DML

DML

DML

DML(Data Manipulation Language)

- 데이터 조작 언어
- 테이블에 값을 삽입, 수정, 삭제 하는 역할
- INSERT(삽입), UPDATE(수정), DELETE(삭제)

INSERT

INSERT

INSERT

- 테이블에 새로운 행을 추가하는 구문
- 추가할 때 마다 테이블의 행 개수가 증가

표현식

1. INSERT INTO 테이블명(컬럼명1,컬럼명2,…) VALUES(값1,값2,…..);
   - 값을 넣을 컬럼명을 입력하고 입력한 순서에 맞춰서 값을 입력
   - 명시한 컬럼에만 값이 들어가고 명시되지 않은 컬럼은 null

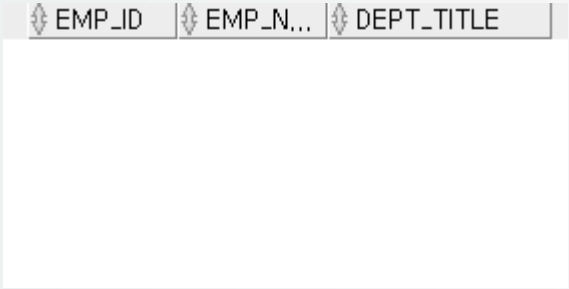
2. INSERT INTO 테이블명 VALUES(값1,값2,….);
   - 테이블 생성 시 만든 컬럼 순서대로 모든 값을 입력(null도 값으로 입력)
   - 컬럼 수와 입력한 값의 수가 맞지 않는 경우 에러 발생

INSERT

INSERT

테스트를 위한 테이블 생성

```
CREATE TABLE EMP_01(
    EMP_ID NUMBER,
    EMP_NAME VARCHAR2(30),
    DEPT_TITLE VARCHAR2(20)
);
```



INSERT

INSERT
- 2가지 방법을 통한 INSERT

INSERT INTO EMP_01 VALUES(100, '홍길동', '인사관리부');

INSERT INTO EMP_01 (EMP_ID,EMP_NAME, DEPT_TITLE)
VALUES(101, '고길동', '마케팅부');

INSERT INTO EMP_01 (DEPT_TITLE,EMP_ID,EMP_NAME)
VALUES('총무부',102, '홍길똥');

	EMP_ID	EMP_N...	DEPT_TITLE
1	100	홍길동	인사관리부
2	101	고길동	마케팅부
3	102	홍길똥	총무부

INSERT

INSERT

- INSERT시에 VALUES 대신 서브쿼리 사용 가능

```
INSERT INTO EMP_01(
    SELECT EMP_ID,
           EMP_NAME,
           DEPT_TITLE
    FROM EMPLOYEE
    LEFT JOIN DEPARTMENT
    ON (DEPT_CODE=DEPT_ID)
);
```

	EMP_ID	EMP_N...	DEPT_TITLE
1	100	홍길동	인사관리부
2	101	고길동	마케팅부
3	102	홍길녕	총무부
4	217	전지연	인사관리부
5	216	차태연	인사관리부
6	214	방명수	인사관리부
7	221	유하진	회계관리부
8	220	이중석	회계관리부
9	219	임시환	회계관리부
10	215	대북혼	해외영업1부
11	210	윤은해	해외영업1부
12	209	심봉선	해외영업1부
13	208	김해술	해외영업1부
14	207	하이유	해외영업1부
15	206	박나라	해외영업1부
16	205	정중하	해외영업2부
17	204	유재식	해외영업2부
18	203	송은희	해외영업2부
19	222	이태림	기술지원부
20	212	장쯔위	기술지원부
21	211	전형돈	기술지원부
22	202	노옹철	총무부
23	201	송송기	총무부
24	200	선농일	총무부
25	218	이오리	(null)
26	213	하농운	(null)

INSERT

INSERT ALL

- INSERT시 사용하는 서브쿼리의 테이블이 동일한 경우, 2개이상의 테이블에 INSERT ALL을 이용하여 한번에 삽입이 가능
- 단, 서브쿼리의 조건절이 같아야 한다.

테스트를 위한 테이블 생성

```
CREATE TABLE EMP_02
AS
SELECT EMP_ID,
       EMP_NAME,
       DEPT_CODE
FROM EMPLOYEE
WHERE 1 = 0;
```

```
CREATE TABLE EMP_03
AS
SELECT EMP_ID,
       EMP_NAME,
       JOB_CODE
FROM EMPLOYEE
WHERE 1 = 0;
```

