



Join

1. 조인의 필요성

- ❖ 특정 부서 번호에 대한 부서이름이 무엇인지는 부서(DEPT) 테이블에 있는 경우 특정 사원에 대한 부서명을 알아내기 위해서는 부서 테이블에서 정보를 가져와야 함

```
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>
```

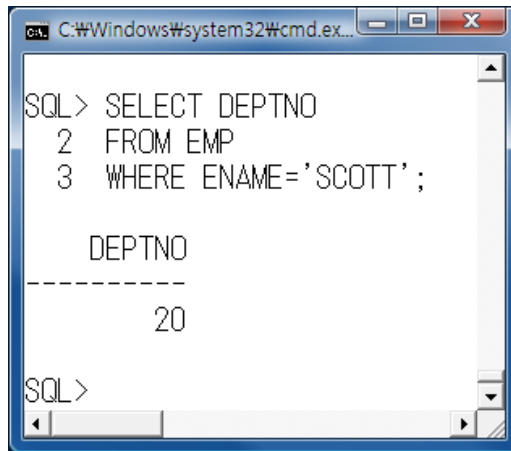
```
C:\Windows\system32\cmd.exe
SQL> SELECT DEPTNO, DNAME
2 FROM DEPT;
```

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

```
SQL>
```

1. 조인의 필요성

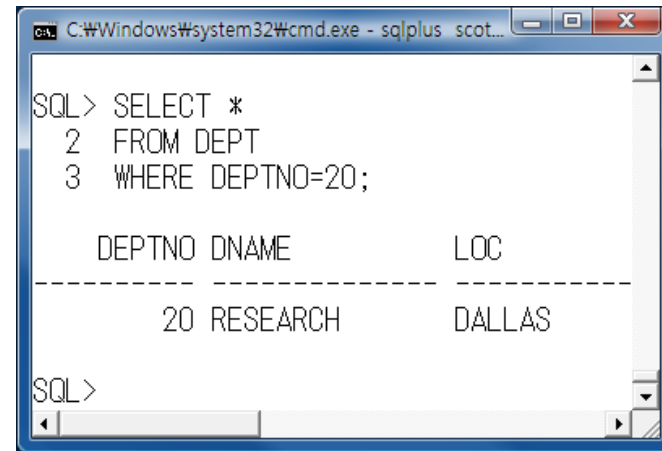
- ❖ MILLER인 사원이 소속되어 있는 부서의 이름이 무엇인지 알아보고자 하는 경우 MILLER란 사원의 부서명을 알아내는 작업 역시 사원 테이블에서 MILLER이 소속된 부서 번호를 알아낸 후에 부서 테이블에서 해당 부서 번호에 대한 부서명을 가져와야 함



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

DEPTNO
-----
      20

SQL>
```



```
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 하지 않고 SQL에서는 두 개 이상의 테이블을 결합해야만 원하는 결과를 얻을 수 있을 때 한 번의 질의로 원하는 결과를 얻을 수 있는 조인 기능을 제공

1. 조인의 필요성

❖ 조인의 종류

종 류	설 명
CROSS JOIN	2개 테이블의 모든 조합
EQUI JOIN	동일한 의미를 갖는 컬럼의 값이 같을 때 만 조인
NON-EQUI JOIN	동일한 의미를 갖는 컬럼에 상관없이 = 이외의 연산자를 이용해서 조인
Outer Join	조인 조건에 만족하지 않는 행도 나타남 어느 한쪽 테이블에만 존재하는 데이터도 조인에 참여
Self Join	하나의 테이블을 가지고 조인
Semi Join	서브쿼리를 이용해서 조인을 수행

2. CROSS JOIN

- ❖ CROSS JOIN은 특별한 키워드 없이 SELECT 문의 FROM 절에 사원(EMP) 테이블과 부서(DEPT) 테이블을 콤마로 연결하여 연속하여 기술하는 것으로 Cartesian Product 라고도 함

예

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

```

C:\WINDOWS\system32\Wcmd.exe - salplus scott/tiger

SQL> SELECT *
2 FROM EMP, DEPT;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO DEPTNO DNAME LOC
-----
7369 SMITH CLERK 7802 80/12/17 800 0 10 ACCOUNTING NEW YORK
7439 ALLEN SALESMAN 7838 81/02/20 1600 300 30 10 ACCOUNTING NEW YORK
7521 WARD SALESMAN 7838 81/02/22 1250 500 30 10 ACCOUNTING NEW YORK
7566 JONES MANAGER 7838 81/04/02 2975 0 20 10 ACCOUNTING NEW YORK
7578 MARTIN SALESMAN 7838 81/09/28 1250 500 30 10 ACCOUNTING NEW YORK
7698 BLAKE MANAGER 7838 81/05/01 2850 1400 30 10 ACCOUNTING NEW YORK
7782 CLARK MANAGER 7838 81/06/09 2450 10 10 ACCOUNTING NEW YORK
7788 SCOTT ANALYST 7568 87/04/19 3000 0 20 10 ACCOUNTING NEW YORK
7838 KING PRESIDENT 81/11/17 5000 0 10 ACCOUNTING NEW YORK
7844 TURNER SALESMAN 7838 81/09/08 1500 0 30 10 ACCOUNTING NEW YORK
7864 ADAMS CLERK 7788 87/05/23 1100 0 20 10 ACCOUNTING NEW YORK

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO DEPTNO DNAME LOC
-----
7900 JAMES CLERK 7568 81/12/03 950 0 30 10 ACCOUNTING NEW YORK
7902 FORD ANALYST 7838 81/12/03 3000 0 20 10 ACCOUNTING NEW YORK
7938 MILLER CLERK 7782 81/12/23 1300 10 10 ACCOUNTING NEW YORK
7369 SMITH CLERK 7802 80/12/17 800 0 20 20 RESEARCH DALLAS
7439 ALLEN SALESMAN 7838 81/02/20 1600 300 30 20 RESEARCH DALLAS
7521 WARD SALESMAN 7838 81/02/22 1250 500 30 20 RESEARCH DALLAS
7566 JONES MANAGER 7838 81/04/02 2975 0 20 20 RESEARCH DALLAS
7578 MARTIN SALESMAN 7838 81/09/28 1250 500 30 20 RESEARCH DALLAS
7698 BLAKE MANAGER 7838 81/05/01 2850 1400 30 20 RESEARCH DALLAS
7782 CLARK MANAGER 7838 81/06/09 2450 10 20 RESEARCH DALLAS
7788 SCOTT ANALYST 7568 87/04/19 3000 0 20 20 RESEARCH DALLAS

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO DEPTNO DNAME LOC
-----
7838 KING PRESIDENT 81/11/17 5000 0 10 20 RESEARCH DALLAS
7844 TURNER SALESMAN 7838 81/09/08 1500 0 30 20 RESEARCH DALLAS
7864 ADAMS CLERK 7788 87/05/23 1100 0 20 20 RESEARCH DALLAS
7900 JAMES CLERK 7568 81/12/03 950 0 30 20 RESEARCH DALLAS
7902 FORD ANALYST 7568 81/12/03 3000 0 20 20 RESEARCH DALLAS
7938 MILLER CLERK 7782 82/01/23 1300 10 20 RESEARCH DALLAS
7369 SMITH CLERK 7802 80/12/17 800 0 20 80 SALES CHICAGO
7439 ALLEN SALESMAN 7838 81/02/20 1600 300 30 80 SALES CHICAGO
7521 WARD SALESMAN 7838 81/02/22 1250 500 30 80 SALES CHICAGO
7566 JONES MANAGER 7838 81/04/02 2975 0 30 80 SALES CHICAGO
7578 MARTIN SALESMAN 7838 81/09/28 1250 500 30 80 SALES CHICAGO

```

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

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
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
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	81/03/28	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

56 개의 행이 선택되었습니다.

