< 데이터를 추가, 수정, 삭제하는 데이터 조작어 >

데이터 조작어로 부르는 DML(Data Manipulation Language) 명령어는 select문으로 조회한 테이블에 데이터를 추가•변경•삭제할 때 사용하는 명령어로 이루어져 있음. 데이터 조회를 위해 사용하는 SELECT문 다음으로 자주 사용하는 명령어이므로 반드시 알아야 함.

• 테이블에 데이터 추가하기

- 테이블 생성하기 : 회원 가입, 새 글 쓰기, 새로운 이체 내역 등 새로운 데이터가 발생하는 기능은 관련 테이블에 새 데이터를 추가해 줌으로써 구현할 수 있음. 특정 테이블에 데이터를 새로 추가할 때 INSERT문을 사용함.

• INSERT문 실습 전 유의점

① 테이블을 잘못 만들었을 때 : DROP TABLE 테이블 이름;

② 실습하는 중에 프로그램이 종료되었을 때 : 만약 실습 도중 프로그램을 종료하면 경고 창이 나타남. 이 경우에 먼저 COMMIT 버튼을 클릭하고 끝내면 됨.

• 테이블에 데이터를 추가하는 INSERT문

```
INSERT INTO 테이블 이름 [(열1, 열2, ..., 열N)]
VALUES (열1에 들어갈 데이터, 열2에 들어갈 데이터, ..., 열N에 들어갈 데이터);
```

키워드	필수 요소	선택 요소	설명
INSERT INTO	테이블 이름	테이블의 열	새로운 데이터를 입력할 대상 테이블과 열을 입력
VALUES	입력할 데이터	-	INSERT INTO에서 지정한 테이블의 열 순서와 자료형에 맞는 입력 데이터를 지정

• 서브쿼리를 사용하여 한 번에 여러 데이터 추가하기

INSERT문에 서브쿼리를 사용하면 SELECT문으로 한 번에 여러 행의 데이터를 추가할 수 있음.

예) EMP 테이블에서 SALGRADE 테이블을 참조하여 급여 등급(SALGRADE)이 1인 사원만을 EMP_TEMP 테이블에 집어넣고 싶다면 서브쿼리를 포함한 INSERT문을 사용하면 됨.

INSERT문에서 서브쿼리를 사용할 때 유의할 점

① VALUES절은 사용하지 않음.

② 데이터가 추가되는 테이블의 열 개수와 서브쿼리의 열 개수가 일치해야 함.

③ 데이터가 추가되는 테이블의 자료형과 서브쿼리의 자료형이 일치해야 함.

• 테이블에 있는 데이터 수정하기

오라클에서는 특정 테이블에 저장되어 있는 데이터 내용을 수정할 때 UPDATE문을 사용함.

UPDATE문의 기본 사용법

```
UPDATE [변경할 테이블]
SET [변경할 열1]=[데이터], [변경할 열2]=[데이터], ..., [변경할 열n]=[데이터]
[WHERE 데이터를 변경할 대상 행을 선별하기 위한 조건];
```

키워드	필수 요소	선택 요소	설명
UPDATE	테이블 이름	-	데이터를 수정할 테이블을 지정
SET	변경할 열의 이름과 데이터	-	변경할 열을 선택하고 변경할 데이터를 입력
WHERE	-	변경 데이터를 선별하기 위한 조건식	테이블의 변경할 데이터 선별 조건식을 지정. 생략할 경우 테이블 내 지정된 모든 열의 데이터가 변경됨.

UPDATE문 사용할 때 유의점

테이블에 이미 존재하는 데이터를 수정하거나 삭제하는 기능을 수행하므로 SELECT문이나 INSERT문에 비해 위험성이 큰 명령어임. 실제로 실무에서도 UPDATE에 사용하는 WHERE 조건식이 정확한 데이터를 대상으로 하는지 따져봐야 함.

• 테이블에 있는 데이터 삭제하기

```
DELETE [FROM] [테이블 이름]
[WHERE 삭제할 대상 행을 선별하기 위한 조건식];
```

키워드	필수 요소	선택 요소	설명
DELETE	테이블 이름	FROM	데이터를 삭제할 테이블을 지정
WHERE	-	삭제 데이터를 선별하는 조건식	테이블의 삭제할 데이터를 선별하는 조건식을 지정. 생략할 경우 테이블의 모든 데이터를 삭제함.

