

DML

(Data Manipulation Language)

DML(Data Manipulation Language)

DML이란?

데이터 조작 언어이다. 테이블에 값을 삽입(INSERT), 수정(UPDATE), 삭제(DELETE)를 한다.

DML(Data Manipulation Language)

INSERT

새로운 행을 테이블에 추가하는 구문이다. 테이블의 행 개수가 증가한다.

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, EMP_NO, EMAIL, PHONE, DEPT_CODE, JOB_CODE, SAL_LEVEL, SALARY, BONUS, MANAGER_ID, HIRE_DATE, ENT_DATE, ENT_YN)  
VALUES(900, '장채현', '901123-1080503', 'jang_ch@kh.or.kr', '01055569512', 'D1', 'J8', 'S3', 4300000, 0.2, '200', SYSDATE, NULL, DEFAULT);
```

또는

```
INSERT INTO EMPLOYEE  
VALUES(900, '장채현', '901123-1080503', 'jang_ch@kh.or.kr', '01055569512', 'D1', 'J8', 'S3', 4300000, 0.2, '200', SYSDATE, NULL, DEFAULT);
```

** INSERT하고자 하는 컬럼이 모든 컬럼인 경우 컬럼명 생략이 가능하다.
단, 컬럼의 순서를 지켜서 VALUES에 값을 기입해야 한다.

```
SELECT * FROM EMPLOYEE  
WHERE EMP_NAME = '장채현';
```

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	DEPT_CODE	JOB_CODE	SAL_LEVEL	SALARY	BONUS	MANAGER_ID	HIRE_DATE	ENT_DATE	ENT_YN
1 900	장채현	901123-1080503	jang_ch@kh.or.kr	01055569512	D1	J8	S3	4300000	0.2 200		17/09/19	(null)	N

DML(Data Manipulation Language)

INSERT

INSERT시에 VALUES 대신 서브쿼리를 이용할 수 있다.

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

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

```
SELECT * FROM EMPLOYEE;
```

SQL | 인출된 모든 행: 24(0초)

	EMP_ID	EMP_NAME	DEPT_TITLE
1	900	장채현	인사관리부
2	217	전지연	인사관리부
3	216	차태연	인사관리부
4	214	방명수	인사관리부
5	221	유하진	회계관리부
6	220	이종석	회계관리부
7	219	임시환	회계관리부
8	215	대복훈	해외영업1부
9	210	윤은해	해외영업1부
10	209	심봉선	해외영업1부
11	208	김해슬	해외영업1부
12	207	하미유	해외영업1부
13	206	박나라	해외영업1부
14	205	정중하	해외영업2부
15	204	유재식	해외영업2부
16	203	송은희	해외영업2부
17	222	이태림	기술지원부
18	212	장프위	기술지원부
19	211	전형돈	기술지원부
20	202	노용철	총무부
21	201	송종기	총무부
22	200	선동일	총무부
23	218	이오리	(null)
24	213	하동운	(null)

DML(Data Manipulation Language)

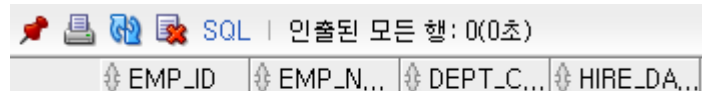
INSERT ALL

INSERT시 사용하는 서브쿼리가 사용하는 테이블이 같은 경우, 두 개 이상의 테이블에 INSERT ALL을 이용하여 한번에 삽입을 할 수 있다.

단, 각 서브쿼리의 조건절이 같아야 한다.

```
CREATE TABLE EMP_DEPT_D1
AS SELECT EMP_ID,
        EMP_NAME,
        DEPT_CODE,
        HIRE_DATE
FROM EMPLOYEE
WHERE 1 = 0;
```

```
SELECT * FROM EMP_DEPT_D1;
```

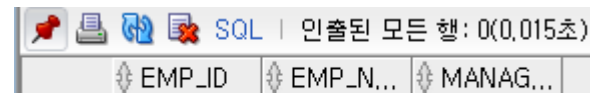


SQL | 인출된 모든 행: 0(0초)

EMP_ID	EMP_N...	DEPT_C...	HIRE_DA...
--------	----------	-----------	------------

```
CREATE TABLE EMP_MANAGER
AS SELECT EMP_ID,
        EMP_NAME,
        MANAGER_ID
FROM EMPLOYEE
WHERE 1 = 0;
```

```
SELECT * FROM EMP_MANAGER;
```



SQL | 인출된 모든 행: 0(0,015초)

EMP_ID	EMP_N...	MANAG...
--------	----------	----------

DML(Data Manipulation Language)

INSERT ALL

[EMP_DEPT_D1 테이블에 EMPLOYEE테이블의 부서 코드가 D2인 직원을 조회해
사번, 이름, 소속부서, 입사일을 삽입하고,
EMP_MANAGER 테이블에 EMPLOYEE 테이블의 부서 코드가 D2인 직원의
사번, 이름, 관리자사번을 조회해서 삽입하세요]

