Chapter
11

SECTION II: BASICS OF SQL

Mutli-Table Insert Statements

The multi-table insert feature allows the **`INSERT . . . SELECT`** statement to use multiple tables as targets. In addition, it can distribute data among target tables based on logical attributes of the new rows. Multi-table insert, thus enables a single scan and transformation of source data to insert data into multiple tables, sharply increasing performance.

The multi-table INSERT statement:

- ❑ Allows records to be inserted into multiple tables at the same time
- ❑ Can be used to copy data from one or more tables to a different set of target tables

The multi-table INSERT statement inserts computed rows derived from the rows returned by a sub query.

There are two forms of the multi-table INSERT statement i.e. unconditional and conditional.

In the unconditional form:

- ❑ An INTO clause list is executed once for each row returned by the sub query

Syntax:

```
INSERT ALL INTO <TableName1> [VALUES(. . .)]
    INTO <TableName1> [VALUES(. . .)] . . .
        SELECT . . .
```

In the conditional form:

- An INTO clause lists are guarded by WHEN clauses that determine whether the corresponding INTO clause list is executed or not

Syntax:

```
INSERT {ALL|FIRST}
    WHEN <Expression1> THEN INTO <TableName1> [VALUES(. . .)]
    WHEN <Expression1> THEN INTO <TableName1> [VALUES(. . .)] . . .
        [ ELSE INTO <TableNameN> [VALUES(. . .)]]
            SELECT . . .
```

An INTO clause list consists of one or more INTO clauses. The execution of an INTO clause list causes the insertion of one row for each INTO clause in the list.

An INTO clause specifies the target into which a computed row is inserted. The target specified, can be any table expression that is legal for an INSERT… SELECT statement. However aliases cannot be used. The same table can be specified as the target for more than one INTO clause.

An INTO clause also provides the value of the row to be inserted using a VALUES clause. An expression used in the VALUES clause can be any legal expression, but can only refer to columns returned by the select list of the sub query.

If the VALUES clause is omitted, the select list of the sub query provides the values to be inserted. If a column list is given, each column in the list is assigned a corresponding value from the VALUES clause or the sub query. If no column list is given, the computed row must provide values for all columns in the target table.

The following are the clauses that are used with INSERT statements for multi-table inserts:

ALL ... INSERT INTO Clause

Specify ALL followed by multiple INSERT INTO clauses to perform an unconditional multi-table insert.

Oracle executes each INSERT INTO clause once for each row returned by the subquery.

Conditional INSERT Clause

Specify the CONDITIONAL INSERT clause to perform a conditional multi-table insert.

Oracle filters each INSERT INTO clause through the corresponding WHEN condition, which determines whether that INSERT INTO CLAUSE is executed. A single multi-table insert statement can contain up to 127 WHEN clauses.

ALL Clause

In case of ALL, Oracle evaluates each WHEN clause, regardless of the results of the evaluation of any other WHEN clause. For each WHEN clause whose condition evaluates to true Oracle executes the corresponding INTO clause list.

FIRST Clause

In case of FIRST, Oracle evaluates each WHEN clause in the order in which it appears in the statement. For the first WHEN clause that evaluates to true, Oracle executes the corresponding INTO clause and skips subsequent WHEN clauses for a given row.

ELSE Clause

Oracle reaches the ELSE CLAUSE, if no WHEN clause evaluates to true. If ELSE CLAUSE is specified, Oracle executes the INTO clause list associated with the ELSE clause. If ELSE CLAUSE is not specified, Oracle takes no action for that row.

SUBQUERY

Acts as a source to data held in the table. It is responsible to return rows from the desired table. The subquery can refer to any table, view or materialized view including the target tables of the INSERT statement. If the subquery selects no rows, Oracle inserts no rows into the table.

Example of Unconditional INSERT

The following table structures are used in the example that follow:

Table Name: OrderMaster

Column Name	Data Type	Size
OrderNo	Varchar2	10
OrderDate	Date	
ProductNo	Varchar2	10
CostPrice	Number	12, 2
Qty	Number	5
SellPrice	Number	12, 2
OrderStatus	Char	2

This table holds the following data:

OrderNo	OrderDate	ProductNo	CostPrice	Quantity	SellPrice	OrderStatus
O10001	30-MAY-07	P0003	50	100	75	F
O10002	03-JUN-07	P0002	35	50	80	F
O10003	03-JUN-07	P0001	120	20	150	P
O10004	15-JUN-07	P0005	450	2	700	F
O10005	30-JUN-07	P0005	450	10	700	P
O10006	29-JUN-07	P0004	145	15	200	P
O10007	20-JUN-07	P0004	145	10	200	P
O10008	03-JUL-07	P0002	35	15	80	F
O10009	09-AUG-07	P0002	35	20	80	F
O10010	09-AUG-07	P0001	120	10	150	F
O10011	09-AUG-07	P0005	450	5	700	P

