Chapter 20. Oracle 9i에서 향상된 DML과 DDL

이번 장에서는 Oracle 9i부터 추가된 다중 테이블 INSERT와 EXTERNAL TABLE에 대하여 설명한다. 다중 테이블 INSERT는 INSERT INTO ... SELECT문을 향상시킨 것으로 이전의 INSERT 문장은 한 번에 한 개의 테이블에 행을 입력 할 수 있었지만 다중 테이블 INSERT는 동시에 여러 개의 테이블에 행을 입력 할 수 있다. EXTERNAL TABLE은 데이터베이스 외부의 텍스트 파일을 테이블처럼 선언하여 검색하는 방법이다.

다중 테이블 INSERT

다중 테이블 INSERT 문장은 서브쿼리의 결과를 1개 이상의 테이블에 입력 할 수 있기 때문에 기존의 데이터베이스에 구축된 많은 데이터를 데이터웨어 하우스로 구축하는 업무에 매우 유용하다. 다중 테이블 INSERT 문장의 종류는 다음과 같다.

- 무조건 INSERT
- 조건부 INSERT ALL
- 조건부 INSERT FIRST
- 피벗팅(Pivoting) INSERT

다중 테이블 INSERT 문장의 문법은 다음과 같다.

```
INSERT [ALL] [conditional_insert_clause]
[insert_into_clause values_clause] (subquery)
```

여기서, *conditional_insert_clause*는 다음과 같다.

```
[ALL] [FIRST]
[WHEN condition THEN] [insert_into_clause values_clause]
[ELSE] [insert_into_clause values_clause]
```

무조건 INSERT

무조건 INSERT는 INSERT 뒤에 ALL을 기술하고 *insert_into_clause* 절에 서브쿼리에서 리턴된 행들을 입력할 테이블과 컬럼명을 필요한 만큼 기술하면 된다. Oracle 서버는 서브쿼리에서 리턴된 행들을 INTO 문장에 기술된 모든 테이블에 각각 입력한다.

무조건 INSERT 문장을 실행하기에 앞서 다음과 같은 임시테이블을 생성한다.

```
SQL> CREATE TABLE T1
  2  AS SELECT EMPNO, ENAME, HIREDATE
  3  FROM EMP
  4  WHERE 0 = 1;

테이블이 생성되었습니다.

SQL> CREATE TABLE T2
  2  AS SELECT EMPNO, ENAME, SAL
  3  FROM EMP
  4  WHERE 0 = 1;

테이블이 생성되었습니다.
```

사원 테이블에서 10번 부서에 근무하는 사원의 사번, 사원명, 입사일은 T1 테이블, 사번, 사원명, 급여는 T2 테이블에 입력하는 방법은 다음과 같다.

```
SQL> INSERT ALL
  2    INTO T1 VALUES (EMPNO, ENAME, HIREDATE)
  3    INTO T2 VALUES (EMPNO, ENAME, SAL)
  4    SELECT EMPNO, ENAME, HIREDATE, SAL
  5    FROM EMP
  6    WHERE DEPTNO = 10;

6 개의 행이 만들어졌습니다.
```

조건부 INSERT ALL

조건부 INSERT ALL은 *conditional_insert_clause*에 ALL을 기술하고 *insert_into_clause*에는 WHEN *condition*이 만족하는 경우에 입력 될 테이블과 컬럼명을 작성한다. Oracle 서버는 서브쿼리에서 리턴된 행들을 WHEN 조건과 비교하여 만족하는 경우 해당 테이블에 입력한다. 만약, INSERT ALL 문장에 ELSE 조건이 포함되어 있으면 어떤 조건에도 만족하지 않는 행들을 ELSE 조건에 기술된 테이블에 저장한다.

