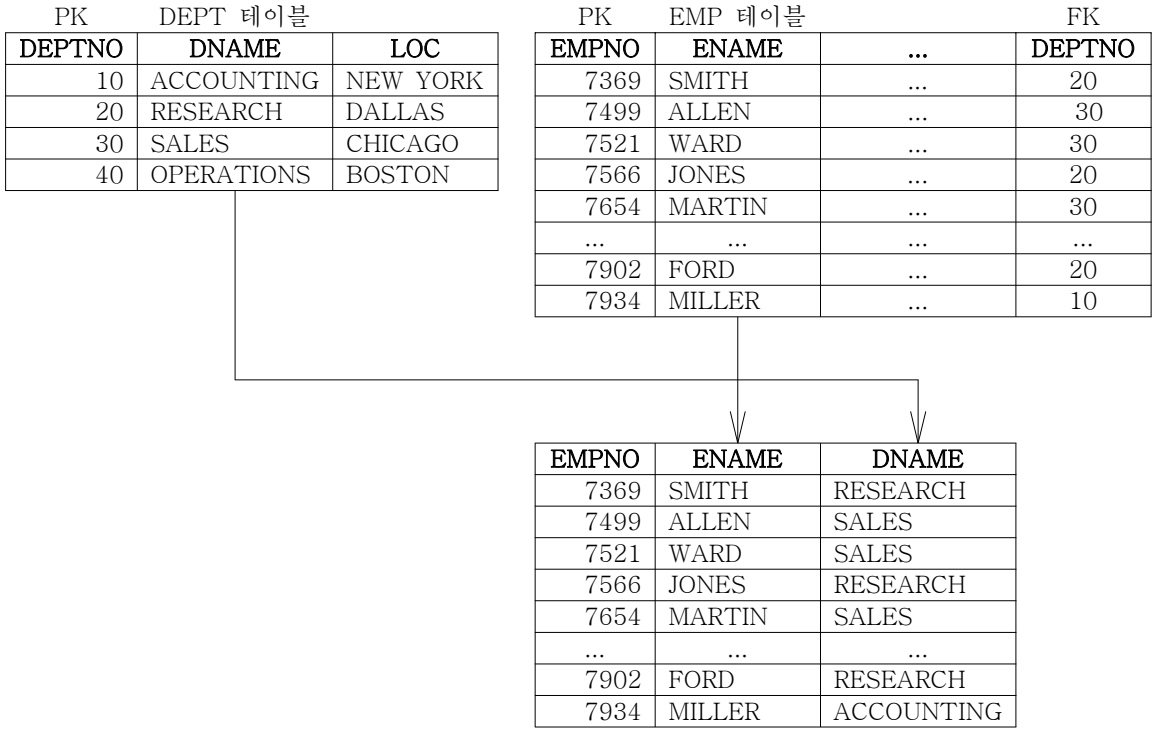
Chapter 5. Join

이전 장까지는 한 개의 테이블만을 검색하는 경우를 살펴보았다. 이번 장에서는 검색하고자 하는 컬럼이 여러 개의 테이블에 존재하는 경우, 데이터를 검색할 수 있는 Join에 대하여 설명한다.

Join의 개념

여러 개의 테이블로부터 데이터를 검색하는 경우를 "테이블을 Join 한다"라고 부른다. 예를 들어, 다음 그림과 같이 사용자가 검색하고자 하는 컬럼이 EMP 테이블의 EMPNO, ENAME, DEPT 테이블의 DNAME이라고 가정했을 때, EMPNO가 7369인 SMITH의 DNAME을 검색하기 위해서는 EMP 테이블의 외래키(FK)인 DEPTNO 컬럼과 DEPT 테이블의 기본키(PK)인 DEPTNO가 일치되는 DNAME을 DEPT 테이블에서 검색해야 한다. 즉, SMITH의 DEPTNO는 20이므로, DEPT 테이블에서 DEPTNO가 20인 DNAME을 검색하여 SMITH의 부서명은 RESEARCH임을 알 수가 있게 된다. 테이블이 Join 되기 위해서는 반드시 FK와 FK가 참조하는 PK가 반드시 존재하여야 함을 알 수가 있다. 여기서, FK와 PK가 일치해야하는 조건을 Join 조건이라고 부르며 SELECT 문장의 WHERE절에 기술한다.



