



DML

1. 테이블에 새로운 행을 추가

- ❖ INSERT 문은 테이블에 새로운 데이터를 입력하기 위해 사용하는 데이터 조작용어
- ❖ INSERT 문의 기본 형식
INSERT INTO table_name
(column_name, ...)
VALUES(column_value, ...);



실습하기

- ❖ 부서 테이블(DEPT)과 동일한 구조를 갖되 데이터는 복사하지 않는 부서 테이블(DEPT01)을 생성하기 위해서 서브 쿼리문을 사용하여 테이블을 생성

```
DROP TABLE DEPT01;
```


```
CREATE TABLE DEPT01  
AS SELECT * FROM DEPT WHERE 1=0;
```



1.1 데이터 삽입

- ❖ 새로운 데이터를 추가하기 위해서 사용할 명령어 INSERT INTO ~ VALUES ~는 컬럼 명에 기술된 목록의 수와 VALUES 다음에 나오는 괄호에 기술한 값의 개수가 동일해야 함
- ❖ 컬럼 DEPTNO에 10을 컬럼 DNAME에는 'ACCOUNTING'을, 컬럼 LOC에는 'NEW YORK'을 추가

```
INSERT INTO DEPT01  
(DEPTNO, DNAME, LOC)  
VALUES(10, 'ACCOUNTING', 'NEW YORK');
```



1.2 INSERT 구문에서 오류

- ❖ 컬럼 명에 기술된 목록의 수보다 VALUES 다음에 나오는 괄호 안에 기술한 값의 개수가 적으면 에러가 발생

```
INSERT INTO DEPT01  
(DEPTNO, DNAME, LOC)  
VALUES (10, 'ACCOUNTING');
```

- ❖ 컬럼 명에 기술된 목록의 수보다 VALUES 다음에 나오는 괄호에 기술한 값의 개수가 많으면 에러가 발생

```
INSERT INTO DEPT01  
(DEPTNO, DNAME, LOC)  
VALUES(10, 'ACCOUNTING', 'NEW YORK', 20);
```



1.3 INSERT 구문에서 오류

- ❖ 컬럼 명이 잘못 입력되었을 때에도 에러가 발생

```
INSERT INTO DEPT01  
(NUM, DNAME, LOC)  
VALUES(10, 'ACCOUNTING', 'NEW YORK');
```

- ❖ 컬럼과 입력할 값의 데이터 타입이 서로 맞지 않을 경우에도 에러가 발생

```
INSERT INTO DEPT01  
(DEPTNO, DNAME, LOC)  
VALUES(10, ACCOUNTING, 'NEW YORK');
```



1.4 컬럼 명을 생략한 INSERT

- ❖ 테이블에 데이터를 추가할 때 모든 컬럼에 모두 자료를 입력하는 경우에는 굳이 컬럼 목록을 기술하지 않아도 됨
- ❖ 컬럼 목록이 생략되면 VALUES 절 다음의 값들이 테이블의 기본 컬럼 순서대로 입력
- ❖ 테이블의 컬럼 순서는 CREATE TABLE로 테이블을 생성할 때의 순서를 따름
- ❖ 테이블의 기본 컬럼 순서는 DESC 문으로 조회했을 때 보여 지는 순서
- ❖ 컬럼명을 생략한 채 테이블이 갖은 모든 컬럼에 데이터를 추가

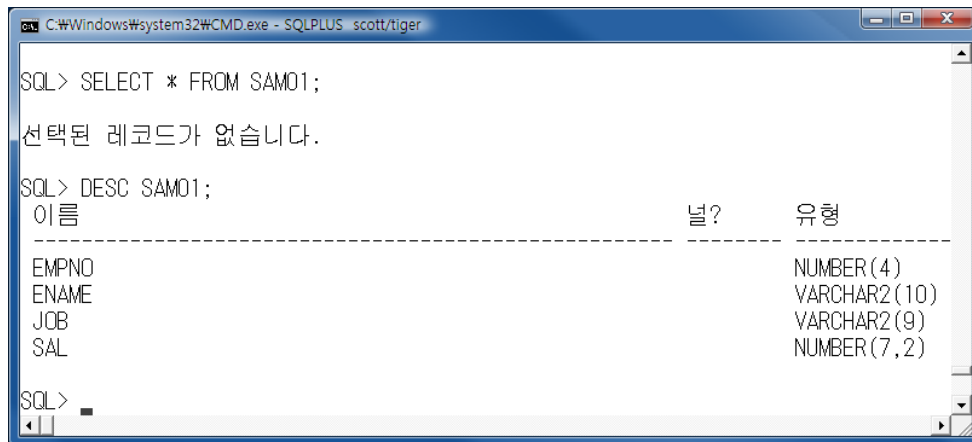
```
INSERT INTO DEPT01
```

```
VALUES (20, 'RESEARCH', 'DALLAS');
```



연습문제

- ❖ 서브 쿼리문을 이용하여 다음과 같은 구조로 SAM01 테이블을 생성
존재할 경우 DROP TABLE로 삭제 후 생성



A screenshot of a Windows command prompt window titled "C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger". The window shows the following text:

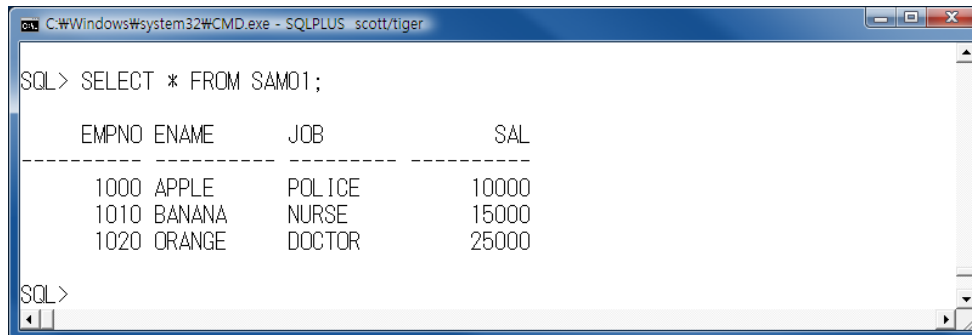
```
SQL> SELECT * FROM SAM01;

선택된 레코드가 없습니다.

SQL> DESC SAM01;
이름                                널?       유형
-----
EMPNO                                NUMBER(4)
ENAME                                VARCHAR2(10)
JOB                                  VARCHAR2(9)
SAL                                  NUMBER(7,2)

SQL>
```

- ❖ SAM01 테이블에 다음과 같은 데이터를 추가



A screenshot of a Windows command prompt window titled "C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger". The window shows the following text:

```
SQL> SELECT * FROM SAM01;

EMPNO  ENAME      JOB          SAL
-----
1000   APPLE      POL ICE      10000
1010   BANANA     NURSE        15000
1020   ORANGE     DOCTOR       25000

SQL>
```

