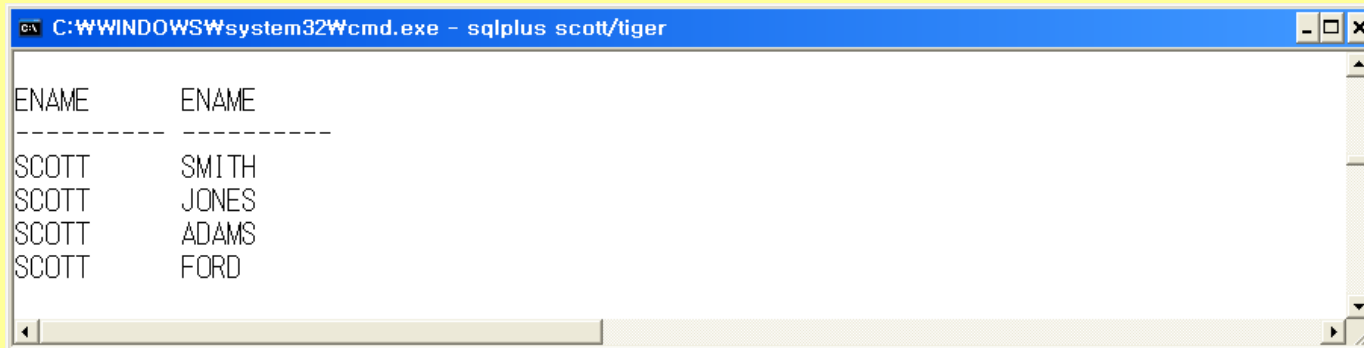
<탄탄히 다지기>

4. 매니저가 KING인 직원들의 이름과 직급을 출력하시오.



ENAME	JOB
BLAKE	MANAGER
JONES	MANAGER
CLARK	MANAGER

5. SCOTT과 동일한 근무지에서 근무하는 직원의 이름을 출력하시오.



ENAME	ENAME
SCOTT	SMITH
SCOTT	JONES
SCOTT	ADAMS
SCOTT	FORD

06. Outer Join



- ❖ Seif Join을 학습하면 특정 사원의 매니저 이름을 구했습니다. 결과를 꼼꼼히 살펴보면 이름이 KING인 사원 한사람의 정보가 빠져 있음을 확인할 수 있습니다.
- ❖ KING은 이 회사의 사장(PRESIDENT)으로 매니저가 존재하지 않으므로 MGR 컬럼 값이 NULL 입니다. 사원 번호(EMPNO)가 NULL인 사원은 없으므로 조인 조건에 만족하지 않아서 KING은 Seif Join의 결과에서 배제되었습니다.

06. Outer Join



- ❖ Self Join을 학습하면 특정 사원의 매니저 이름을 구했습니다.
- ❖ 결과를 꼼꼼히 살펴보면 이름이 KING인 사원 한사람의 정보가 빠져 있음을 확인할 수 있습니다.
- ❖ KING은 이 회사의 사장(PRESIDENT)으로 매니저가 존재하지 않으므로 MGR 컬럼 값이 NULL 입니다. 사원 번호(EMPNO)가 NULL인 사원은 없으므로 조인 조건에 만족하지 않아서 KING은 Self Join의 결과에서 배제되었습니다.
- ❖ 조인 조건에 만족하지 못하였더라도 해당 로우를 나타내고 싶을 때에 사용하는 것이 외부 조인(Outer Join)입니다.
- ❖ 외부 조인은 NULL 값이기에 배제된 행을 결과에 포함시킬 수 있으며 다음과 같이 “(+)” 기호를 조인 조건에서 정보가 부족한 컬럼 이름 뒤에 덧붙입니다.
- ❖ 사원 번호(EMPNO)가 NULL인 사원은 없으므로 manager.empno 뒤에 “(+)” 기호를 덧붙입니다.