PK DEPT 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

PK EMP 테이블 FK

EMPNO	ENAME	...	DEPTNO
7369	SMITH	...	20
7499	ALLEN	...	30
7521	WARD	...	30
7566	JONES	...	20
7654	MARTIN	...	30
...	...	...	...
7902	FORD	...	20
7934	MILLER	...	10

EMPNO	ENAME	DNAME
7369	SMITH	RESEARCH
7499	ALLEN	SALES
7521	WARD	SALES
7566	JONES	RESEARCH
7654	MARTIN	SALES
...	...	...
7902	FORD	RESEARCH
7934	MILLER	ACCOUNTING

그림 5-1. Equi-Join

카르테시안 프로덕트(Cartesian Product)

카르테시안 프로덕트란 SELECT 문장에서 Join 조건을 생략하여 첫 번째 테이블의 모든 행들이 두 번째 테이블의 모든 행들과 Join되는 경우이다.

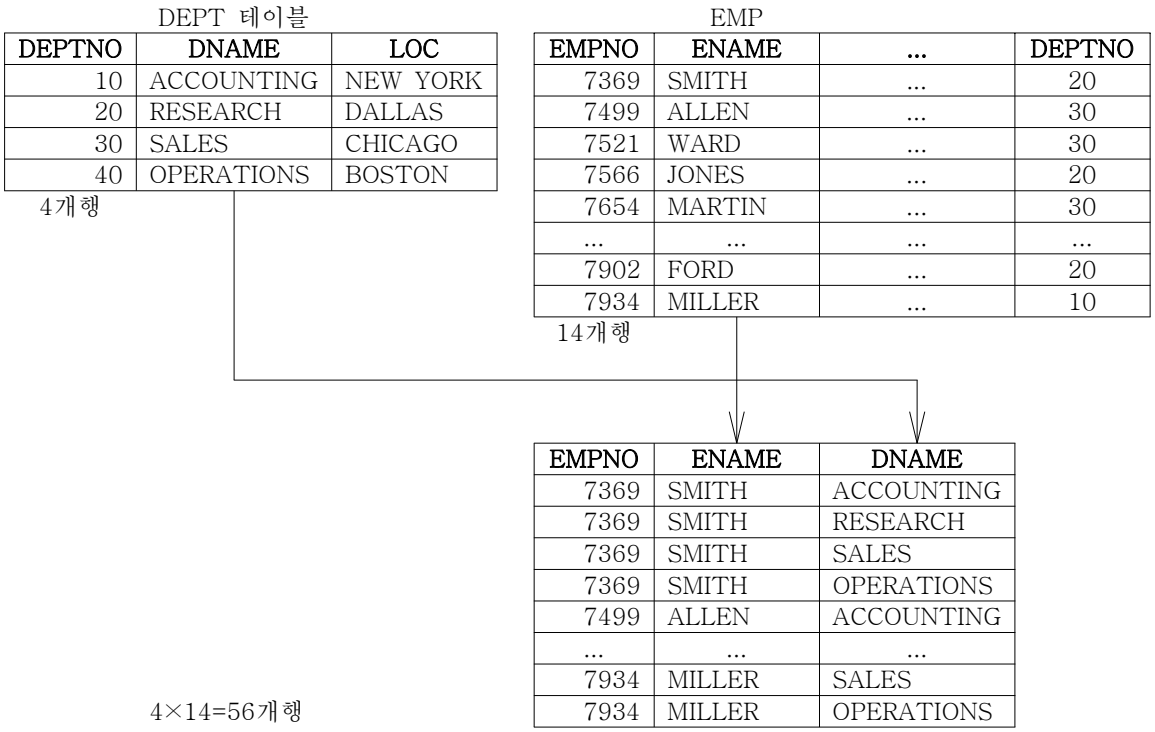


그림 5-2. 카르테시안 프로덕트

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

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      ACCOUNTING
      7369 SMITH      RESEARCH
      7369 SMITH      SALES
      7369 SMITH      OPERATIONS
      7499 ALLEN      ACCOUNTING
        ...
      7934 MILLER     OPERATIONS