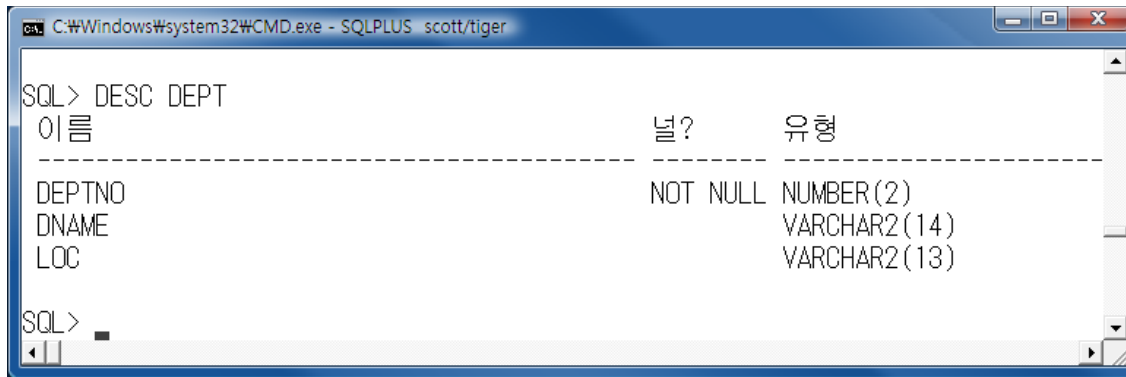

1.5 NULL 값 삽입

- ❖ 데이터를 입력하는 시점에서 해당 컬럼 값을 모르거나 확정되지 않았을 경우에는 NULL값을 입력
- ❖ NULL 값 삽입은 암시적인 방법과 명시적인 방법이 있음
- ❖ 암시적 방법은 컬럼 명 리스트에 컬럼을 생략하는 것으로 다른 컬럼은 값을 입력하지만 이렇게 생략한 컬럼에는 암시적으로 NULL 값이 할당되는 것
- ❖ 명시적 방법은 VALUES 리스트에 명시적으로 NULL을 입력



1.5 NULL 값 삽입

- ❖ 부서 테이블에 컬럼이 NULL값을 허용하는지 살펴보기 위해서 DESC 명령을 실행
- ❖ DEPT 테이블의 DEPTNO 컬럼은 NOT NULL 제약조건이 지정되어 있음



```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> DESC DEPT
이름                널?       유형
-----
DEPTNO              NOT NULL   NUMBER(2)
DNAME               VCHAR2(14)
LOC                 VCHAR2(13)

SQL>
```

- ❖ NOT NULL 제약조건이 지정된 DEPTNO 컬럼은 NULL 값을 입력하지 못함
- ❖ 오라클이 제공해 주는 DEPT 테이블의 DEPTNO 컬럼에 NULL값을 허용하지 못하도록 오라클 내부에서 이미 컬럼에 제약조건을 지정해 놓은 상태
- ❖ 컬럼에 NULL값을 허용하지 못하도록 하려면 컬럼에 제약조건을 지정해야 함

암시적으로 NULL 값의 삽입

- ❖ 지역명이 결정되지 않은 30번 부서에 부서명만 입력하고자 하는 경우 저장할 값을 명확하게 알고 있는 컬럼 명만 명시적으로 기술한 후에 그에 매칭되는 값을 VALUES 절 다음에 기술

```
INSERT INTO DEPT01  
(DEPTNO, DNAME)  
VALUES (30, 'SALES');
```



명시적으로 NULL 값의 삽입

- ❖ 컬럼명을 명시적으로 기술하지 않으면 테이블이 갖고 있는 모든 컬럼에 값을 지정해야 함
- ❖ 지역명이 결정되어 지지 않았더라도 반드시 값을 3개 지정해야 하기 때문에 명시적으로 VALUES 리스트에서 지역명에 NULL을 입력

```
INSERT INTO DEPT01  
VALUES (40, 'OPERATIONS', NULL);
```



명시적으로 NULL 값의 삽입

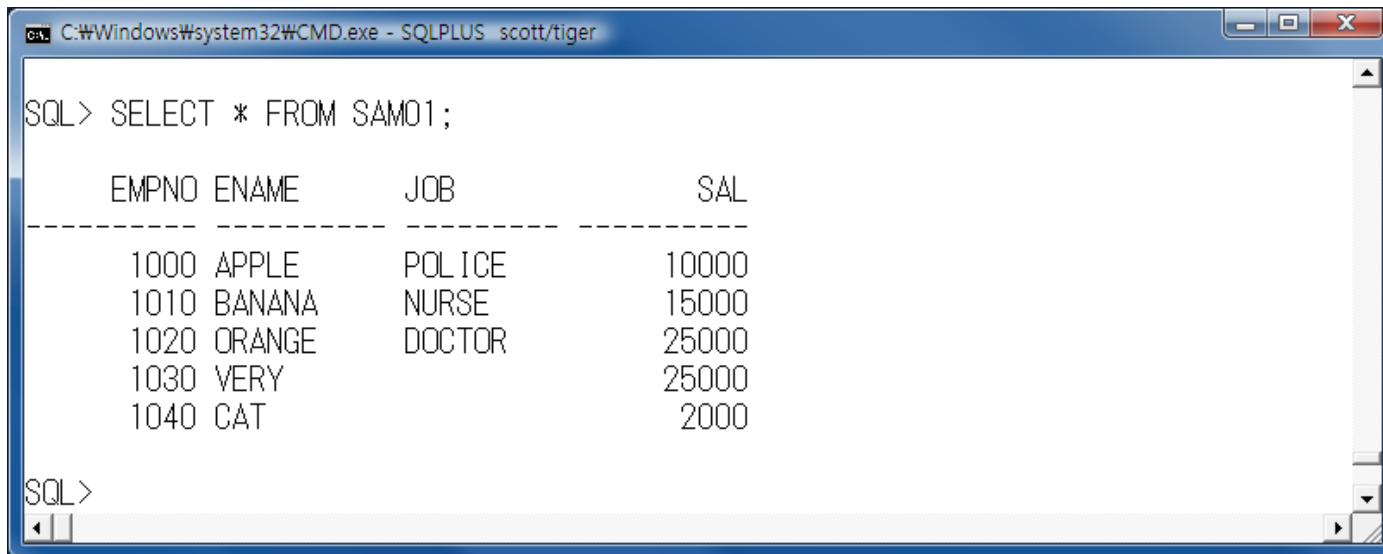
- ❖ NULL 값을 갖는 컬럼을 추가하기 위해서 NULL 대신 ''를 사용할 수 있음
- ❖ 지역명이 아닌 부서명이 결정되지 않아 부서명에 NULL 값을 입력

```
INSERT INTO DEPT01  
VALUES (50, '', 'CHICAGO');
```



연습문제

- ❖ SAM01 테이블에 다음과 같이 NULL 값을 갖는 행을 추가



```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAM01;

  EMPNO ENAME      JOB          SAL
-----
  1000  APPLE      POLICE      10000
  1010  BANANA     NURSE       15000
  1020  ORANGE     DOCTOR      25000
  1030  VERY              25000
  1040  CAT              2000

SQL>
```

1.6 서브 쿼리로 데이터 삽입

- ❖ INSERT INTO 다음에 VALUES 절을 사용하는 대신에 서브 쿼리를 사용할 수 있음
- ❖ 기존의 테이블에 있던 여러 행을 복사해서 다른 테이블에 삽입할 수 있음
- ❖ INSERT 명령문에서 지정한 컬럼의 개수나 데이터 타입이 서브 쿼리를 수행한 결과와 동일해야 함



