Oracle12*c*: SQL

Chapter 12
Subqueries and Merge Statements

Objectives

- Determine when using a subquery is appropriate
- Identify which clauses can contain subqueries
- Distinguish between an outer query and a subquery
- Use a single-row subquery in a WHERE clause
- Use a single-row subquery in a HAVING clause
- Use a single-row subquery in a SELECT clause

Objectives (continued)

- Distinguish between single-row and multiple-row comparison operators
- Use a multiple-row subquery in a WHERE clause
- Use a multiple-row subquery in a HAVING clause
- Use a multiple-column subquery in a WHERE clause

Objectives (continued)

- Create an inline view using a multiple-column subquery in a FROM clause
- Compensate for NULL values in subqueries
- Distinguish between correlated and uncorrelated subqueries
- Nest a subquery inside another subquery
- Use a subquery in a DML action
- Process multiple DML actions with a MERGE statement

Subqueries and Their Uses

- Subquery – a query nested inside another query
- Used when a query is based on an unknown value
- Requires SELECT and FROM clauses
- Must be enclosed in parentheses
- Place on right side of comparison operator

Types of Subqueries

SUBQUERY	DESCRIPTION
Single-row subquery	Returns to the outer query one row of results that consists of one column
Multiple-row subquery	Returns to the outer query more than one row of results
Multiple-column subquery	Returns to the outer query more than one column of results
Correlated subquery	References a column in the outer query, and executes the subquery once for every row in the outer query
Uncorrelated subquery	Executes the subquery first and passes the value to the outer query

Single-Row Subqueries

- Can only return one result to the outer query
- Operators include =, >, <, >=, <=, < >

Single-Row Subquery in a WHERE Clause

- Used for comparison against individual data

Enter SQL Statement:

```
SELECT category, title, cost
 FROM books
 WHERE cost >
       (SELECT cost
        FROM books
        WHERE title = 'DATABASE IMPLEMENTATION')
   AND category = 'COMPUTER'
```

Results | Script Output | Explain | Autotrace | DBMS Output

Results:

	CATEGORY	TITLE	COST
1	COMPUTER	HOLY GRAIL OF ORACLE	47.25
2	COMPUTER	E-BUSINESS THE EASY WAY	37.9

Single-Row Subquery in a HAVING Clause

- Required when returned value is compared to grouped data

Enter SQL Statement:

```
SELECT category, AVG(retail-cost) "Average Profit"
 FROM books
 GROUP BY category
 HAVING AVG(retail-cost) > (SELECT AVG(retail-cost)
                            FROM books
                            WHERE category = 'LITERATURE');
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	CATEGORY	Average Profit
1	COMPUTER	18.2625
2	FAMILY LIFE	24.875

Single-Row Subquery in a SELECT Clause

- Replicates subquery value for each row displayed

Enter SQL Statement:

```
SELECT title, retail,
              (SELECT TO_CHAR(AVG(retail),999.99)
               FROM books) "Overall Average"
FROM books;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	TITLE	RETAIL	Overall Average
1	BODYBUILD IN 10 MINUTES A DAY	30.95	40.98
2	REVENGE OF MICKEY	22	40.98
3	BUILDING A CAR WITH TOOTHPICKS	59.95	40.98
4	DATABASE IMPLEMENTATION	55.95	40.98
5	COOKING WITH MUSHROOMS	19.95	40.98
6	HOLY GRAIL OF ORACLE	75.95	40.98
7	HANDCRANKED COMPUTERS	25	40.98
8	E-BUSINESS THE EASY WAY	54.5	40.98
9	PAINLESS CHILD-REARING	89.95	40.98
10	THE WOK WAY TO COOK	28.75	40.98
11	BIG BEAR AND LITTLE DOVE	8.95	40.98
12	HOW TO GET FASTER PIZZA	29.95	40.98
13	HOW TO MANAGE THE MANAGER	31.95	40.98
14	SHORTEST POEMS	39.95	40.98

