Nested Queries and Performance Issues in SQL
ITM 354

Nested Queries are queries that contain another complete SELECT statements nested within it, that is, in the WHERE clause. The nested SELECT statement is called an "inner query" or an "inner SELECT." The main query is called "outer SELECT" or "outer query." Many nested queries are equivalent to a simple query using JOIN operation. The use of nested query in this case is to avoid explicit coding of JOIN which is a very expensive database operation and to improve query performance. However, in many cases, the use of nested queries is necessary and cannot be replaced by a JOIN operation.

I. Nested queries that can be expressed using JOIN operations:

Example 1: (Library DB Query A) How many copies of the book titled the lost tribe are owned by the library branch whose name is "Sharptown"?

Single Block Query Using Join:

```
SELECT  No_Of_Copies
     FROM    BOOK_COPIES, BOOK, LIBRARY_BRANCH
     WHERE  BOOK_COPIES.BranchId = LIBRARY_BRANCH.BranchId  AND
                BOOK_COPIES.BookId = BOOK.BookId        AND
                BOOK.Title = "The Lost Tribe"   AND
                LIBRARY_BRANCH.BranchName = "Sharpstown"
```

Using Nested Queries:

```
SELECT  No_Of_Copies
FROM     BOOK_COPIES
 WHERE      BranchID  IN
                (SELECT  BranchID  from LIBRARY_BRANCH   WHERE
                LIBRARY_BRANCH.BranchName = "Sharpstown")
  AND          BookID IN
                 (SELECT  BookID from  BOOK  WHERE
                  BOOK.Title = "The Lost Tribe" )
```

Performance considerations: The nested queries in this example involves simpler and faster operations. Each subquery will be executed once and then a simple select operation will be performed. On the other hands, the operations using join require Cartesian products of three tables and have to evaluate 2 join conditions and 2 selection conditions. Nested queries in this example also save internal temporary memory space for holding Cartesian join results.

==========

Rule of thumb:

1) ***Correlated queries*** *where the inner query references some attribute of a relation declared in the outer query and use the" =" or IN operators.*
2) *Conversely, if the attributes in the projection operation of a single block query that joins several tables are from only one table, this query can always be translated into a nested query.*

==========

Example 2: see Query 12 and Query 12A
Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

Single Block query using JOIN operation

```
select A.fname, A.lname
from employee A, dependent B
where A.ssn = B.essn  and
        A.sex = B.sex and A.fname = B.dependent_name
```

Correlated Query:

```
select  A.fname, A.lname
from  employee A
where A.ssn  IN (SELECT essn
                 FROM dependent
                 WHERE  essn = A.ssn  and  dependent_name = A.fname and sex = A.sex)
```

Computer Procedures:
Conceptually, think of this query as stepping through EMPLOYEE table one row at a time, and then executing the inner query each time. The first row has A.fname = "John" and A.sex = "M" so that the inner query becomes **SELECT Essn FROM dependent where essn = 12345678, dependent_name = "John" and sex = "M"**; The first run of the subquery returns nothing so it continues to proceed to the next tuple and executes the inner query again with the values of A.SSN, A.fname and A.sex for the second row, and so on for all rows of EMPLOYEE.
The term *correlated subquery* is used because its value depends on a variable (or variables) that receives its value from an outer query (e.g., A.SSN, A.fname, A.sex in this example; they are called **correlation variables**.). A correlated subquery thus cannot be evaluated once and for all. It must be evaluated repeatedly -- once for each value of the variable received from the outer query. This is different from non-correlated subqueries explained below.

Non-correlated Subquery:
A non-correlated subquery needs to be evaluated only once. For example:
Query EMP-NQ2: find an employee that has the highest salary of the company.

```
SELECT fname, lname, bdate
FROM  EMPLOYEE
WHERE salary =  (SELECT max (salary) FROM Employee);
```

Here inner query returns a value: 55000. The inner query will be executed first and only *once* and then the entire query becomes

```
SELECT fname, lname, bdate
FROM  EMPLOYEE
WHERE salary =  55000;
```

II. Nested Queries that cannot be directly translated into Join Operations

Rule of thumb:

1) Unknown selection criteria: WHERE clause examines unknown value.
For example shown above (Query EMP-NQ2): find everybody in a department which has an employee that has the highest salary of the company.
Another example in section 7.2.5. finds employees who has salary higher than the highest salary in Department 5.

```
SELECT ssn, salary, dno
```

from Employee
where salary > (SELECT max (salary) from employee where dno = 5);

2) Relational **set** operations such as Division or other comparison that involves EXISTS, NOT EXISTS, > , etc. (This may involve using paradox SET operation operators, such as NO, ONLY, EXACTLY and EVERY.)

3) Outer Join that involves Null value operations. This is the equivalent of using NOT EXISTS. (See *SQL solution for queries on Library DB*: query C and C').

III. General Discussion on SQL query formulation: (See page 212)

There are many ways to specify the same query in SQL. This flexibility in specifying queries has advantage and disadvantages.

- Advantage: You can choose a way to express the query that you prefer. **It is general preferable to write a query with as little nesting and implied ordering as possible.**
- Disadvantages:
  1. the user may be confused
  2. user may have the burden to figure out which way is more efficient due to different DBMS query optimization strategies. (Performance issues.)

Correlated and Non-correlated Subqueries

Write SQL statements for the following queries on the Company Database and determine whether it's a correlated or non-correlated query. (Please translate your SQL single-block join, if applicable, to subqueries.)

Tip: the term *correlated subquery* is used because its value depends on a variable (or variables) that receives its value from an outer query (e.g., A.SSN, A.fname, A.sex in the example shown in the previous handout; they are called **correlation variables**.). A correlated subquery thus cannot be evaluated once and for all. It must be evaluated repeatedly -- once for each value of the variable received from the outer query. A non-correlated subquery needs to be evaluated only once.

1. Retrieve the names of employees whose salary is higher than the average salary of all female employees.

2. Find the names of all employees who have a supervisor with the same birthday as theirs.

3. Find the names of all employees who are directly supervised by "Franklin Wong."

4. Find the location of the department which has the employee of the highest pay in the company.