SQL Nesting

Business Data Management and Analytics

Contributions by Arthur Adamopoulos and Vince Bruno

Nested Queries – introduction

- Let's write a query to find mobile phones whose owner is from WERRIBEE. We will use the IN keyword and a list of customerIDs (assuming we already know who they are):

```
SELECT mobileID, phonenumber, joined, cancelled
FROM mobile
WHERE customerID  IN (20004, 20164, 20318, 20546, 21010, 21500);
```

- This is OK but not very flexible.
- What if a new customer from WERRIBEE was added ?

Nested Queries

- If we have customerIDs in the CUSTOMER table, we can get them instead of the hard coded list.
- We replace the hard coded list with a nested (inner) query.
- The inner query is enclosed in brackets.
- The inner most query is executed first before the outer ones are executed.

Nested Queries - example

```
SELECT mobileID, phonenumber, joined, cancelled
FROM mobile
WHERE customerid IN (SELECT customerid
                     FROM customer
                     WHERE suburb = 'WERRIBEE');
```

MobileID	PhoneNumber	Joined	Cancelled
4502	410717359	1999-06-03	2001-01-24
4583	413497522	1998-09-12	<NULL>
4650	410289237	1999-01-13	<NULL>
4651	411894122	1998-12-05	2000-03-16
4774	413604204	1998-12-06	<NULL>
4775	410197562	1999-01-28	<NULL>
5006	412563658	1999-02-03	<NULL>

- Could also be done using a join in SQL:

```
SELECT mobileID, phonenumber, joined, cancelled
FROM mobile m, customer c
WHERE m.customerid = c.customerid AND suburb = 'WERRIBEE';
```

Nested Queries – execution order

- Execution of nested query:
  1. inner sub-query is processed, producing value(s).

  2. outer query uses the resulting values of inner sub-query in its execution.

Nested Queries – more examples

- Show youngest customer(s) name and date of birth:

```
SELECT surname, given, dob
FROM customer
WHERE dob=  (SELECT MAX(dob)
                FROM customer);
```

Surname	Given	DOB
DIGHTON	STEPHEN FRANCIS	2002-12-30

Note: Can we do the above query with a join ?

Nested Queries – more examples

- Show all customers that do not have a mobile phone as yet.

```
SELECT surname, given
FROM   customer
WHERE customerID NOT IN
            (SELECT DISTINCT customerID
             FROM mobile);
```

Surname	Given
BINDEVIS	CATHERINE RAE
QUAH	VICKI
TAYLOR	AMIT
COATH	JON

Note: Can we do the above query with a join?

Correlated Sub-Query

- This is also a nested query with one difference. The nested query refers to a field in an outer query.
- Since a reference is made to an outer query field, the query can NOT be executed once before outer query.

Correlated Sub-Query - example

- The inner query refers to a field from the outer query:

```
SELECT surname, given, phonenumber, joined, cancelled
FROM customer c, mobile m
WHERE c.customerid = m.customerid
AND 3 > (SELECT count(*)
         FROM calls
         WHERE calls.mobileid = m.mobileid);
```

Same Table

Less than 3 calls

38 rows returned

Surname	Given	PhoneNumber	Joined	Cancelled
SPENCER	BENJAMIN G...	413881812	1998-02-17	<NULL>
FINDLAY	STEVEN AN...	411801933	1999-06-03	<NULL>
PARSONS	DAVID GOR...	411212499	1998-10-06	<NULL>
SHIPARD	MAUREEN	413605507	1999-04-26	<NULL>

Correlated Sub-Query - execution

- Execution of correlated queries:

1. For each record in the outer query:
   A) execute inner sub-query
   B) produce result
   C) use result in the execution of the outer query

Correlated Sub-Query - example

- The previous sub-correlated query could be done using a JOIN and GROUP BY/HAVING. Differences?

```
SELECT surname, given, m.phonenumber, joined, cancelled
FROM customer c, mobile m, calls
WHERE c.customerid = m.customerid
AND calls.mobileid = m.mobileid
GROUP BY surname, given, m.phonenumber, joined, cancelled
HAVING count(*) < 3;
```

33 rows returned

Surname	Given	PhoneNumber	Joined	Cancelled	count(*)
BROWN	MUK MIKE	413114309	1997-11-13	<NULL>	1
CHEN	MING-RU	412795481	1998-01-01	<NULL>	1
CHUI	TECK CHENG	412883436	1998-08-15	<NULL>	2
D CRUZ	WAI KEY	413228193	1999-03-13	1999-12-09	2

Nested Queries

- Show mobiles that have made more than 50 calls in 2006:

```
SELECT mobileID, phonenumber
FROM mobile
WHERE 50 < (SELECT count(*) FROM calls
                    WHERE mobile.mobileid = calls.mobileID
                    AND year(calldate) = 2006);
```

NOTE: Can also use EXISTS, NOT EXISTS, IN, NOT IN, "=" instead of "<".

