Data Manipulation Language

데이터 조작어

INSERT 구문

UPDATE 구문

DELETE 구문

MERGE 구문

데이터 조작어

- 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 관리하는데 사용되는 언어
- DML문은 다음의 경우에 실행된다.
  - 테이블에 새로운 행을 추가 – INSERT
  - 테이블의 행 변경 – UPDATE
  - 테이블의 행 삭제 – DELETE
- 트랜잭션이란 작업의 논리적인 단위를 형성하게 하는 DML문의 집합으로 구성된다.

INSERT문 구문

```
INSERT INTO table [(column [, column …])]
VALUES (value [, value …]);
```

- 한번에 오직 한 행씩만 삽입이 가능하다.

새 행 삽입

- 각각의 컬럼에 대해 값을 갖는 새로운 행을 추가한다.
- 테이블에 정의된 컬럼의 순서에 따라 값을 나열한다.
- 선택적으로, INSERT절에 컬럼을 기입할 수 있다.

```
INSERT INTO departments(department_id, department_name,
manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);


1 row created.
```

- 문자와 날짜 값은 ‘ ’로 둘러싸야 한다.

널 값을 갖는 행 삽입

- 묵시적인 방법 : 컬럼 리스트에서 컬럼을 생략한다.

```
INSERT INTO departments(department_id, department_name)
VALUES (30, 'Purchasing');


1 row created.
```

- 명시적인 방법 : VALUE 절에 NULL 키워드를 기입한다.

```
INSERT INTO departments
VALUES (100, 'Finance', NULL, NULL);


1 row created.
```

특정 값 삽입

- SYSDATE 함수는 현재의 날짜와 시각을 기록한다.

```
INSERT INTO employees(employee_id, first_name, last_name,
                      email, phone_number, hire_date, job_id, salary,
                      commission_pct, manager_id, department_id)
VALUES (113, 'Louis', 'Popp', 'LPOPP', '515.124.4567',
                SYSDATE, 'AC_ACCOUNT', 6900, NULL, 205, 100);
```

```
1 row created.
```

특정 날짜 값 삽입

■ 새로운 사원 추가

```
INSERT INTO employees
VALUES (114, 'Den', 'Raphealy', 'DRAPHEAL', '515.127.4561',
                TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
                'AC_ACCOUNT', 11000, NULL, 100, 30);


      1 row created.
```

다른 테이블에서 행 복사

- 서브 쿼리로 INSERT 문을 작성할 수 있다.

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
        SELECT employee_id, last_name, salary, commission_pct
        FROM employees
        WHERE job_id LIKE '%REP%';


    4 rows created.
```

- VALUES 절은 사용하지 않는다.
- 서브 쿼리에 있는 컬럼의 수와 INSERT 절에 있는 컬럼의 수는 일치해야 한다.

CREATE TABLE … AS 문

- 존재하는 테이블의 복사본을 생성하기 위하여 CREATE TABLE 문과 SELECT 문을 결합하여 사용
- 새로 생성되는 테이블은 SELECT 문의 결과와 동일한 컬럼 정의들을 갖음
- 새로 생성되는 테이블은 SELECT 문의 결과 (행과 열)로 채워짐

```
CREATE TABLE new_table_name AS
        SELECT column1, column2, …
        FROM existing_table_name
        WHERE where_conditions;
```

- 예제: employees 테이블을 cpy_emp 테이블로 복사한다.

```
CREATE TABLE cpy_emp AS
SELECT *
FROM employees;
```

UPDATE문 구문

- UPDATE 문을 사용하여 행의 값을 변경할 수 있다.

```
UPDATE table
SET column = value [, column = value, …]
[WHERE condition];
```

- 만약 필요하다면, 한번에 하나 이상의 행을 변경할 수 있다.

테이블의 행 갱신

- WHERE 절을 사용하여 특정한 행을 변경할 수 있다.

```
UPDATE employees
SET department_id = 70
WHERE employee_id = 113;


      1 row updated.
```

- WHERE 절을 생략하면 테이블에 있는 모든 행을 변경한다.

```
UPDATE copy_emp
SET department_id = 110;


      22 rows updated.
```

서브쿼리로 두 열 갱신

■ 사원 번호 114인 사원의 업무와 봉급을 사원 번호 205인 사원과 동일하게 변경하는 DML 예제

```
UPDATE employees
SET job_id = (SELECT job_id
              FROM employees
              WHERE employee_id = 205),
    salary = (SELECT salary
              FROM employees
              WHERE employee_id = 205)
WHERE employee_id = 114;
```

```
1 row updated.
```

다른 테이블을 기반으로 행 갱신

- 다른 테이블의 값을 이용하여 테이블의 행을 변경하기 위하여 UPDATE 문 안에 서브 쿼리를 사용할 수 있다.