56 개의 행이 선택되었습니다.
```

Join에는 Equi-Join, Non-EquiJoin, Outer Join, Self Join이 있으며, 각각 살펴보도록 한다.

Equi-Join

Equi-Join이란 테이블을 Join할 때, FK와 FK가 참조하는 PK가 Equal 조건에 의해 정확히 일치하는 경우만 검색하는 Join 방식이다. 먼저, Equi-Join의 문법은 다음과 같다.

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

위 문법에서 보듯이 Join을 하기 위해서는 어떤 테이블에 어떤 컬럼이 있는지, 테이블간의 관계는 어떻게 되는지, FK와 PK의 컬럼명은 무엇인지를 정확히 알아야만 Join 문장을 오류없이 정확하게 기술할 수 있다. 그러나, 데이터베이스에는 수십~수백개의 테이블이 존재하기 때문에 이러한 정보를 일일이 검색하는 것은 불가능하다고 할 수 있다. 그러면 이러한 정보를 어떻게 하면 손쉽게 확인할 수 있을까? 해법은 1장에서 설명한 개체관계도(ERD)이다. 이렇듯 ERD는 데이터베이스 관리자나 개발자에게 여러모로 쓸모 있는 것임에 틀림없다.

실습에 사용할 DEPT, EMP, SALGRADE 테이블의 ERD는 다음 그림과 같다.

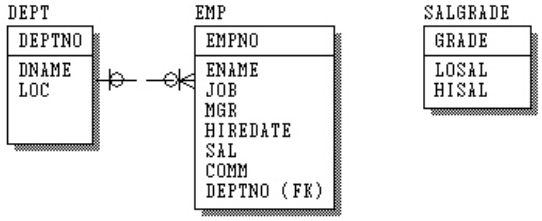


그림 5-3. 개체관계도(ERD)

그림의 ERD를 보면서 Join 문장을 작성해보자. DEPT, EMP 테이블에서 EMPNO, ENAME, DNAME을 검색하려면 ERD를 확인하면서 다음과 같은 절차에 의해 SQL 문장을 작성하면 된다.

(1) 검색하고자 하는 컬럼을 SELECT절 뒤에 기술한다.
(2) 검색하고자 하는 컬럼이 소속되어 있는 테이블들의 이름을 FROM절 뒤에 기술한다.
(3) DEPT 테이블과 EMP 테이블의 관계에서 실선으로 되어 있는 개체 쪽의 기본키 컬럼과 까마귀발 모양으로 되어 있는 개체 쪽의 외래키 컬럼을 WHERE절에 Equal 조건으로 기술한다.

```
SQL> SELECT EMP.EMPNO, EMP.ENAME, DEPT.DNAME
  2  FROM EMP, DEPT
  3  WHERE EMP.DEPTNO = DEPT.DEPTNO;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      RESEARCH
      7499 ALLEN      SALES
      7521 WARD       SALES
        ...
      7934 MILLER     ACCOUNTING

14 개의 행이 선택되었습니다.
```

만약, 위의 Join 결과에서 EMPNO가 7698인 행들만 검색하려면 WHERE절의 Join 조건에 검색 조건을 추가하면 된다.