06. Outer Join



예

```
SELECT employee.ename || '의 매니저는 '
      || manager.ename || '입니다.'
FROM emp employee, emp manager
WHERE employee.mgr = manager.empno(+);
```

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

SQL> SELECT EMPLOYEE.ENAME || '의 매니저는 ' || MANAGER.ENAME || '입니다.'
2  FROM EMP EMPLOYEE, EMP MANAGER
3  WHERE EMPLOYEE.MGR=MANAGER.EMPNO(+);

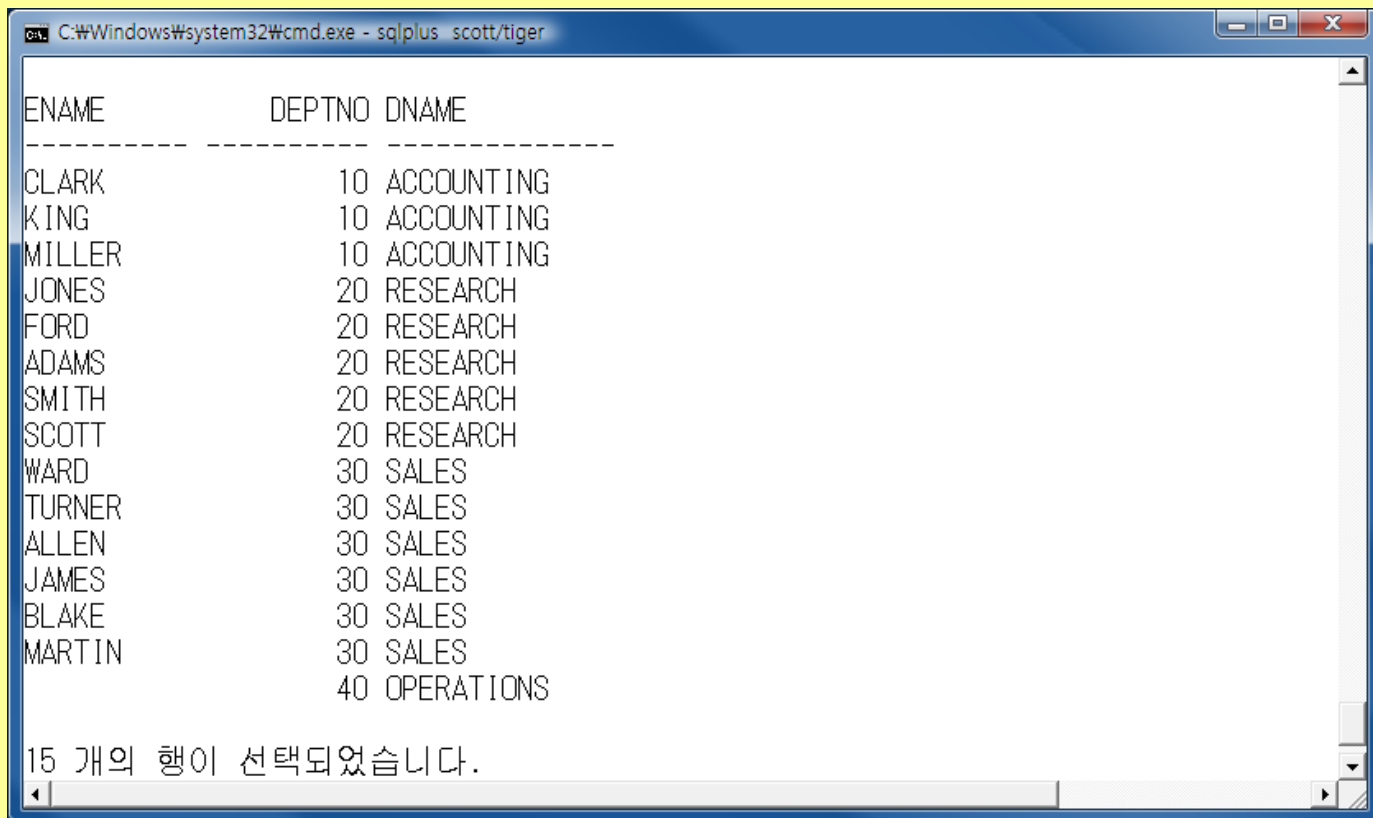
EMPLOYEE.ENAME || '의 매니저는 ' || MANAGER.EN
-----
FORD의 매니저는 JONES 입니다.
SCOTT의 매니저는 JONES 입니다.
JAMES의 매니저는 BLAKE 입니다.
TURNER의 매니저는 BLAKE 입니다.
MARTIN의 매니저는 BLAKE 입니다.
WARD의 매니저는 BLAKE 입니다.
ALLEN의 매니저는 BLAKE 입니다.
MILLER의 매니저는 CLARK 입니다.
ADAMS의 매니저는 SCOTT 입니다.
CLARK의 매니저는 KING 입니다.
BLAKE의 매니저는 KING 입니다.