사원 테이블에서 입사일이 1981년 이후인 직원의 사번, 사원명, 입사일을 T2, 급여가 3000이 넘는 직원의 사번, 사원명, 급여를 T2 테이블에 입력하는 방법은 다음과 같다.

```
SQL> INSERT ALL
  2    WHEN HIREDATE >= '81/01/01' THEN
  3      INTO T1 VALUES (EMPNO, ENAME, HIREDATE)
  4    WHEN SAL > 3000 THEN
  5      INTO T2 VALUES (EMPNO, ENAME, SAL)
  6    SELECT EMPNO, ENAME, HIREDATE, SAL
  7    FROM EMP;

4 개의 행이 만들어졌습니다.
```

위의 조건부 INSERT ALL에서 주의할 사항은 입사일이 1981년 이후이고 급여가 3000이 초과하는 사원의 데이터는 T1과 T2 테이블에 모두 저장된다는 사실이다.

조건부 INSERT FIRST

조건부 INSERT FIRST는 조건부 INSERT ALL과 사용방법이 유사하다. 조건부 INSERT FIRST는 서브쿼리에서 리턴된 행을 INSERT FIRST 문장에서 기술된 WHEN 조건과 차례로 비교하면서 조건에 만족하면 해당 테이블에 입력하고 INSERT FIRST 문장을 종료한다. 만약, INSERT FIRST 문장에 ELSE 조건이 포함되어 있으면 어떤 조건에도 만족하지 않는 행들은 ELSE 조건에 기술된 INSERT 문장에 의해 해당 테이블에 저장된다.

조건부 INSERT FIRST 문장을 실행하기에 앞서 다음과 같은 임시테이블을 생성한다.

```
SQL> CREATE TABLE SAL1000
  2  AS SELECT EMPNO, ENAME, SAL FROM EMP
  3  WHERE 0 = 1;

테이블이 생성되었습니다.

SQL> CREATE TABLE SAL3000
  2  AS SELECT EMPNO, ENAME, SAL FROM EMP
  3  WHERE 0 = 1;

테이블이 생성되었습니다.

SQL> CREATE TABLE SALMAX
  2  AS SELECT EMPNO, ENAME, SAL FROM EMP
  3  WHERE 0 = 1;

테이블이 생성되었습니다.
```

사원 테이블에서 직원의 급여가 1000 이하이면 SAL1000, 3000 이하이면 SAL3000, 3000이 초과되면 SALMAX 테이블에 사번, 사원명, 급여를 입력하는 방법은 다음과 같다.

```
SQL> INSERT FIRST
  2    WHEN SAL <= 1000 THEN
  3      INTO SAL1000 VALUES (EMPNO, ENAME, SAL)
  4    WHEN SAL <= 3000 THEN
  5      INTO SAL3000 VALUES (EMPNO, ENAME, SAL)
  6    ELSE
  7      INTO SALMAX VALUES (EMPNO, ENAME, SAL)
  8    SELECT EMPNO, ENAME, SAL FROM EMP;

14 개의 행이 만들어졌습니다.
```

피벗팅(Pivoting) INSERT

피벗팅 INSERT는 한 개의 행을 한 개의 테이블내 여러 개의 행으로 입력하는 방법으로서 비관계형 데이터베이스의 테이블을 관계형 데이터베이스의 테이블로 변환하는데 유용하다. 사용 방법은 무조건 INSERT ALL 문장과 동일하지만 INTO 절에는 모두 같은 테이블을 기술하는 것이 다르다.

피벗팅 INSERT 문장을 실행하기에 앞서 다음과 같은 임시테이블을 생성한다.

```
CREATE TABLE SALES_DATA
(EMPNO NUMBER(4),
SPRING_SALE NUMBER(9),
SUMMER_SALE NUMBER(9),
AUTUMN_SALE NUMBER(9),
WINTER_SALE NUMBER(9));

INSERT INTO SALES_DATA VALUES (100, 3000, 5000, 6000, 7000);
INSERT INTO SALES_DATA VALUES (200, 4000, 2000, 3000, 5000);
INSERT INTO SALES_DATA VALUES (300, 6000, 3000, 4000, 4000);
INSERT INTO SALES_DATA VALUES (400, 3000, 1000, 5000, 2000);

CREATE TABLE SALES_SEASON
(EMPNO NUMBER(4),
SEASON_CODE CHAR,
SALES NUMBER(9));
```