```
SQL> SELECT EMP.EMPNO, EMP.ENAME, DEPT.DNAME
  2  FROM EMP, DEPT
  3  WHERE EMP.DEPTNO = DEPT.DEPTNO
  4  AND EMP.EMPNO = 7698;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7698 BLAKE      SALES
```

SELECT절 뒤의 컬럼명에는 해당 컬럼이 소속되어 있는 테이블명을 정확히 기술해주는 것이 바람직하지만 해당 컬럼명이 FROM절 뒤의 모든 테이블내에서 유일하다면 반드시 기술해 줄 필요는 없다.

```
SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM EMP, DEPT
  3  WHERE EMP.DEPTNO = DEPT.DEPTNO
  4  AND EMP.EMPNO = 7698;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7698 BLAKE      SALES
```

그러나, SELECT절 뒤에 기술되는 컬럼명이 FROM절 뒤의 일부 테이블들에 중복되어 있다면 다음과 같은 오류 메시지가 발생하게 된다.

```
SQL> SELECT EMPNO, ENAME, DEPTNO, DNAME
  2  FROM EMP, DEPT
  3  WHERE EMP.DEPTNO = DEPT.DEPTNO
  4  AND EMP.EMPNO = 7698;
SELECT EMPNO, ENAME, DEPTNO, DNAME
                     *
1행에 오류:
ORA-00918: 열의 정의가 애매합니다
```

FROM절 뒤에 기술되는 테이블명이 길거나 너무 복잡한 경우에는 테이블 별칭을 사용하여 SQL 문장을 간략하게 작성할 수 있다. 테이블 별칭을 사용할 때는 다음의 작성지침을 준수하도록 한다.

- 테이블 별칭의 길이는 30문자까지 가능하지만 짧을수록 좋다.
- 테이블 별칭이 사용되면 SQL 문장내에서 모든 테이블명은 지정된 테이블 별칭을 사용해야만 한다.
- 테이블 별칭은 기술된 SELECT 문장내에서만 유효하다.

```
SQL> SELECT E.EMPNO, E.ENAME, D.DNAME
  2  FROM EMP E, DEPT D
  3  WHERE E.DEPTNO = D.DEPTNO
  4  AND E.EMPNO = 7698;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7698 BLAKE      SALES
```

이번에는 검색하려는 테이블의 개수가 2개 이상일 경우, Join하는 방법을 알아보자. ERD가 다음과 같다면, A1, B1, C1 컬럼을 검색할 수 있는 SQL 문장은 다음과 같다.

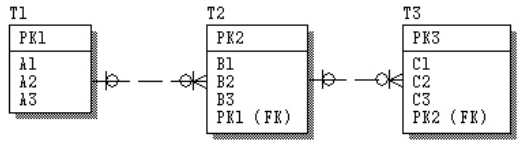


그림 5-4. Join 예제 1

```
SELECT T1.A1, T2.B1, T3.C1
FROM T1, T2, T3
WHERE T1.PK1 = T2.PK1
AND T2.PK2 = T3.PK2;
```

앞에서 설명한 바와 같이 SELECT절 뒤에는 검색하고자 하는 컬럼들을 나열하고, FROM절 뒤에는 해당 컬럼들이 포함된 테이블들을 기술한 뒤에 Join 조건을 차례로 작성한다. 위에서 보는 것처럼 Join할 테이블이 N개라면 Join 조건은 N-1개가 됨을 확인할 수 있다.

만약, 검색하고자 하는 컬럼이 A1, C1이라면 어떻게 Join 문장을 작성해야 되는지 생각해보자. 해당 컬럼이 포함되어 있는 테이블은 T1, T3이지만 직접적인 Join 조건이 존재하지 않으므로 이런 경우도 위와 마찬가지로 어쩔 수 없이 3개의 테이블을 모두 Join 해야 한다.

