

Chapter 7 Subqueries

Subquery Concepts

- A SELECT statement nested within the FROM, WHERE, or HAVING clause of another SQL statement

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT  select_list
         FROM    table);
```

- The subquery (inner query) executes **once**, **before** the main query
- The **result** of the subquery is passed to the main query (outer query)

2

Subquery Concepts

- Subqueries are an alternative to running a **series** of queries where the result of one query is the input for another
 - subqueries are useful when a query is based on unknown values
- Incremental development
 - what work should the subquery do?
- Which writers have same freelancer status as K Cox?

```
SQL> SELECT freelancer
      FROM writer
      WHERE writerid = 'C200';
```

```
SQL> SELECT ln, fn, freelancer
      FROM writer
      WHERE freelancer = 'Y';
```

3

Simple Subquery

- Which writers have the same freelancer status as Kelly Cox?

```
SQL> SELECT ln, fn, freelancer
       FROM writer
       WHERE freelancer = 'Y'
         (SELECT freelancer
          FROM writer
          WHERE writerid = 'C200')
```

LN	FN	F
Cox	Kelly	Y
Epstein	Diane	Y
Johnson	Leroy W.	Y
Kim	Chong	Y
Nilsson	Tonya	Y
Seeger	Wilhelm	Y

- Note: the **outer query's** columns are displayed

4

Simple Subquery Practice

- Show title, type and length for articles that are the same type as 'Dot-Coms Go Bust'

5

Types of Subqueries

- Scalar subquery



- Multiple-row subquery



- Multiple-column subquery



6

Guidelines for Using Subqueries

- Enclose a subquery in parentheses
- Place subquery to the right of a comparison operator
- Do not use an ORDER BY clause within a subquery
- Use a **single-row** operator with a single-row subquery
 - = < > <>
- Use a **multiple-row** operator with a multi-row subquery
 - IN ANY/SOME ALL

7

IN Operator (review...)

- Condition is True if field's value equals any of the values in a **comma separated list**

```
SQL> SELECT title, type, length
      FROM article
      WHERE type IN('LAW', 'BUS');
```

TITLE	TYP	LENGTH
The Supreme Court's Business Attitude	LAW	1773
Trans-Alaskan Oil Pipeline Opening	BUS	803
25% Tax Cut Bill Approved	LAW	1515
AT&T Antitrust Settlement	BUS	1600
Building Trade Outlook	BUS	1437
You and the Pension Reform Act	LAW	1698
...		

Subqueries Returning Multiple Rows

- Use a multi-row operator to handle results of a multi-row subquery (IN, ANY/SOME, ALL)
- What are the names of the writers who serve as contacts?

```
SQL> SELECT writerid, ln, fn
      FROM writer
      WHERE writerid IN(SELECT contact FROM writer);
```

WRIT LN	FN
L350 Lawton	Pat
W432 Waldeck	Kristine

- Note that IN() autodistincts and suppresses nulls,

9

Subqueries Returning Multiple Rows

- Use a multi-row operator to handle results of a multi-row subquery (IN, ANY/SOME, ALL)
- What are the names of the writers who serve as contacts?

```
SQL> SELECT writerid, ln, fn
       FROM writer
       WHERE writerid = ANY(SELECT contact FROM writer);
```

WRIT	LN	FN
L350	Lawton	Pat
W432	Waldeck	Kristine

10

Subqueries Returning Multiple Rows

- Use a multi-row operator to handle results of a multi-row subquery (IN, ANY/SOME, ALL)
- Which students are not enrolled in any section?

```
SQL> SELECT last_name, first_name
       FROM student
       WHERE student_id NOT IN(SELECT student_id
                               FROM enrollment);
```

LAST_NAME	FIRST_NAME
Lindeman	Salewa
Sikinger	Paul
Kelly	Robin
...	

11

Multi-Row Subquery Practice

- Warm Up:

```
SQL> descr scott.emp
SQL> SELECT *
       FROM scott.emp
```

- Show the name, salary and department number of each employee who earns less than the average salary.

12

Subqueries Returning Multiple Columns

- Use parentheses in the WHERE clause to designate the columns
- For each section, show the student with the highest project grade (pg. 336)

```
SQL> SELECT student_id, section_id, numeric_grade
FROM grade
WHERE grade_type_code = 'PJ'
AND (section_id, numeric_grade) IN
    (SELECT section_id, max(numeric_grade)
     FROM grade
     WHERE grade_type_code = 'PJ'
     GROUP BY section_id)
```

Correlated Subquery Concepts

- Used when the subquery needs a value from each row processed by the outer query
- The inner query refers to column(s) from a table in outer query
 - inner query is executed once for **each row** in the outer query
- Example: Show the name, salary, and department number of each employee who earns less than **their department's average salary**
 - incremental development: we need each department's average salary