EMPLOYEE.ENAME || '의 매니저는 ' || MANAGER.EN
-----
JONES의 매니저는 KING 입니다.
SMITH의 매니저는 FORD 입니다.
KING의 매니저는   입니다.

14 개의 행이 선택되었습니다.
```

<탄탄히 다지기>

6. 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 작성해 보시오.



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.

07. ANSI Join



❖ 7.1 ANSI Cross Join

```
SELECT *  
FROM EMP CROSS JOIN DEPT;
```

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

```
SQL> SELECT *  
2 FROM EMP CROSS JOIN DEPT;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7369	SMITH	CLERK	7902	80/12/17	800		20	10	ACCOUNTING	NEW YORK
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									
7369	SMITH									

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

7934	MILLER	CLERK	7782	82/01/23	1300		10	30	SALES	CHICAGO
7369	SMITH	CLERK	7902	80/12/17	800		20	40	OPERATIONS	BOSTON
7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30	40	OPERATIONS	BOSTON
7521	WARD	SALESMAN	7698	81/02/22	1250	500	30	40	OPERATIONS	BOSTON
7566	JONES	MANAGER	7839	81/04/02	2975		20	40	OPERATIONS	BOSTON
7654	MARTIN	SALESMAN	7698	81/09/28	1250		30	40	OPERATIONS	BOSTON
7698	BLAKE	MANAGER	7839	81/05/01	2850		30	40	OPERATIONS	BOSTON
7782	CLARK	MANAGER	7839	81/06/09	2450		10	40	OPERATIONS	BOSTON
7788	SCOTT	ANALYST	7566	87/04/19	3000		20	40	OPERATIONS	BOSTON
7839	KING	PRESIDENT		81/11/17	5000		10	40	OPERATIONS	BOSTON
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30	40	OPERATIONS	BOSTON
7876	ADAMS	CLERK	7788	87/05/23	1100		20	40	OPERATIONS	BOSTON
7900	JAMES	CLERK	7698	81/12/03	950		30	40	OPERATIONS	BOSTON
7902	FORD	ANALYST	7566	81/12/03	3000		20	40	OPERATIONS	BOSTON
7934	MILLER	CLERK	7782	82/01/23	1300		10	40	OPERATIONS	BOSTON

56 개의 행이 선택되었습니다.

7.2 ANSI Inner Join



- ❖ 새로운 ANSI 조인은 FROM 다음에 INNER JOIN 이란 단어를 사용하여 조인할 테이블 이름을 명시하고 ON 절을 사용하여 조인 조건을 명시하여 다음과 같이 작성합니다.

```
SELECT * FROM table1 INNER JOIN table2  
ON table1.column1 = table2.column2
```

- ❖ ANSI 조인에서는 조인 정보를 ON절에 기술하여 조인 조건을 명확하게 지정하고 다른 조건에 대해서는 WHERE 구문에서 지정하면 됩니다.