```
UPDATE copy_emp
SET department_id = (SELECT department_id
                     FROM employees
                     WHERE employee_id = 100)
WHERE job_id = (SELECT job_Id
                FROM employees
                WHERE employee_id = 200);
```

```
1 row updated.
```

행 갱신 : 무결성 제약조건 오류

```
UPDATE employees
SET department_id = 55
WHERE department_id = 110;
```

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

부서 번호 55는 부서 테이블에 없기 때문에
error가 발생한다.

DELETE문

- DELETE 문을 이용하여 테이블에서 행을 삭제할 수 있다.

```
DELETE [FROM] table
[WHERE condition];
```

테이블에서 행 삭제

- WHERE 절을 사용하여 삭제할 행을 기술할 수 있다.

```
DELETE FROM departments
WHERE departments_name = 'Finance';

        1 row deleted.
```

- WHERE 절을 생략하면 테이블에 있는 모든 행을 삭제한다.

```
DELETE FROM copy_emp;

        22 rows deleted.
```

다른 테이블을 기반으로 행 삭제

- 다른 테이블에 있는 값을 이용하여 테이블에 있는 행을 삭제하기 위하여 DELETE 문에 서브 쿼리를 사용할 수 있다.

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

```
1 row deleted.
```

행 삭제 : 무결성 제약조건 오류

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

```
DELETE FROM departments
            *
    ERROR at line 1 :
    ORA – 02292 : integrity constraint (HR.EMP_DEPT_FK)
    violated – child record found
```

다른 테이블에서 참조되어지는 primary key를 포함하고 있는 행을 삭제하려고 할 때 발생한다.

명시적(Explicit) 기본 기능 개요

- 명시적 기본 기능을 사용하면 열 기본값이 필요한 경우 DEFAULT 키워드를 열 값으로 사용할 수 있습니다.
- 이 기능은 SQL:1999 표준을 준수하기 위해 추가되었습니다.
- 이 기능을 통해 데이터에 기본값이 적용될 위치 및 시기를 제어할 수 있습니다.
- 명시적 기본값은 INSERT 및 UPDATE문에 사용할 수 있습니다.

명시적(Explicit) 기본 값 사용

- DEFAULT with INSERT :

```
INSERT INTO departments (department_id, department_name, manager_id)
VALUES (300, 'Engineering', DEFAULT);
```

- DEFAULT with UPDATE :

```
UPDATE departments
SET manager_id = DEFAULT WHERE department_id = 10;
```

MERGE 문

- 데이터 베이스 테이블로 데이터를 제한적으로 변경하거나 삽입할 수 있는 방법을 제공한다.
- 이미 존재하는 행이면 UPDATE가 실행되고, 새로운 행이면 INSERT가 실행된다.
  - 여러 번의 변경을 피할 수 있다.
  - 성능이 향상되고 사용하기 쉽다.
  - 데이터 웨어하우징 환경에 사용할 수 있다.

MERGE 문 구문

- MERGE 문을 사용하여 테이블에 있는 행들을 제한적으로 삽입하거나 변경할 수 있다.

```
MERGE INTO table_name table_alias
        USING (table | view | sub_query) alias
        ON (join condition)
        WHEN MATCHED THEN
            UPDATE SET
            col1 = col_val1,
            col2 = col2_val2
        WHEN NOT MATCHED THEN
            INSERT (column_list)
            VALUES (column_values);
```

행 병합

■ EMPLOYEES 테이블과 일치하도록 COPY_EMP 테이블에 행을 삽입하거나 변경한다.

```
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,
            …
            c.department_id = e.department_id
    WHEN NOT MATCHED THEN
        INSERT VALUES (e.employee_id, e.first_name, e.last_name, e.email,
            e.phone_number, e.hire_date, e.job_id, e.salary, e.commission_pct,
            e.manager_id, e.department_id);
```

실습

1. employees 테이블을 복사하여 cpy_emp 테이블을 생성하시오. describe 문을 사용하여 employees와 cpy_emp의 스키마가 동일한지 확인하기 바랍니다.
2. 다음 예제 데이터의 첫 번째 데이터 행을 cpy_emp 테이블에 추가하십시오. Insert 절에 열을 나열하지 마십시오.

Employee_id	First_name	Last_name	Email	Hire_date	Job_id
300	Ralph	Patel	Rpatel	Now	SA_MAN
301	Dancs	Betty	Bdancs	Now	SA_REP

3. 위의 목록에 있는 예제 데이터의 두 번째 행을 cpy_emp 테이블에 추가하십시오. 이번에는 Insert 절에 열을 명시적으로 나열하십시오.
4. 301번 사원의 Last_name을 Drexler로 변경하십시오.
5. 300, 301번 사원의 나머지 사원 정보를 Vance Jones와 동일하게 변경하십시오.
6. 급여가 3000미만인 모든 사원의 급여를 3000으로 변경하십시오.
7. Dancs Drexler 사원의 정보를 삭제하십시오.