```
SELECT T1.A1, T3.C1
FROM T1, T2, T3
WHERE T1.PK1 = T2.PK1
AND T2.PK2 = T3.PK2;
```

이번에는 다음과 같은 ERD에서 A1, C1 컬럼을 검색하는 경우, Join 문장은 어떻게 작성되어야 하는지 생각해보자.

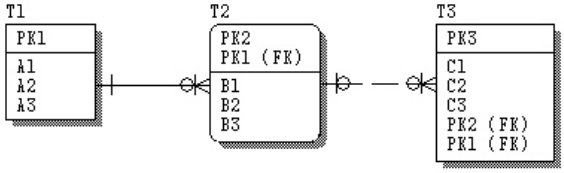


그림 5-5. Join 예제 2

ERD에서 보는 것처럼 A1, C1 컬럼이 포함되어 있는 T1, T3 테이블간에는 앞의 경우처럼 직접적인 관계가 존재하지 않아서 T1과 T3간에 Join 조건을 작성할 수 없을 것 같이 보이지만 T3 테이블에는 T1의 기본키인 PK1이 T3의 외래키로 추가되어 있기 때문에 직접 Join이 가능하다. 이와 같이 되는 이유는 T1과 T2간에 식별관계로 설정되어 T1의 기본키인 PK1이 T2의 외래키이면서 기본키에 참여했기 때문에 T2의 기본키(PK2, PK1)가 T3의 외래키로 추가되었기 때문이다.

```
SELECT T1.A1, T3.C1
FROM T1, T3
WHERE T1.PK1 = T3.PK1;
```

Non-EquiJoin

Non-EquiJoin은 테이블간에 Join 조건으로 Equal이 아닌 경우이다.

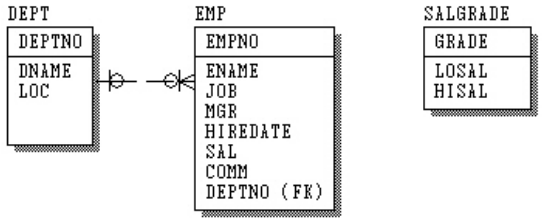


그림 5-6. 개체관계도(ERD)

실습 ERD에서 사원 테이블내의 급여가 급여등급 테이블내의 하한값과 상한값의 범위에 포함되는 경우 해당 등급를 검색하는 SQL 문장은 다음과 같다.

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

     EMPNO ENAME             SAL      GRADE
---------- ---------- ---------- ----------
      7369 SMITH             800          1
      7876 ADAMS            1100          1
      7900 JAMES             950          1
      7521 WARD             1250          2
        ...
      7902 FORD             3000          4
      7839 KING             5000          5

14 개의 행이 선택되었습니다.
```

Outer-Join

앞에서 설명한 Equi-Join과 Non-EquiJoin의 경우 Join 조건이 반드시 만족하는 경우에만 해당 테이블의 행들이 검색된다. 즉, DEPT 테이블과 EMP 테이블을 Equi-Join한 결과를 살펴보면 DEPT 테이블에 저장되어 있는 DEPTNO가 40번인 OPERATIONS 부서는 절대

로 검색되지 않는다. 그 이유는 EMP 테이블에 저장되어 있는 사원 데이터중에 DEPTNO가 40인 사원이 존재하지 않기 때문이다.

```
SQL> SELECT DEPTNO, DNAME
  2  FROM DEPT
  3  WHERE DEPTNO = 40;

    DEPTNO DNAME
---------- --------------
        40 OPERATIONS

SQL> SELECT EMPNO, ENAME, DEPTNO
  2  FROM EMP
  3  WHERE DEPTNO = 40;

선택된 레코드가 없습니다.
```

이런 경우, Outer-Join을 이용하면 40번 OPERATIONS 부서를 출력할 수 있다. 먼저, Outer-Join의 문법은 다음과 같다.

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column = table2.column(+);
```

위에서 보는 것처럼 Join 조건에 (+) 기호를 포함하는데, Join 조건의 양단에 모두 사용할 수는 없다. (+) 기호를 붙이는 쪽은 Join 할 때, 일치되는 행이 존재하지 않는 쪽에 붙이도록 한다.

즉, DEPT와 EMP 테이블에서 DEPT 테이블의 DEPTNO가 40인 행과 일치되는 행이 EMP 테이블에는 존재하지 않으므로 SQL 문장은 다음과 같이 작성하면 된다.