실습하기

- ❖ 서브 쿼리로 데이터 삽입하기 위해서 우선 테이블을 생성하되 데이터는 복사하지 않고 빈 테이블만 생성

```
DROP TABLE DEPT02;
```

```
CREATE TABLE DEPT02  
AS  
SELECT * FROM DEPT WHERE 1=0;
```

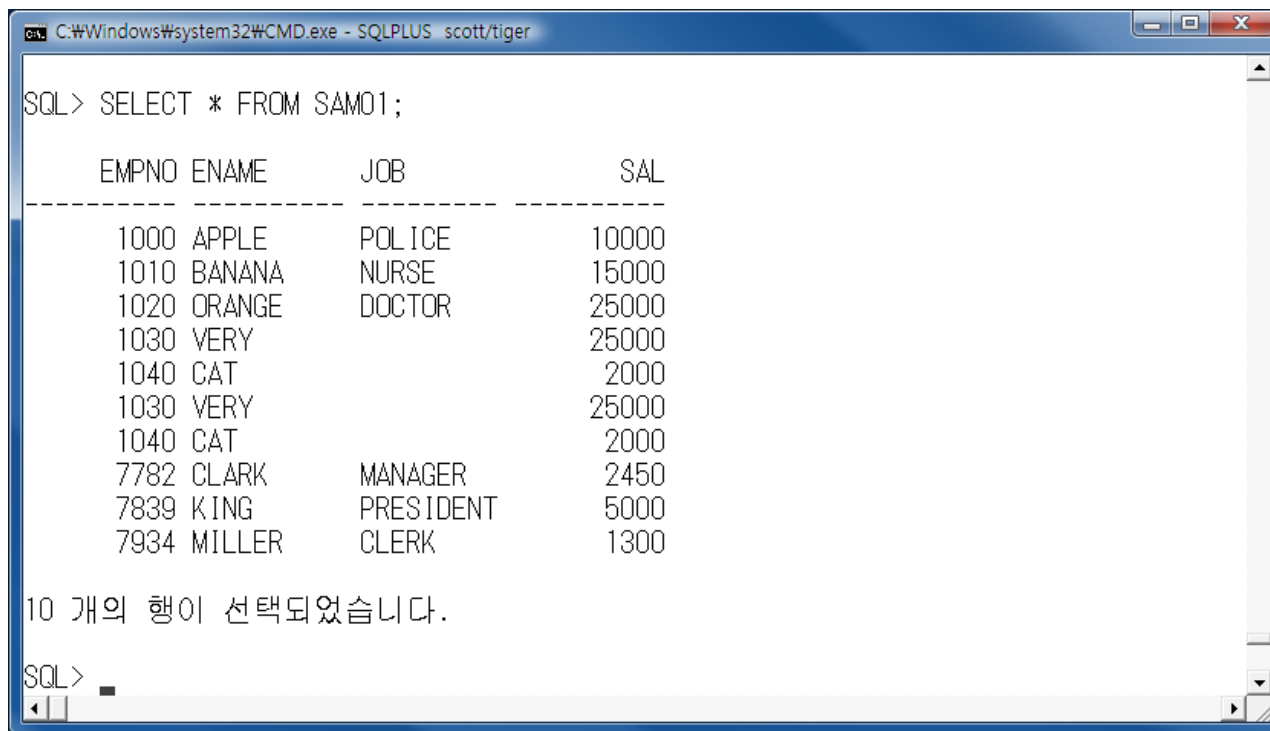
- ❖ 테이블 구조만을 복사해서 내용을 갖지 않는 테이블에 서브 쿼리로 로우를 입력

```
INSERT INTO DEPT02  
SELECT * FROM DEPT;
```



연습문제

- ❖ SAM01 테이블에 서브 쿼리문을 사용하여 EMP 에 저장된 사원 중 deptno가 10번 사원의 정보를 추가



```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAM01;

  EMPNO ENAME      JOB              SAL
-----
1000 APPLE        POLICE           10000
1010 BANANA       NURSE            15000
1020 ORANGE       DOCTOR           25000
1030 VERY                     25000
1040 CAT                      2000
1030 VERY                     25000
1040 CAT                      2000
7782 CLARK        MANAGER          2450
7839 KING         PRESIDENT        5000
7934 MILLER       CLERK            1300

10 개의 행이 선택되었습니다.

SQL>
```

2. 다중 테이블에 다중 행 입력

- ❖ INSERT ALL문을 사용하면 서브 쿼리의 결과를 조건 없이 여러 테이블에 동시에 입력 가능
- ❖ 사원번호, 사원명, 입사일자로 구성된 EMP_HIR 테이블과 사원번호, 사원명, 해당관리자(상관))로 구성된 EMP_MGR 테이블이 빈 테이블로 존재하는 경우에 사원 테이블(EMP)에서 부서 번호가 20인 사원들을 검색하여 EMP_HIR 테이블에는 사원 번호, 사원 명, 급여를 EMP_MGR 테이블에는 사원 번호, 사원 명, 해당관리자(상관)를 입력하고자 할 때 INSERT ALL 명령문을 사용하면 두 번의 쿼리문을 수행하지 않고도 하나의 쿼리문으로 두 개의 테이블에 원하는 컬럼 값을 삽입 가능
- ❖ INSERT ALL 명령문은 서브 쿼리의 결과 집합을 조건 없이 여러 테이블에 동시에 입력하기 위한 명령문
- ❖ 서브 쿼리의 컬럼명과 데이터가 입력되는 테이블의 컬럼명이 동일해야 가능

```
INSERT ALL  
  INTO EMP_HIR VALUES(EMPNO, ENAME, HIREDATE)  
  INTO EMP_MGR VALUES(EMPNO, ENAME, MGR)  
SELECT EMPNO, ENAME, HIREDATE, MGR  
FROM EMP  
WHERE DEPTNO=20;
```



2. 다중 테이블에 다중 행 입력

- ❖ INSERT ALL 명령문에 WHEN 절을 추가해서 조건을 제시하여 조건에 맞는 행만 추출하여 테이블에 추가
- ❖ EMP_HIR02 테이블에는 1982년 01월 01일 이후에 입사한 직원들의 번호, 직원 명, 입사일을 추가
- ❖ EMP_SAL 테이블에는 급여가 2000 이상인 직원들의 번호, 직원 명, 급여를 추가

```
INSERT ALL  
  WHEN HIREDATE > '1982/01/01' THEN  
  INTO EMP_HIR02 VALUES(EMPNO, ENAME, HIREDATE)  
  WHEN SAL > 2000 THEN  
  INTO EMP_SAL VALUES(EMPNO, ENAME, SAL)  
SELECT EMPNO, ENAME, HIREDATE, SAL FROM EMP;
```