```
SQL> SELECT ename, sal, deptno
FROM scott.emp outer
WHERE sal < (SELECT AVG(sal)
            FROM scott.emp inner
            WHERE inner.deptno = outer.deptno)
```

ENAME	SAL	DEPTNO
SMITH	800	20
ALLEN	1600	10
WARD	1250	30
MARTIN	1250	30
CLARK	2450	10
TURNER	1500	30
ADAMS	1100	20
JAMES	950	30
MILLER	1300	10

- Although serviceable, this correlated subquery is *inefficient* since it calculates each department's average salary for each employee in the department.

Correlated Subquery Example

- Which sections have an enrollment that exceeds the capacity of the section? (p359)

correlated subquery solution

```
SQL> SELECT section_id, count(*)
FROM enrollment e
GROUP BY section_id
HAVING COUNT(*) >
    (SELECT capacity FROM section s
     WHERE s.section_id = e.section_id)
```

SECTION_ID	COUNT(*)
101	12

Equijoin Solution

- Which sections have an enrollment that exceeds the capacity of the section?

```
SQL> SELECT e.section_id, COUNT(*), s.capacity
       FROM enrollment e, section s
       WHERE e.section_id = s.section_id
       GROUP BY e.section_id, s.capacity
       HAVING COUNT(*) > s.capacity
```

equijoin solution

SECTION_ID	COUNT(*)	CAPACITY
101	12	10

- An equijoin (inner join) can return columns from any of the joined tables

16

EXISTS Operator

- Used to check whether a **correlated subquery** returns any rows
 - returns True if the subquery returns at least one row
 - returns False if the subquery doesn't return any rows
- Column(s) listed in the subquery's SELECT clause are irrelevant
- Show the description of each type of article present in the article table

```
SQL> SELECT descr
       FROM type t
       WHERE EXISTS
         (SELECT 'X'
          FROM article a
          WHERE t.type = a.type)
```

17

NOT EXISTS Operator

- Used to check whether a **correlated subquery** returns any rows
 - returns True if the subquery doesn't return any rows
 - returns False if the subquery returns at least one row
- Column(s) in the subquery's SELECT clause are irrelevant
- Show the description of each type of article not present in the article table

```
SQL> SELECT descr
       FROM type t
       WHERE NOT EXISTS
         (SELECT 'X'
          FROM article a
          WHERE t.type = a.type)
```

18

ANY/SOME Operators

- Compares a value to each value in a list
- Condition is True if it is satisfied by any of the values produced by the subquery
 - returns False when the subquery returns no rows
- Must be preceded by =, !=, >, <, <=, >=
 - = ANY is equivalent to IN
 - < ANY means less than the maximum
 - > ANY means more than the minimum
- Which articles were written before any of W. Seeger's articles?

```
SQL> SELECT title, issue
FROM article
WHERE issue < ANY(SELECT issue FROM article WHERE writerid='S260')
```

```
SQL> SELECT title, issue
FROM article
WHERE issue < (SELECT MAX(issue) FROM article
WHERE writerid='S260')
```

19

ALL Operator

- Compares a value to each value in a list
- The condition is True only if it is satisfied by all values produced by the subquery
 - returns True if the subquery returns no rows
- Must be preceded by =, !=, >, <, <=, >=
 - > ALL means more than the maximum
 - < ALL mean less than the minimum
- Which articles are longer than all of the articles by W Seeger?

```
SQL> SELECT title, length
FROM article
WHERE length>ALL(SELECT length FROM article WHERE writerid='S260')
```

```
SQL> SELECT title, length
FROM article
WHERE length>(SELECT MAX(length) FROM article WHERE writerid='S260')
```

20

Inline View Concepts

- A subquery in the FROM clause
- Used to break complex problems into simpler parts
 - incremental development
- Creates a **virtual table**
 - results are **calculated on-the-fly** and are held in memory while statement is processed
 - is not a persistent schema object (like a **base table** is)
- Can have multiple inline views embedded within an SQL statement
- An inline view can have an alias that you use to qualify a reference any of its columns

21

Inline View Example #1

- Show each writer's name and number of articles written
 - incremental development

Author	Articles Written
Johnson, Leroy W.	4
Hall, Valerie	3
Leavitt, Morris	3
Kim, Chong	3
...	

