

# DML

INSERT  
UPDATE  
DELETE  
MARGE

## □ SQL 명령어들

- **DML** (Data Manipulation Language) : INSERT(입력) , UPDATE(변경) ,  
DELETE(삭제) , MERGE(병합)
- **DDL** (Data Definition Language) : CREATE (생성) , ALTER (수정) ,  
TRUNCATE (잘라내기) , DROP (삭제)
- \* **DCL** (Data Control Language) : GRANT (권한 주기) , REVOKE (권한 뺏기)
- \* **TCL** (Transaction Control Language): COMMIT (확정) , ROLLBACK (취소)
- \* **SELECT** : 어떤 분류에서는 DQL (Data Query Language) 라고 하기도 합니다.

# 복습

- 테이블명 : panmae
- 컬럼명
  - ▣ no : number 1바이트, 기본키 설정
  - ▣ pcode : char 4바이트, 중복 불가
  - ▣ pdate : char 8바이트 기본값 시스템 데이터
  - ▣ pqty : number 3바이트
  - ▣ area : number 1, chekck : 1~4

# 1. INSERT

## □ INSERT : 데이터 입력 명령어

### 1) INSERT 를 사용하여 단일 행 입력하기

```
INSERT INTO table [(column1, column2,.....)]  
VALUES (value 1 , value 2,....) ;
```

#### - 사용 예 1:

Dept2 테이블에 아래와 같은 내용으로 새로운 부서 정보를 입력하세요.

- \* 부서번호 : 9000
- \* 부서명 : 특판1팀
- \* 상위부서 : 영업부
- \* 지역 : 임시지역

```
INSERT INTO dept2(dcode , dname , pdept ,area )  
VALUES (9000 , '특판1팀','영업부','임시지역') ;
```

```
INSERT INTO dept2  
VALUES(9001 , '특판2팀','영업부','임시지역') ;
```

# 1. INSERT

## - 사용 예 2: 특정 칼럼만 입력하기

부서번호와 부서명, 상위부서 값만 아래의 값으로 입력하세요.

- \* 부서번호 : 9002
- \* 부서명 : 특판3팀
- \* 상위부서 : 영업부

```
INSERT INTO dept2(dcode,dname,pdept)  
VALUES(9002, '특판3팀' , '영업부') ;
```

# 1. INSERT

## - 사용 예 3: 날짜 데이터 입력하기

아래 정보를 professor 테이블에 입력하세요.

- \* 교수번호 : 5001
- \* 교수이름 : 김설희
- \* ID : Love\_me
- \* POSITION : 정교수
- \* PAY : 510
- \* 입사일 : 2011년 11월 14일 <- 이 부분을 주의 깊게 보세요.

```
INSERT INTO professor (profno , name , id , position , pay , hiredate)
VALUES (5001,'김설희','Love_me','정교수',510,'2011-11-14');
```

- 윈도우 용과 유닉스 용은 날짜 포맷이 다르므로 주의해야 함.

# 1. INSERT

## - 사용 예 4: Null 값 입력하기

### \* 자동 NULL 값 입력하기

데이터를 입력할 때 칼럼에 값을 안 주면 자동으로 NULL 값이 들어 감

### \* 수동 NULL 값 입력하기

데이터부분에 NULL 값을 적어주면 됨.

# 1. INSERT

## 2) INSERT 를 사용하여 여러 행 입력하기

```
CREATE TABLE professor2  
AS SELECT * FROM professor ;
```

실습을 위해 professor2 테이블을 생성합니다.

```
INSERT INTO professor2  
SELECT * FROM professor ;
```

이 방식은 이미 생성되어 있는 테이블에서 대량의 데이터를 복사 해 올 때 아주 많이 사용하는 방법입니다. ITAS 라고 부르기도 합니다.



# 1. INSERT

## - 사용 예 2 : 다른 테이블의 데이터를 가져와서 입력하기

Professor 테이블에서 교수번호가 1000 번 에서 1999번까지 인 교수의 번호와 교수이름은 p\_01 테이블에 입력하고 교수번호가 2000 번에서 2999 번까지 인 교수의 번호와 이름은 p\_02 테이블에 입력하세요.