SALES_DATA 테이블은 각 사원들의 계절별 판매액을 저장한 테이블인데, 계절별로 별도 컬럼이 있으며, 각 컬럼에 계절별 판매액이 저장되어 있다. 만약, 계절별로 컬럼을 두지 않고 SALES_SEASON 테이블과 같이 한 개의 컬럼을 가진 테이블을 만들어 데이터를 이관하고자 한다면, 피벗팅 INSERT 문장을 사용하여 수월하게 데이터를 이관 할 수 있다.

```
SQL> INSERT ALL
  2  INTO SALES_SEASON VALUES (EMPNO, '1', SPRING_SALE)
  3  INTO SALES_SEASON VALUES (EMPNO, '2', SUMMER_SALE)
  4  INTO SALES_SEASON VALUES (EMPNO, '3', AUTUMN_SALE)
  5  INTO SALES_SEASON VALUES (EMPNO, '4', WINTER_SALE)
  6  SELECT EMPNO, SPRING_SALE, SUMMER_SALE, AUTUMN_SALE, WINTER_SALE
  7  FROM SALES_DATA;

16 개의 행이 만들어졌습니다.
```

EXTERNAL TABLE

EXTERNAL TABLE은 데이터베이스에는 테이블에 대한 정의만 저장되고 실제 데이터는 데이터베이스 외부에 저장되는 읽기 전용 테이블로서 데이터베이스에 저장 시킬 필요 없이 직접 검색 또는 조인이 가능하다. 그러나 EXTERNAL TABLE은 읽기 전용으로 DML 작업이 불가능하며 인덱스를 생성할 수도 없다.

EXTERNAL TABLE을 사용하는 방법은 다음과 같다. 먼저, 외부 파일이 저장되어 있는 디렉터리 명으로 디렉터리 객체를 생성한다. 디렉터리 객체를 생성하기 위해서는 CREATE ANY DIRECTORY 권한을 부여 받아야 한다. CREATE ANY DIRECTORY 권한을 부여 받았으며 외부 파일이 C:₩ORACLE 디렉터리에 저장되어 있다면 디렉터리 객체를 만드는 방법은 다음과 같다.

```
SQL> CONNECT / AS SYSDBA
연결되었습니다.
SQL> GRANT CREATE ANY DIRECTORY TO SCOTT;

권한이 부여되었습니다.

SQL> CONNECT SCOTT/TIGER
연결되었습니다.
SQL> CREATE DIRECTORY EMP_DIR AS 'C:WORACLE';

디렉토리가 생성되었습니다.
```

C:₩ORACLE 디렉터리의 외부 파일 EMP.TXT의 내용은 다음과 같다.

```
7369,SMITH,CLERK,7902,80/12/17
7499,ALLEN,SALESMAN,7698,81/02/20
7521,WARD,SALESMAN,7698,81/02/22
7566,JONES,MANAGER,7839,81/04/02
7654,MARTIN,SALESMAN,7698,81/09/28
7698,BLAKE,MANAGER,7839,81/05/01
7782,CLARK,MANAGER,7839,81/06/09
7788,SCOTT,ANALYST,7566,87/04/19
7839,KING,PRESIDENT,,81/11/17
7844,TURNER,SALESMAN,7698,81/09/08
7876,ADAMS,CLERK,7788,87/05/23
7900,JAMES,CLERK,7698,81/12/03
7902,FORD,ANALYST,7566,81/12/03
7934,MILLER,CLERK,7782,82/01/23
```

외부 파일을 기반으로 EXTERNAL TABLE을 생성하는 방법은 다음과 같다.

```
SQL> CREATE TABLE EXTEMP (
  2  EMPNO NUMBER, ENAME VARCHAR2(10), JOB VARCHAR2(10), MGR NUMBER, HIREDATE DATE)
  3  ORGANIZATION EXTERNAL
  4  (TYPE ORACLE_LOADER
  5  DEFAULT DIRECTORY EMP_DIR
  6  ACCESS PARAMETERS
  7  (RECORDS DELIMITED BY NEWLINE
  8  BADFILE 'BAD_EMP'
  9  LOGFILE 'LOG_EMP'
 10  FIELDS TERMINATED BY ','
 11  (EMPNO CHAR,
 12  ENAME CHAR,
 13  JOB CHAR,
 14  MGR CHAR,
 15  HIREDATE CHAR DATE_FORMAT DATE MASK "YY/MM/DD"))
 16  LOCATION('EMP.TXT'))
 17  PARALLEL 5
 18  REJECT LIMIT 200;

테이블이 생성되었습니다.
```