2. CROSS JOIN

- ❖ CROSS JOIN의 결과 얻어지는 컬럼의 수는 사원 테이블의 컬럼의 수(8)와 부서 테이블의 컬럼의 수(3)를 더한 것이므로 11
- ❖ 행의 수는 사원 한 명에 대해서 DEPT 테이블의 모든 행과 결합되기에 2개 테이블의 행의 개수를 곱한 것 만큼 조회
- ❖ CROSS JOIN의 결과를 보면 사원 테이블에 부서에 대한 상세정보가 결합되긴 했지만, 조인 될 때 아무런 조건을 제시하지 않았기에 사원 한 명에 대해서 DEPT 테이블의 모든 행의 데이터와 결합된 형태이기에 CROSS JOIN의 결과는 아무런 의미를 갖지 못하는 경우가 많음
- ❖ 조인 결과가 의미를 갖으려면 조인할 때 조건을 지정해야 하는 경우가 대부분



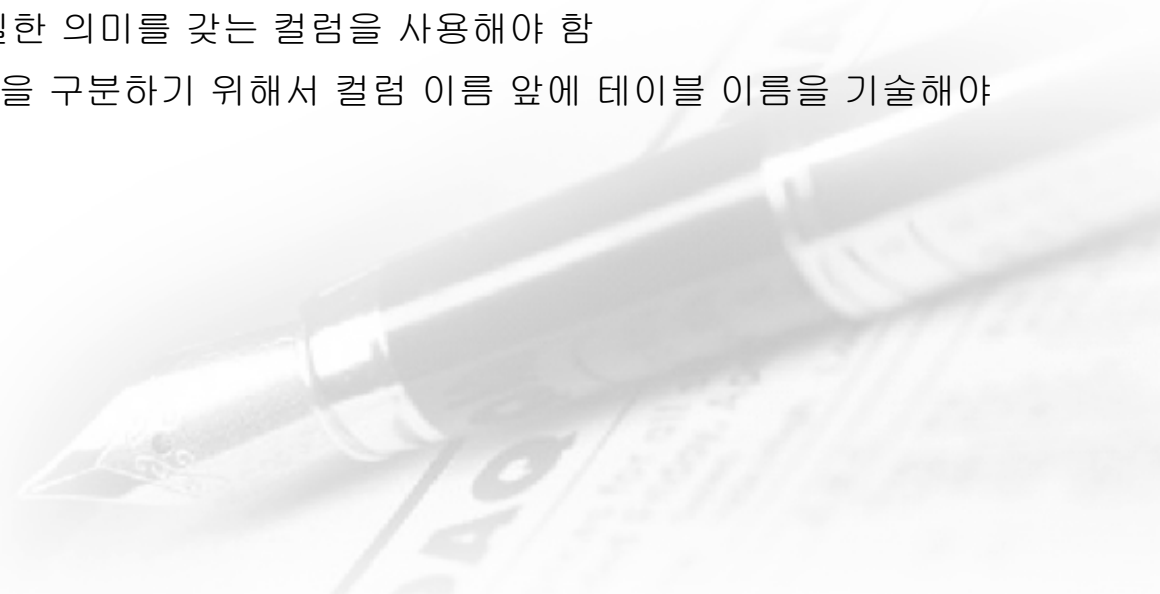
3. EQUI JOIN

- ❖ EQUI JOIN은 가장 많이 사용하는 조인 방법으로서 조인 대상이 되는 두 테이블에서 동일한 의미를 갖는 컬럼의 값이 일치되는 행을 연결하여 결과를 생성하는 조인 방법
- ❖ 다음은 사원 정보를 출력할 때 각 사원들이 소속된 부서의 상세 정보를 출력하기 위해서 두 개의 테이블을 조인한 경우

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 사용
- ❖ 테이블을 조인하려면 일치되는 동일한 의미를 갖는 컬럼을 사용해야 함
- ❖ 컬럼의 이름이 같은 경우 컬럼 이름을 구분하기 위해서 컬럼 이름 앞에 테이블 이름을 기술해야 함



3. EQUI JOIN

- ❖ 다음은 두 테이블을 조인한 결과로 살펴보면 다음과 같이 부서 번호를 기준으로 같은 값을 가진 사원 테이블의 컬럼과 부서 테이블의 컬럼이 결합

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 연산

- ❖ EMP 테이블의 ENAME이 MILLER인 사원의 ENAME과 DEPT 테이블의 DNAME을 출력

예

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

	ENAME	DNAME
1	MILLER	ACCOUNTING

3.2 컬럼명의 모호성 해결

- ❖ 양쪽 테이블에 동일한 컬럼이 존재하는 경우 컬럼 이름을 기재할 때 테이블 이름을 기재해야 하는데 그렇지 않으면 애매한 컬럼 이름이라고 에러가 발생

예

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

ORA-00918: column ambiguously defined

00918. 00000 - "column ambiguously defined"

*Cause:

*Action:

70행, 22열에서 오류 발생



3.2 컬럼명의 모호성 해결

- ❖ 동일한 이름의 컬럼은 컬럼명 앞에 테이블 이름을 명시적으로 기술함으로써 컬럼이 어느 테이블 소속인지 구분할 수 있음

예

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

	ENAME	DNAME	DEPTNO
1	MILLER	ACCOUNTING	10



3.3 테이블에 별칭 부여하기

- ❖ 테이블 이름에 별칭을 붙이는 방법은 FROM 절 다음에 테이블 이름을 명시하고 공백을 둔 다음에 별칭을 지정
- ❖ 이후 절에서 테이블 이름을 사용할 때는 별칭을 사용해야 함

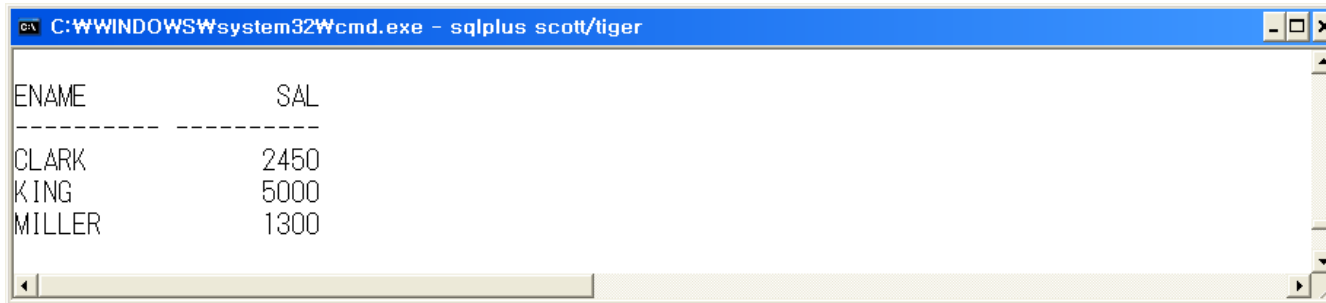
예

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



연습문제

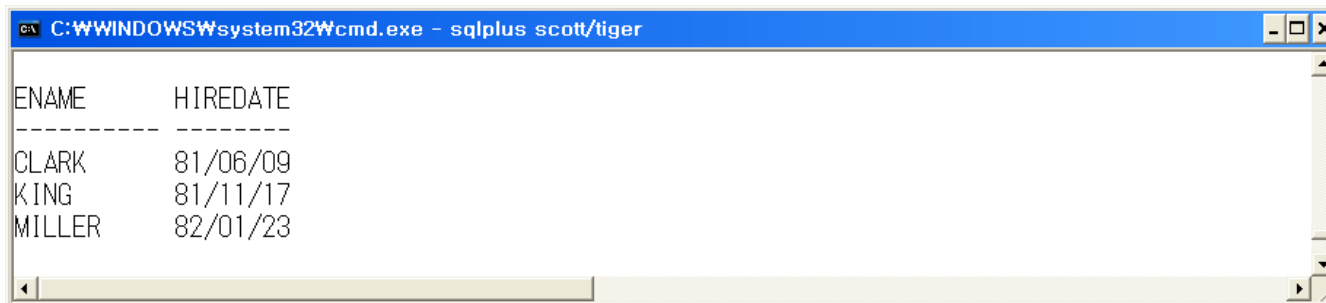
1. DEPT 테이블의 LOC가 'NEW YORK' 인 사원의 EMP 테이블의 이름(ename)과 급여(sal)를 조회



