20

Oracle9*i* Extensions to DML and DDL Statements

Objectives

After completing this lesson, you should be able to do the following:

- Describe the features of multitable inserts
- Use the following types of multitable inserts
  - Unconditional `INSERT`
  - Pivoting `INSERT`
  - Conditional `ALL INSERT`
  - Conditional `FIRST INSERT`
- Create and use external tables
- Name the index at the time of creating a primary key constraint

Review of the INSERT Statement

- Add new rows to a table by using the INSERT statement.

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

- Only one row is inserted at a time with this syntax.

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

Review of the UPDATE Statement

- Modify existing rows with the UPDATE statement.

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

- Update more than one row at a time, if required.
- Specific row or rows are modified if you specify the WHERE clause.

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 142;
1 row updated.
```

Overview of Multitable INSERT Statements

- The INSERT...SELECT statement can be used to insert rows into multiple tables as part of a single DML statement.
- Multitable INSERT statements can be used in data warehousing systems to transfer data from one or more operational sources to a set of target tables.
- They provide significant performance improvement over:
  - Single DML versus multiple INSERT...SELECT statements
  - Single DML versus a procedure to do multiple inserts using IF...THEN syntax

Types of Multitable `INSERT` Statements

Oracle9*i* introduces the following types of multitable insert statements:

- Unconditional `INSERT`
- Conditional `ALL INSERT`
- Conditional `FIRST INSERT`
- Pivoting `INSERT`

Multitable INSERT Statements

Syntax

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

conditional_insert_clause

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

Unconditional INSERT ALL

- Select the EMPLOYEE_ID, HIRE_DATE, SALARY, and MANAGER_ID values from the EMPLOYEES table for those employees whose EMPLOYEE_ID is greater than 200.
- Insert these values into the SAL_HISTORY and MGR_HISTORY tables using a multitable INSERT.

```
INSERT  ALL
   INTO sal_history VALUES(EMPID,HIREDATE,SAL)
   INTO mgr_history VALUES(EMPID,MGR,SAL)
   SELECT employee_id EMPID, hire_date HIREDATE,
          salary SAL, manager_id MGR
   FROM  employees
   WHERE employee_id > 200;
8 rows created.
```

Conditional INSERT ALL

- Select the EMPLOYEE_ID, HIRE_DATE, SALARY and MANAGER_ID values from the EMPLOYEES table for those employees whose EMPLOYEE_ID is greater than 200.
- If the SALARY is greater than $10,000, insert these values into the SAL_HISTORY table using a conditional multitable INSERT statement.
- If the MANAGER_ID is greater than 200, insert these values into the MGR_HISTORY table using a conditional multitable INSERT statement.

Conditional INSERT ALL

```
INSERT ALL
  WHEN SAL > 10000 THEN
    INTO sal_history VALUES(EMPID,HIREDATE,SAL)
  WHEN MGR > 200   THEN
    INTO mgr_history VALUES(EMPID,MGR,SAL)
    SELECT employee_id EMPID,hire_date HIREDATE,
           salary SAL, manager_id MGR
    FROM   employees
    WHERE  employee_id > 200;
4 rows created.
```

Conditional FIRST INSERT

- Select the DEPARTMENT_ID , SUM(SALARY) and MAX(HIRE_DATE) from the EMPLOYEES table.
- If the SUM(SALARY) is greater than $25,000 then insert these values into the SPECIAL_SAL, using a conditional FIRST multitable INSERT.
- If the first WHEN clause evaluates to true, the subsequent WHEN clauses for this row should be skipped.
- For the rows that do not satisfy the first WHEN condition, insert into the HIREDATE_HISTORY_00, or HIREDATE_HISTORY_99, or HIREDATE_HISTORY tables, based on the value in the HIRE_DATE column using a conditional multitable INSERT.

Conditional FIRST INSERT

```
INSERT FIRST
   WHEN SAL  > 25000          THEN
    INTO special_sal VALUES(DEPTID, SAL)
  WHEN HIREDATE like ('%00%') THEN
    INTO hiredate_history_00 VALUES(DEPTID,HIREDATE)
  WHEN HIREDATE like ('%99%') THEN
    INTO hiredate_history_99 VALUES(DEPTID, HIREDATE)
  ELSE
  INTO hiredate_history VALUES(DEPTID, HIREDATE)
  SELECT department_id DEPTID, SUM(salary) SAL,
         MAX(hire_date) HIREDATE
  FROM   employees
  GROUP BY department_id;
8 rows created.
```

Pivoting INSERT

- Suppose you receive a set of sales records from a nonrelational database table, SALES_SOURCE_DATA in the following format:

```
EMPLOYEE_ID, WEEK_ID, SALES_MON,
SALES_TUE, SALES_WED, SALES_THUR,
SALES_FRI
```

- You would want to store these records in the SALES_INFO table in a more typical relational format:

```
EMPLOYEE_ID, WEEK, SALES
```

- Using a pivoting INSERT, convert the set of sales records from the nonrelational database table to relational format.

Pivoting INSERT