### INSERT ALL

```
2 WHEN profno BETWEEN 1000 AND 1999 THEN
3   INTO p_01 VALUES(profno,name)
4 WHEN profno BETWEEN 2000 AND 2999 THEN
5   INTO p_02 VALUES(profno,name)
6 SELECT profno,name
7 FROM professor ;
```

# 1. INSERT

- 사용 예 3 : 다른 테이블에 동시에 같은 데이터 입력하기

Professor 테이블에서 교수번호가 3000번 에서 3999 번인 교수들의 교수 번호와 이름을 p\_01 테이블과 p\_02 테이블에 동시에 입력하세요.

```
SCOTT>INSERT ALL
2   INTO p_01 VALUES (profno,name)
3   INTO p_02 VALUES (profno,name)
4   SELECT profno,name
5   FROM professor
6   WHERE profno BETWEEN 3000 AND 3999 ;
```

## 2. UPDATE

### □ UPDATE : 데이터 수정하기

**UPDATE** table

**SET** column = value

**WHERE** 조건 ;

#### - 사용 예 1:

**Professor 테이블에서 직급이 조교수 인 교수들의 BONUS 를 100 만원으로 인상하세요.**

```
SCOTT>UPDATE professor
```

```
2 SET bonus = 100
```

```
3 WHERE position ='조교수';
```

## 2. UPDATE

- 사용 예 2:

**Professor 테이블에서 차범철 교수의 직급과 동일한 직급을 가진 교수들 중 현재 급여가 250 만원이 안 되는 교수들의 급여를 15% 인상하세요.**

```
UPDATE professor
2 SET pay = pay * 1.15
3 WHERE position = ( SELECT position
4                     FROM professor
5                     WHERE name = '차범철')
6 AND pay < 250 ;
```

### 3. DELETE

#### □ DELETE : 데이터 삭제하기

```
DELETE FROM table  
WHERE 조건 ;
```

- 사용 예 :

Dept2 테이블에서 부서번호(DCODE)가 9000 번에서 9100 번 사이인 매장들을 삭제하세요.

```
DELETE FROM dept2  
2 WHERE dcode between 9000 and 9100 ;
```

**DELETE 는 데이터는 삭제되나 용량은 변함이 없다는 것 !!!**

## 4. MERGE

### □ MERGE: 테이블 합치기

```
MERGE INTO Table1
  USING Table2
  ON ( 병합 조건절 )
  WHEN MATCHED THEN
  UPDATE SET 업데이트 내용
  DELETE WHERE 조건
  WHEN NOT MATCHED THEN
  INSERT VALUES(컬럼 이름) ;
```

## 4. MERGE

### - Merge 실습

pt\_01

판매번호	제품번호	수량	금액
12010101	1000	1	500
12010102	1001	1	400
12010103	1003	1	300

pt\_02

판매번호	제품번호	수량	금액
12010201	1004	1	600
12010202	1000	1	500
12010203	1005	1	700

P\_total

판매번호	제품번호	수량	금액
12010101	1000	1	500
12010102	1001	1	400
12010103	1003	1	300
12010201	1004	1	600
12010202	1000	1	500
12010203	1005	1	700

## 4. MERGE

### - Merge 전 테이블 내용 확인

```
SCOTT>SELECT * FROM pt_01 ;
```

판매번호	제품번호	수량	금액
12010101	1000	1	500
12010102	1001	1	400
12010103	1003	1	300

```
SCOTT>SELECT * FROM pt_02 ;
```

판매번호	제품번호	수량	금액
12010201	1004	1	600
12010202	1000	1	500
12010203	1005	1	700

```
SCOTT>SELECT * FROM p_total ;
```

no rows selected



## 4. MERGE

### - MERGE 작업 QUERY 1 (pt\_01 테이블과 p\_total 테이블 병합)

```
MERGE INTO p_total total
2 USING pt_01 p01
3 ON (total.판매번호=p01.판매번호)
4 WHEN MATCHED THEN
5   UPDATE SET total.제품번호 = p01.제품번호
6 WHEN NOT MATCHED THEN
7   INSERT VALUES(p01.판매번호 , p01.제품번호 , p01.수량 , p01.금액) ;
```