앞으로 진행할 문제는 다음 SQL문을 실행하여 EMP, DEPT, SALGRADE 테이블을 복사한 테이블을 진행하세요.
복사해서 만들어지는 테이블명은 CHAP10HW_EMP, CHAP10HW_DEPT, CHAP10HW_SALGRADE입니다.
실습1) CHAP10HW_DEPT 테이블에 50, 60, 70, 80번 부서를 등록하는 쿼리를 작성하세요.

50	ORACLE	BUSAN
60	SQL	ILSAN
70	SELECT	INCHEON
80	DML	BUNDANG

```
INSERT INTO CHAP10HW_DEPT (DEPTNO, DNAME, LOC) VALUES (50, 'ORACLE', 'BUSAN');
INSERT INTO CHAP10HW_DEPT (DEPTNO, DNAME, LOC) VALUES (60, 'SQL', 'ILSAN');
INSERT INTO CHAP10HW_DEPT (DEPTNO, DNAME, LOC) VALUES (70, 'SELECT', 'INCHEON');
INSERT INTO CHAP10HW_DEPT (DEPTNO, DNAME, LOC) VALUES (80, 'DML', 'BUNDANG');
```

실습2) 다음과 같이 CHAP10HW_EMP 테이블에 다음 8명의 사원 정보를 등록하는 쿼리를 작성하세요.

7201	TEST_USER1	MANAGER	7788	2016-01-02	4500		50
7202	TEST_USER2	CLERK	7201	2016-02-21	1800		50
7203	TEST_USER3	ANALYST	7201	2016-04-11	3400		60
7204	TEST_USER4	SALESMAN	7201	2016-05-31	2700	300	60
7205	TEST_USER5	CLERK	7201	2016-07-20	2600		70
7206	TEST_USER6	CLERK	7201	2016-09-08	2600		70
7207	TEST_USER7	LECTURER	7201	2016-10-28	2300		80
7208	TEST_USER8	STUDENT	7201	2018-03-09	1200		80

```
INSERT INTO CHAP10HW_EMP
  VALUES(7201, 'TEST_USER1', 'MANAGER', 7788, TO_DATE('2016-01-02', 'YYYY-MM-DD'), 4500, NULL, 50);
INSERT INTO CHAP10HW_EMP
  VALUES(7202, 'TEST_USER2', 'CLERK', 7201, TO_DATE('2016-02-21', 'YYYY-MM-DD'), 1800, NULL, 50);
INSERT INTO CHAP10HW_EMP
  VALUES(7203, 'TEST_USER3', 'ANALYST', 7201, TO_DATE('2016-04-11', 'YYYY-MM-DD'), 3400, NULL, 60);
INSERT INTO CHAP10HW_EMP
  VALUES(7204, 'TEST_USER4', 'SALESMAN', 7201, TO_DATE('2016-05-31', 'YYYY-MM-DD'), 2700, 300, 60);
INSERT INTO CHAP10HW_EMP
  VALUES(7205, 'TEST_USER5', 'CLERK', 7201, TO_DATE('2016-07-20', 'YYYY-MM-DD'), 2600, NULL, 70);
INSERT INTO CHAP10HW_EMP
```

```
  VALUES(7206, 'TEST_USER6', 'CLERK', 7201, TO_DATE('2016-09-08', 'YYYY-MM-DD'), 2600,
NULL, 70);
INSERT INTO CHAP10HW_EMP
  VALUES(7207, 'TEST_USER7', 'LECTURER', 7201, TO_DATE('2016-10-28', 'YYYY-MM-DD'),
2300, NULL, 80);
INSERT INTO CHAP10HW_EMP
  VALUES(7208, 'TEST_USER8', 'STUDENT', 7201, TO_DATE('2018-03-09', 'YYYY-MM-DD'),
1200, NULL, 80);
```

실습3) CHAP10HW_EMP에 속한 사원 중 50번 부서에서 근무하는 사원들의 평균 급여보다 많은 급여를 받고 있는 사원들을 70번 부서로 옮기는 쿼리를 작성하세요.

```
UPDATE CHAP10HW_EMP
   SET DEPTNO = 70
 WHERE SAL > (SELECT AVG(SAL)
                FROM CHAP10HW_EMP
               WHERE DEPTNO = 50);
```

< 여러 테이블을 하나의 테이블처럼 사용하는 조인 >

관계형 데이터베이스는 여러 종류의 데이터가 다양한 테이블에 나뉘어 저장되는 특성이 있음. 응용 프로그램이나 업무에 사용하는 SQL문은 대부분 단일 테이블의 조회보다 여러 테이블의 데이터를 조합하여 출력해야 하는 경우가 많음. 이를 가능하게 해 주는 조회 방식이 바로 조인.