실습하기

1. PIVOTING INSERT문을 실습하기 위해 한 주차 월요일부터 금요일까지 매일 매일의 판매 실적을 기록하는 테이블을 생성

```
CREATE TABLE SALES(  
  SALES_ID NUMBER(4),  
  WEEK_ID NUMBER(4),  
  MON_SALES NUMBER(8, 2),  
  TUE_SALES NUMBER(8, 2),  
  WED_SALES NUMBER(8, 2),  
  THU_SALES NUMBER(8, 2),  
  FRI_SALES NUMBER(8, 2));
```



실습하기

2. PIVOTING INSERT문의 결과를 저장할 SALES_DATA 테이블을 생성

```
CREATE TABLE SALES_DATA(  
    SALES_ID NUMBER(4),  
    WEEK_ID NUMBER(4),  
    DAILY_ID NUMBER(4),  
    SALES NUMBER(8, 2));
```

2. 새롭게 생성된 SALES 테이블에 주간 판매 실적을 추가

```
INSERT INTO SALES VALUES  
(1001, 1, 200, 100, 300, 400, 500);  
INSERT INTO SALES VALUES  
(1002, 2, 100, 300, 200, 500, 350);
```



<실습하기>

4. 각 요일을 구분할 수 있는 컬럼을 추가하여 매일 매일의 판매 실적을 기록
- ```
INSERT ALL
 INTO SALES_DATA VALUES(SALES_ID, WEEK_ID, 1, MON_SALES)
 INTO SALES_DATA VALUES(SALES_ID, WEEK_ID, 2, TUE_SALES)
 INTO SALES_DATA VALUES(SALES_ID, WEEK_ID, 3, WED_SALES)
 INTO SALES_DATA VALUES(SALES_ID, WEEK_ID, 4, THU_SALES)
 INTO SALES_DATA VALUES(SALES_ID, WEEK_ID, 5, FRI_SALES)
SELECT SALES_ID, WEEK_ID, MON_SALES, TUE_SALES, WED_SALES,
 THU_SALES, FRI_SALES
FROM SALES;
```



### 3. 테이블의 내용을 수정

- ❖ UPDATE 문은 테이블에 저장된 데이터를 수정하기 위해서 사용

```
UPDATE table_name
SET column_name1 = value1, column_name2 = value2, ...
WHERE conditions;
```

- ❖ UPDATE 문은 기존의 행을 수정하는 것으로 어떤 행의 데이터를 수정하는지 WHERE 절을 이용하여 조건을 지정할 수 있음
- ❖ WHERE 절을 사용하지 않을 경우는 테이블에 있는 모든 행이 수정
- ❖ 테이블의 전체 행을 수정하려고 했던 것이 아니라면 WHERE 절의 사용 유무를 판단



# 실습하기

- ❖ 모든 사원의 부서번호를 30번으로 수정

```
UPDATE EMP01
SET DEPTNO=30;
```

- ❖ 모든 사원의 급여를 10% 인상시키는 UPDATE

```
UPDATE EMP01
SET SAL = SAL * 1.1;
```

- ❖ 모든 사원의 입사일을 오늘로 수정

```
UPDATE EMP01
SET HIREDATE = SYSDATE;
```





## 3.2 테이블의 특정 행만 변경

- ❖ UPDATE 문에 WHERE 절을 추가하면 테이블의 특정 행이 변경
- ❖ 사원 테이블(EMP01)을 제거한 후 다시 기존에 있던 사원 테이블(EMP)과 동일한 구조와 데이터를 갖는 사원 테이블(EMP01)을 생성

```
DROP TABLE EMP01;
```

```
CREATE TABLE EMP01
AS
SELECT *
FROM EMP
WHERE 1=0;
```



## 3.2 테이블의 특정 행만 변경

- ❖ 부서번호가 10번인 사원의 부서번호를 30번으로 수정

```
UPDATE EMP01
SET DEPTNO=30
WHERE DEPTNO=10;
```

- ❖ 급여가 3000 이상인 사원만 급여를 10% 인상

```
UPDATE EMP01
SET SAL = SAL * 1.1
WHERE SAL >= 3000;
```



# 실습하기

- ❖ 1982년에 입사한 사원의 입사일을 오늘로 수정

```
UPDATE EMP01
SET HIREDATE = SYSDATE
WHERE SUBSTR(HIREDATE, 1, 2)='82';
```



# 연습문제

- ❖ SAM01 테이블에 저장된 사원 중 급여가 10000 이상인 사원들의 급여만 5000원씩 삭감

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAM01;
```

| EMPNO | ENAME  | JOB       | SAL   |
|-------|--------|-----------|-------|
| 1000  | APPLE  | POLICE    | 10000 |
| 1010  | BANANA | NURSE     | 15000 |
| 1020  | ORANGE | DOCTOR    | 25000 |
| 1030  | VERY   |           | 25000 |
| 1040  | CAT    |           | 2000  |
| 1030  | VERY   |           | 25000 |
| 1040  | CAT    |           | 2000  |
| 7782  | CLARK  | MANAGER   | 2450  |
| 7839  | KING   | PRESIDENT | 5000  |
| 7934  | MILLER | CLERK     | 1300  |

10 개의 행이 선택되었습니다.

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAM01;
```

| EMPNO | ENAME  | JOB       | SAL   |
|-------|--------|-----------|-------|
| 1000  | APPLE  | POLICE    | 5000  |
| 1010  | BANANA | NURSE     | 10000 |
| 1020  | ORANGE | DOCTOR    | 20000 |
| 1030  | VERY   |           | 20000 |
| 1040  | CAT    |           | 2000  |
| 1030  | VERY   |           | 20000 |
| 1040  | CAT    |           | 2000  |
| 7782  | CLARK  | MANAGER   | 2450  |
| 7839  | KING   | PRESIDENT | 5000  |
| 7934  | MILLER | CLERK     | 1300  |

10 개의 행이 선택되었습니다.

## 3.3 2개 이상의 컬럼 값 변경

- ❖ 테이블에서 하나의 컬럼이 아닌 복수 개 컬럼의 값을 변경하려면 기존 SET 절에 콤마를 추가하고 컬럼=값을 추가 기술
- ❖ SCOTT 사원의 부서번호는 20번으로, 직급은 MANAGER로 한꺼번에 수정

```
UPDATE EMP01
SET DEPTNO=20, JOB='MANAGER'
WHERE ENAME='SCOTT';
```

- ❖ SCOTT 사원의 입사일자는 오늘로, 급여를 50 으로 커미션을 4000 으로 수정

```
UPDATE EMP01
SET HIREDATE = SYSDATE, SAL=50, COMM=4000
WHERE ENAME='SCOTT';
```