Multiple-Row Subqueries

- Return more than one row of results
- Require use of IN, ANY, ALL, or EXISTS operators

ANY and ALL Operators

- Combine with arithmetic operators

OPERATOR	DESCRIPTION
>ALL	More than the highest value returned by the subquery
<ALL	Less than the lowest value returned by the subquery
<ANY	Less than the highest value returned by the subquery
>ANY	More than the lowest value returned by the subquery
=ANY	Equal to any value returned by the subquery (same as IN)

FIGURE 12-11 Descriptions of ALL and ANY operator combinations

Multiple-Row Subquery in a WHERE Clause

Enter SQL Statement:

```
SELECT title, retail, category
 FROM books
 WHERE retail IN (SELECT MAX(retail)
                  FROM books
                  GROUP BY category)
 ORDER BY category;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	TITLE	RETAIL	CATEGORY
1	HOW TO MANAGE THE MANAGER	31.95	BUSINESS
2	BUILDING A CAR WITH TOOTHPICKS	59.95	CHILDREN
3	HOLY GRAIL OF ORACLE	75.95	COMPUTER
4	THE WOK WAY TO COOK	28.75	COOKING
5	PAINLESS CHILD-REARING	89.95	FAMILY LIFE
6	BODYBUILD IN 10 MINUTES A DAY	30.95	FITNESS
7	SHORTEST POEMS	39.95	LITERATURE
8	HOW TO GET FASTER PIZZA	29.95	SELF HELP

Note: Could use IN operator or =ANY

Multiple-Row Subquery in a WHERE Clause (continued)

Enter SQL Statement:

```
SELECT title, retail
 FROM books
 WHERE retail <ANY (SELECT retail
                    FROM books
                    WHERE category = 'COOKING');
```

Results | Script Output | Explain | Autotrace | DBMS Output

Results:

	TITLE	RETAIL
1	BIG BEAR AND LITTLE DOVE	8.95
2	COOKING WITH MUSHROOMS	19.95
3	REVENGE OF MICKEY	22
4	HANDCRANKED COMPUTERS	25

Multiple-Row Subquery in a HAVING Clause

Enter SQL Statement:

```
SELECT order#, SUM(quantity*paideach)
 FROM orderitems
 HAVING SUM(quantity*paideach) >ALL (SELECT SUM(quantity*paideach)
                                     FROM customers JOIN orders USING (customer#)
                                      JOIN orderitems USING (order#)
                                     WHERE state = 'FL'
                                     GROUP BY order#)
 GROUP BY order#;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	ORDER#	SUM(QUANTITY*PAIDEACH)
1	1001	117.4
2	1002	111.9
3	1007	335.85
4	1004	170.9
5	1012	166.4

Multiple-Column Subqueries

- Return more than one column in results
- Can return more than one row
- Column list on the left side of operator must be in parentheses
- Use the IN operator for WHERE and HAVING clauses

Multiple-Column Subquery in a FROM Clause

- Creates a temporary table

Enter SQL Statement:

```
SELECT b.title, b.retail, a.category, a.cataverage
 FROM books b, (SELECT category, AVG(retail) cataverage
                  FROM books
                  GROUP BY category) a
 WHERE b.category = a.category
   AND b.retail > a.cataverage;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	TITLE	RETAIL	CATEGORY	CATAVERAGE
1	E-BUSINESS THE EASY WAY	54.5	COMPUTER	52.85
2	HOLY GRAIL OF ORACLE	75.95	COMPUTER	52.85
3	DATABASE IMPLEMENTATION	55.95	COMPUTER	52.85
4	THE WOK WAY TO COOK	28.75	COOKING	24.35
5	BUILDING A CAR WITH TOOTHPICKS	59.95	CHILDREN	34.45
6	PAINLESS CHILD-REARING	89.95	FAMILY LIFE	55.975

Multiple-Column Subquery in a WHERE Clause

- Returns multiple columns for evaluation

Enter SQL Statement:

```
SELECT title, retail, category
 FROM books
 WHERE (category, retail) IN (SELECT category, MAX(retail)
                              FROM books
                              GROUP BY category)
 ORDER BY category;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	TITLE	RETAIL	CATEGORY
1	HOW TO MANAGE THE MANAGER	31.95	BUSINESS
2	BUILDING A CAR WITH TOOTHPICKS	59.95	CHILDREN
3	HOLY GRAIL OF ORACLE	75.95	COMPUTER
4	THE WOK WAY TO COOK	28.75	COOKING
5	PAINLESS CHILD-REARING	89.95	FAMILY LIFE
6	BODYBUILD IN 10 MINUTES A DAY	30.95	FITNESS
7	SHORTEST POEMS	39.95	LITERATURE
8	HOW TO GET FASTER PIZZA	29.95	SELF HELP

NULL Values

- When a subquery might return NULL values, use NVL function

Enter SQL Statement:

```
SELECT customer#
 FROM customers
 WHERE NVL(referred, 0) = (SELECT NVL(referred, 0)
                           FROM customers
                           WHERE customer# = 1005);
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	CUSTOMER#
1	1001
2	1002
3	1003
4	1004
5	1005
6	1006
7	1008
8	1010
9	1011
10	1012
11	1014
12	1015
13	1017
14	1018
15	1020

Uncorrelated Subqueries

- Processing sequence
  - Inner query is executed first
  - Result is passed to outer query
  - Outer query is executed

Correlated Subqueries

- Inner query is executed once for each row processed by the outer query
- Inner query references the row contained in the outer query

Correlated Subqueries (continued)

Enter SQL Statement:

```
SELECT title
 FROM books
 WHERE EXISTS (SELECT isbn
               FROM orderitems
               WHERE books.isbn = orderitems.isbn);
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

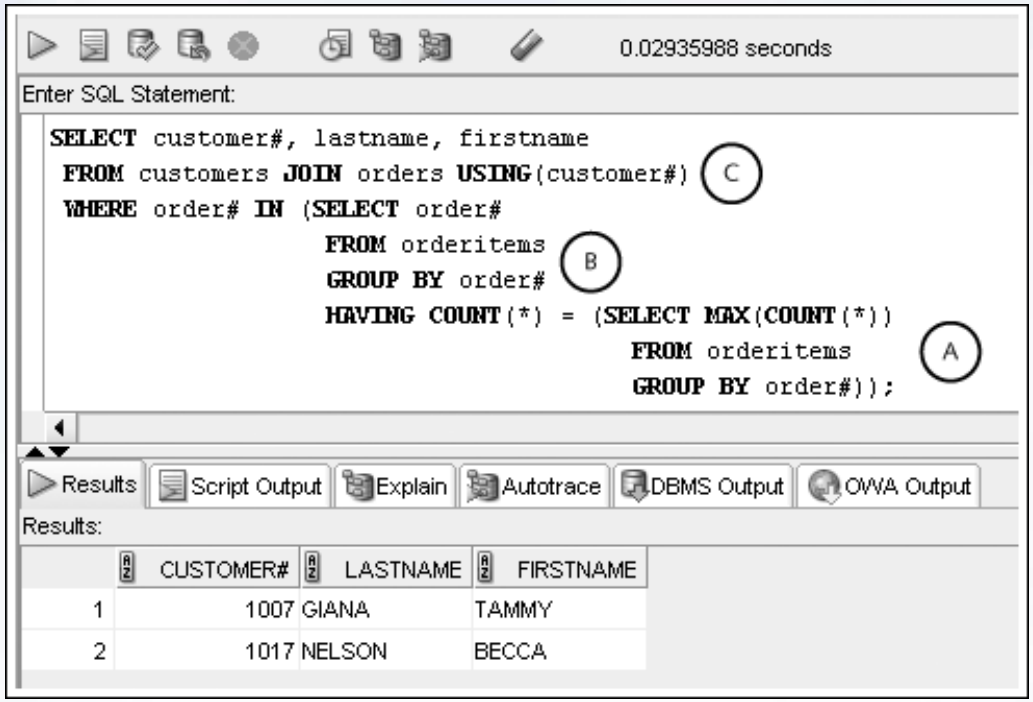
	TITLE
1	COOKING WITH MUSHROOMS
2	HOW TO MANAGE THE MANAGER
3	PAINLESS CHILD-REARING
4	DATABASE IMPLEMENTATION
5	BODYBUILD IN 10 MINUTES A DAY
6	SHORTEST POEMS
7	E-BUSINESS THE EASY WAY
8	HOLY GRAIL OF ORACLE
9	BIG BEAR AND LITTLE DOVE
10	REVENGE OF MICKEY
11	HANDCRANKED COMPUTERS