```
INSERT ALL  
  INTO EMP_DEPT_D1 VALUES(EMP_ID, EMP_NAME, DEPT_CODE, HIRE_DATE)  
  INTO EMP_MANAGER VALUES(EMP_ID, EMP_NAME, DEPT_CODE, HIRE_DATE)  
  SELECT EMP_ID, EMP_NAME, DEPT_CODE, HIRE_DATE, MANAGER_ID  
  FROM EMPLOYEE  
  WHERE DEPT_CODE = 'D1';
```

| 8개 행 이 (가) 삽입되었습니다.

SQL | 인출된 모든 행: 4(0초)

	EMP_ID	EMP_NAME	DEPT_CODE	HIRE_DATE
1	214	방명수	D1	10/04/04
2	216	차태연	D1	13/03/01
3	217	전지연	D1	07/03/20
4	900	장채현	D1	17/09/19

SQL | 인출된 모든 행: 4(0초)

	EMP_ID	EMP_NAME	DEPT_CODE	MANAGER_ID
1	214	방명수	D1	200
2	216	차태연	D1	214
3	217	전지연	D1	214
4	900	장채현	D1	200


DML(Data Manipulation Language)

INSERT ALL

[EMPLOYEE 테이블의 구조를 복사하여 사번, 이름, 입사일, 급여를 기록할 수 있는 테이블 EMP_OLD와 EMP_NEW를 생성하세요.]


```
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;
```



SQL | 인출된 모든 행: 0(0초)

EMP_ID	EMP_N...	HIRE_DA...	SALARY
--------	----------	------------	--------



SQL | 인출된 모든 행: 0(0초)

EMP_ID	EMP_N...	HIRE_DA...	SALARY
--------	----------	------------	--------




DML(Data Manipulation Language)

INSERT ALL

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




```
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;
```

```
SELECT * FROM EMP_OLD;
```

   SQL | 인출된 모든 행: 8(0초)

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	하미유	94/07/07	2200000
5 213	하동운	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

```
SELECT * FROM EMP_NEW;
```

   SQL | 인출된 모든 행: 16(0초)

EMP_ID	EMP_NAME	HIRE_DATE	SALARY
1 201	송종기	01/09/01	6000000
2 202	노용철	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
15 220	이종석	14/09/18	2490000
16 900	장채현	17/09/19	4300000

DML(Data Manipulation Language)

MERGE

구조가 같은 두 개의 테이블을 하나의 테이블로 합치는 기능을 한다. 두 테이블에서 지정하는 조건의 값이 존재하면 UPDATE되고, 조건의 값이 없으면 INSERT 되도록 한다.

```
CREATE TABLE EMP_M01  
AS SELECT *  
FROM EMPLOYEE;
```

```
CREATE TABLE EMP_M02  
AS SELECT *  
FROM EMPLOYEE  
WHERE JOB_CODE = 'J4';
```

```
INSERT INTO EMP_M02  
VALUES (999, '곽두원', '561016-1234567', 'kwack_dw@kh.or.kr', '01011112222', 'D9', 'J1', 'S1',  
9000000, 0.5, NULL, SYSDATE, DEFAULT, DEFAULT);
```

```
UPDATE EMP_M02 SET SALARY = 0;
```

MERGE

```
MERGE INTO EMP_M01 USING EMP_M02 ON(EMP_M01.EMP_ID = EMP_M02.EMP_ID)
WHEN MATCHED THEN
UPDATE SET
EMP_M01.EMP_NAME = EMP_M02.EMP_NAME,
EMP_M01.EMP_NO = EMP_M02.EMP_NO,
EMP_M01.EMAIL = EMP_M02.EMAIL,
EMP_M01.PHONE = EMP_M02.PHONE,
EMP_M01.DEPT_CODE = EMP_M02.DEPT_CODE,
EMP_M01.JOB_CODE = EMP_M02.JOB_CODE,
EMP_M01.SAL_LEVEL = EMP_M02.SAL_LEVEL,
EMP_M01.SALARY = EMP_M02.SALARY,
EMP_M01.BONUS = EMP_M02.BONUS,
EMP_M01.MANAGER_ID = EMP_M02.MANAGER_ID,
EMP_M01.HIRE_DATE = EMP_M02.HIRE_DATE,
EMP_M01.ENT_DATE = EMP_M02.ENT_DATE,
EMP_M01.ENT_YN = EMP_M02.ENT_YN
WHEN NOT MATCHED THEN
INSERT VALUES(EMP_M02.EMP_ID, EMP_M02.EMP_NAME, EMP_M02.EMP_NO, EMP_M02.EMAIL,
EMP_M02.PHONE, EMP_M02.DEPT_CODE, EMP_M02.JOB_CODE, EMP_M02.SAL_LEVEL, EMP_M02.SALARY,
EMP_M02.BONUS, EMP_M02.MANAGER_ID, EMP_M02.HIRE_DATE, EMP_M02.ENT_DATE, EMP_M02.ENT_YN);
```

DML(Data Manipulation Language)

MERGE

