Introduction to Data Management
CSE 344

Lecture 7: Nested Queries in SQL

Announcements

- Today's office hour in CSE 218 (Vaspol), 3:30 – 4:20 pm

Lecture Goals

- Today we will learn how to write more powerful SQL queries

- They are needed in Homework 3

- Reading: Ch. 6.3

Subqueries

- A subquery is a SQL query nested inside a larger query

- Such inner-outer queries are called nested queries

- A subquery may occur in:
  - A SELECT clause
  - A FROM clause
  - A WHERE clause

- Rule of thumb: avoid writing nested queries when possible; keep in mind that sometimes it's impossible

1. Subqueries in SELECT

Product (<u>pname</u>, price, cid)
Company(<u>cid</u>, cname, city)

For each product return the city where it is manufactured

```
SELECT X.pname, (SELECT Y.city
                 FROM Company Y
                 WHERE Y.cid=X.cid) as City
FROM  Product X
```

1. Subqueries in SELECT

Product (pname, price, cid)
Company(cid, cname, city)

For each product return the city where it is manufactured

```
SELECT X.pname, (SELECT Y.city
                        FROM Company Y
                        WHERE Y.cid=X.cid) as City
FROM  Product X
```

What happens if the subquery returns more than one city ?

1. Subqueries in SELECT

Product (pname, price, cid)
Company(cid, cname, city)

For each product return the city where it is manufactured

```
SELECT X.pname, (SELECT Y.city
                 FROM Company Y
                 WHERE Y.cid=X.cid) as City
FROM  Product X
```

What happens if the subquery returns more than one city ?

We get a runtime error
(SQLite simply ignores the extra values)

1. Subqueries in SELECT

Product (<u>pname</u>, price, cid)
Company(<u>cid</u>, cname, city)

For each product return the city where it is manufactured

SELECT X.pname, (SELECT Y.city
                 FROM Company Y
                 WHERE Y.cid=X.cid) as City
FROM Product X

"correlated subquery"

What happens if the subquery returns more than one city ?

We get a runtime error
(SQLite simply ignores the extra values)

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

Whenever possible, don't use nested queries:

```
SELECT X.pname, (SELECT Y.city
                 FROM Company Y
                 WHERE Y.cid=X.cid) as City
FROM  Product X
```

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

Whenever possible, don't use nested queries:

```
SELECT X.pname, (SELECT Y.city
                 FROM Company Y
                 WHERE Y.cid=X.cid) as City
FROM  Product X
```

=

```
SELECT X.pname, Y.city
FROM   Product X, Company Y
WHERE X.cid=Y.cid
```

We have "unnested" the query

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

Compute the number of products made by each company

```
SELECT DISTINCT C.cname, (SELECT count(*)
                          FROM Product P
                          WHERE P.cid=C.cid)
FROM  Company C
```

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

Compute the number of products made by each company

```
SELECT DISTINCT C.cname, (SELECT count(*)
                          FROM Product P
                          WHERE P.cid=C.cid)
FROM  Company C
```

Better: we can
unnest by using
a GROUP BY

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

Compute the number of products made by each company

```
SELECT DISTINCT C.cname, (SELECT count(*)
                          FROM Product P
                          WHERE P.cid=C.cid)
FROM  Company C
```

Better: we can unnest by using a GROUP BY

```
SELECT C.cname, count(*)
FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname
```

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

But are these really equivalent?

SELECT DISTINCT C.cname, (SELECT count(*)
                                            FROM Product P
                                            WHERE P.cid=C.cid)

FROM Company C

SELECT C.cname, count(*)
FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

But are these really equivalent?

```
SELECT DISTINCT C.cname, (SELECT count(*)
                          FROM Product P
                          WHERE P.cid=C.cid)

FROM  Company C
```

```
SELECT C.cname, count(*)
FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname
```

No! Different results if a
company has no products

Product (pname, price, cid)
Company(cid, cname, city)

1. Subqueries in SELECT

But are these really equivalent?

```
SELECT DISTINCT C.cname, (SELECT count(*)
                          FROM Product P
                          WHERE P.cid=C.cid)
FROM  Company C
```

```
SELECT C.cname, count(*)
FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname
```

No! Different results if a company has no products

```
SELECT C.cname, count(pname)
FROM Company C LEFT OUTER JOIN Product P
ON C.cid=P.cid
GROUP BY C.cname
```

Product (pname, price, cid)
Company(cid, cname, city)

2. Subqueries in FROM

Find all products whose prices is > 20 and < 500

```
SELECT X.pname
FROM (SELECT * FROM Product AS Y WHERE price > 20) as X
WHERE X.price < 500
```

Unnest this query !