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 with salary 2450, KING with salary 5000, and MILLER with salary 1300. The window has a blue title bar and standard Windows window controls.

ENAME	SAL
CLARK	2450
KING	5000
MILLER	1300

2. DEPT 테이블의 DNAME 컬럼의 값이 'ACCOUNTING' 인 사원의 EMP 테이블의 이름(ENAME)과 입사일(HIREDATE)을 조회

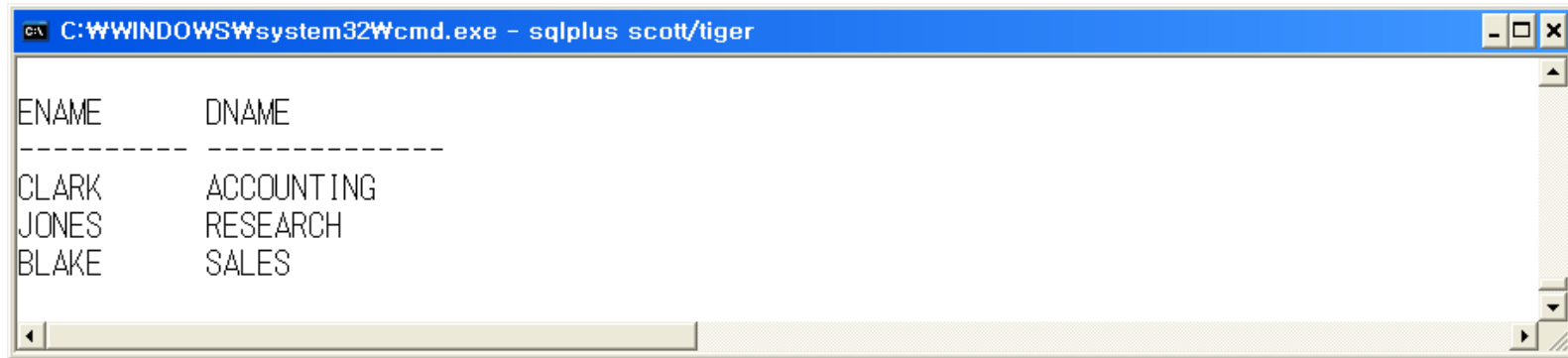


A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The window displays the output of a SQL query. The output is a table with two columns: ENAME and HIREDATE. The data rows are CLARK with hire date 81/06/09, KING with hire date 81/11/17, and MILLER with hire date 82/01/23. The window has a blue title bar and standard Windows window controls.

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

<연습문제>

3. EMP 테이블의 JOB이 'MANAGER'인 사원의 EMP 테이블의 ename, DEPT 테이블의 dname을 조회



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

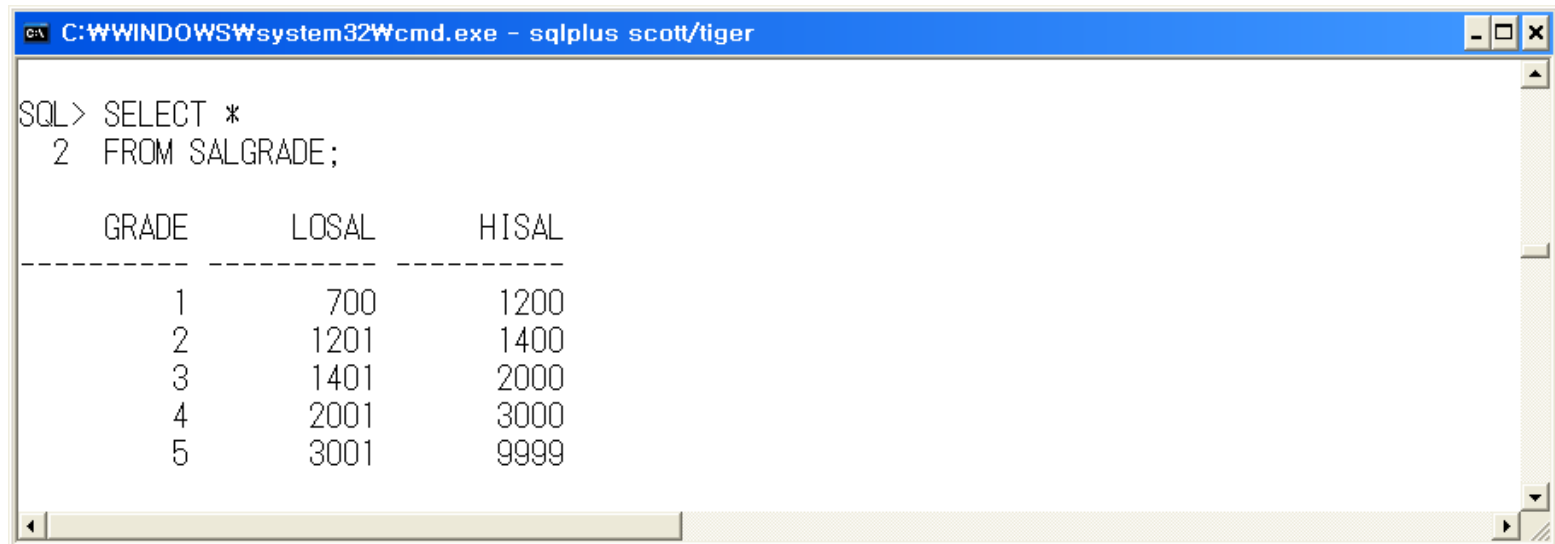
ENAME      DNAME
-----
CLARK      ACCOUNTING
JONES      RESEARCH
BLAKE      SALES
```

4. NON-EQUI JOIN

- ❖ NON-EQUI JOIN은 특정 범위 내에 있는지를 조사하기 위해서 WHERE 절에 조인 조건을 = 연산자 이외의 비교 연산자를 사용
- ❖ 급여 등급 테이블(SALGRADE)을 데이터 확인

예