위에서 EXTERNAL TABLE을 정의 할 때는 CREATE TABLE에 ORGANIZATION EXTERNAL을 지정하여 데이터가 데이터베이스 외부에 저장되어 있으며 읽기 전용이라는 것을 데이터베이스에 알려주어야 한다. TYPE절 뒤에는 외부 파일 접근을 위한 드라이버

이름을 기술하는데 Oracle은 SQL*Loader 유틸리티 기반의 ORACLE_LOADER 드라이버와 Import/Export 유틸리티 기반의 ORACLE_INTERVAL 드라이버를 제공하며, 별도로 기술하지 않으면 디폴트로 ORACLE_LOADER 드라이버가 사용된다. DEFAULT DIRECTORY는 외부 파일이 저장되어 있는 디렉터리 위치를 기술한 디렉터리 객체의 이름을 지정하고 ACCESS PARAMETERS에는 외부 파일의 형식을 기술한다. LOCATION에는 외부 파일의 이름을 지정하지만 반드시 기술 할 필요는 없으며 REJECT LIMIT는 외부 파일을 읽으면서 발생 할 수 있는 오류의 최대 갯수를 적어주면 된다.

CREATE TABLE 문장에서 CREATE INDEX 사용

Oracle 서버는 테이블에 PRIMARY KEY 제약조건을 설정하면 자동으로 제약조건의 이름과 동일한 고유 인덱스가 생성된다. Oracle 9i에서는 PRIMARY KEY 제약조건과 함께 CREATE INDEX 문장을 사용하면 인덱스 이름을 사용자 임의로 지정 할 수 있다.

```
SQL> CREATE TABLE NEW_EMP
  2  (EMPNO NUMBER(4)
  3  PRIMARY KEY USING INDEX
  4  (CREATE INDEX EMP_EMPNO_IDX ON
  5  NEW_EMP(EMPNO)),
  6  ENAME VARCHAR2(10),
  7  JOB VARCHAR2(10));

테이블이 생성되었습니다.

SQL> SELECT INDEX_NAME, TABLE_NAME
  2  FROM USER_INDEXES
  3  WHERE TABLE_NAME = 'NEW_EMP';

INDEX_NAME                     TABLE_NAME
------------------------------ ------------------------------
EMP_EMPNO_IDX                  NEW_EMP
```

복습

1. 다음과 같은 임시 테이블을 생성하고, 사원 테이블의 사원정보를 업무별로 ANALYST, CLERK, MANAGER, PRESIDENT, SALESMAN 테이블에 각각 입력하시오

```
CREATE TABLE ANALYST AS SELECT * FROM EMP WHERE 0 = 1;
CREATE TABLE CLERK AS SELECT * FROM EMP WHERE 0 = 1;
CREATE TABLE MANAGER AS SELECT * FROM EMP WHERE 0 = 1;
CREATE TABLE PRESIDENT AS SELECT * FROM EMP WHERE 0 = 1;
CREATE TABLE SALESMAN AS SELECT * FROM EMP WHERE 0 = 1;
```

2. 다음과 같은 임시 테이블을 생성하고, 부서번호가 10이고 급여가 2000이 초과되는 직원의 사원정보는 T10, 부서번호가 20이고 급여가 1000 미만인 직원의 사원정보는 T20, 부서번호가 30이고 급여가 2000에서 3000사이인 직원의 사원정보는 T30 테이블에 각각 입력하시오.

```
CREATE TABLE T10 AS SELECT * FROM EMP WHERE 0 = 1;
CREATE TABLE T20 AS SELECT * FROM EMP WHERE 0 = 1;
CREATE TABLE T30 AS SELECT * FROM EMP WHERE 0 = 1;
```