```
SELECT * FROM EMP_M01;
```

SQL | 인출된 모든 행: 25(0초)

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	DEPT_CODE	JOB_CODE	SAL_LEVEL	SALARY	BONUS	MANAGER_ID	HIRE_DATE	ENT_DATE	ENT_YN
1 200	선동일	621235-1985634	sun_di@kh.or.kr	01099546325	D9	J1	S1	8000000	0.3 (null)		90/02/06	(null)	N
2 201	송종기	631156-1548654	song_jk@kh.or.kr	01045686656	D9	J2	S1	6000000	(null) 200	200	01/09/01	(null)	N
3 202	노용철	861015-1356452	no_hc@kh.or.kr	01066656263	D9	J2	S4	3700000	(null) 201	201	01/01/01	(null)	N
4 203	송은희	631010-2653546	song_eh@kh.or.kr	01077607879	D6	J4	S5	0	(null) 204	204	96/05/03	(null)	N
5 204	유재식	660508-1342154	yoo_js@kh.or.kr	01099999129	D6	J3	S4	3400000	0.2 200		00/12/29	(null)	N
6 205	정중하	770102-1357951	jung_jh@kh.or.kr	01036654875	D6	J3	S4	3900000	(null) 204	204	99/09/09	(null)	N
7 206	박나라	630709-2054321	pack_nr@kh.or.kr	01096935222	D5	J7	S6	1800000	(null) 207	207	08/04/02	(null)	N
8 207	하미유	690402-2040612	ha_iy@kh.or.kr	01036654488	D5	J5	S5	2200000	0.1 200	200	94/07/07	(null)	N
9 208	김해술	870927-1313564	kim_hs@kh.or.kr	01078634444	D5	J5	S5	2500000	(null) 207	207	04/04/30	(null)	N
10 209	심봉선	750206-1325546	sim_bs@kh.or.kr	0113654485	D5	J3	S4	3500000	0.15 207	207	11/11/11	(null)	N
11 210	윤은해	650505-2356985	youn_eh@kh.or.kr	0179964233	D5	J7	S5	2000000	(null) 207	207	01/02/03	(null)	N
12 211	전형도	830807-1121321	jun_hd@kh.or.kr	01044432222	D8	J6	S5	2000000	(null) 200	200	12/12/12	(null)	N
13 212	장프위	780923-2234542	jang_zw@kh.or.kr	01066682224	D8	J6	S5	2550000	0.25 211	211	15/06/17	(null)	N
14 213	하동운	621111-1785463	ha_dh@kh.or.kr	01158456632	(null)	J6	S5	2320000	0.1 (null)	(null)	99/12/31	(null)	N
15 214	방명수	856795-1313513	bang_ms@kh.or.kr	01074127545	D1	J7	S6	1380000	(null) 200	200	10/04/04	(null)	N
16 215	대복존	881130-1050911	dae_bh@kh.or.kr	01088808584	D5	J5	S4	3760000	(null) (null)	(null)	17/06/19	(null)	N
17 216	차태연	770808-1364897	cha_ty@kh.or.kr	01064643212	D1	J6	S5	2780000	0.2 214	214	13/03/01	(null)	N
18 217	전지연	770808-2665412	jun_jy@kh.or.kr	01033624442	D1	J6	S4	3660000	0.3 214	214	07/03/20	(null)	N
19 218	미오리	870427-2232123	loo_or@kh.or.kr	01022306545	(null)	J7	S5	2890000	(null) (null)	(null)	16/11/28	(null)	N
20 219	임시환	660712-1212123	im_sw@kh.or.kr	(null)	D2	J4	S6	0	(null) (null)	(null)	99/09/09	(null)	N
21 220	이종석	770823-1113111	lee_js@kh.or.kr	(null)	D2	J4	S5	0	(null) (null)	(null)	14/09/18	(null)	N
22 221	유하진	800808-1123341	yoo_hj@kh.or.kr	(null)	D2	J4	S5	0	(null) (null)	(null)	94/01/20	(null)	N
23 222	이태림	760918-2854697	lee_tr@kh.or.kr	01033000002	D8	J6	S5	2436240	0.35 100	100	97/09/12	17/09/12	Y
24 900	장채현	901123-1080503	jang_ch@kh.or.kr	01055569512	D1	J8	S3	4300000	0.2 200	200	17/09/19	(null)	N
25 999	곽두원	561016-1234567	kwack_dw@kh.or.kr	01011112222	D9	J1	S1	0	0.5 (null)	(null)	17/09/20	(null)	(null)

DML(Data Manipulation Language)

UPDATE

테이블에 기록된 컬럼의 값을 수정하는 구문이다. 테이블의 전체 행 개수에는 변화가 없다.

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

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

1 행 이 (가) 업데이트되었습니다.

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	전략기획팀	L1

** WHERE 조건을 설정하지 않으면 모든 행의 컬럼 값이 변경된다.

DML(Data Manipulation Language)

UPDATE

UPDATE시에도 서브쿼리를 이용할 수 있다.

[평상시 유재식 사원을 부러워하던 방명수 사원의 급여와 보너스율을 유재식 사원과 동일하게 변경해 주기로 했다. 이를 반영하는 UPDATE문을 작성하세요.]

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