Nested Subqueries

- Maximum of 255 subqueries if nested in the WHERE clause
- No limit if nested in the FROM clause
- Innermost subquery is resolved first, then the next level, etc.

Nested Subqueries (continued)

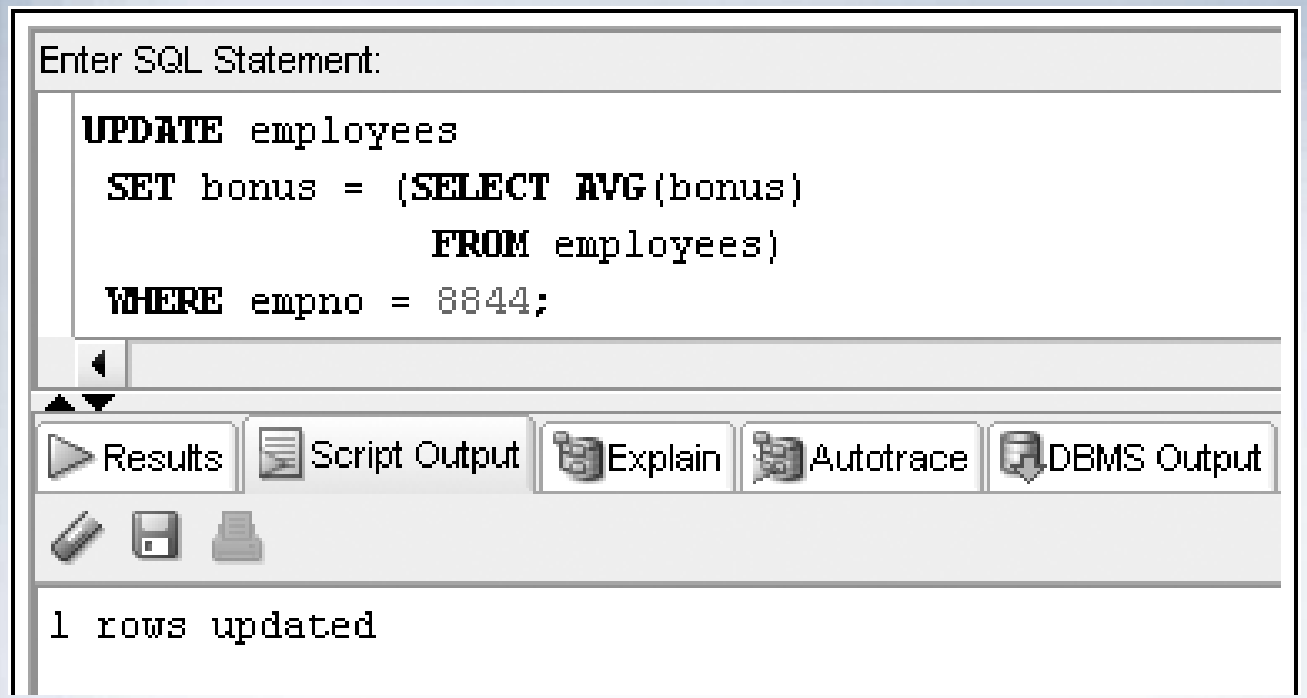
- Innermost is resolved first (A), then the second level (B), then the outer query (C)