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

7.2 ANSI Inner Join



❖ USING을 이용한 조인 조건 지정하기

- 두 테이블에 각각 조인을 정의한 컬럼의 이름이 동일하다면 USING 절에서 조인할 컬럼을 지정하여 구문을 더 간단하게 표현할 수 있습니다.

```
SELECT * FROM table1 JOIN table2  
USING (공통컬럼)
```

- EMP와 DEPT에 DEPTNO 라는 같은 이름의 컬럼이 있기 때문에 다음과 같이 간단하게 조인문을 기술할 수 있습니다.

```
SELECT EMP.ENAME, DEPT.DNAME  
FROM EMP INNER JOIN DEPT  
USING (DEPTNO);
```

7.2 ANSI Inner Join



❖ NATURAL Join

- 두 테이블에 각각 조인을 정의한 컬럼의 이름이 동일하다면 USING 절에서 조인할 컬럼을 지정하여 구문을 더 간단하게 표현할 수 있습니다.

```
SELECT * FROM table1 NATURAL JOIN table2
```

- EMP와 DEPT에 DEPTNO 라는 같은 이름의 컬럼이 있기 때문에 다음과 같이 간단하게 조인문을 기술할 수 있습니다.

```
SELECT EMP.ENAME, DEPT.DNAME  
FROM EMP NATURAL JOIN DEPT;
```

7.3 ANSI Outer Join



- ❖ 새로운 ANSI 구문에서 Outer Join은 LEFT Outer Join, RIGHT Outer Join 그리고 FULL Outer Join 세 가지 타입의 조인을 제공합니다.

```
SELECT * FROM table1  
[LEFT | RIGHT | FULL] Outer Join table2
```

- ❖ Outer Join은 이미 설명했듯이 어느 한쪽 테이블에는 해당하는 데이터가 존재하는데 다른 쪽 테이블에는 데이터가 존재하지 않을 경우 그 데이터가 출력되지 않는 문제점을 해결하기 위해 사용하는 조인 기법입니다.

〈실습하기〉 다양한 Outer Join

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

```
DROP TABLE DEPT01;
```

2. 부서번호와 부서명을 컬럼으로 갖는 DEPT01 테이블을 생성합니다.

```
CREATE TABLE DEPT01(  
  DEPTNO NUMBER(2),  
  DNAME VARCHAR2(14));
```

3. 데이터를 추가합니다.

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

〈실습하기〉 다양한 Outer Join

4. 동일한 방법으로 DEPT02 테이블을 생성합니다.

```
DROP TABLE DEPT01;
```

```
CREATE TABLE DEPT02(  
  DEPTNO NUMBER(2),  
  DNAME VARCHAR2(14));
```

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

```
SELECT * FROM DEPT02;
```

LEFT OUTER JOIN

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



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

```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger  
  
SQL> SELECT *  
2 FROM DEPT01 LEFT OUTER JOIN DEPT02  
3 USING(DEPTNO);  
  
DEPTNO DNAME      DNAME  
-----  
10 ACCOUNTING    ACCOUNTING  
20 RESEARCH
```

SQL>

RIGHT OUTER JOIN

- ❖ DEPT02 테이블에만 있는 30번 부서까지 출력되도록 하기 위해서 RIGHT OUTER JOIN을 사용합시다.



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

```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger  
  
SQL> SELECT *  
2 FROM DEPT01 RIGHT OUTER JOIN DEPT02  
3 USING(DEPTNO);  
  
DEPTNO DNAME      DNAME  
-----  
10 ACCOUNTING    ACCOUNTING  
30              SALES  
  
SQL>
```

FULL OUTER JOIN

- ❖ FULL OUTER JOIN은 LEFT OUTER JOIN, RIGHT OUTER JOIN 을 합한 형태라고 볼 수 있습니다.



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

```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger  
  
SQL> SELECT *  
2 FROM DEPT01 FULL OUTER JOIN DEPT02  
3 USING(DEPTNO);  
  
DEPTNO DNAME      DNAME  
-----  
10 ACCOUNTING    ACCOUNTING  
30              SALES  
20 RESEARCH
```



Thank You !

Dynamic_오라클 11g + PL/SQL 입문 7장