3장 대형 데이터 셋 다루기

9. 서브쿼리를 사용한 테이블 생성(1)

- CREATE TABLE 문장과 AS subquery옵션을 조합하여 테이블을 생성하고 행을 삽입함.

```
CLEATE  TABLE  table
      [column( ,  column…..)]
As  subquery;
```

- 서브쿼리 열의 개수와 명시된 열의 개수를 일치시킴.
- 열 이름과 디폴트 값을 가진 열을 정의

9. 서브쿼리를 사용한 테이블 생성(2)

```
SQL> CREATE  TABLE  dept30
  2   AS
  3        SELECT  empno,  ename,  sal* 12  annsal,  hiredate
  4        FORM   emp
  5        WHERE    deptno = 30;
Table created.
```

```
SQL> DESCRIBE   dept30;
```

```
 Name                          Null?           Type
 ----------------------------  --------------  ---------------
EMPNO                          NOT NULL        NUMBER(4)
ENAME                                          VARCHAR2(10)
ANNSAL                                          NUMBER
HIREDATE                                        DATE
```

DEFAULT 옵션

- 삽입 시에 열에 대한 디폴트 값 명시.

```
_ hiredate  DATE   DEFAULT  SYSDATE, _
```

- 가능한 값은 리터럴 값, 표현식 또는 SQL 함수
- 불가능한 값은 다른 열의 이름이나 의사열 ((ex)NEXTVAL, CURRVAL)
- 디폴트 데이터형은 열의 데이터형과 일치해야 함.

복수 테이블에 대한 INSERT 문장

- **INSERT... SELECT** 문장을 수행할 때 하나의 **DML** 문장만으로 복수의 테이블들에 행들을 삽입할 수 있습니다.
- 데이터 웨어하우스를 구축함에 있어서 원천 데이터로부터 여러 개의 목표 테이블로 데이터를 전송하는 데 사용할 수 있습니다.

복수 테이블 **INSERT**의 장점

- 복수 개의 테이블에 데이터를 채우기 위해 복수의 **INSERT... SELECT** 문장을 수행할 필요가 없습니다.
- 복수 삽입을 위한 조건 논리를 강제하기 위해 프로시져를 정의할 필요가 없습니다.
- 원천 데이터에 대한 반복된 스캔이 일어나지 않아도 된다는 점에서 상기한 두 방법에 비해 성능상의 이점이 있습니다.

복수 테이블 **INSERT** 문장의 종류

- 무조건적 **INSERT ALL**
- 조건부 **INSERT ALL**
- 조건부 **INSERT FIRST**

예: 무조건적 INSERT ALL

```
INSERT ALL
INTO product_activity VALUES(
  today, product_id, quantity
)
INTO product_sales VALUES(
  today, product_id, total
)
SELECT trunc(order_date) today, product_id,
       SUM(unit_price) total, SUM(quantity)
       quantity
  FROM orders, order_items
 WHERE orders.order_id = order_items.order_id
   AND order_date = TRUNC(SYSDATE)
GROUP BY product_id;
```

예: 무조건적 INSERT ALL - Pivoting INSERT

```
INSERT ALL
INTO sales VALUES (
  product_id, TO_DATE(week_id, 'WW'), sales_sun
)
INTO sales VALUES (
  product_id, TO_DATE(week_id, 'WW'), sales_mon
)
INTO sales VALUES (
  product_id, TO_DATE(week_id, 'WW'), sales_tue
)
INTO sales VALUES (
  product_id, TO_DATE(week_id, 'WW'), sales_wed
)
```

예: 무조건적 INSERT ALL - Pivoting INSERT(계속)

```
INTO sales VALUES (
  product_id, TO_DATE(week_id, 'WW'), sales_thu
)
INTO sales VALUES (
  product_id, TO_DATE(week_id, 'WW'), sales_fri
)
INTO sales VALUES (
  product_id, TO_DATE(week_id, 'WW'), sales_sat
)
SELECT product_id, week_id, sales_sun, sales_mon,
       sales_tue, sales_wed, sales_thu, sales_fri,
       sales_sat
  FROM sales_source_data;
```

조건부 INSERT ALL

```
INSERT ALL
   WHEN product_id IN (
     select product_id FROM promotional_items
   )
    INTO promotional_sales
    VALUES (product_id, list_price)
   WHEN order_mode = 'online'
    INTO web_orders
    VALUES (product_id, order_total)
SELECT product_id, list_price, order_total,
       order_mode
  FROM orders;
```

예: 조건부 INSERT FIRST

```
INSERT FIRST
   WHEN order_total> 10000 THEN
     INTO priority_handling VALUES (id)
   WHEN order_total > 5000 THEN
     INTO special_handling VALUES (id)
   WHEN total > 3000 THEN
     INTO privilege_handling VALUES (id)
   ELSE
     INTO regular_handling VALUES (id)
SELECT order_total, order_id id FROM orders;
```

MERGE 문

- 테이블에 대해 조건적으로 갱신이나 삽입을 할 수 있도록 하는 **“upsert”** 기능을 제공합니다.
- 만일 그 행이 테이블에 이미 존재한다면 갱신을 수행하고, 그렇지 않으면 새로 행을 삽입합니다.
- 데이터 웨어하우징 애플리케이션의 **ETL** 과정에 유용합니다.

MERGE 문의 이점

- **MERGE** 문을 사용함으로써 하나의 **SQL** 문으로 하여금 갱신이나 삽입, 또는 둘 다를 수행할 수 있습니다.
- **MERGE** 문은 사용자에게 투명한 방식으로 병렬 수행됩니다.
- 원천 테이블에 대한 스캔이 덜 일어나므로 성능의 향상이 있습니다.

예: 데이터 웨어 하우징에서의 MERGE 문의 사용

```
MERGE INTO customer C
USING cust_src s
ON (c.customer_id = s.src_customer_id)
WHEN MATCHED THEN
  UPDATE SET c.cust_address = s.cust_address
WHEN NOT MATCHED THEN
  INSERT (customer_id, cust_first_name, ...)
  VALUES (src_customer_id, src_first_name, …);
```