```
INSERT ALL
  INTO sales_info VALUES (employee_id,week_id,sales_MON)
  INTO sales_info VALUES (employee_id,week_id,sales_TUE)
  INTO sales_info VALUES (employee_id,week_id,sales_WED)
  INTO sales_info VALUES (employee_id,week_id,sales_THUR)
  INTO sales_info VALUES (employee_id,week_id, sales_FRI)
  SELECT EMPLOYEE_ID, week_id, sales_MON, sales_TUE,
         sales_WED, sales_THUR,sales_FRI
  FROM sales_source_data;
5 rows created.
```

External Tables

- External tables are read-only tables in which the data is stored outside the database in flat files.
- The metadata for an external table is created using a `CREATE TABLE` statement.
- With the help of external tables, Oracle data can be stored or unloaded as flat files.
- The data can be queried using SQL, but you cannot use DML and no indexes can be created.

Creating an External Table

- Use the `external_table_clause` along with the `CREATE TABLE` syntax to create an external table.
- Specify `ORGANIZATION` as `EXTERNAL` to indicate that the table is located outside the database.
- The `external_table_clause` consists of the access driver `TYPE`, `external_data_properties`, and the `REJECT LIMIT`.
- The `external_data_properties` consist of the following:
  - `DEFAULT DIRECTORY`
  - `ACCESS PARAMETERS`
  - `LOCATION`

Example of Creating an External Table

Create a `DIRECTORY` object that corresponds to the directory on the file system where the external data source resides.

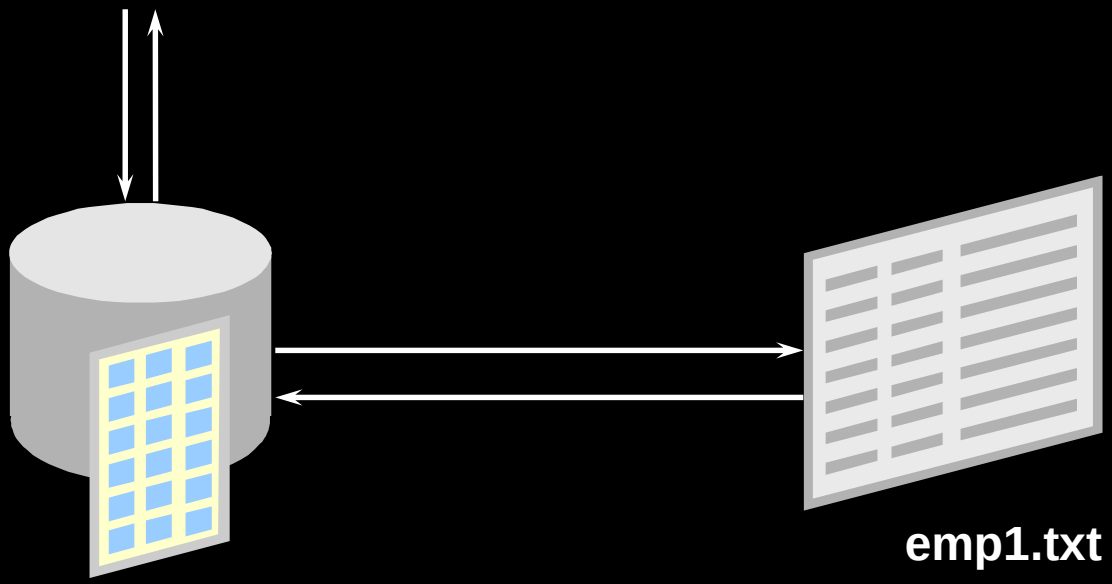
```
CREATE DIRECTORY emp_dir AS '/flat_files' ;
```

Example of Creating an External Table

```
CREATE TABLE oldemp (
  empno NUMBER, empname CHAR(20), birthdate DATE)
  ORGANIZATION EXTERNAL
  (TYPE ORACLE_LOADER
  DEFAULT DIRECTORY emp_dir
  ACCESS PARAMETERS
  (RECORDS DELIMITED BY NEWLINE
  BADFILE 'bad_emp'
  LOGFILE 'log_emp'
  FIELDS TERMINATED BY ','
  (empno CHAR,
  empname CHAR,
  birthdate CHAR date_format date mask "dd-mon-yyyy"))
  LOCATION ('emp1.txt'))
  PARALLEL 5
  REJECT LIMIT 200;
Table created.
```

Querying External Tables

```
SELECT  *
FROM oldemp
```



CREATE INDEX with CREATE TABLE Statement

```
CREATE TABLE NEW_EMP
(employee_id NUMBER(6)
             PRIMARY KEY USING INDEX
             (CREATE INDEX emp_id_idx ON
             NEW_EMP(employee_id)),
first_name   VARCHAR2(20),
last_name    VARCHAR2(25));
Table created.
```

```
SELECT INDEX_NAME, TABLE_NAME
FROM   USER_INDEXES
WHERE  TABLE_NAME = 'NEW_EMP';
```

INDEX_NAME	TABLE_NAME
EMP_ID_IDX	NEW_EMP

Summary

In this lesson, you should have learned how to:

- Use the `INSERT…SELECT` statement to insert rows into multiple tables as part of a single DML statement
- Create external tables
- Name indexes using the `CREATE INDEX` statement along with the `CREATE TABLE` statement

Practice 20 Overview

This practice covers the following topics:

- Writing unconditional `INSERT` statements
- Writing conditional `ALL INSERT` statements
- Pivoting `INSERT` statements
- Creating indexes along with the `CREATE TABLE` command