Murach Chapter 6 Part 2

How to Code Subqueries

Week 5, Week 6

Knowledge Points in this lecture

- Subquery in WHERE clause, ALL, ANY/SOME
- Correlated subquery, EXISTS, NOT EXISTS
- Inline View
- Subquery factoring

ALL, ANY, SOME and Subquery

- ALL (subquery) – all rows in the result set of the subquery
- ANY (subquery) – any row in the result set of the subquery
- SOME is equivalent to ANY
- value > ALL (subquery)
  - Value > Maximum in the list of results returned by the subquery
- value > ANY (subquery)
  - Value > Minimum in the list of results returned by the subquery
- value > SOME (subquery)
  - Value > Minimum in the list of results returned by the subquery

ALL, ANY, SOME and Subquery

- Examples: The following are all true
  - 20000 > ALL (116, 1083, 14092, 14092, 14092,14092)
  - 200 > ANY (116, 1083, 14092, 14092, 14092,14092)
  - 200 > SOME (116, 1083, 14092, 14092, 14092,14092)

The value returned by the subquery

```
1669.906
```

The result set

	INVOICE_NUMBER	INVOICE_DATE	BALANCE_DUE
1	31359783	23-MAY-14	1575
2	97/553	27-APR-14	904.14
3	I77271-001	05-JUN-14	662
4	31361833	23-MAY-14	579.42
5	9982771	03-JUN-14	503.2
6	97/553B	26-APR-14	313.55

```
(33 rows)
```

A query that uses ALL

```
SELECT vendor_name, invoice_number, invoice_total
FROM invoices i JOIN vendors v
     ON i.vendor_id = v.vendor_id
WHERE invoice_total > ALL
     (SELECT invoice_total
     FROM invoices
     WHERE vendor_id = 34)
ORDER BY vendor_name
```

The result of the subquery

	INVOICE_TOTAL
1	116.54
2	1083.58

The result set

	VENDOR_NAME	INVOICE_NUMBER	INVOICE_TOTAL
1	Bertelsmann Industry Svcs. Inc	509786	6940.25
2	Cahners Publishing Company	587056	2184.5
3	Computerworld	367447	2433
4	Data Reproductions Corp	40318	21842

(25 rows)

A query that uses ANY/SOME

```
SELECT vendor_name, invoice_number, invoice_total
FROM invoices i JOIN vendors v
     ON i.vendor_id = v.vendor_id
WHERE invoice_total > SOME
     (SELECT invoice_total
     FROM invoices
     WHERE vendor_id = 34)
ORDER BY vendor_name
```

The result of the subquery

	INVOICE_TOTAL
1	116.54
2	1083.58

The result set

	VENDOR_NAME	INVOICE_NUMBER	INVOICE_TOTAL
1	Bertelsmann Industry Svcs. Inc	509786	6940.25
2	Blue Cross	547480102	224
3	Blue Cross	547481328	224
4	Cahners Publishing Company	587056	2184.5
5	Cardinal Business Media, Inc.	133560	175
6	Compuserve	456789	8344.5

(64 rows)

Correlated Subquery

- Correlated subquery executes once for each row processed by the outer query.
- Non-correlated subquery executed only once for all rows processed by the outer query.

A query that uses a correlated subquery

```
SELECT vendor_id, invoice_number, invoice_total
FROM invoices inv_main
WHERE invoice_total >
     (SELECT AVG(invoice_total)
     FROM invoices inv_sub
     WHERE inv_sub.vendor_id = inv_main.vendor_id)
ORDER BY vendor_id, invoice_total
```

The value returned by the subquery for vendor 95

```
28.50166...
```

The result set

	VENDOR_ID	INVOICE_NUMBER	INVOICE_TOTAL
6	83	31359783	1575
7	95	111-92R-10095	32.7
8	95	111-92R-10093	39.77
9	95	111-92R-10092	46.21
10	110	P-0259	26881.4

(36 rows)

The syntax of a subquery with EXISTS operator

```
WHERE [NOT] EXISTS (subquery)
```