```
SQL> SELECT EMP.EMPNO, EMP.ENAME, DEPT.DNAME
  2  FROM EMP, DEPT
  3  WHERE EMP.DEPTNO(+) = DEPT.DEPTNO;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7782 CLARK      ACCOUNTING
      7839 KING       ACCOUNTING
      7934 MILLER     ACCOUNTING
        ...
      7521 WARD       SALES
                      OPERATIONS

15 개의 행이 선택되었습니다.
```

Self-Join

지금까지의 Join은 2개 이상의 테이블이 사용되었으나, Self-Join은 테이블 자신을 자신이 Join하는 방법이다. 실습 테이블인 EMP 테이블의 데이터를 살펴보면 MGR 컬럼이 있는데, 이 컬럼은 해당 사원의 담당 사수인 EMPNO를 참조하고 있다.

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

     EMPNO ENAME             MGR
---------- ---------- ----------
      7369 SMITH            7902
        ...
      7566 JONES            7839
        ...
      7839 KING
        ...
      7902 FORD             7566
      7934 MILLER           7782

14 개의 행이 선택되었습니다.
```

결과에서 보듯이 SMITH의 사수는 7902번 FORD이고, FORD의 사수는 7566번 JONES, JONES의 사수는 7839번 KING, KING의 사수는 존재하지 않으므로 NULL을 테이블에 저장한 것이다. 그렇다면, 사원명과 사수명을 동시에 검색하는 방법을 생각해보자. 사원명과 사수명은 모두 ENAME 이므로, Join을 사용하지 않은 일반 SQL 문장으로는 절대 검색할 수 없다. 이런 경우는 테이블 별칭을 사용하여 EMP 테이블을 사원 테이블과 사수 테이블로 명명하여 Join 하면 된다. 즉, 사원 테이블의 MGR이 사수 테이블의 EMPNO를 Equal 조건으로 Join 하면 다음과 같이 검색할 수 있다.

```
SQL> SELECT W.ENAME "사원", M.ENAME "사수"
  2  FROM EMP W, EMP M
  3  WHERE W.MGR = M.EMPNO;

사원       사수
---------- ----------
SMITH      FORD
ALLEN      BLAKE
WARD       BLAKE
JONES      KING
MARTIN     BLAKE
BLAKE      KING
CLARK      KING
SCOTT      JONES
TURNER     BLAKE
ADAMS      SCOTT
JAMES      BLAKE
FORD       JONES
MILLER     CLARK

13 개의 행이 선택되었습니다.
```

추가적으로 위의 결과에서 사원 컬럼에 KING을, 사수 컬럼에 NULL을 표시하려면 Outer-Join을 사용하면 된다.

```
SQL> SELECT W.ENAME "사원", M.ENAME "사수"
  2  FROM EMP W, EMP M
  3  WHERE W.MGR = M.EMPNO(+);

사원       사수
---------- ----------
SMITH      FORD
ALLEN      BLAKE
WARD       BLAKE
JONES      KING
MARTIN     BLAKE
BLAKE      KING
CLARK      KING
SCOTT      JONES
KING
TURNER     BLAKE
ADAMS      SCOTT
JAMES      BLAKE
FORD       JONES
MILLER     CLARK

14 개의 행이 선택되었습니다.
```

SQL : 1999 Join

지금까지 살펴본 Join 구문은 Oracle 8i와 이전 버전에서 사용하는 문법이었다. 그러나, Oracle 9i 버전부터는 SQL : 1999 표준과의 호환성을 위해 Join 문장이 다수 추가되었다.