• 조인 : 두 개 이상의 테이블을 연결하여 하나의 테이블처럼 출력할 때 사용하는 방식

- 집합 연산자와 조인의 차이점 : 집합 연산자를 사용한 결과는 두 개 이상 SELECT문의 결과값을 세로로 연결한 것이고, 조인을 사용한 결과는 두 개 이상의 테이블 데이터를 가로로 연결한 것.
- 여러 테이블을 사용할 때의 FROM절
  지금까지 사용한 SELECT문은 다음과 같이 FROM절에 EMP 테이블 하나만을 명시했음. 하지만 FROM절에는 여러 개 테이블을 지정하는 것이 가능함. 꼭 테이블이 아니더라도 테이블 형태, 즉 열과 행으로 구성된 데이터 집합이면 모두 FROM절에 지정 가능함. 뷰(view), 서브쿼리(subquery) 등이 이에 해당함.

```
SELECT 열1, 열2, ..., 열N
   FROM EMP
  WHERE 조건식
GROUP BY 그룹식
   HAVING 그룹조건식
ORDER BY 정렬식
```

```
SELECT FROM 테이블1, 테이블2, ..., 테이블N
```

• 조인 종류

① 등가 조인(equi join) : 테이블을 연결한 후에 출력 행을 각 테이블의 특정 열에 일치한 데이터를 기준으로 선정하는 방식. 내부 조인(inner join) 또는 단순 조인(simple join)으로 부르기도 함. 일반적으로 가장 많이 사용되는 조인 방식. 따라서 외부 조인과 같이 이름을 특별히 명시하지 않으면 '조인을 사용한다'는 것은 대부분 등가 조인, 즉 특정 열 값이 일치한 출력 결과를 사용하는 방식이라고 보면 됨.

- 여러 테이블의 열 이름이 같을 때 유의점 : 등가 조인을 사용할 때 조인 조건이 되는 각 테이블의 열 이름이 같을 경우에 해당 열 이름을 테이블 구분 없이 명시하면 오류가 발생하여 실행되지 못함.

예제) EMP 테이블 별칭을 E로, DEPT 테이블 별칭은 D로 하여 등가 조인을 했을 때 급여가 2500이하이고 사원 번호가 9999이하인 사원의 정보가 출력되도록 쿼리를 작성하시오.

- 조인 테이블 개수와 조건식 개수의 관계

A와 B 테이블을 조인할 경우에 A와 B를 정확히 연결해 주는 열이 하나 필요함. 앞에서 EMP 테이블과 DEPT 테이블의 DEPTNO 열과 같은 역할을 함. 만약 테이블이 A, B, C라면 A와 B를 연결해 줄 열 하나, A와 B가 연결된 상태에서 C를 연결해 줄 열 하나가 추가로 더 필요함. WHERE절의 조건식을 사용해 테이블을 조인할 때 반드시 각 테이블을 정확히 연결하는 조건식이 최소한 전체 테이블 수보다 하나 적은 수만큼은 있어야 함.

② 비등가 조인(non-equi join) : 등가 조인 방식 외의 방식을 의미함. 등가 조인 방식에 비해 자주 사용하는 방식이 아님. 하지만 조인 조건이 특정 열의 일치 여부를 검사하는 방식 외에 다른 방식도 사용할 수 있음을 기억할 것.

③ 자체 조인

EMP 테이블에는 직속 상관의 사원 번호가 저장된 MGR 열이 있음. EMP 테이블 사원 정보와 해당 사원의 직속 상관의 사원 번호를 나란히 함께 출력해야 하는 경우를 생각해 봅시다.

