SECTION II: BASICS OF SQL

Multi-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:

```sql
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:

```sql
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**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
<tr>
<td>OrderDate</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>ProductNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
<tr>
<td>CostPrice</td>
<td>Number</td>
<td>12, 2</td>
</tr>
<tr>
<td>Qty</td>
<td>Number</td>
<td>5</td>
</tr>
<tr>
<td>SellPrice</td>
<td>Number</td>
<td>12, 2</td>
</tr>
<tr>
<td>OrderStatus</td>
<td>Char</td>
<td>2</td>
</tr>
</tbody>
</table>

This table holds the following data:

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

**Table Name: ProductDemand**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateSold</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>ProductNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
<tr>
<td>Qty</td>
<td>Number</td>
<td>5</td>
</tr>
</tbody>
</table>

**Table Name: ProductSales**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateSold</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>ProductNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
<tr>
<td>TotalSales</td>
<td>Number</td>
<td>12, 2</td>
</tr>
</tbody>
</table>
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
```

 Output:
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**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderTotal</td>
<td>Number</td>
<td>12, 2</td>
</tr>
<tr>
<td>OrderNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
<tr>
<td>ProductNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
</tbody>
</table>
Table Name: SpecialProducts

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderTotal</td>
<td>Number</td>
<td>12, 2</td>
</tr>
<tr>
<td>OrderNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
<tr>
<td>ProductNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
</tbody>
</table>

Table Name: EconomicProducts

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderTotal</td>
<td>Number</td>
<td>12, 2</td>
</tr>
<tr>
<td>OrderNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
<tr>
<td>ProductNo</td>
<td>Varchar2</td>
<td>10</td>
</tr>
</tbody>
</table>

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        010004    P0005
1200        010008    P0002
```

Solution:

```
SELECT * FROM PriorityProducts;
```
Solution:

```
SELECT * FROM SpecialProducts;
```

Output:

<table>
<thead>
<tr>
<th>ORDERTOTAL</th>
<th>ORDERNO</th>
<th>PRODUCTNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7500</td>
<td>010001</td>
<td>P0003</td>
</tr>
<tr>
<td>4000</td>
<td>010002</td>
<td>P0002</td>
</tr>
<tr>
<td>7000</td>
<td>010005</td>
<td>P0005</td>
</tr>
<tr>
<td>3500</td>
<td>010011</td>
<td>P0005</td>
</tr>
</tbody>
</table>

**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 \textit{SpecialProducts} table

\hspace{1.5cm} \textbf{If OrderTotal is greater than 3000}

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

\hspace{1.5cm} \textbf{Example of Conditional INSERT}

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

<table>
<thead>
<tr>
<th>Table Name: PriorityProductsSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
</tr>
<tr>
<td>ProductNo</td>
</tr>
<tr>
<td>CostPrice</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table Name: RegularProductsSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
</tr>
<tr>
<td>ProductNo</td>
</tr>
<tr>
<td>CostPrice</td>
</tr>
</tbody>
</table>

\hspace{1.5cm} \textbf{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;
```

\hspace{1.5cm} \textbf{Output:}

```
11 rows created.
```

\hspace{1.5cm} \textbf{Test:}

\hspace{1.5cm} \textbf{PriorityProductsSales and RegularProductsSales now hold the following data:}

\hspace{1.5cm} \textbf{Solution:}

```
SELECT * FROM PriorityProductsSales;
```
Output:

<table>
<thead>
<tr>
<th>PRODUCTNO</th>
<th>COSTPRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0002</td>
<td>35</td>
</tr>
<tr>
<td>P0004</td>
<td>145</td>
</tr>
<tr>
<td>P0004</td>
<td>145</td>
</tr>
<tr>
<td>P0002</td>
<td>35</td>
</tr>
<tr>
<td>P0002</td>
<td>35</td>
</tr>
</tbody>
</table>

Solution:

```sql
SELECT * FROM RegularProductsSales;
```

Output:

<table>
<thead>
<tr>
<th>PRODUCTNO</th>
<th>COSTPRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0003</td>
<td>50</td>
</tr>
<tr>
<td>P0002</td>
<td>35</td>
</tr>
<tr>
<td>P0005</td>
<td>450</td>
</tr>
<tr>
<td>P0002</td>
<td>35</td>
</tr>
<tr>
<td>P0002</td>
<td>35</td>
</tr>
<tr>
<td>P0001</td>
<td>120</td>
</tr>
</tbody>
</table>

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
Mutli-Table Insert Statements

- 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