EXISTS and Sub-Queries

- Check if the sub-query returns any answer.
- NOT version also possible

```
SELECT *
FROM plan
WHERE exists (SELECT *
                    FROM mobile
                    WHERE mobile.planname = plan.planname);
```

Views, Grouping and Nesting

- Lets look at showing how popular the phone colours are.

```
SELECT phonecolour, count(*)
    FROM mobile
    GROUP BY phonecolour;
```

- How can we show the MOST popular or LEAST popular?

PhoneColour	count(*)
Azzuro	44
Black	123
Blue	55
Brown	56
Gold	56
Green	61
Grey	101
KittyKat	53
Pink	70
Purple	58
Rainbow	64
Red	62
Silver	64
Tiger	67
Transparent	61
Unknown	47
White	54
Yellow	55

Views, Grouping and Nesting

- Place colour summary into a view:

```
CREATE VIEW phonecolour AS
    SELECT phonecolour as colour, count(*) as num_phones
    FROM mobile
    GROUP BY phonecolour;
```

- The MOST popular?

```
SELECT colour, num_phones
    FROM phonecolour
    WHERE num_phones =
      (SELECT max(num_phones) from phonecolour);
```

colour	num_phones
Black	123

Query Build

- List ALL the mobile phone plans showing the count of mobiles on them.

SELECT planname, connectFee

FROM plan;

PlanName	ConnectFee
Yes10	1.00
Yes20	1.20
Yes30	1.50
Yes40	1.75
Weekender	2.50
FreeStyle	3.95

- Add the count and GROUP BY and COUNT(*):

```
SELECT p.planname, connectFee, count(*)
FROM plan p, mobile m
WHERE p.planname = m.planname
GROUP BY planname, connectFee;
```

PlanName	ConnectFee	count(*)
FreeStyle	3.95	246
Yes10	1.00	240
Yes20	1.20	214
Yes30	1.50	230
Yes40	1.75	221

Note: Missing a plan?

Missing Alternatives - 1

- Including Missing plan can be done in various ways:

```
SELECT planname, connectFee
FROM plan
WHERE not exists (SELECT planname FROM mobile
                  WHERE mobile.planname = plan.planname);
```

OR

```
SELECT planname, connectFee
FROM plan
WHERE planname not in (SELECT planname FROM mobile);
```

Missing Alternatives - 2

- Including Missing plan can be done in various ways:

```
SELECT p.planname, connectFee
FROM plan p
LEFT JOIN mobile m ON p.planname = m.planname
WHERE m.planname is NULL;
```

OR

```
SELECT planname, connectFee
FROM plan
WHERE 0 = (SELECT count(*) FROM mobile
                    WHERE mobile.planname = plan.planname);
```

Query Solution - 1

- UNION solution

```
SELECT p.planname, connectFee, count(*)
FROM plan p, mobile m
WHERE p.planname = m.planname
GROUP BY planname, connectFee
UNION
SELECT planname, connectFee, 0
FROM plan
WHERE planname not in (SELECT planname FROM mobile);
```

planname	connectFee	count(*)
FreeStyle	3.95	246
Yes10	1.00	240
Yes20	1.20	214
Yes30	1.50	230
Yes40	1.75	221
Weekender	2.50	0

Query Solution - 2

- OUTER JOIN solution or not?

```
SELECT p.planname, connectFee, count(*)
FROM plan p
LEFT JOIN mobile m ON p.planname = m.planname
GROUP BY planname, connectFee;
```

PlanName	ConnectFee	count(*)
FreeStyle	3.95	246
Weekender	2.50	1
Yes10	1.00	240
Yes20	1.20	214
Yes30	1.50	230
Yes40	1.75	221

Summary

- Two types of Nesting that can occur
  - Nested – not referring to field in outside query
  - Sub-Correlated – which is referring to fields in outside query
- Placement of nested queries, anywhere in a query. The nested query MUST return an appropriate value(s) to work in a given situation. For example:
  - FROM – query must return a table
  - WHERE – query must return a value or list of values (IN)