```
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';
```

```
SELECT * FROM EMP_SALARY  
WHERE EMP_NAME IN ('유재식', '방명수');
```

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

	EMP_ID	EMP_NAME	SALARY	BONUS
1	204	유재식	3400000	0.2
2	214	방명수	3400000	0.2

DML(Data Manipulation Language)

UPDATE

[각각 쿼리문 작성한 것을 다중행 다중열 서브쿼리로 변경하세요]

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

```
SELECT * FROM EMP_SALARY  
WHERE EMP_NAME IN ('유재식', '방명수');
```

	EMP_ID	EMP_NAME	SALARY	BONUS
1	204	유재식	3400000	0.2
2	214	방명수	3400000	0.2

DML(Data Manipulation Language)

UPDATE

[EMP_SALARY 테이블에서 아시아 지역에 근무하는 직원의 보너스
포인트를 0.3으로 변경하세요]

```
UPDATE EMP_SALARY
SET BONUS = 0.3
WHERE EMP_ID IN (SELECT EMP_ID
                  FROM EMPLOYEE
                  JOIN DEPARTMENT ON(DEPT_ID = DEPT_CODE)
                  JOIN LOCATION ON(LOCATION_ID = LOCAL_CODE)
                  WHERE LOCAL_NAME LIKE 'ASIA%');
```

19개 행 이 (가) 업데이트되었습니다.

EMP_ID	EMP_NAME	SALARY	BONUS
1 200	선동일	8000000	0.3
2 201	송종기	6000000	(null)
3 202	노홍철	3700000	(null)
4 203	송은희	2800000	(null)
5 204	유재식	3400000	0.2
6 205	정종하	3900000	(null)
7 206	박나라	1800000	(null)
8 207	하미유	2200000	0.1
9 208	김해솔	2500000	(null)
10 209	심봉선	3500000	0.15
11 210	윤은혜	2000000	(null)
12 211	전철돈	2000000	(null)
13 212	장프위	2550000	0.25
14 213	하동운	2320000	0.1
15 214	방영수	1380000	(null)
16 215	대복존	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	이종석	2490000	(null)
22 221	유하진	2480000	(null)
23 222	이태림	2436240	0.35
24 900	장채현	4300000	0.2



EMP_ID	EMP_NAME	SALARY	BONUS
1 200	선동일	8000000	0.3
2 201	송종기	6000000	0.3
3 202	노홍철	3700000	0.3
4 203	송은희	2800000	0.3
5 204	유재식	3400000	0.3
6 205	정종하	3900000	0.3
7 206	박나라	1800000	0.3
8 207	하미유	2200000	0.3
9 208	김해솔	2500000	0.3
10 209	심봉선	3500000	0.3
11 210	윤은혜	2000000	0.3
12 211	전철돈	2000000	(null)
13 212	장프위	2550000	0.25
14 213	하동운	2320000	0.1
15 214	방영수	1380000	0.3
16 215	대복존	3760000	0.3
17 216	차태연	2780000	0.3
18 217	전지연	3660000	0.3
19 218	이오리	2890000	(null)
20 219	임시환	1550000	0.3
21 220	이종석	2490000	0.3
22 221	유하진	2480000	0.3
23 222	이태림	2436240	0.35
24 900	장채현	4300000	0.3

DML(Data Manipulation Language)

DELETE

테이블의 행을 삭제하는 구문이다. 테이블의 행 개수가 줄어든다.