Table Name: ProductDemand

Column Name	Data Type	Size
DateSold	Date	
ProductNo	Varchar2	10
Qty	Number	5

Table Name: ProductSales

Column Name	Data Type	Size
DateSold	Date	
ProductNo	Varchar2	10
TotalSales	Number	12, 2

Insert Data into ProductDemand and ProductSales tables from OrderMaster table using Unconditional INSERT Technique:

```
INSERT ALL
  INTO ProductDemand VALUES (DateSold, ProductNo, Qty)
  INTO ProductSales VALUES (DateSold, ProductNo, TotalSales)
  SELECT TRUNC(OrderDate) DateSold, ProductNo,
      SUM(SellPrice * Qty) TotalSales, SUM(Qty) Qty FROM OrderMaster
          GROUP BY TRUNC(OrderDate), ProductNo;
```

Output:

```
22 rows created.
```

Test:

ProductDemand and ProductSales now hold the following data:

Solution:

```
SELECT * FROM ProductDemand;
```

Output:

```
DATESOLD  PRODUCTNO         QTY
--------- ---------- ----------
09-AUG-07 P0002              20
09-AUG-07 P0001              10
30-MAY-07 P0003             100
03-JUN-07 P0002              50
03-JUN-07 P0001              20
30-JUN-07 P0005              10
15-JUN-07 P0005               2
29-JUN-07 P0004              15
20-JUN-07 P0004              10
03-JUL-07 P0002              15
09-AUG-07 P0005               5

11 rows selected.
```

Solution:

```
SELECT * FROM ProductSales;
```

Output:

```
DATESOLD  PRODUCTNO  TOTALSALES
--------- ---------- ----------
09-AUG-07 P0002            1600
```

```
09-AUG-07 P0001            1500
30-MAY-07 P0003            7500
03-JUN-07 P0002            4000
03-JUN-07 P0001            3000
30-JUN-07 P0005            7000
15-JUN-07 P0005            1400
29-JUN-07 P0004            3000
20-JUN-07 P0004            2000
03-JUL-07 P0002            1200
09-AUG-07 P0005            3500

11 rows selected.
```

REMINDER

Be informed about the following:

- ❑ The VALUES clause, can only contain columns referenced in the SELECT statement in the subquery
- ❑ If the VALUES clause is not included, the SELECT statement in the subquery will define the values to be inserted
- ❑ ALL indicates evaluation of all WHEN clauses. For each WHEN clause that evaluates to true, Oracle processes the corresponding INTO clause

Explanation:

The above example:

- ❑ Inserts OrderDate, ProductNo and Qty into the **ProductDemand** table
- ❑ Inserts OrderDate, ProductNo and TotalSales into the **ProductSales** table

The values are obtained through a subquery retrieving the desired data from the **OrderMaster** table.

Example of Conditional INSERT FIRST

The following table structures are used in the examples that follow:

Table Name: PriorityProducts

Column Name	**Data Type**	**Size**
OrderTotal	Number	12, 2
OrderNo	Varchar2	10
ProductNo	Varchar2	10

Table Name: SpecialProducts

Column Name	Data Type	Size
OrderTotal	Number	12, 2
OrderNo	Varchar2	10
ProductNo	Varchar2	10

Table Name: EconomicProducts

Column Name	Data Type	Size
OrderTotal	Number	12, 2
OrderNo	Varchar2	10
ProductNo	Varchar2	10

Solution:

```
INSERT FIRST
   WHEN OrderTotal < 1500
        THEN INTO EconomicProducts VALUES (OrderTotal, OrderNo, ProductNo)
   WHEN (OrderTotal > 1500 AND OrderTotal < 3000)
        THEN INTO PriorityProducts VALUES (OrderTotal, OrderNo, ProductNo)
   WHEN OrderTotal > 3000
        THEN INTO SpecialProducts VALUES (OrderTotal, OrderNo, ProductNo)
   SELECT (Qty * SellPrice) OrderTotal, OrderNo, ProductNo FROM OrderMaster;
```

Output:

```
8 rows created.
```

Test:

EconomicProducts, PriorityProducts and SpecialProducts now hold the following data:

Solution:

```
SELECT * FROM EconomicProducts;
```

Output:

```
ORDERTOTAL ORDERNO    PRODUCTNO
---------- ---------- ----------
      1400 O10004     P0005
      1200 O10008     P0002
```

Solution:

```
SELECT * FROM PriorityProducts;
```

Output:

```
ORDERTOTAL ORDERNO    PRODUCTNO
---------- ---------- ----------
      2000 O10007     P0004
      1600 O10009     P0002
```

Solution:

```
SELECT * FROM SpecialProducts;
```

Output:

```
ORDERTOTAL ORDERNO    PRODUCTNO
---------- ---------- ----------
      7500 O10001     P0003
      4000 O10002     P0002
      7000 O10005     P0005
      3500 O10011     P0005
```

REMINDER

Be informed about the following:

- The VALUES clause can only contain columns referenced in the SELECT statement in the subquery
- If the VALUES clause is not included, the SELECT statement in the subquery will define the values to be inserted
- The INSERT FIRST will execute the FIRST INTO clause that matches on the expression. It will, then, skip all following WHEN clauses

REMINDER

The same table can be specified for multiple target INTO clauses.

A multi table INSERT statement, can have up to 127 WHEN clauses in the statement.

Explanation:

The above example:

- Inserts OrderTotal, OrderNo, ProductNo into the **EconomicProducts** table

  If OrderTotal is less than 1500
- Inserts OrderTotal, OrderNo, ProductNo into the **PriorityProducts** table

  If OrderTotal is greater than 1500 but is less than 3000

- Inserts OrderTotal, OrderNo, ProductNo into the **SpecialProducts** table

  If OrderTotal is greater than 3000

The values are obtained through a subquery retrieving the desired data from the OrderMaster table. The INSERT FIRST simply indicates that it will execute the FIRST INTO clause that matches on the expression. It will, then, skip all following WHEN clauses.

Example of Conditional INSERT

The following table structures are used in the examples that follow:

Table Name: PriorityProductsSales

Column Name	Data Type	Size	Attribute
ProductNo	Number	10	
CostPrice	Number	12, 2	

Table Name: RegularProductsSales

Column Name	Data Type	Size	Attribute
ProductNo	Number	10	
CostPrice	Number	12, 2	

Solution:

```
INSERT ALL
  WHEN ProductNo IN(SELECT ProductNo FROM PriorityProducts) THEN
      INTO PriorityProductsSales VALUES(ProductNo, CostPrice)
  WHEN OrderStatus = 'F' THEN
      INTO RegularProductsSales VALUES (ProductNo, CostPrice)
  SELECT ProductNo, CostPrice, OrderStatus FROM OrderMaster;
```

Output:

```
11 rows created.
```

Test:

PriorityProductsSales and RegularProductsSales now hold the following data:

Solution:

```
SELECT * FROM PriorityProductsSales;
```

Output:

```
PRODUCTNO   COSTPRICE
---------- ----------
P0002              35
P0004             145
P0004             145
P0002              35
P0002              35
```

Solution:

```
SELECT * FROM RegularProductsSales;
```

Output:

```
PRODUCTNO   COSTPRICE
---------- ----------
P0003              50
P0002              35
P0005             450
P0002              35
P0002              35
P0001             120

6 rows selected.
```

REMINDER

Be informed about the following:

- ❑ The VALUES clause can only contain columns referenced in the SELECT statement in the subquery
- ❑ If the VALUES clause is not included, the SELECT statement in the subquery will define the values to be inserted
- ❑ ALL indicates evaluation of all WHEN clauses. For each WHEN clause that evaluates to true, Oracle will process the corresponding INTO clause

Explanation:

The above example:

- ❑ Inserts ProductNo, CostPrice into the **PriorityProductsSales** table

  If an entry exists in the **PriorityProducts** table

- Inserts ProductNo, CostPrice into the **RegularProductsSales** table

  If the **OrderStatus** column holds the value **F**

The values are obtained through a subquery retrieving the desired data from the OrderMaster table. The INSERT ALL simply indicates evaluation of all WHEN clauses. For each WHEN clause that evaluates to true, Oracle will process the corresponding INTO clause.

Multi-Table Restrictions

The following are a few restrictions on using multi-table INSERTS.

Multi-table inserts**:**

- Cannot be used with views and materialized. Only tables are supported
- Cannot be performed on a remote table
- Cannot use a collection expression
- Cannot be parallelized in a Real Application Cluster environment
- Cannot use plan stability
- Cannot use a sequence in a subquery