Chapter 13: Multitable INSERTs and MERGE Statement

Multitable INSERT Operations

Overview

Multitable INSERT statements allow you to insert data into multiple tables from a single subquery. This feature provides an efficient way to distribute data across tables in a single operation.

Key Requirements:

- The statement must include a subquery
- Two types available: Unconditional and Conditional

Unconditional Multitable INSERT

Syntax

```sql
INSERT ALL
    INTO table1 (col_list1) VALUES (val_list1)
    INTO table2 (col_list2) VALUES (val_list2)
    ...
subquery;
```

Key Points

- The keyword `ALL` is mandatory (unless using WHEN conditions)
- Every row returned by the subquery is inserted into all specified tables
- Multiple INTO clauses are allowed
- Each INTO can specify different VALUES, including expressions
- If VALUES list is omitted, column list from subquery must match target table
- Values can be transformed before insertion (e.g., SYSDATE, mathematical operations, literals)

Conditional Multitable INSERT

Syntax

sql

```sql
INSERT [ALL | FIRST]
    WHEN condition1 THEN INTO table1 VALUES (...)
    WHEN condition2 THEN INTO table2 VALUES (...)
    ...
    ELSE INTO table3 VALUES (...)
subquery;
```

Key Points

- `ALL` (default): evaluates all WHEN clauses
- `FIRST`: executes only the first true WHEN clause and skips the rest
- `ELSE` (optional): used if none of the WHEN conditions are true
- Each WHEN condition is evaluated per row of the subquery
- Multiple INTO clauses can follow a single WHEN
- ELSE must be last and also followed by its own INTO

Best Practices and Common Issues

General Guidelines

- Subquery must be a standalone SELECT
- Table aliases in subqueries are not visible in WHEN or VALUES clauses
    - Use column aliases if needed
- You cannot use a view as the target of a multitable INSERT—only base tables

Sequence Generators (NEXTVAL)

- Using NEXTVAL in subquery = syntax error
- Using NEXTVAL in VALUES clause is allowed but:
    - Increments once per subquery row, not per target table
    - Executes even if WHEN evaluates to false
- Avoid using the same sequence in multiple INTOs within one INSERT

Pivoting with Multitable INSERT

Conditional multitable INSERTs can pivot data (turn columns into rows). This is useful for converting spreadsheet-like data into normalized tables.

Example

```sql
INSERT ALL
    WHEN OCEAN IS NOT NULL THEN
        INTO stats (room_type, window_type, sq_ft) VALUES (room_type, 'OCEAN', ocean)
    WHEN BALCONY IS NOT NULL THEN
        INTO stats (room_type, window_type, sq_ft) VALUES (room_type, 'BALCONY', balcony)
    ...
SELECT ...
FROM source_table;
```

Exam Tips for Multitable INSERT

- ✅ Multitable INSERTs always require a subquery
- ❌ Do not use table aliases from subqueries outside the subquery
- ❌ Do not omit ALL in unconditional inserts
- ❗ Any failure in any INTO clause = entire statement fails and rolls back

MERGE Statement

Overview

The MERGE statement combines INSERT, UPDATE, and limited DELETE functionality into one DML statement. It provides more efficiency than multiple separate DMLs through a single pass through the data.

Syntax

```sql
MERGE INTO target_table
USING source_table_or_subquery
ON (condition)
WHEN MATCHED THEN
    UPDATE SET col = expr
    DELETE WHERE condition
WHEN NOT MATCHED THEN
    INSERT (cols) VALUES (exprs)
WHERE condition;
```

Clause Functions

1. **INTO clause**: Specifies the target table (where to insert/update). Required.
2. **USING clause**: Source of new data. Can be a table, view, or subquery. Required.
3. **ON clause**: Join logic that compares source to target. Required.
4. **WHEN MATCHED clause**:
   - UPDATE SET modifies existing rows in the target
   - Optional DELETE WHERE removes updated rows matching the delete condition
5. **WHEN NOT MATCHED clause**:
   - INSERT new rows into target when no match is found
6. **WHERE clause**:
   - Optional filter on source data after USING but before any action is taken

Key Rules and Notes

- You cannot use UPDATE or DELETE directly in the ON condition
- DELETE only affects rows that were updated, not those inserted
- MERGE cannot delete rows unless they are also matched and updated
- Efficient for distributed systems and ETL-like operations

Example Use Case

Merge ONTARIO_ORDERS into WWA_INVOICES:

```sql
MERGE INTO WWA_INVOICES WWA
USING ONTARIO_ORDERS ONT
ON (WWA.CUST_PO = ONT.PO_NUM)
WHEN MATCHED THEN
    UPDATE SET WWA.NOTES = ONT.SALES_REP
WHEN NOT MATCHED THEN
    INSERT (INV_ID, CUST_PO, INV_DATE, NOTES)
    VALUES (SEQ_INV_ID.NEXTVAL, ONT.PO_NUM, SYSDATE, ONT.SALES_REP)
WHERE SUBSTR(ONT.PO_NUM, 1, 3) <> 'NBC';
```

Important Considerations

- USING can be based on subqueries
- DELETE clause syntax example:

```sql
DELETE WHERE target_col < TO_DATE('01-SEP-09')
```

After MERGE Execution (based on example)

- If CUST_PO = WWA-001 → row updated
- If CUST_PO = WWA-017 → row inserted
- If PO_NUM starts with NBC → row skipped due to final WHERE

Summary

- MERGE combines INSERT + UPDATE, and sometimes DELETE operations
- Syntax order, clause rules, and logical flow are critical for exam
- Understand how to filter, match, and act based on ON and WHERE conditions