MGR 열이 특정 사원의 직속 상관의 사원 번호를 가리키는 데이터이므로 이 열의 데이터와 사원번호를 잘 이용하면 사원 정보와 직속 상관의 정보를 연결할 수 있음. 즉 현재 행에 MGR 열 값을 EMPNO 열 값으로 사용하고 있는 데이터를 연결해 주면 됨. 하지만 지금까지 사용한 SELECT문 방식으로는 현재 행을 벗어나 다른 행의 데이터를 가져올 수 없음. 특정 행의 MGR 열 데이터와 일치한 데이터가 EMPNO 열에 저장된 데이터를 가져와야 사원과 직속 상관을 나란히 출력할 수 있음. 이를 해결할 가장 손쉬운 방법은 EMP 테이블과 완전히 똑같은 테이블 하나 더 만들어 조인해 주는 것. 만약 EMP 테이블과 완전히 같은 COPY_EMP 테이블이 존재한다면 아래와 같이 SELECT문을 작성할 수 있고, WHERE절에 조인 조건을 작성할 수 있음.

```
SELECT * FROM EMP E, COPY_EMP C
  WHERE E.MGR = C.EMPNO;
```

④ 외부 조인

조인 조건 데이터 중 어느 한쪽이 NULL임에도 결과를 출력할 때 포함시켜야 하는 경우가 종종 있음. 어떤 두 테이블간 조인 수행에서 조인 기준 열의 어느 한쪽이 NULL이어도 강제로 출력하는 방식.

왼쪽 외부 조인(Left Outer Join)	WHERE TABLE1.COL1 = TABLE2.COL1(+)
오른쪽 외부 조인(Right Outer Join)	WHERE TABLE1.COL1(+) = TABLE2.COL1

외부 조인은 좌우를 따로 나누어 지정하는데 WHERE절에 조인 기준 열 중 한쪽에 (+) 기호를 붙여줌.

외부 조인은 조인 기준 열의 NULL을 처리하는 것을 목적으로 자주 사용하는 조인 방식. 하지만 (+) 기호를 붙이는 외부 조인 방식으로는 양쪽 모든 열이 외부 조인되는 '전체 외부 조인' 사용은 불가능함.

• SQL-99 표준 문법으로 배우는 조인

SQL문은 ISO/ANSI에서 관계형 데이터베이스 표준 언어로 지정(SQL-82)된 후 SQL-92를 거쳐 SQL-99 표준 문법이 나왔음. 오라클은 9i 버전부터 SQL-99 방식의 문법을 지원하고 있음. SQL-99 조인은 앞에서 배운 조인 방식과 기능은 같지만 조인을 사용하는 문법에서 다소 차이가 남. 다른 DBMS 제품에서도 사용할 수 있고, 앞에서 배운 조인 방식과 더불어 SQL-99 방식의 조인도 많이 사용하기 때문에 사용법을 알아야 함.

- NATURAL JOIN

앞에서 소개한 등가 조인을 대신해 사용할 수 있는 조인 방식으로 조인 대상이 두 테이블에 이름과 자료형이 같은 열을 찾은 후 그 열을 기준으로 등가 조인을 해 주는 방식.

- JOIN ~ USING

기존 등가 조인을 대신하는 조인 방식. NATURAL JOIN이 자동으로 조인 기준 열을 지정하는 것과 달리 USING 키워드에 조인 기준으로 사용할 열을 명시하여 사용함.

```
FROM TABLE1 JOIN TABLE2 USING (조인에 사용할 기준 열)
```

- JOIN ~ ON

JOIN ~ ON 방식에서는 기존 WHERE절에 있는 조인 조건식을 ON 키워드 옆에 작성함. 조인 기준 조건식은 ON에 명시하고 그 밖에 출력 행을 걸러 내기 위해 WHERE 조건식을 따로 사용하는 방식.

```
FROM TABLE1 JOIN TABLE2 ON (조인 조건식)
```

- OUTER JOIN

외부 조인에 사용. 다른 SQL-99 방식의 조인과 마찬가지로 WHERE절이 아닌 FROM절에서 외부 조인을 선언함.