## 3.4 서브 쿼리로 데이터 수정

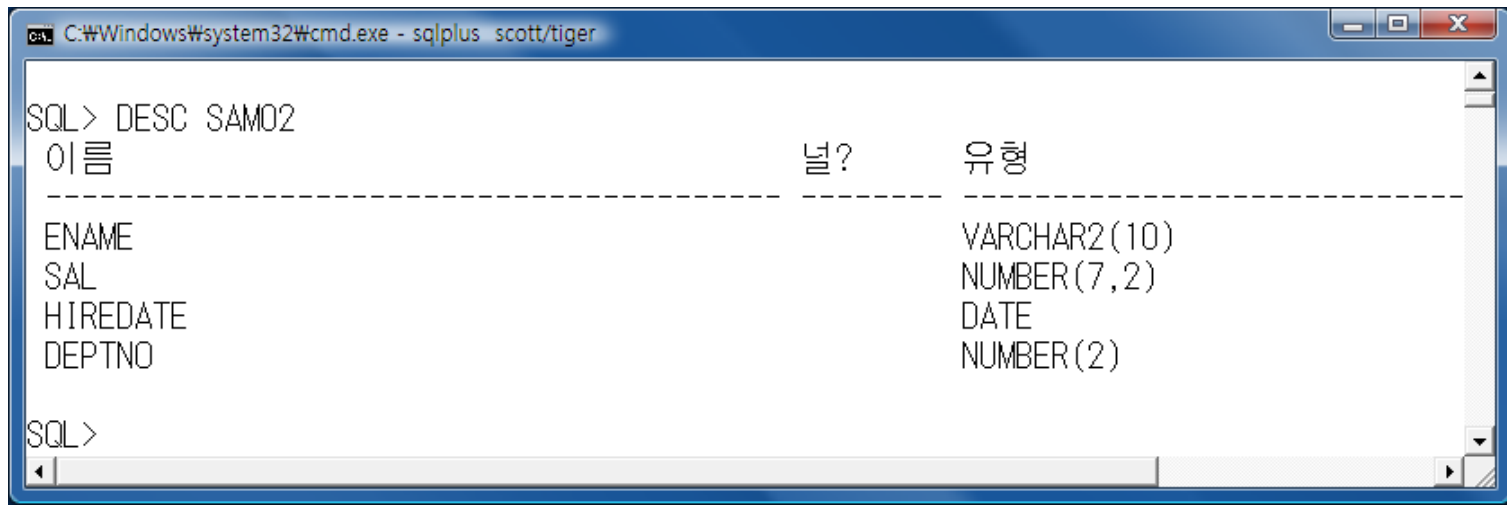
- ❖ UPDATE 문의 SET 절에서 서브 쿼리를 기술하면 서브 쿼리를 수행한 결과로 내용이 변경
- ❖ 이러한 방법으로 다른 테이블에 저장된 데이터로 해당 컬럼 값을 변경할 수 있음
- ❖ 20번 부서의 지역명을 40번 부서의 지역명으로 변경하기 위한 서브 쿼리

```
UPDATE DEPT01
SET LOC=(SELECT LOC
 FROM DEPT01
 WHERE DEPTNO=40)
WHERE DEPTNO=20;
```



# 연습문제

- ❖ 서브 쿼리를 사용하여 EMP 테이블의 저장된 데이터의 특정 컬럼만으로 구성된 SAM02 테이블을 생성



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

SQL> DESC SAM02
이름 널? 유형

ENAME VARCHAR2(10)
SAL NUMBER(7,2)
HIREDATE DATE
DEPTNO NUMBER(2)

SQL>
```

# 연습문제

❖ 생성 후 DALLAS 에 위치한 부서 소속 사원들의 급여를 1000 인상하시오.

[변경 전]

[변경 후]

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAMO2;
```

| ENAME  | SAL  | HIREDATE | DEPTNO |
|--------|------|----------|--------|
| SMITH  | 800  | 80/12/17 | 20     |
| ALLEN  | 1600 | 81/02/20 | 30     |
| WARD   | 1250 | 81/02/22 | 30     |
| JONES  | 2975 | 81/04/02 | 20     |
| MARTIN | 1250 | 81/09/28 | 30     |
| BLAKE  | 2850 | 81/05/01 | 30     |
| CLARK  | 2450 | 81/06/09 | 10     |
| SCOTT  | 3000 | 87/04/19 | 20     |
| KING   | 5000 | 81/11/17 | 10     |
| TURNER | 1500 | 81/09/08 | 30     |
| ADAMS  | 1100 | 87/05/23 | 20     |
| JAMES  | 950  | 81/12/03 | 30     |
| FORD   | 3000 | 81/12/03 | 20     |
| MILLER | 1300 | 82/01/23 | 10     |

14 개의 행이 선택되었습니다.

```
SQL>
```

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAMO2;
```

| ENAME  | SAL  | HIREDATE | DEPTNO |
|--------|------|----------|--------|
| SMITH  | 1800 | 80/12/17 | 20     |
| ALLEN  | 1600 | 81/02/20 | 30     |
| WARD   | 1250 | 81/02/22 | 30     |
| JONES  | 3975 | 81/04/02 | 20     |
| MARTIN | 1250 | 81/09/28 | 30     |
| BLAKE  | 2850 | 81/05/01 | 30     |
| CLARK  | 2450 | 81/06/09 | 10     |
| SCOTT  | 4000 | 87/04/19 | 20     |
| KING   | 5000 | 81/11/17 | 10     |
| TURNER | 1500 | 81/09/08 | 30     |
| ADAMS  | 2100 | 87/05/23 | 20     |
| JAMES  | 950  | 81/12/03 | 30     |
| FORD   | 4000 | 81/12/03 | 20     |
| MILLER | 1300 | 82/01/23 | 10     |

14 개의 행이 선택되었습니다.