※ 서브쿼리를 통한 테이블 생성 시 조건절을 FALSE로 만들면 구조는 복사되고 데이터는 복사되지 않음

INSERT

INSERT ALL

```
INSERT ALL
INTO EMP_02 VALUES(EMP_ID,EMP_NAME,DEPT_CODE)
INTO EMP_03 VALUES(EMP_ID,EMP_NAME,JOB_CODE)
SELECT EMP_ID,EMP_NAME,DEPT_CODE,JOB_CODE
FROM EMPLOYEE
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE
1	200	선동일	D9
2	201	송송기	D9
3	202	노옹절	D9
4	204	유재식	D6
5	205	정중하	D6
6	209	심봉선	D5
7	215	대북혼	D5
8	217	전지연	D1

	EMP_ID	EMP_NAME	JOB_CODE
1	200	선동일	J1
2	201	송송기	J2
3	202	노옹절	J2
4	204	유재식	J3
5	205	정중하	J3
6	209	심봉선	J3
7	215	대북혼	J5
8	217	전지연	J6

INSERT

INSERT ALL – 활용 예제

[EMPLOYEE 테이블의 입사일 기준으로 2000년 1월 1일 이전에 입사한 사원의 사번, 이름, 입사일, 급여를 조회해서 EMP_OLD 테이블에 삽입하고, 그 이후에 입사한 사원의 정보는 EMP_NEW에 삽입]

테스트를 위한 테이블 생성

```
CREATE TABLE EMP_OLD
AS
SELECT EMP_ID,
       EMP_NAME,
       HIRE_DATE,
       SALARY
FROM EMPLOYEE
WHERE 1 = 0;
```

```
CREATE TABLE EMP_NEW
AS
SELECT EMP_ID,
       EMP_NAME,
       HIRE_DATE,
       SALARY
FROM EMPLOYEE
WHERE 1 = 0;
```

INSERT

INSERT ALL

```
INSERT ALL
WHEN HIRE_DATE <'2000/01/01' THEN
INTO EMP_OLD VALUES(EMP_ID,EMP_NAME,HIRE_DATE,SALARY)
WHEN HIRE_DATE >='2000/01/01' THEN
INTO EMP_NEW VALUES(EMP_ID,EMP_NAME,HIRE_DATE,SALARY)
SELECT EMP_ID,EMP_NAME,HIRE_DATE,SALARY
FROM EMPLOYEE;
```

	EMP_ID	EMP_NAME	HIRE_DATE	SALARY
1	200	선동일	90/02/06	8000000
2	203	송은희	96/05/03	2800000
3	205	정중하	99/09/09	3900000
4	207	하이유	94/07/07	2200000
5	213	하농운	99/12/31	2320000
6	219	임시환	99/09/09	1550000
7	221	유하진	94/01/20	2480000
8	222	이태림	97/09/12	2436240

	EMP_ID	EMP_NAME	HIRE_DATE	SALARY
1	201	송송기	01/09/01	6000000
2	202	노옹철	01/01/01	3700000
3	204	유재식	00/12/29	3400000
4	206	박나라	08/04/02	1800000
5	208	김해술	04/04/30	2500000
6	209	심봉선	11/11/11	3500000
7	210	윤은해	01/02/03	2000000
8	211	전형돈	12/12/12	2000000
9	212	장쯔위	15/06/17	2550000
10	214	방명수	10/04/04	1380000
11	215	대북혼	17/06/19	3760000
12	216	차태연	13/03/01	2780000
13	217	전지연	07/03/20	3660000
14	218	이오리	16/11/28	2890000
15	220	이승석	14/09/18	2490000

UPDATE

UPDATE

UPDATE

- 테이블에 기록된 컬럼의 값을 수정하는 구문
- 테이블의 전체 행 개수 변화 없음

표현식

UPDATE 테이블명 SET 컬럼명1 = 변경값1, 컬럼명2 = 변경값2 WHERE 조건식
- 조건식을 적지 않는 경우 테이블 전체 데이터 일괄 변경이므로 주의할 것

테스트를 위한 테이블 생성

```
CREATE TABLE DEPT_COPY
AS
SELECT * FROM DEPARTMENT;
```

	DEPT_ID	DEPT_TITLE	LOCATION_ID
1	D1	인사관리부	L1
2	D2	회계관리부	L1
3	D3	마케팅부	L1
4	D4	국내영업부	L1
5	D5	해외영업1부	L2
6	D6	해외영업2부	L3
7	D7	해외영업3부	L4
8	D8	기술지원부	L5
9	D9	총무부	L1

UPDATE

UPDATE

[총무부의 부서이름을 전략기획본부 로 변경]

```
UPDATE DEPT_COPY SET DEPT_TITLE='전략기획본부'
WHERE DEPT_ID = 'D9';
```

	DEPT_ID	DEPT_TITLE	LOCATION_ID
1	D1	인사관리부	L1
2	D2	회계관리부	L1
3	D3	마케팅부	L1
4	D4	국내영업부	L1
5	D5	해외영업1부	L2
6	D6	해외영업2부	L3
7	D7	해외영업3부	L4
8	D8	기술지원부	L5
9	D9	총무부	L1

	DEPT_ID	DEPT_TITLE	LOCATION_ID
1	D1	인사관리부	L1
2	D2	회계관리부	L1
3	D3	마케팅부	L1
4	D4	국내영업부	L1
5	D5	해외영업1부	L2
6	D6	해외영업2부	L3
7	D7	해외영업3부	L4
8	D8	기술지원부	L5
9	D9	전략기획본부	L1

UPDATE

UPDATE

- 업데이트시에도 서브쿼리 사용이 가능

테스트를 위한 테이블 생성