```
SQL> SELECT w.ln || ', ' || w.fn "Author",
           a.writer_art "Articles Written"
FROM       writer w,
           (SELECT writerid, COUNT(*) writer_art
            FROM article
            GROUP by writerid) a
WHERE      w.writerid = a.writerid
ORDER BY 2 DESC
```

22

Inline View Example #2

- Show each writer's name, number of articles written, and their percentage of the total articles

Author	Articles Written	%
Johnson, Leroy W.	4	13.33
Hall, Valerie	3	10
Leavitt, Morris	3	10
Kim, Chong	3	10
Seeger, Wilhelm	3	10
Ngo, Thuy	2	6.67
Waldeck, Kristine	2	6.67
Nilsson, Tonya	2	6.67
Cox, Kelly	1	3.33
Martinez, Maria	1	3.33
...		
Epstein, Diane	1	3.33

23

Inline View Example #2

- Show each writer's name, number of articles written, and their percentage of the total articles

```
SQL> SELECT w.ln || ', ' || w.fn "Author",
           a.writer_art "Articles Written",
           round(100*(a.writer_art/b.total_art),2) "%"
FROM       writer w,
           (SELECT writerid, COUNT(*) writer_art
            FROM article
            GROUP BY writerid) a,
           (SELECT COUNT(*) total_art
            FROM article) b
WHERE      w.writerid = a.writerid
ORDER BY 2 DESC;
```

- Notice: inline view **b** is joined neither to **a** nor **w**
 - so... what happens?

24

Inline View Example #3

- Show the name and salary of each employee who earns less than their department's average salary
 - incremental development

- This inline view solution is more efficient than the correlated subquery solution since the inline view calculates each department's average salary only once.

ENAME	SAL	DEPTNO	DEPT_AVG_SAL
SMITH	800	20	2175
ALLEN	1600	10	2587.5
MARTIN	1250	30	1637.5
CLARK	2450	10	2587.5
TURNER	1500	30	1637.5
ADAMS	1100	20	2175
JAMES	950	30	1637.5
MILLER	1300	10	2587.5

```
SQL> SELECT e.ename, e.sal, e.deptno, iv.dept_avg_sal
       FROM scott.emp e,
       (SELECT AVG(sal) dept_avg_sal, deptno
        FROM scott.emp
        GROUP BY deptno) iv
       WHERE e.sal < iv.dept_avg_sal
       AND e.deptno = iv.deptno;
```

25

ROWNUM Pseudocolumn

- Assigns a sequential value to each row returned by a query

```
SQL> SELECT ROWNUM, title, length
       FROM article;
```

ROWNUM	TITLE	LENGTH
1	The Supreme Court's Business Attitude	1773
2	Wages and Prices Holding Steady	1429
3	U.S. Plans Gas Rationing	1068
4	Milton Friedman Interview	1994
5	Trans-Alaskan Oil Pipeline Opening	803
6	Farm Aid Abuses	1866
7	Chrysler Asks U.S. For \$1 Billion	975
...		

26

Top-N Query Concepts

- By default, SELECT queries return all rows meeting the specified criteria (WHERE or HAVING)
- Top-N queries return only those rows which contain the *n* largest (or smallest) column values
 - eg: What are the 5 *shortest* articles?

27

Structure of a Top-N Query

```
SELECT [* | column_list, ROWNUM]
FROM (SELECT column_list FROM table
      ORDER BY Top-N_column)
WHERE ROWNUM <= N
```

- The inline view selects and sorts the rows
- The outer query uses ROWNUM to limit the rows to the top-N
 - what if there are ties at the cut-off point?

28

Top-N Query

- What are the 5 shortest articles?

```
SQL> SELECT ROWNUM, title, length
FROM (SELECT title, length
      FROM article
      ORDER BY length)
WHERE ROWNUM <=5;
```

ROWNUM	TITLE	LENGTH
1	Trans-Alaskan Oil Pipeline Opening	803
2	Chrysler Asks U.S. For \$1 Billion	975
3	The Economy Under Sub-Zero Population Growth	1020
4	U.S. Plans Gas Rationing	1068
5	Cola Advertising War	1096

29

Top-N Query

- What are the 5 longest articles?

```
SQL> SELECT *
FROM (SELECT title, length
      FROM article
      ORDER BY length DESC)
WHERE ROWNUM <=5;
```

TITLE	LENGTH
Bush Finally Wins!	6533
Advertising Over the Past 25 Years	3285
Law Over the Past 25 Years	2834
The BCCI Scandal	2779
Stock Market's Black Monday	2395

- Notice: outer query uses * and does not display ROWNUM

30

Top-N Practice

- Show the 4 most heavily used locations (i.e., the locations that are used by the most sections).

LOCATION	# of Sections
L509	25
L214	15
L507	15
L210	10

- Modify the previous query to show only the 2 most heavily used locations.

31

Top-N Query using RANK

- Use the RANK function to handle ties at the cut-off point

- ROWNUM mechanism doesn't detect ties

```
SQL>SELECT location, COUNT(*) num_sections,  
         RANK() OVER (ORDER BY COUNT(*) DESC) ranking  
FROM section  
GROUP BY location;
```

```
SQL>SELECT *  
FROM (SELECT location, COUNT(*) num_sections,  
         RANK() OVER (ORDER BY COUNT(*) DESC) ranking  
      FROM section  
      GROUP BY location)  
WHERE ranking <= 2;
```

32