Subquery Factoring Clause

```
WITH dcount AS (
  SELECT deptno, COUNT(*) AS dcount
  FROM   employees
  GROUP BY deptno)
SELECT e.lname Emp_Lastname,
       e.deptno e_dept,
       d1.dcount edept_count,
       m.lname manager_name,
       m.deptno mdept,
       d2.dcount mdept_count
FROM   employees e,
       dcount d1,
       employees m,
       dcount d2
WHERE  e.deptno = d1.deptno
AND    e.mgr = m.empno
AND    m.deptno = d2.deptno
   AND e.mgr = '7839';
```

Subquery in a DML action



MERGE Statement

- With a MERGE statement, a series of DML actions can occur with a single SQL statement
- Conditionally updates one data source based on another

MERGE Statement (continued)

Enter SQL Statement:

```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category);

SELECT *
 FROM books_1;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
5 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL               CATEGORY
---------- ------------------------------ ------------------------- -------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                 29.95                COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                 25                   COMPUTER
0299282519 THE WOK WAY TO COOK            11-SEP-00                 28.75                COOKING

5 rows selected
```

No change

Retail updated

Retail and Category updated

Row added

MERGE Statement (continued)

- The following explains each part of the previous MERGE statement:
  - MERGE INTO books_1 a: The BOOKS_1 table is to be changed and a table alias of “a” is assigned to this table
  - USING books_2 b: The BOOKS_2 table will provide the data to update and/or insert into BOOKS_1 and a table alias of “b” is assigned to this table
  - ON (a.isbn = b.isbn): The rows of the two tables will be joined or matched based on isbn
  - WHEN MATCHED THEN: If a row match based on ISBN is discovered, execute the UPDATE action in this clause. The UPDATE action instructs the system to modify only two columns (Retail and Category)
  - WHEN NOT MATCHED THEN: If no match is found based on the ISBN (a books exists in BOOKS_2 that is not in BOOKS_1), then perform the INSERT action in this clause

MERGE with WHERE conditions

Enter SQL Statement:

```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
   WHERE b.category  = 'COMPUTER'
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category)
    WHERE b.category = 'COMPUTER';

SELECT *
 FROM books_1;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
3 rows merged
ISBN       TITLE                          PUBDATE                  RETAIL               CATEGORY
---------- ------------------------------ ------------------------ -------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                55.95                COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                19.95                COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                75.95                COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                25                   COMPUTER

4 rows selected
```

MERGE with DELETE

Enter SQL Statement:

```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
  DELETE WHERE (b.retail < 50);

SELECT *
 FROM books_1;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
3 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL               CATEGORY
---------- ------------------------------ ------------------------- -------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                COMPUTER
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                COMPUTER

2 rows selected
```

Summary

- A subquery is a complete query nested in the SELECT, FROM, HAVING, or WHERE clause of another query
  - The subquery must be enclosed in parentheses and have a SELECT and a FROM clause, at a minimum
- Subqueries are completed first; the result of the subquery is used as input for the outer query
- A single-row subquery can return a maximum of one value
- Single-row operators include =, >, <, >=, <=, and <>
- Multiple-row subqueries return more than one row of results

Summary (continued)

- Operators that can be used with multiple-row subqueries include IN, ALL, ANY, and EXISTS
- Multiple-column subqueries return more than one column to the outer query
- NULL values returned by a multiple-row or multiple-column subquery will not present a problem if the IN or =ANY operator is used
- Correlated subqueries reference a column contained in the outer query
- Subqueries can be nested to a maximum depth of 255 subqueries in the WHERE clause of the parent query

Summary (continued)

- With nested subqueries, the innermost subquery is executed first, then the next highest level subquery is executed, and so on, until the outermost query is reached
- A MERGE statement allows multiple DML actions to be conditionally performed while comparing data of two tables