A query that returns vendors without invoices

```
SELECT vendor_id, vendor_name, vendor_state
FROM vendors
WHERE NOT EXISTS
    (SELECT *
    FROM invoices
    WHERE invoices.vendor_id = vendors.vendor_id)
```

The result set

	VENDOR_ID	VENDOR_NAME	VENDOR_STATE
53	33	Nielson	OH
54	35	Cal State Termite	CA
55	36	Graylift	CA
56	38	Venture Communications Int'l	NY
57	39	Custom Printing Company	MO
58	40	Nat Assoc of College Stores	OH

(88 rows)

EXISTS, NOT EXISTS

- Use EXISIT to test if 1 or N rows are returned by the subquery.
  - The subquery returns just an indication of any rows satisfy the specified condition.
- Use NOT EXISTS to test if no rows are returned by the subquery.
- Typically use SELECT * in the subquery, because it does not matter what columns are listed in SELECT clause of the subquery.
- EXISTS operator is mostly used in correlated subqueries.

Inline View

- Inline view
  - The result set of a subquery in FROM clause
  - Most useful when you need to further summarize the results of a summary query.
- Must assign a table alias to the inline view.
- Must assign column alias to calculated values in subquery.

A query that uses an inline view

```
SELECT i.vendor_id,
    MAX(invoice_date) AS last_invoice_date,
    AVG(invoice_total) AS average_invoice_total
FROM invoices i JOIN
    (
    SELECT vendor_id,
        AVG(invoice_total) AS average_invoice_total
    FROM invoices
    HAVING AVG(invoice_total) > 4900
    GROUP BY vendor_id
    ) v
    ON i.vendor_id = v.vendor_id
GROUP BY i.vendor_id
ORDER BY MAX(invoice_date) DESC
```

The result of the subquery (an inline view)

	VENDOR_ID	AVERAGE_INVOICE_TOTAL
1	72	10963.655
2	99	6940.25
3	104	7125.34
4	110	23978.482
5	119	4901.26

The result set

	VENDOR_ID	LAST_INVOICE_DATE	AVERAGE_INVOICE_TOTAL
1	72	18-JUL-14	10963.655
2	119	04-JUN-14	4901.26
3	104	03-JUN-14	7125.34
4	99	31-MAY-14	6940.25
5	110	08-MAY-14	23978.482

Subquery Factoring

Subquery factoring

- Use a WITH clause to name the result set of a subquery, then use the named result set as a temporary table in the main query
- Can be used to simplify writing a complex query

Subquery Factoring

WITH Clause Syntax:

```
WITH rs_name AS (subquery_sql_text)
mainquery_sql_text;
```

Meaning:

The result set of the subquery: `subquery_sql_text` is named as `rs_name`.

`rs_name` can be used in the main query: `mainquery_sql_text`.

Subquery Factoring Example

```
WITH vendor_invoice_summary AS
(
    SELECT vendor_id, AVG(invoice_total) AS average_invoice_total
    FROM invoices
    GROUP BY vendor_id
    HAVING AVG(invoice_total) > 4900
)
SELECT i.vendor_id, MAX(invoice_date) AS last_invoice_date,
     AVG(invoice_total) AS average_invoice_total
FROM invoices i JOIN vendor_invoice_summary v
        ON i.vendor_id = v.vendor_id
GROUP BY i.vendor_id
ORDER BY MAX(invoice_date) DESC;
```

Result Set:

	VENDOR_ID	LAST_INVOICE_DATE	AVERAGE_INVOICE_TOTAL
1	34	30-AUG-14	9595.08
2	72	18-JUL-14	10963.655
3	119	04-JUN-14	4901.26
4	104	03-JUN-14	7125.34
5	99	31-MAY-14	6940.25
6	110	08-MAY-14	23978.482

A procedure for building complex queries

1. State the problem to be solved in English.
2. Use pseudocode to outline the query.
3. If necessary, use pseudocode to outline each subquery.
4. Code the subqueries and test them.
5. Code and test the final query.