```
SELECT * FROM SALGRADE;
```



SQL> SELECT *

2 FROM SALGRADE;

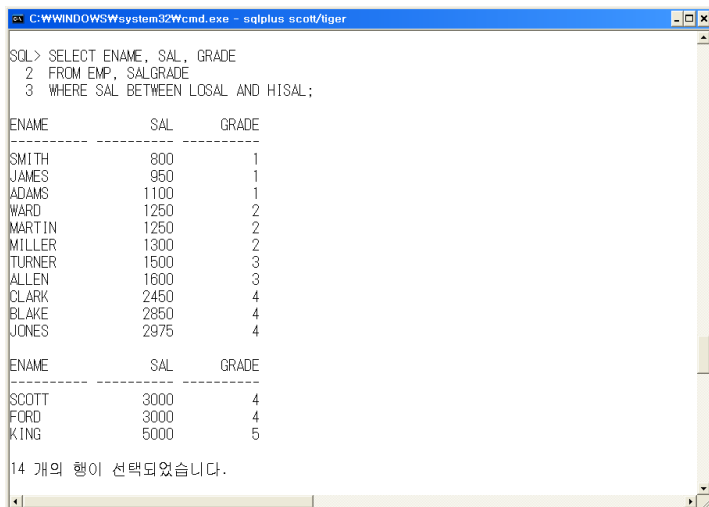
GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

4. NON-EQUI JOIN

- ❖ 급여 등급 테이블(salgrade)에는 급여에 대한 등급을 나누어 놓음
- ❖ 급여의 등급은 총 5등급으로 나누어져 있으며, 1등급은 급여가 700부터 1200 사이이고, 2등급은 1201부터 1400 사이이고, 3등급은 1401부터 2000 사이이고, 4등급은 2001부터 3000사이이고, 5등급이면 3001부터 9999사이
- ❖ 급여 등급을 5개로 나누어 놓은 salgrade에서 정보를 얻어 와서 각 사원의 급여 등급을 조회하고자 하는 경우에 사원(emp) 테이블과 급여 등급(salgrade) 테이블을 조인해야 함

예

```
SELECT ENAME, SAL, GRADE
FROM EMP, SALGRADE
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.

5. Self Join

- ❖ 조인은 두 개 이상의 서로 다른 테이블을 서로 연결하는 것뿐만 아니라 하나의 테이블 내에서 조인을 해야만 원하는 자료를 얻는 경우가 발생할 수 있음 - 동일한 의미를 갖는 컬럼이 하나의 테이블에 2개 이상 존재하는 경우
- ❖ Self Join이란 자기 자신과 조인을 맺는 것
- ❖ SMITH의 매니저 이름이 무엇인지 알아내려면 어떻게?

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.

5. Self Join

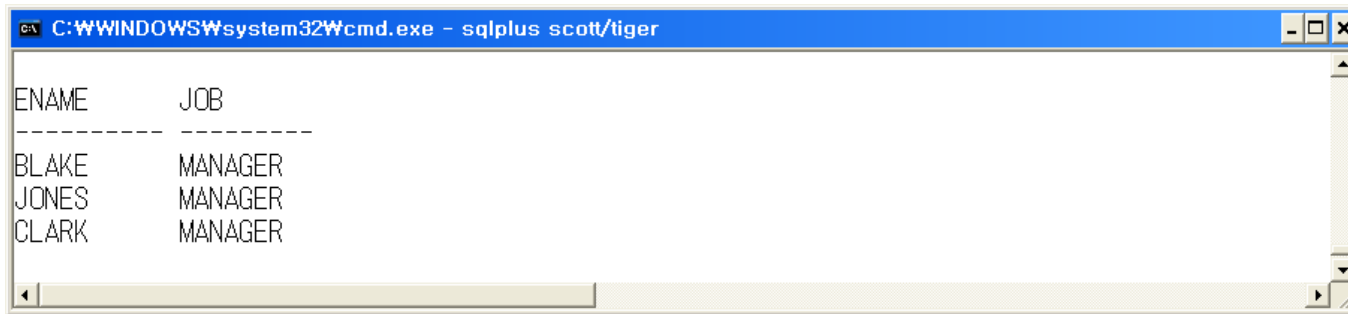
예

```
select employee.ename || '의 매니저는 ' || manager.ename  
from emp employee, emp manager  
where employee.mgr = manager.empno
```

Results:	
	EMPLOYEE,ENAME '의매니저는' MANAGER,ENAME
1	FORD의 매니저는 JONES
2	SCOTT의 매니저는 JONES
3	JAMES의 매니저는 BLAKE
4	TURNER의 매니저는 BLAKE
5	MARTIN의 매니저는 BLAKE
6	WARD의 매니저는 BLAKE
7	ALLEN의 매니저는 BLAKE
8	MILLER의 매니저는 CLARK
9	ADAMS의 매니저는 SCOTT
10	CLARK의 매니저는 KING
11	BLAKE의 매니저는 KING
12	JONES의 매니저는 KING
13	SMITH의 매니저는 FORD