SQL : 1999 호환을 위해 추가된 Join 문법은 다음과 같다.

```
SELECT table1.column, table2.column
FROM table1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)];
```

■ Cross Join

Cross Join은 카르테시안 프로덕트와 동일하다.

```
SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM DEPT
  3  CROSS JOIN EMP;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      ACCOUNTING
      7369 SMITH      RESEARCH
      7369 SMITH      SALES
      7369 SMITH      OPERATIONS
      7499 ALLEN      ACCOUNTING
        ...
      7934 MILLER     OPERATIONS

56 개의 행이 선택되었습니다.
```

■ Natural Join

Natural Join은 Equi-Join과 동일하지만 기술하는 방법이 약간 다르다. 즉, Join 조건을 기술 할 필요가 없으며, FROM 절 뒤의 테이블내에서 동일한 이름을 같는 컬럼들을 찾아서 Equal 조건으로 Join 해준다.

```
SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM DEPT
  3  NATURAL JOIN EMP;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      RESEARCH
      7499 ALLEN      SALES
      7521 WARD       SALES
      7566 JONES      RESEARCH
        ...
      7934 MILLER     ACCOUNTING

14 개의 행이 선택되었습니다.
```

■ USING 구문을 이용한 Join

Natural Join에서 Join 해야 할 테이블 간에 동일한 이름을 갖는 컬럼이 여러 개인 경우에는 원하는 컬럼으로 Join 할 수가 없다. 이러한 경우, Join 하고자 하는 컬럼을 USING 구문 뒤에 기술하면 원하는 결과를 검색 할 수 있다.

```
SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM DEPT
  3  JOIN EMP
  4  USING (DEPTNO);

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      RESEARCH
      7499 ALLEN      SALES
      7521 WARD       SALES
      7566 JONES      RESEARCH
        ...
      7934 MILLER     ACCOUNTING

14 개의 행이 선택되었습니다.
```

한 가지 주의할 점은 USING 구문 뒤의 컬럼에는 테이블 이름 또는 테이블 별칭을 참조하면 에러가 발생할 수 있으니 주의하여야 하며, NATURAL JOIN과 USING 구문은 동시에 사용할 수 없다.

```
SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM DEPT
  3  JOIN EMP
  4  USING (DEPTNO)
  5  WHERE DEPT.DEPTNO =10;
WHERE DEPT.DEPTNO =10
      *
5행에 오류:
ORA-25154: USING 절의 열 부분은 식별자를 가질 수 없음
```

■ ON 구문을 이용한 Join

Natural Join은 테이블간의 동일한 이름을 갖는 컬럼에 대하여 Equi-Join을 수행하기 때문에 Non-EquiJoin이나 PK와 FK 컬럼의 이름이 서로 다른 경우는 Natural Join을 사용할 수 없다. 이러한 경우, ON 구문을 사용하여 임의의 조건 및 컬럼을 지정할 수 있다. 또한, ON 구문을 사용하면 다른 검색 조건과 분리되어 SQL 문장을 이해하기가 쉬워진다.

```
SQL> SELECT EMPNO, ENAME, DNAME
   2  FROM DEPT JOIN EMP
   3  ON DEPT.DEPTNO = EMP.DEPTNO
   4  AND DEPT.DEPTNO = 10;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7782 CLARK      ACCOUNTING
      7839 KING       ACCOUNTING
      7934 MILLER     ACCOUNTING
```

다음과 같은 ERD에서 ON 구문으로 여러 개의 테이블을 Join하는 방법은 다음과 같다.

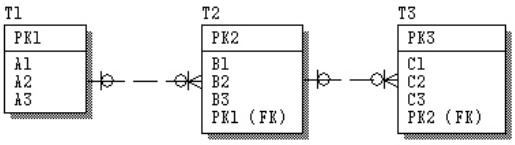


그림 5-4. Join 예제 3

```
SELECT T1.A1, T2.B1, T3.C1
FROM T1 JOIN T2
ON T1.PK1 = T2.PK1
JOIN T3
ON T2.PK2 = T3.PK2;
```

■ LEFT OUTER JOIN

LEFT OUTER JOIN은 LEFT OUTER JOIN 구문의 좌측에 기술한 테이블내의 모든 행들이 우측에 기술한 테이블내 행들과 일치 여부에 상관 없이 모두 출력된다.

```
SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM DEPT LEFT OUTER JOIN EMP
  3  ON DEPT.DEPTNO = EMP.DEPTNO;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      RESEARCH
      7499 ALLEN      SALES
      7521 WARD       SALES
        ...
      7934 MILLER     ACCOUNTING
                      OPERATIONS

15 개의 행이 선택되었습니다.
```

위의 문장은 다음 문장과 동일하다.