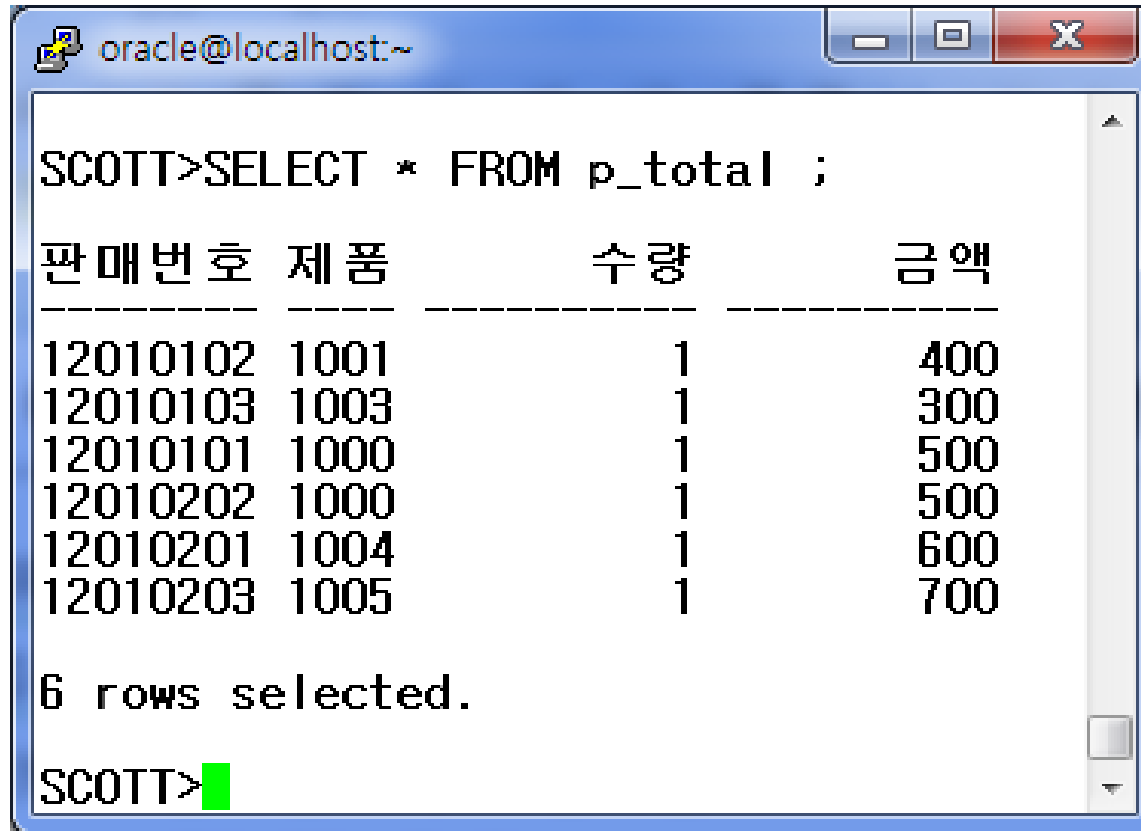
## 4. MERGE

### - MERGE 작업 QUERY 2 (pt\_02 테이블과 p\_total 테이블 병합)

```
MERGE INTO p_total total
2 USING pt_02 p02
3 ON (total.판매번호=p02.판매번호)
4 WHEN MATCHED THEN
5   UPDATE SET total.제품번호 = p02.제품번호
6 WHEN NOT MATCHED THEN
7   INSERT VALUES(p02.판매번호 , p02.제품번호 , p02.수량 , p02.금액) ;
```

## 4. MERGE

- Merge 작업 완료 후 결과 조회하기



The screenshot shows a terminal window titled 'oracle@localhost:~'. The user 'SCOTT' has executed the command 'SELECT \* FROM p\_total ;'. The output is a table with four columns: '판매번호' (Sales Number), '제품' (Product), '수량' (Quantity), and '금액' (Amount). The table contains six rows of data. Below the table, it states '6 rows selected.' and the prompt 'SCOTT>' is visible with a green cursor.

```
SCOTT>SELECT * FROM p_total ;
```

판매번호	제품	수량	금액
12010102	1001	1	400
12010103	1003	1	300
12010101	1000	1	500
12010202	1000	1	500
12010201	1004	1	600
12010203	1005	1	700

6 rows selected.

```
SCOTT>
```