연습문제

4. EMP 테이블에서 manager가 'KING'인 직원들의 이름(ename)과 직급(job)을 조회



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

ENAME      JOB
-----
BLAKE      MANAGER
JONES      MANAGER
CLARK      MANAGER
```

6. Outer Join

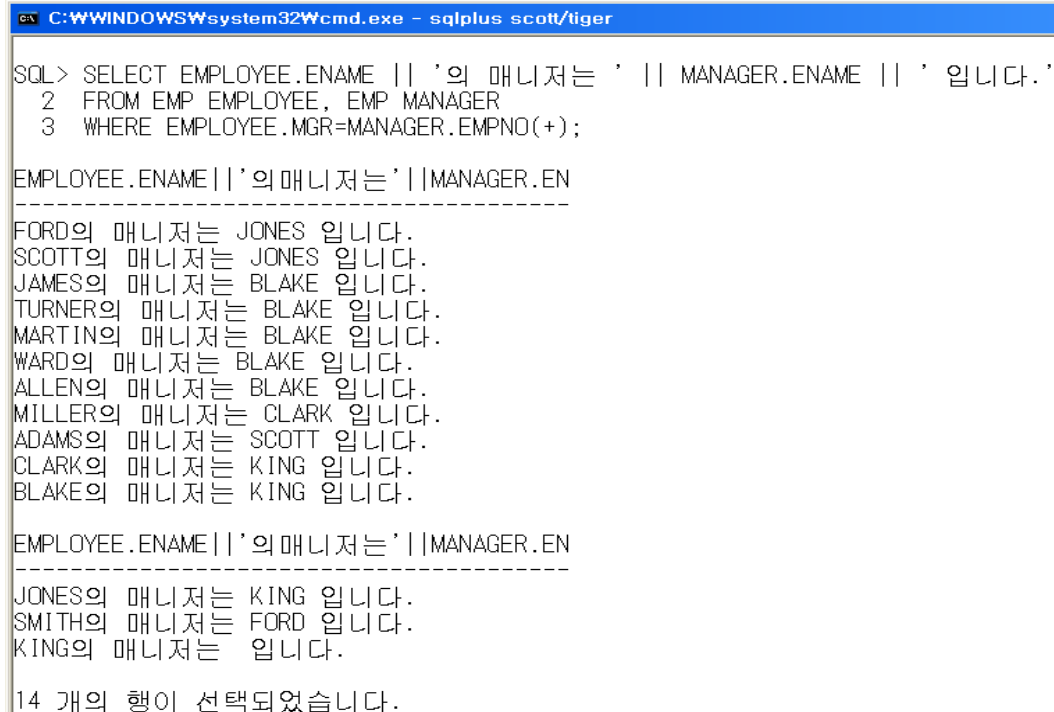
- ❖ Self Join을 이용해서 특정 사원의 매니저 이름을 구했는데 결과를 살펴보면 이름이 KING인 사원 한 사람의 정보가 빠져 있음을 확인할 수 있음
- ❖ KING은 이 회사의 사장(PRESIDENT)으로 매니저가 존재하지 않으므로 MGR 컬럼 값이 NULL
- ❖ 사원 번호(EMPNO)가 NULL인 사원은 없으므로 조인 조건에 만족하지 않아서 KING은 Self Join의 결과에서 배제
- ❖ 조인 조건에 만족하지 못하였더라도 해당 행을 나타내고 싶을 때에 사용하는 것이 외부 조인(Outer Join)
- ❖ 외부 조인은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



6. Outer Join

예

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



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The user has entered an SQL query to perform an outer join between the EMP and EMP_MANAGER tables. The results are displayed in two sections, separated by dashed lines. The first section shows the manager for employees FORD, SCOTT, JAMES, TURNER, MARTIN, and WARD, all of whom are managed by BLAKE. The second section shows the manager for employees ALLEN, MILLER, ADAMS, CLARK, and BLAKE, all of whom are managed by KING. At the bottom, a message indicates that 14 rows were selected.

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

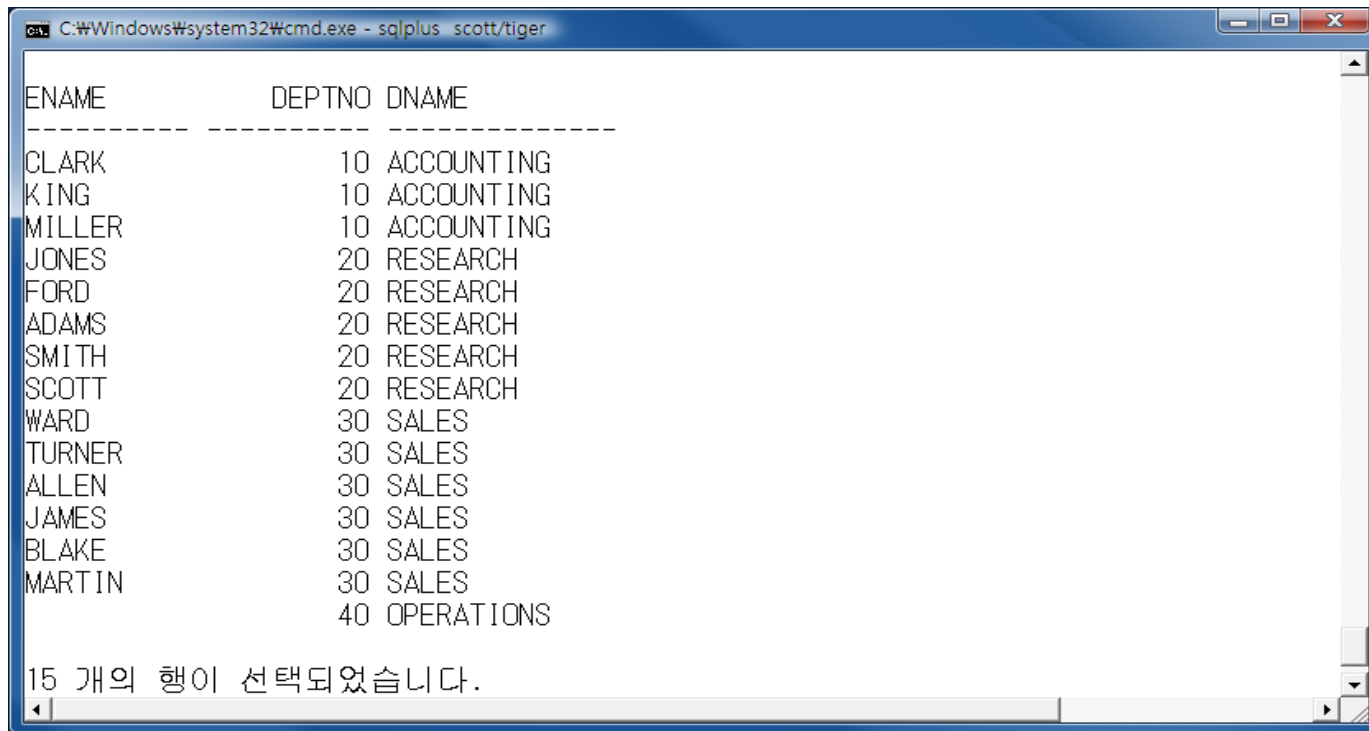
EMPLOYEE.ENAME || '의 매니저는 ' || MANAGER.EN
-----
FORD의 매니저는 JONES 입니다.
SCOTT의 매니저는 JONES 입니다.
JAMES의 매니저는 BLAKE 입니다.
TURNER의 매니저는 BLAKE 입니다.
MARTIN의 매니저는 BLAKE 입니다.
WARD의 매니저는 BLAKE 입니다.
ALLEN의 매니저는 BLAKE 입니다.
MILLER의 매니저는 CLARK 입니다.
ADAMS의 매니저는 SCOTT 입니다.
CLARK의 매니저는 KING 입니다.
BLAKE의 매니저는 KING 입니다.

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

14 개의 행이 선택되었습니다.
```

연습문제

5. 사원(EMP) 테이블과 부서(DEPT) 테이블을 조인하여 사원이름(ename)과 부서번호(deptno)와 부서명(dname)을 출력하도록 합시다. 부서 테이블의 40번 부서와 조인할 사원 테이블의 부서번호가 없지만, 아래 그림과 같이 40번 부서의 부서 이름도 출력되도록 쿼리문을 작성