```
CREATE TABLE EMP_SALARY
AS
SELECT EMP_ID,
       EMP_NAME,
       SALARY,
       BONUS FROM
EMPLOYEE;
```

	EMP_ID	EMP_NAME	SALARY	BONUS
1	200	선동일	8000000	0.3
2	201	송송기	6000000	(null)
3	202	노옹철	3700000	(null)
4	203	송은희	2800000	(null)
5	204	유재식	3400000	0.2
6	205	정중하	3900000	(null)
7	206	박나라	1800000	(null)
8	207	하이유	2200000	0.1
9	208	김해술	2500000	(null)
10	209	심봉선	3500000	0.15
11	210	윤은해	2000000	(null)
12	211	전형돈	2000000	(null)
13	212	장쯔위	2550000	0.25
14	213	하동운	2320000	0.1
15	214	방명수	1380000	(null)
16	215	대북혼	3760000	(null)
17	216	차태연	2780000	0.2
18	217	전지연	3660000	0.3
19	218	이오리	2890000	(null)
20	219	임시환	1550000	(null)
21	220	이중석	2490000	(null)
22	221	유하진	2480000	(null)
23	222	이태림	2436240	0.35

UPDATE

UPDATE

[방명수 사원의 급여와 보너스를 유재식사원과 동일하게 변경]

```
UPDATE EMP_SALARY SET
SALARY =
(SELECT SALARY FROM EMP_SALARY WHERE EMP_NAME='유재식'),
BONUS =
(SELECT BONUS FROM EMP_SALARY WHERE EMP_NAME='유재식')
WHERE EMP_NAME ='방명수';
```

	EMP_ID	EMP_NAME	SALARY	BONUS
1	204	유재식	3400000	0.2
2	214	방명수	1380000	(null)

	E...	EMP_NAME	SALARY	BONUS
1	204	유재식	3400000	0.2
2	214	방명수	3400000	0.2

DELETE

DELETE

DELETE

- 테이블의 행을 삭제하는 구문
- 테이블의 전체 행 개수 감소

표현식

DELETE FROM 테이블명 WHERE 조건식
- 조건식을 적지 않는 경우 테이블 전체 데이터가 삭제되므로 주의할 것

DELETE FROM DEPT_COPY WHERE DEPT_ID = 'D9';

	DEPT_ID	DEPT_TITLE	LOCATION_ID
1	D1	인사관리부	L1
2	D2	회계관리부	L1
3	D3	마케팅부	L1
4	D4	국내영업부	L1
5	D5	해외영업1부	L2
6	D6	해외영업2부	L3
7	D7	해외영업3부	L4
8	D8	기술지원부	L5
9	D9	전략기획본부	L1

	DEPT_ID	DEPT_TITLE	LOCATION_ID
1	D1	인사관리부	L1
2	D2	회계관리부	L1
3	D3	마케팅부	L1
4	D4	국내영업부	L1
5	D5	해외영업1부	L2
6	D6	해외영업2부	L3
7	D7	해외영업3부	L4
8	D8	기술지원부	L5

DELETE

DELETE

- FOREIGN KEY 제약조건이 설정되어 있는 경우 삭제 불가능
  (ON DELETE RESTRICTED 인 경우)
- 제약조건을 비활성화 후 삭제 가능

테스트를 위한 테이블 생성

```
CREATE TABLE TEST01(
ID VARCHAR2(20) PRIMARY KEY,
PW VARCHAR2(20) NOT NULL
);
```

```
CREATE TABLE TEST02(
ID VARCHAR2(20),
TESTDATE DATE,
FOREIGN KEY (ID) REFERENCES
TEST01 (ID)
);
```

DELETE

DELETE

테스트 데이터 입력

```
INSERT INTO TEST01 VALUES (‘user01’, ‘pass01’);
INSERT INTO TEST01 VALUES (‘user02’, ‘pass02’);
INSERT INTO TEST02 VALUES (‘user01’, SYSDATE);
INSERT INTO TEST02 VALUES (‘user02’, SYSDATE);
```

```
DELETE FROM TEST01 WHERE ID = ‘user02’;
```

→ 외래키 제약조건으로 삭제 불가능

	ID	PW
1	user01	pass01
2	user02	pass02

	ID	TESTDATE
1	user01	17/09/24
2	user02	17/09/24

DELETE

DELETE

제약조건으로 인한 삭제 불가 시 강제 삭제

1. 제약조건 비활성화
   ALTER TABLE TEST02
   DISABLE CONSTRAINT SYS_C008161 CASCADE;
2. 데이터 삭제
   DELETE FROM TEST01 WHERE ID='user02';
3. 비활성화 된 제약조건 활성화(다시 제약조건 활성화 하는 경우 TEST02 테이블에서 user02 데이터를 처리해야함 삭제 또는 null로 변경)
   ALTER TABLE TEST02
   ENABLE CONSTRAINT SYS_C008161;

	ID	PW
1	user01	pass01

	ID	TESTDATE
1	user01	17/09/24
2	user02	17/09/24