왼쪽 외부 조인 (Left Outer Join)	기존	WHERE TABLE1.COL1 = TABLE2.COL1(+)
	SQL-99	**FROM** TABLE1 LEFT OUTER JOIN TABLE2 ON (조인 조건식)
오른쪽 외부 조인 (Right Outer Join)	기존	WHERE TABLE1.COL1(+) = TABLE2.COL1
	SQL-99	**FROM** TABLE1 RIGHT OUTER JOIN TABLE2 ON (조인 조건식)
전체 외부 조인 (Full Outer Join)	기존	기본 문법은 없음 (UNION 집합 연산자를 활용)
	SQL-99	**FROM** TABLE1 FULL OUTER JOIN TABLE2 ON (조인 조건식)

- SQL-99 조인 방식에서 세 개 이상의 테이블을 조인할 때

기존 조인 방식은 FROM절에 조인 테이블을 명시하고 조인 관련 조건식을 WHERE절에 명시하기 때문에 테이블 수가 두 개를 넘더라도 다음과 같이 작성하면 문제가 없음.

```
FROM TABLE1, TABLE2, TABLE3
WHERE TABLE1.COL = TABLE2.COL
AND TABLE2.COL = TABLE3.COL
```

하지만 FROM절에 조인 관련 내용을 작성해야 하는 SQL-99 방식에서는 테이블의 개수가 두 개를 넘어갈 때 어떻게 조인해야 할지 막막할 수 있음. 여러 가지 조인 키워드 방식이 있지만 다음과 같이 FROM절에 두 개 테이블을 키워드로 조인한 바로 옆에 SQL-99 방식의 조인 내용을 추가로 작성하면 세 개 이상의 테이블도 조인할 수 있음.

```
FROM TABLE1 JOIN TABLE2 ON (조건식)
JOIN TABLE3 ON (조건식)
```

예제) 다음 실습은 JOIN~USING 키워드를 사용한 등가 조인 문제입니다. 다음 조건에 알맞도록 쿼리를 작성하시오.

조건1. EMP 테이블과 DEPT 테이블의 조인 조건은 부서 번호(DEPTNO)가 같을 때입니다.

조건2. 급여는 3000이상이며 직송상관이 반드시 있어야 합니다.

실습1) 급여가 2000 초과인 사원들의 부서 정보, 사원 정보를 출력하는 쿼리를 작성하시오.
(단 SQL-99 방식을 사용하여 작성하시오.)

```
SELECT DEPTNO, D.DNAME, E.EMPNO, E.ENAME, E.SAL
  FROM EMP E NATURAL JOIN DEPT D
 WHERE E.SAL > 2000;
```

실습2) 각 부서별 평균 급여, 최대 급여, 최소 급여, 사원수를 출력하는 쿼리를 작성하시오.
(단 SQL-99 방식을 사용하여 작성하시오.)

```
SELECT DEPTNO,
       D.DNAME,
       TRUNC(AVG(SAL)) AS AVG_SAL,
       MAX(SAL) AS MAX_SAL,
       MIN(SAL) AS MIN_SAL,
       COUNT(*) AS CNT
  FROM EMP E JOIN DEPT D USING (DEPTNO)
```

```
GROUP BY DEPTNO, D.DNAME;
```

실습3) 모든 부서 정보와 사원 정보를 부서 번호, 사원 이름순으로 정렬하여 출력하는 쿼리를 작성하시오. (단 SQL-99 방식을 사용하여 작성하시오.)

```
SELECT D.DEPTNO, D.DNAME, E.EMPNO, E.ENAME, E.JOB, E.SAL
  FROM EMP E RIGHT OUTER JOIN DEPT D ON (E.DEPTNO = D.DEPTNO)
ORDER BY D.DEPTNO, E.ENAME;
```

실습4) 모든 부서 정보, 사원 정보, 급여 등급 정보, 각 사원의 직속상관의 정보를 부서 번호, 사원 번호 순서로 정렬하여 출력하는 쿼리를 작성하시오. (단 SQL-99 방식을 사용하여 작성하시오.)

```
SELECT D.DEPTNO, D.DNAME,
       E.EMPNO, E.ENAME, E.MGR, E.SAL, E.DEPTNO,
       S.LOSAL, S.HISAL, S.GRADE,
       E2.EMPNO AS MGR_EMPNO,
       E2.ENAME AS MGR_ENAME
  FROM EMP E RIGHT OUTER JOIN DEPT D ON (E.DEPTNO = D.DEPTNO)
                LEFT OUTER JOIN SALGRADE S ON (E.SAL BETWEEN S.LOSAL AND S.HISAL)
                LEFT OUTER JOIN EMP E2 ON (E.MGR = E2.EMPNO)
ORDER BY D.DEPTNO, E.EMPNO;
```