```
SQL>
```



## 3.5 두개 이상의 컬럼 수정

❖ 서브 쿼리를 사용한 UPDATE 형식은 2가지

✓ 형식 1

UPDATE table\_name

SET column\_name1 = (sub\_query1), column\_name2 = (sub\_query2), ...

WHERE 조건

✓ 형식 2

UPDATE table\_name

SET (column\_name1, column\_name2, ...) = (sub\_query)

WHERE 조건



# 실습하기

- ❖ 서브 쿼리를 이용해서 부서번호가 20인 부서의 부서명과 지역명을 부서 번호가 40번인 부서와 동일하게 수정
- ❖ 부서 번호가 20번인 부서의 이름과 지역은 RESEARCH와 DALLAS이며 부서번호가 20인 부서의 부서명과 지역명을 부서 번호가 40번인 부서와 동일하게 변경하기 위한 UPDATE 명령문

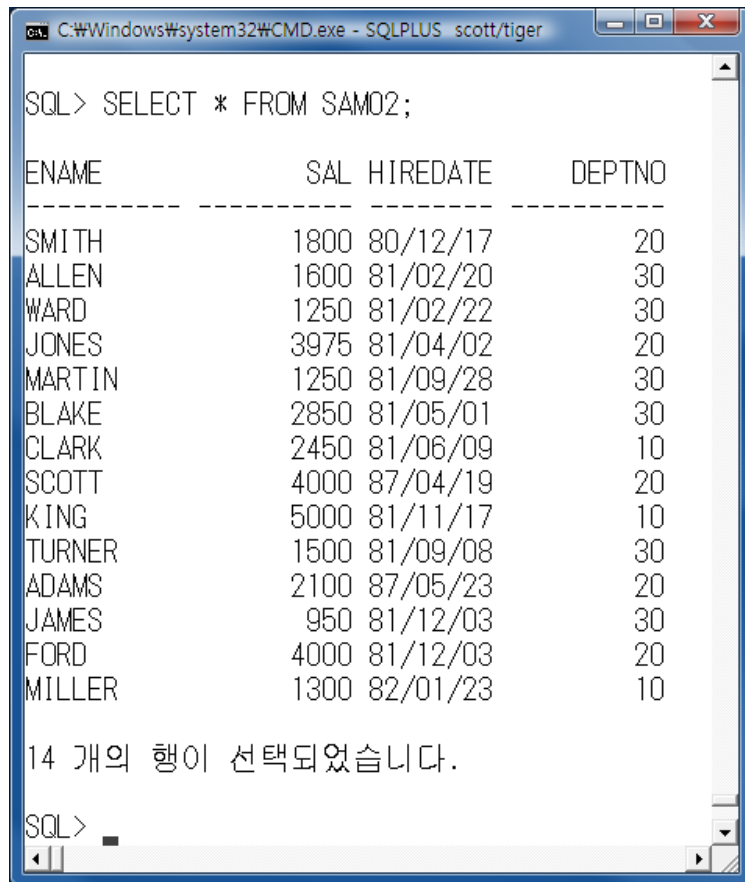
```
UPDATE DEPT01
SET (DNAME, LOC)=(SELECT DNAME, LOC
 FROM DEPT
 WHERE DEPTNO=40)
WHERE DEPTNO=20;
```



# 연습문제

- ❖ 서브 쿼리문을 사용하여 SAM02 테이블의 모든 사원의 급여와 입사일을 이름이 KING 인 사원의 급여와 입사일로 수정

[변경 전]



C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

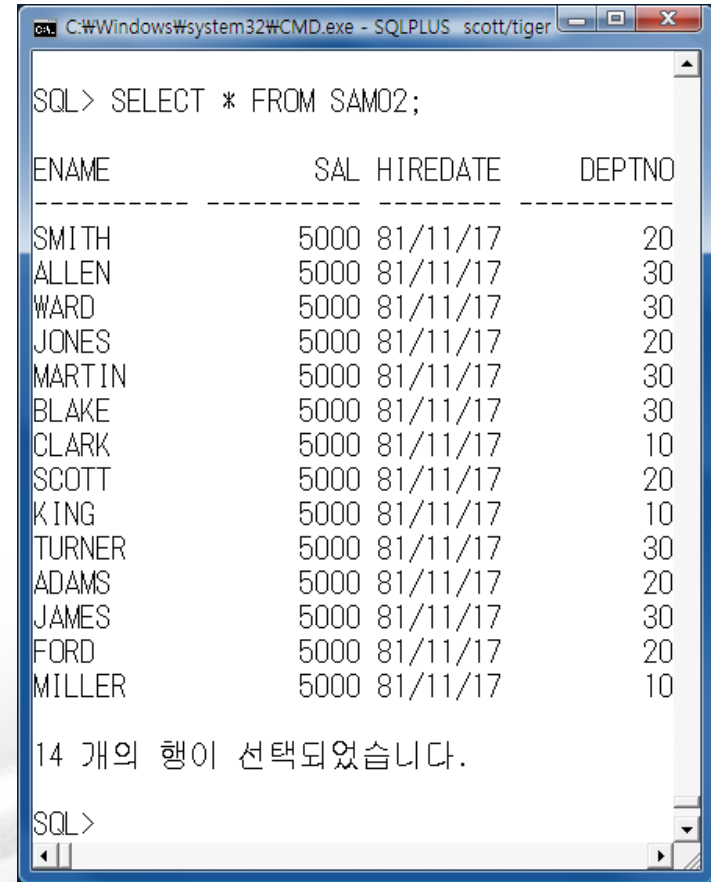
```
SQL> SELECT * FROM SAM02;
```

| ENAME  | SAL  | HIREDATE | DEPTNO |
|--------|------|----------|--------|
| SMITH  | 1800 | 80/12/17 | 20     |
| ALLEN  | 1600 | 81/02/20 | 30     |
| WARD   | 1250 | 81/02/22 | 30     |
| JONES  | 3975 | 81/04/02 | 20     |
| MARTIN | 1250 | 81/09/28 | 30     |
| BLAKE  | 2850 | 81/05/01 | 30     |
| CLARK  | 2450 | 81/06/09 | 10     |
| SCOTT  | 4000 | 87/04/19 | 20     |
| KING   | 5000 | 81/11/17 | 10     |
| TURNER | 1500 | 81/09/08 | 30     |
| ADAMS  | 2100 | 87/05/23 | 20     |
| JAMES  | 950  | 81/12/03 | 30     |
| FORD   | 4000 | 81/12/03 | 20     |
| MILLER | 1300 | 82/01/23 | 10     |

14 개의 행이 선택되었습니다.

SQL>

[변경 후]



C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

```
SQL> SELECT * FROM SAM02;
```

| ENAME  | SAL  | HIREDATE | DEPTNO |
|--------|------|----------|--------|
| SMITH  | 5000 | 81/11/17 | 20     |
| ALLEN  | 5000 | 81/11/17 | 30     |
| WARD   | 5000 | 81/11/17 | 30     |
| JONES  | 5000 | 81/11/17 | 20     |
| MARTIN | 5000 | 81/11/17 | 30     |
| BLAKE  | 5000 | 81/11/17 | 30     |
| CLARK  | 5000 | 81/11/17 | 10     |
| SCOTT  | 5000 | 81/11/17 | 20     |
| KING   | 5000 | 81/11/17 | 10     |
| TURNER | 5000 | 81/11/17 | 30     |
| ADAMS  | 5000 | 81/11/17 | 20     |
| JAMES  | 5000 | 81/11/17 | 30     |
| FORD   | 5000 | 81/11/17 | 20     |
| MILLER | 5000 | 81/11/17 | 10     |

14 개의 행이 선택되었습니다.

SQL>

## 4. 테이블의 행 삭제

- ❖ DELETE 문은 테이블에 저장되어 있는 데이터를 삭제

```
DELETE FROM table_name
WHERE conditions;
```

- ❖ DELETE 문은 테이블의 기존 행을 삭제하며 특정한 로우(행)을 삭제하기 위해서 WHERE 절을 이용하여 조건을 지정
- ❖ DELETE 문에 WHERE 절을 사용하지 않을 경우 테이블에 있는 모든 행이 삭제



# 실습하기

- ❖ DELETE 문으로 부서 테이블의 모든 행을 삭제  
DELETE FROM DEPT01;

The image shows two overlapping SQL\*Plus command windows. The top window shows the process of dropping and recreating the DEPT01 table. The bottom window shows the deletion of all data from the DEPT01 table.

**Top Window:**

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> DROP TABLE DEPT01;

테이블이 삭제되었습니다.

SQL> CREATE TABLE DEPT01
 2 AS
 3 SELECT * FROM DEPT;

테이블이 생성되었습니다.

SQL> SELECT * FROM DEPT01;
```

| DEPTNO | DNAME | LOC |
|--------|-------|-----|
| 10     | AC    |     |
| 20     | RE    |     |
| 30     | SA    |     |
| 40     | OP    |     |