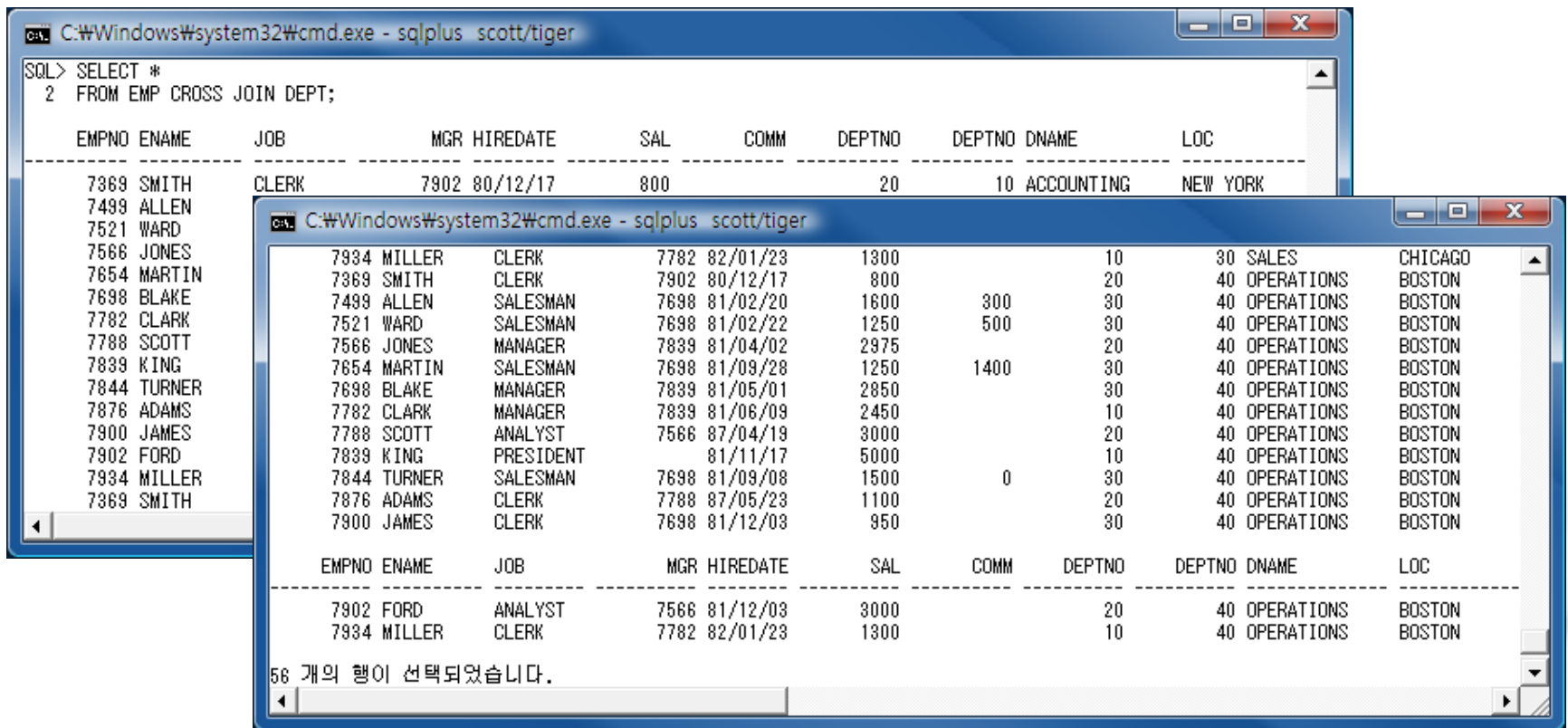


ENAME	DEPTNO	DNAME
CLARK	10	ACCOUNTING
KING	10	ACCOUNTING
MILLER	10	ACCOUNTING
JONES	20	RESEARCH
FORD	20	RESEARCH
ADAMS	20	RESEARCH
SMITH	20	RESEARCH
SCOTT	20	RESEARCH
WARD	30	SALES
TURNER	30	SALES
ALLEN	30	SALES
JAMES	30	SALES
BLAKE	30	SALES
MARTIN	30	SALES
	40	OPERATIONS

15 개의 행이 선택되었습니다.

7.1 ANSI CROSS JOIN

SELECT *
FROM EMP CROSS JOIN DEPT;



The image shows two overlapping SQL*Plus windows. The top window displays the result of the query 'SELECT * FROM EMP CROSS JOIN DEPT;'. It shows a list of employees with their details. The bottom window shows a more detailed view of the same query result, displaying all columns for each employee and department combination. At the bottom of the bottom window, it states '56 개의 행이 선택되었습니다.' (56 rows selected).

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									

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
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	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
7934	MILLER	CLERK	7782	82/01/23	1300		10	40	OPERATIONS	BOSTON

56 개의 행이 선택되었습니다.

7.2 ANSI Inner Join

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

```
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 지정

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



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 표현

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

- ✓ EMP와 DEPT에 DEPTNO 라는 같은 이름의 컬럼이 있기 때문에 다음과 같이 간단하게 조인문을 기술

```
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 표현

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

- ✓ EMP와 DEPT에 DEPTNO 라는 같은 이름의 컬럼이 있기 때문에 다음과 같이 간단하게 조인문을 기술

```
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 제공

SELECT *

FROM table1 [LEFT | RIGHT | FULL] Outer Join table2

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



다양한 Outer Join

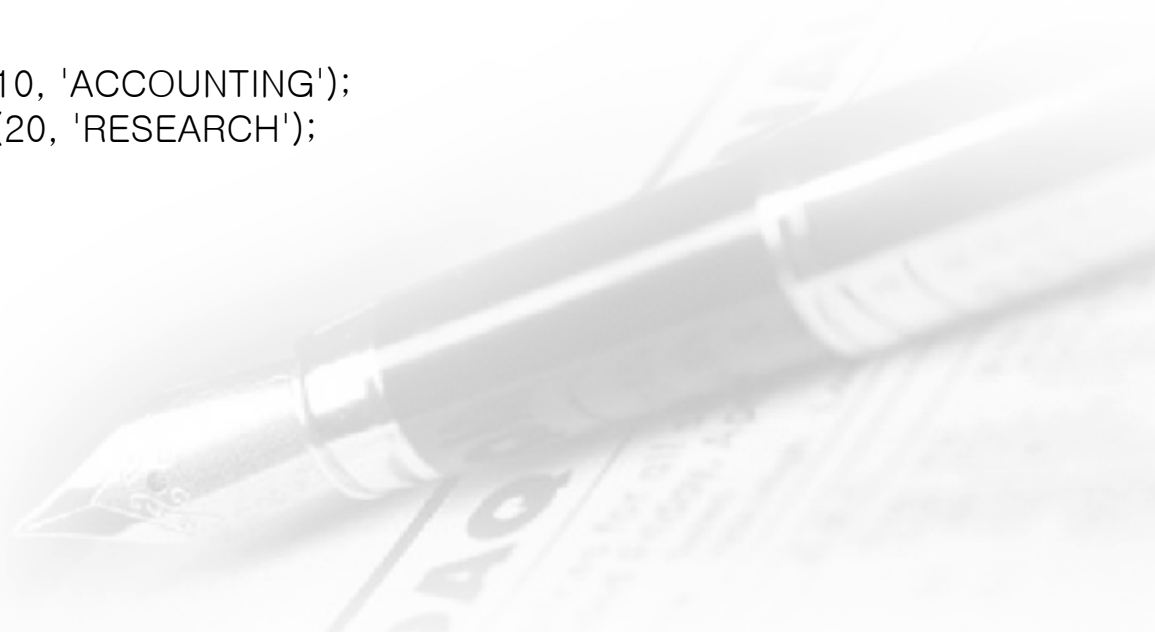
1. DEPT 테이블과 닮은 DEPT01 테이블을 만들어 보겠습니다. 혹시 DEPT01 테이블이 존재한다면 DEPT01 테이블이 생성되지 않으므로 DROP 명령어로 삭제 후 생성

```
DROP TABLE DEPT01;
```

2. 부서번호와 부서명을 컬럼으로 갖는 DEPT01 테이블을 생성

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

3. 데이터를 추가
INSERT INTO DEPT01 VALUES(10, 'ACCOUNTING');
INSERT INTO DEPT01 VALUES (20, 'RESEARCH');



다양한 Outer Join

4. 동일한 방법으로 DEPT02 테이블을 생성

```
DROP TABLE DEPT02;
```

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

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

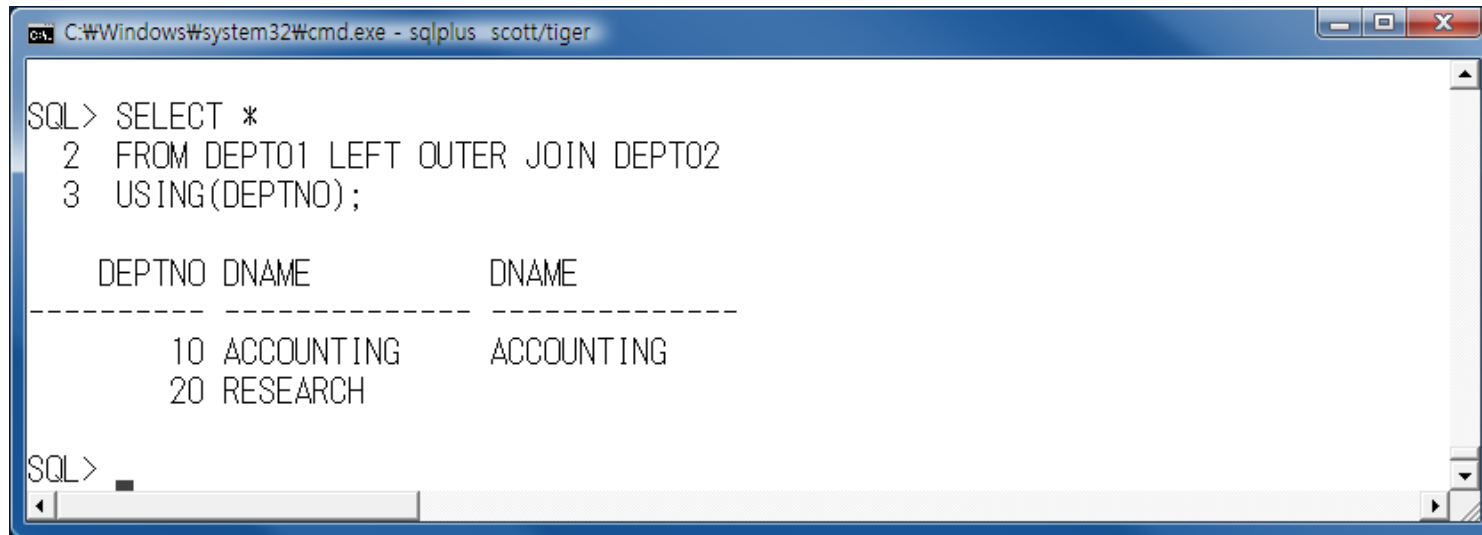
```
SELECT * FROM DEPT02;
```



LEFT OUTER JOIN

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

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



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". Inside the window, the following SQL query is entered and executed:

```
SQL> SELECT *  
2 FROM DEPT01 LEFT OUTER JOIN DEPT02  
3 USING(DEPTNO);
```

The output of the query is displayed as follows:

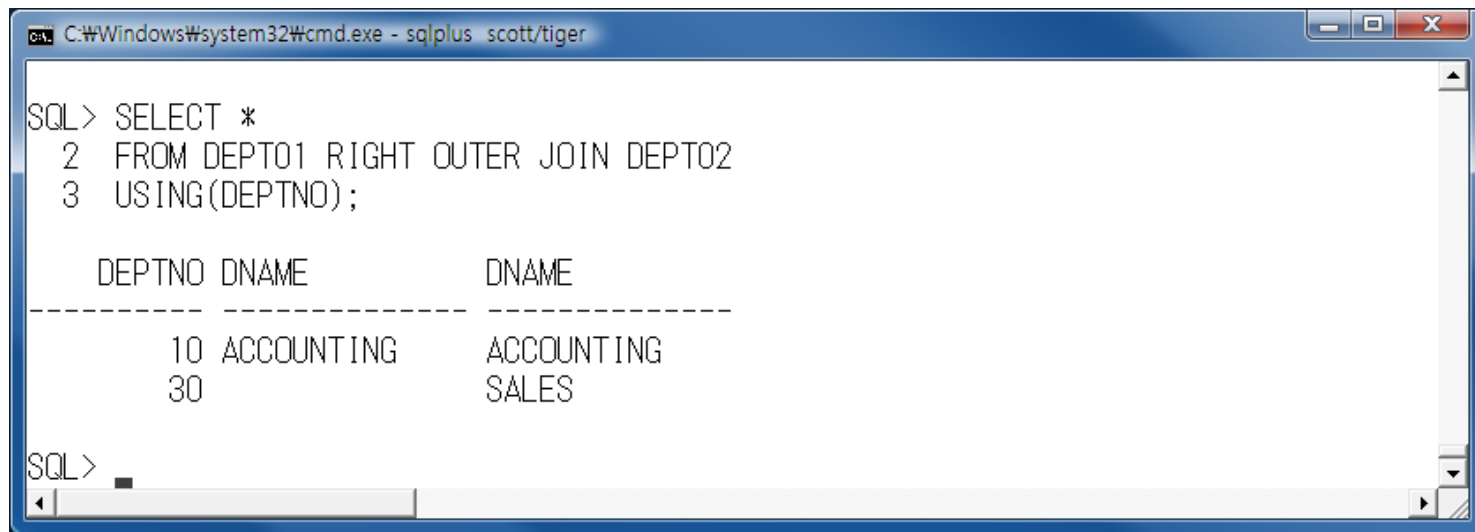
DEPTNO	DNAME	DNAME
10	ACCOUNTING	ACCOUNTING
20	RESEARCH	

The command prompt shows the SQL prompt "SQL>" at the bottom left, and the window has standard Windows window controls (minimize, maximize, close) at the top right.

RIGHT OUTER JOIN

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

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



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The prompt is "SQL>". The user has entered the following SQL query:

```
SQL> SELECT *  
2 FROM DEPT01 RIGHT OUTER JOIN DEPT02  
3 USING(DEPTNO);
```

The query results are displayed in a table format:

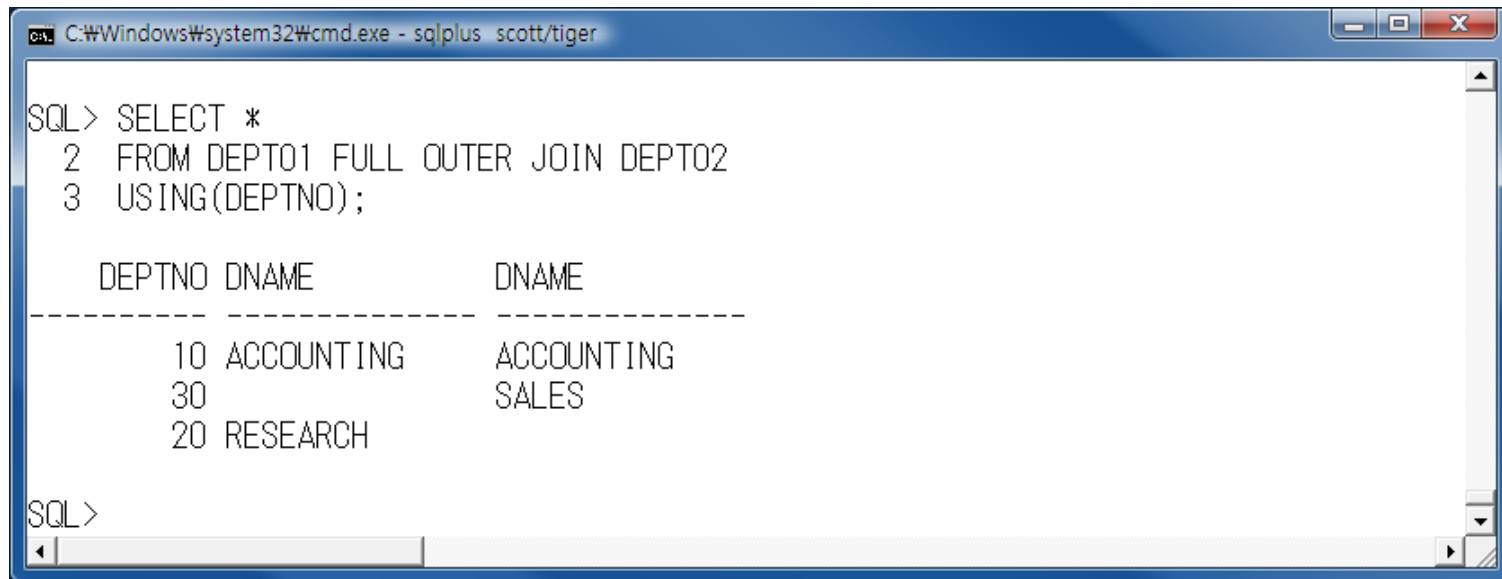
DEPTNO	DNAME	DNAME
10	ACCOUNTING	ACCOUNTING
30		SALES

The prompt "SQL>" is visible at the bottom left of the window.

FULL OUTER JOIN

- ❖ FULL OUTER JOIN은 LEFT OUTER JOIN, RIGHT OUTER JOIN 을 합한 형태

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



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". Inside the window, the following SQL query is entered and executed:

```
SQL> SELECT *  
2 FROM DEPT01 FULL OUTER JOIN DEPT02  
3 USING(DEPTNO);
```

The output of the query is displayed as a table with three columns: DEPTNO, DNAME, and DNAME. The first column (DEPTNO) contains the values 10, 30, and 20. The second column (DNAME) contains the values ACCOUNTING, SALES, and RESEARCH. The third column (DNAME) contains the values ACCOUNTING, SALES, and RESEARCH. The output is formatted as follows:

DEPTNO	DNAME	DNAME
10	ACCOUNTING	ACCOUNTING
30		SALES
20	RESEARCH	

The command prompt window also shows the prompt "SQL>" at the bottom left.

8. Set Operator

- ❖ 하나 이상의 테이블로부터 자료를 검색하는 또 다른 방법은 SET연산자를 이용하는 방법이 있음
- ❖ SET연산자를 이용하여 여러 개의 SELECT문장을 연결하여 작성

- ❖ Syntax

```
SELECT          * | column1[, column2, column3, . . . . ]
FROM            table1
. . . . .
SET             operator
SELECT          * | column1[, column2, column3, . . . . ]
FROM            table2
. . . . .
[ORDER BY      column | expression];
```



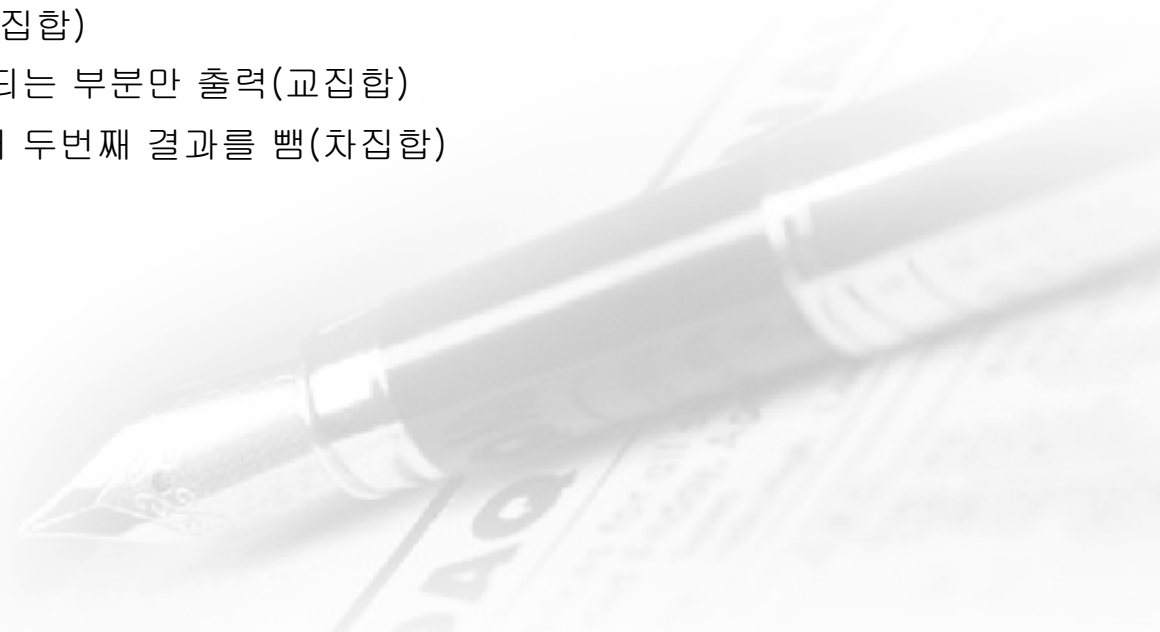
8. Set Operator

❖ Guidelines

- ✓ 첫번째 SELECT 구문에서 기술된 열과 두번째 SELECT 구문에서 기술된 열들은 좌측부터 1대1 대응하며 그 개수와 타입이 일치해야 함
- ✓ FROM절 뒤에 기술되는 테이블은 같을 수도 있고 다를 수도 있음
- ✓ 출력되는 HARDING은 첫번째 SELECT구문에서 기술된 열이 출력
- ✓ ORDER BY는 한번만 기술 가능하고 SELECT 구문의 마지막에 기술
- ✓ BLOB, CLOB, BFILE, LONG 형 컬럼에는 사용할 수 없음

❖ SET 연산자의 종류

- ✓ UNION :각 결과의 합(합집합:중복되는 값은 한번 출력)
- ✓ UNION ALL :각 결과의 합(합집합)
- ✓ INTERSECT :각 결과의 중복되는 부분만 출력(교집합)
- ✓ MINUS :첫번째 결과에서 두번째 결과를 뺀(차집합)



8.1 UNION

- ❖ UNION과 UNION ALL의 차이
- ❖ 양쪽에서 검색된 결과를 모두 출력

```
SELECT deptno  
FROM dept  
UNION  
SELECT deptno  
FROM emp;
```

```
SELECT deptno  
FROM dept  
UNION ALL  
SELECT deptno  
FROM emp;
```



8.2 INTERSECT

❖ 양쪽에서 검색된 자료만 출력

```
SELECT deptno  
FROM dept  
INTERSECT  
SELECT deptno  
FROM emp;
```

DEPTNO

10

20

30



8.3 MINUS

- ❖ 두번째 SELECT문장에서 검색되지 않았던 값을 첫번째 SELECT문장에서 출력
- ❖ 첫번째 SELECT문장에서 두번째 SELECT문장에의 값을 뺀 것을 출력

```
SELECT deptno  
FROM dept  
MINUS  
SELECT deptno  
FROM emp;
```

DEPTNO

40



8.4 Grouping Sets

- ❖ Group by에 기재해서 그룹 별 소계만을 출력할 때 사용

```
select deptno, job, sum(sal)
from emp
group by grouping sets(deptno, job)
```

	DEPTNO	JOB	SUM(SAL)
1	30	(null)	9400
2	20	(null)	10875
3	40	(null)	8750
4	(null)	CLERK	4150
5	(null)	SALESMAN	5600
6	(null)	PRESIDENT	5000
7	(null)	MANAGER	8275
8	(null)	ANALYST	6000

연습문제

1. EMP 테이블에서 모든 사원에 대한 이름(ename), 부서번호(deptno) DEPT 테이블에서 부서명(dname)을 출력하는 SELECT 문장을 작성
2. DEPT 테이블의 LOC가 NEW YORK에서 근무하고 있는 사원에 대하여 EMP 테이블의 이름(ename), 업무(job), 급여(sal), DEPT 테이블의 부서명(dname)을 출력하는 SELECT 문장을 작성
3. EMP 테이블에서 보너스(comm)가 null 이 아닌 사원에 대하여 이름(ename), DEPT 테이블의 부서명(dname), 위치(loc)를 출력하는 SELECT 문장을 작성
4. EMP 테이블에서 이름(ename) 중 L자가 있는 사원에 대하여 이름(ename), 업무(job), DEPT 테이블의 부서명(dname), 위치(loc)를 출력하는 SELECT 문장을 작성
5. EMP 테이블에서 그들의 관리자(mgr) 보다 먼저 입사한 사원에 대하여 이름(ename), 입사일(hiredate), 관리자(mgr) 이름, 관리자(mgr) 입사일을 출력하는 SELECT 문장을 작성

