

# Introduction to Oracle9i : SQL

## Chapter6. DML(Data Manipulation Language)

- ▶ DML문의 종류를 기술한다.
- ▶ 유형별 DML문을 작성하고 실행한다.

## 1. DML의 정의

- DML문은 다음과 같은 상황에 실행된다.
  - 테이블에 새로운 행을 입력할 경우 (INSERT)
  - 테이블에 존재하는 행의 내용을 변경할 경우 (UPDATE)
  - 테이블에 존재하는 행을 삭제할 경우 (DELETE)

## 2. DML의 종류

### (1) INSERT

- 테이블에 새로운 행을 입력할 때 사용
- 하나의 INSERT문을 통해서 한 행만 입력 가능

#### 1) INSERT 구문

```
INSERT INTO 테이블명 [(컬럼명, 컬럼명...)]  
VALUES      (값, 값...);
```

## 2. DML의 종류

### (1) INSERT

#### 2) INSERT문 작성

- 문자와 날짜값을 입력할 때는 작은 따옴표로 묶어서 작성

```
INSERT INTO departments (department_id, department_name, manager_id)
VALUES      (120, 'CRM', 100) ;
```

1행이 입력되었습니다.

## 2. DML의 종류

### (1) INSERT

#### 3) NULL값의 INSERT

- 암시적 방법

```
INSERT INTO departments (department_id, department_name )  
VALUES      (130, 'Finance') ;
```

1행이 입력되었습니다.

- 명시적 방법

```
INSERT INTO departments  
VALUES      (130, 'Finance', NULL, NULL) ;
```

1행이 입력되었습니다.

## 2. DML의 종류

### (1) INSERT

#### 4) 특별한 값의 INSERT

- **SYSDATE** 함수를 통한 날짜값 입력

```
INSERT INTO employees (employee_id, hire_date, ...)  
VALUES      (300, SYSDATE, ...);
```

1행이 입력되었습니다.

## 2. DML의 종류

### (1) INSERT

#### 5) 다른 테이블의 값을 복사하여 INSERT

- 서브쿼리를 사용하여 다른 테이블의 데이터를 복사

```
INSERT INTO sales_reps (id, name, salary, commission_pct )  
SELECT employee_id, last_name, salary, commission_pct  
FROM employees ;
```

107 행이 입력되었습니다.

- **VALUES절을 사용하지 않음**
- **INSERT절의 컬럼 수와 서브쿼리 SELECT절의 컬럼 수가 같아야 함**



## 2. DML의 종류

### (2) UPDATE

- 테이블에 존재하는 행의 내용을 변경할 때 사용
- 하나의 UPDATE문을 통해서 한 행 이상의 행 변경 가능

#### 1) UPDATE 구문

```
UPDATE    테이블명  
SET       컬럼명 = 값  
[WHERE    조건 ] ;
```

## 2. DML의 종류

### (2) UPDATE

#### 2) UPDATE문 작성

- WHERE절을 작성하지 않으면 테이블 전체 행이 변경

```
UPDATE employees  
SET      department_id = 80  
WHERE    employee_id = 100 ;
```

1행이 변경되었습니다.

## 2. DML의 종류

### (2) UPDATE

#### 3) 서브쿼리를 사용하여 동시에 여러 컬럼값을 UPDATE

101번 사원의 직종과 부서번호를 102번과 같게 변경

```
UPDATE employees
SET   job_id = (SELECT job_id
                FROM   employees
                WHERE  employee_id = 102),
      department_id = (SELECT department_id
                      FROM   employees
                      WHERE  employee_id = 102)
WHERE employee_id = 101 ;
```

1행이 변경되었습니다.

## 2. DML의 종류

### (2) UPDATE

#### 4) 다른 테이블의 값을 기반으로 UPDATE

101번 사원의 직종과 부서번호를 102번과 같게 변경

```
UPDATE emp_copy
SET    job_id = (SELECT job_id
                  FROM   employees
                  WHERE  employee_id = 102),
      department_id = (SELECT department_id
                       FROM   employees
                       WHERE  employee_id = 102)
WHERE  employee_id = 101 ;
```

1행이 변경되었습니다.

## 2. DML의 종류

### (2) UPDATE

#### 5) 무결성 제약조건 에러

```
UPDATE employees  
SET    department_id = 55  
WHERE  department_id = 100 ;
```

```
UPDATE employees  
*  
ERROR at line 1 :  
ORA-02291 : integrity constraint (HR.EMP_DEPT_FK) violated –  
parent key not found
```

**부서 테이블에 55번 부서가 존재하지 않음**

## 2. DML의 종류

### (3) DELETE

- 테이블에 존재하는 행을 삭제할 때 사용

#### 1) DELETE 구문

```
DELETE [FROM]   테이블명  
[WHERE          조건 ] ;
```

## 2. DML의 종류

### (3) DELETE

#### 2) DELETE문 작성

- WHERE절을 작성하지 않으면 테이블 전체 행이 삭제

```
DELETE employees  
WHERE employee_id = 100 ;
```

1행이 삭제되었습니다.

## 2. DML의 종류

### (3) DELETE

#### 3) 다른 테이블의 값을 기반으로 DELETE

부서이름에 Finance가 들어가는 부서에 근무하는 직원들을 삭제

```
DELETE employees
WHERE department_id =
      ( SELECT department_id
        FROM departments
        WHERE department_name LIKE '%Finance%' );
```

1행이 삭제되었습니다.



## 2. DML의 종류

### (3) DELETE

#### 4) 무결성 제약조건 에러

```
DELETE departments  
WHERE department_id = 60 ;
```

```
DELETE departments  
*  
ERROR at line 1 :  
ORA-02291 : integrity constraint (HR.EMP_DEPT_FK) violated –  
Child record found
```

사원테이블에 60번 부서값을 참조하는 값이 있음

## 2. DML의 종류

### (4) MERGE

- 테이블의 값을 상황에 따라 조건적으로 UPDATE 또는 INSERT
- 행이 존재할 경우 UPDATE를 수행, 행이 존재하지 않을 경우 INSERT를 수행

## 2. DML의 종류

### (4) MERGE

#### 1) MERGE 구문

```
MERGE INTO 테이블명 테이블별칭
      USING 테이블명(뷰 또는 서브쿼리) 별칭
      ON (조인 조건)
      WHEN MATCHED THEN
            UPDATE SET
            컬럼 1 = 값,
            컬럼 2 = 값
      WHEN NOT MATCHED THEN
            INSERT (컬럼명,...)
            VALUES (값, ...) ;
```

## 2. DML의 종류

### (4) MERGE

#### 2) MERGE문 작성

```
MERGE INTO copy_emp c
  USING employees e
  ON (c.employee_id = e.employee_id )
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name = e.first_name,
      c.last_name = e.last_name
    ...
  WHEN NOT MATCHED THEN
    INSERT
      VALUES (e.employee_id, e.first_name, ...);
```