**Bottom Window:**

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

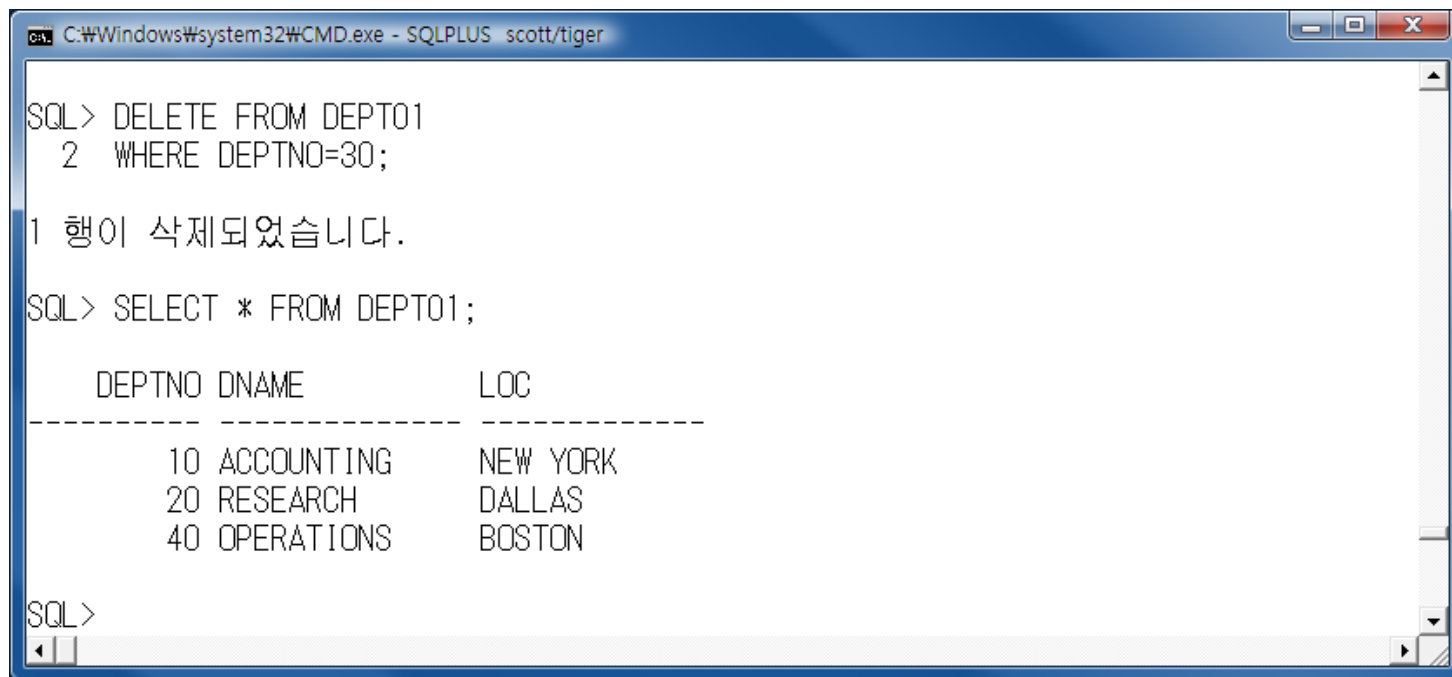
SQL> DELETE FROM DEPT01;

4 행이 삭제되었습니다.

SQL>
```

# 실습하기

- ❖ 부서 테이블에서 30번 부서만 삭제  
DELETE FROM DEPT01  
WHERE DEPTNO=30;



```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> DELETE FROM DEPT01
2 WHERE DEPTNO=30;

1 행이 삭제되었습니다.

SQL> SELECT * FROM DEPT01;