```
DELETE FROM EMPLOYEE  
WHERE EMP_NAME = '장채현';
```

| 1 행 이 (가) 삭제되었습니다.

** WHERE 조건을 설정하지 않으면 모든 행이 삭제된다.

```
DELETE FROM DEPARTMENT  
WHERE DEPT_ID = 'D1';
```

오류 보고 -

SQL 오류: ORA-02292: integrity constraint (EMPLOYEE.EMP_DEPTCODE_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"

*Cause: attempted to delete a parent key value that had a foreign dependency.

*Action: delete dependencies first then parent or disable constraint.

** FOREIGN KEY 제약조건이 설정되어 있는 경우 참조되고 있는 값에 대해서는 삭제 할 수 없다.

```
DELETE FROM DEPARTMENT  
WHERE DEPT_ID = 'D3';
```

| 1 행 이 (가) 삭제되었습니다.

** FOREIGN KEY 제약조건이 설정되어 있는 경우 참조되고 있는 값에 대해서는 삭제 할 수 없다.

DML(Data Manipulation Language)

DELETE

삭제 시 FOREIGN KEY 제약조건으로 컬럼 삭제가 불가능 한 경우, 제약조건을 비활성화 할 수 있다.

```
DELETE FROM DEPARTMENT  
WHERE DEPT_ID = 'D1';
```

오류 보고 -

SQL 오류: ORA-02292: integrity constraint (EMPLOYEE.EMP_DEPTCODE_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"

*Cause: attempted to delete a parent key value that had a foreign
dependency.

*Action: delete dependencies first then parent or disable constraint.

```
ALTER TABLE EMPLOYEE  
DISABLE CONSTRAINT EMP_DEPTCODE_FK CASCADE;
```

```
DELETE FROM DEPARTMENT  
WHERE DEPT_ID = 'D1';
```

| 1 행 미(가) 삭제되었습니다.

** 비활성화 된 제약조건을 다시 활성화 시킬 수 있다.

```
ALTER TABLE EMPLOYEE  
ENABLE CONSTRAINT EMP_DEPTCODE_FK;
```

DML(Data Manipulation Language)

TRUNCATE

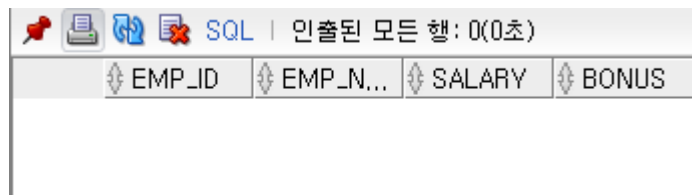
테이블의 전체 행을 삭제 시 사용한다. DELETE문 보다 수행 속도가 빠르지만, ROLLBACK을 통해 복구를 할 수 없다.

또한, DELETE와 마찬가지로 FOREIGN KEY 제약조건일 때는 적용 불가능하기 때문에 제약조건을 비활성화 해야 삭제를 할 수 있다.

```
TRUNCATE TABLE EMP_SALARY;
```

```
SELECT * FROM EMP_SALARY;
```

Table EMP_SALARY이 (가) 잘렸습니다.



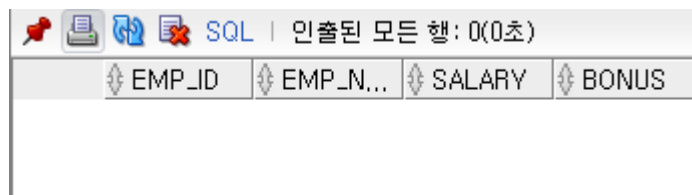
The screenshot shows a SQL query result window. The title bar indicates 'SQL | 인출된 모든 행: 0(0초)'. The table structure is displayed with columns: EMP_ID, EMP_N..., SALARY, and BONUS. The table is currently empty.

EMP_ID	EMP_N...	SALARY	BONUS
--------	----------	--------	-------

** 모든 컬럼이 삭제 되기는 하지만, 테이블의 구조는 남아 있다.

** ROLLBACK 수행 후에도 컬럼이 복구되지 않는다.

```
ROLLBACK;
```



The screenshot shows a SQL query result window. The title bar indicates 'SQL | 인출된 모든 행: 0(0초)'. The table structure is displayed with columns: EMP_ID, EMP_N..., SALARY, and BONUS. The table is currently empty.

EMP_ID	EMP_N...	SALARY	BONUS
--------	----------	--------	-------