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

■ RIGHT OUTER JOIN

RIGHT OUTER JOIN은 RIGHT OUTER JOIN 구문의 우측에 기술한 테이블내의 모든 행들이 좌측에 기술한 테이블내 행들과 일치 여부에 상관 없이 모두 출력된다. 실습을 위해서 먼저 EMP 테이블에 새로운 행을 입력한다.

```
SQL> INSERT INTO EMP
  2  VALUES(8000, 'KIM', 'MANAGER', 7934, '04/06/22', 2000, NULL, NULL);

1 개의 행이 만들어졌습니다.

SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM DEPT RIGHT OUTER JOIN EMP
  3  ON DEPT.DEPTNO = EMP.DEPTNO;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7934 MILLER     ACCOUNTING
      7839 KING       ACCOUNTING
      7782 CLARK      ACCOUNTING
        ...
      7499 ALLEN      SALES
      8000 KIM

15 개의 행이 선택되었습니다.
```

위의 문장은 다음 문장과 동일하다.

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

■ FULL OUTER JOIN

LEFT OUTER JOIN과 RIGHT OUTER JOIN을 결합한 것이다. 즉, FULL OUTER JOIN의 좌측과 우측에 기술된 테이블내의 모든 행들이 일치되는 것은 물론이고 일치되지 않는 행들도 모두 출력된다. 사용 방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, DNAME
  2  FROM DEPT FULL OUTER JOIN EMP
  3  ON DEPT.DEPTNO=EMP.DEPTNO;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      RESEARCH
      7499 ALLEN      SALES
      7521 WARD       SALES
      7566 JONES      RESEARCH
        ...
      7934 MILLER     ACCOUNTING
                      OPERATIONS
      8000 KIM

16 개의 행이 선택되었습니다.
```

복습

1. 부서 테이블과 사원 테이블에서 사번, 사원명, 부서코드, 부서명을 검색하시오. 단, 출력시, 사원명을 기준으로 오름차순으로 정렬하시오.

2. 부서 테이블과 사원 테이블에서 사번, 사원명, 급여, 부서명을 검색하시오. 단, 급여가 2000 이상인 사원에 대하여 급여를 기준으로 내림차순으로 정렬하시오.

3. 부서 테이블과 사원 테이블에서 사번, 사원명, 업무, 급여, 부서명을 검색하시오. 단, 업무가 MANAGER이며 급여가 2500 이상인 사원에 대하여 사번을 기준으로 오름차순으로 정렬하시오.

4. 사원 테이블과 급여등급 테이블에서 사번, 사원명, 급여, 등급을 검색하시오. 단, 등급은 급여가 하한값과 상한값 범위에 포함되고 등급이 4이며 급여를 기준으로 내림차순으로 정렬하시오.

5. 부서 테이블, 사원 테이블, 급여등급 테이블에서 사번, 사원명, 부서명, 급여, 등급를 검색하시오. 단, 등급은 급여가 하한값과 상한값 범위에 포함되며 등급을 기준으로 내림차순으로 정렬하시오.

6. 사원 테이블에서 사원명과 해당 사원의 관리자명을 검색하시오.

7. 사원 테이블에서 사원명, 해당 사원의 관리자명, 해당 사원의 관리자의 관리자명을 검색하시오.

8. 7번 결과에서 상위 관리자가 없는 모든 사원의 이름도 사원명에 출력되도록 수정하시오.