 DEPTNO DNAME LOC

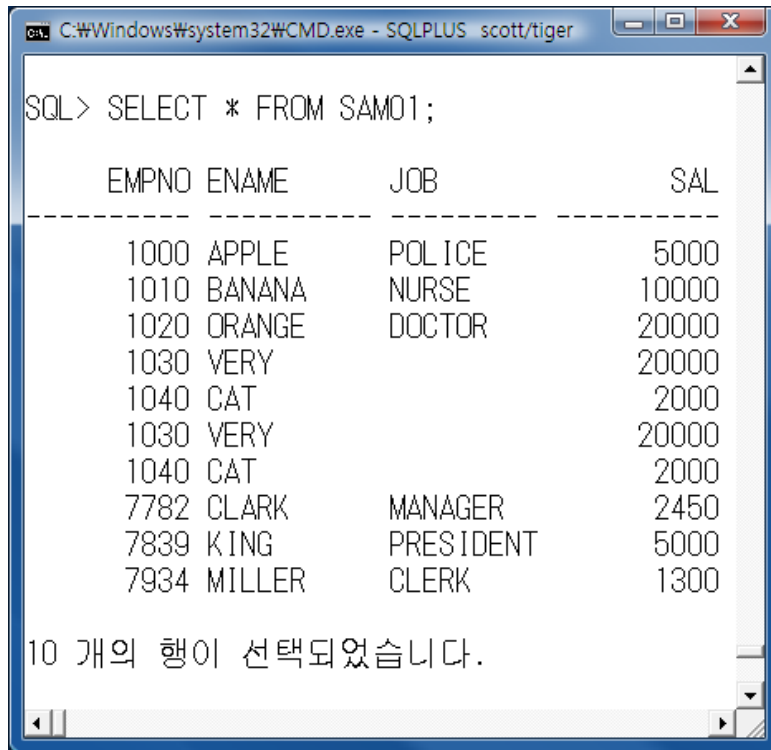
10 ACCOUNTING NEW YORK
20 RESEARCH DALLAS
40 OPERATIONS BOSTON

SQL>
```

# 연습문제

- ❖ SAM01 테이블에서 직급이 정해지지 않은 사원을 삭제

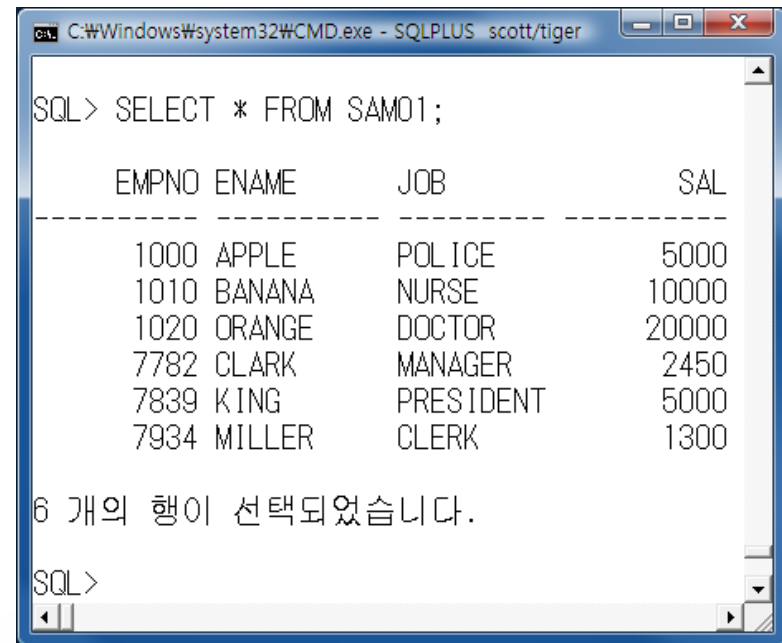
[삭제 전]



| EMPNO | ENAME  | JOB       | SAL   |
|-------|--------|-----------|-------|
| 1000  | APPLE  | POLICE    | 5000  |
| 1010  | BANANA | NURSE     | 10000 |
| 1020  | ORANGE | DOCTOR    | 20000 |
| 1030  | VERY   |           | 20000 |
| 1040  | CAT    |           | 2000  |
| 1030  | VERY   |           | 20000 |
| 1040  | CAT    |           | 2000  |
| 7782  | CLARK  | MANAGER   | 2450  |
| 7839  | KING   | PRESIDENT | 5000  |
| 7934  | MILLER | CLERK     | 1300  |

10 개의 행이 선택되었습니다.

[삭제 후]



| EMPNO | ENAME  | JOB       | SAL   |
|-------|--------|-----------|-------|
| 1000  | APPLE  | POLICE    | 5000  |
| 1010  | BANANA | NURSE     | 10000 |
| 1020  | ORANGE | DOCTOR    | 20000 |
| 7782  | CLARK  | MANAGER   | 2450  |
| 7839  | KING   | PRESIDENT | 5000  |
| 7934  | MILLER | CLERK     | 1300  |

6 개의 행이 선택되었습니다.

## 4.1 서브 쿼리를 이용한 삭제

- ❖ DELETE 문을 사용하기에 앞서 사원 테이블을 복사한 후사원 테이블에서 부서명이 SALES인 사원을 모두 삭제
- ❖ 사원 테이블에는 부서명이 기록되어 있지 않아서 부서명은 부서 테이블에 기록되어 있으므로 부서 테이블에서 부서명이 SALES인 부서의 번호부터 찾아서 이렇게 찾아낸 부서번호를 사원 테이블에 적용하기 위해서는 서브 쿼리를 이용
- ❖ 사원 테이블에서 부서명이 SALES인 사원을 모두 삭제

```
DELETE FROM EMP01
WHERE DEPTNO=(SELECT DEPTNO
 FROM DEPT
 WHERE DNAME='SALES');
```





# 연습문제

❖ SAM02 테이블에서 RESEARCH 부서 소속 사원들만 삭제

[변경 전]

[변경 후]

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAM02;

ENAME SAL HIREDATE DEPTNO

SMITH 5000 81/11/17 20
ALLEN 5000 81/11/17 30
WARD 5000 81/11/17 30
JONES 5000 81/11/17 20
MARTIN 5000 81/11/17 30
BLAKE 5000 81/11/17 30
CLARK 5000 81/11/17 10
SCOTT 5000 81/11/17 20
KING 5000 81/11/17 10
TURNER 5000 81/11/17 30
ADAMS 5000 81/11/17 20
JAMES 5000 81/11/17 30
FORD 5000 81/11/17 20
MILLER 5000 81/11/17 10

14 개의 행이 선택되었습니다.

SQL>
```

```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger

SQL> SELECT * FROM SAM02;

ENAME SAL HIREDATE DEPTNO

ALLEN 5000 81/11/17 30
WARD 5000 81/11/17 30
MARTIN 5000 81/11/17 30
BLAKE 5000 81/11/17 30
CLARK 5000 81/11/17 10
KING 5000 81/11/17 10
TURNER 5000 81/11/17 30
JAMES 5000 81/11/17 30
MILLER 5000 81/11/17 10

9 개의 행이 선택되었습니다.

SQL>
```

## 5. 테이블을 합병

- ❖ MERGE는 합병이란 의미이므로 구조가 같은 두개의 테이블을 하나의 테이블로 합치는 기능
- ❖ MERGE 명령을 수행하기 위해서 수행하는 테이블에 기존에 존재하는 행이 있다면 새로운 값으로 갱신(UPDATE)되고, 존재하지 않으면 새로운 행으로 추가(INSERT)
- ❖ MERGE 문에 의해서 갱신(UPDATE) 되고 추가(INSERT)
  - ✓ 새로운 테이블 EMP01 을 생성  
DROP TABLE EMP01;

```
CREATE TABLE EMP01
AS
SELECT *
FROM EMP;
```

```
SELECT *
FROM EMP01;
```



# 실습하기

- ✓ 직급이 'MANAGER' 인 사원들로만 구성된 EMP02 테이블을 생성

```
CREATE TABLE EMP02
AS
SELECT * FROM EMP
WHERE JOB='MANAGER';
```

- ✓ MERGE 문에 의해서 기존에 존재하는 행이 갱신(UPDATE) 되는지 살펴보기 위해서 새로 생성된 EMP02 테이블의 JOB을 'TEST' 로 변경

```
UPDATE EMP02
SET JOB='TEST';
```


- ✓ MERGE 문에 의해서 새로운 행이 추가(INSERT) 되는지 살펴보기 위해서 새로운 로우를 추가

```
INSERT INTO EMP02
VALUES(8000, 'SYJ', 'TOP', 7566, '2009/01/12', 1200, 10, 20);
```

# 실습하기

✓ EMP01 테이블에 EMP02 테이블을 합병

```
MERGE INTO EMP01
USING EMP02
ON(EMP01.EMPNO=EMP02.EMPNO)
WHEN MATCHED THEN
UPDATE SET
EMP01.ENAME=EMP02.ENAME,
EMP01.JOB=EMP02.JOB,
EMP01.MGR=EMP02.MGR,
EMP01.HIREDATE=EMP02.HIREDATE,
EMP01.SAL=EMP02.SAL,
EMP01.COMM=EMP02.COMM,
EMP01.DEPTNO=EMP02.DEPTNO
WHEN NOT MATCHED THEN
INSERT VALUES(EMP02.EMPNO, EMP02.ENAME, EMP02.JOB,
EMP02.MGR, EMP02.HIREDATE, EMP02.SAL,
EMP02.COMM, EMP02.DEPTNO);
```



# 연습문제

1. 아래의 구조를 만족하는 MY\_DATA 테이블을 생성

```
SQL> DESC MY_DATA
```

| Name   | Null?    | Type         |
|--------|----------|--------------|
| -----  |          |              |
| ID     | NOT NULL | NUMBER(4)    |
| NAME   |          | VARCHAR2(10) |
| USERID |          | VARCHAR2(30) |
| SALARY |          | NUMBER(10,2) |

2. 1번에 의해 생성된 테이블에 아래의 값을 입력

| ID | NAME   | USERID  | SALARY    |
|----|--------|---------|-----------|
| 1  | Scott  | sscott  | 10,000.00 |
| 2  | Ford   | fford   | 13,000.00 |
| 3  | Patel  | ppatel  | 33,000.00 |
| 4  | Report | rreport | 23,500.00 |
| 5  | Good   | ggood   | 44,450.00 |



# <연습문제>

3. 2번에서 입력한 자료를 확인 하여라.
4. ID가 3번인 사람의 급여를 65,000.00으로 갱신하여라.
5. 이름이 Ford인 사원을 삭제하여라.
6. 급여가 15,000이하인 사람의 급여를 15,000로 변경하여라.
7. 1번에서 생성한 테이블을 삭제하여라.