2. Subqueries in FROM

- At the end of the lecture we will see that sometimes we really need a subquery and one option will be to put it in the FROM clause (see "finding witnesses").

Product (pname, price, cid)
Company(cid, cname, city)

3. Subqueries in WHERE

Find all companies that make <u>some</u> products with price < 200

Existential quantifiers

Using EXISTS:

```
SELECT DISTINCT  C.cname
FROM    Company C
WHERE  EXISTS (SELECT *
                FROM Product P
                WHERE C.cid = P.cid and P.price < 200)
```

Product (pname, price, cid)
Company(cid, cname, city)

3. Subqueries in WHERE

Find all companies that make <u>some</u> products with price < 200

Existential quantifiers

Using IN

```
SELECT DISTINCT  C.cname
FROM     Company C
WHERE C.cid IN (SELECT P.cid
                FROM Product P
                WHERE P.price < 200)
```

Product (pname, price, cid)
Company(cid, cname, city)

3. Subqueries in WHERE

Find all companies that make <u>some</u> products with price < 200

Existential quantifiers

Using ANY:

```
SELECT DISTINCT  C.cname
FROM     Company C
WHERE 200 > ANY (SELECT price
                  FROM Product P
                  WHERE P.cid = C.cid)
```

Product (<u>pname</u>, price, cid)
Company(<u>cid</u>, cname, city)

3. Subqueries in WHERE

Find all companies that make <u>some</u> products with price < 200

Existential quantifiers

Now let's unnest it:

```
SELECT DISTINCT  C.cname
FROM    Company C, Product P
WHERE  C.cid= P.cid and P.price < 200
```

Existential quantifiers are easy ! ☺

Product (pname, price, cid)
Company(cid, cname, city)

3. Subqueries in WHERE

Find all companies s.t. <u>all</u> their products have price < 200

same as:

Find all companies that make <u>only</u> products with price < 200

Universal quantifiers

Universal quantifiers are hard ! ☹

Product (pname, price, cid)
Company(cid, cname, city)

3. Subqueries in WHERE

Find all companies s.t. all their products have price < 200

1. Find *the other* companies: i.e. s.t. some product ≥ 200

```
SELECT DISTINCT  C.cname
FROM     Company C
WHERE  C.cid IN (SELECT P.cid
                 FROM Product P
                 WHERE P.price >= 200)
```

2. Find all companies s.t. all their products have price < 200

```
SELECT DISTINCT  C.cname
FROM     Company C
WHERE  C.cid NOT IN (SELECT P.cid
                     FROM Product P
                     WHERE P.price >= 200)
```

Product (pname, price, cid)
Company(cid, cname, city)

3. Subqueries in WHERE

Find all companies s.t. <u>all</u> their products have price < 200

Universal quantifiers

Using EXISTS:

```
SELECT DISTINCT  C.cname
FROM     Company C
WHERE NOT EXISTS (SELECT *
                  FROM Product P
                  WHERE P.cid = C.cid and P.price >= 200)
```

Product (pname, price, cid)
Company(cid, cname, city)

3. Subqueries in WHERE

Find all companies s.t. all their products have price < 200

Universal quantifiers

Using ALL:

```
SELECT DISTINCT  C.cname
FROM     Company C
WHERE 200 > ALL  (SELECT price
                  FROM Product P
                  WHERE P.cid = C.cid)
```

Question for Database Fans and their Friends

- Can we unnest the *universal quantifier* query ?

Product (pname, price, cid)
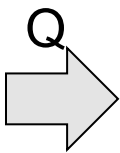Company(cid, cname, city)

Monotone Queries

- Definition A query Q is monotone if:
  - Whenever we add tuples to one or more input tables, the answer to the query will not lose any of of the tuples

Product

pname	price	cid
Gizmo	19.99	c001
Gadget	999.99	c003
Camera	149.99	c001

Company

cid	cname	city
c001	Sunworks	Bonn
c002	DB Inc.	Lyon
c003	Builder	Lodtz

Q →

A	B
149.99	Lodtz
19.99	Lyon

Is the mystery query monotone?

Product

pname	price	cid
Gizmo	19.99	c001
Gadget	999.99	c003
Camera	149.99	c001
iPad	499.99	c001

Company

cid	cname	city
c001	Sunworks	Bonn
c002	DB Inc.	Lyon
c003	Builder	Lodtz

Q →
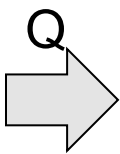
A	B
149.99	Lyon
19.99	Lyon
19.99	Bonn
149.99	Bonn

CSE 344 - Winter 2014

Monotone Queries

- <u>Theorem</u>: If Q is a SELECT-FROM-WHERE query that does not have subqueries, and no aggregates, then it is monotone.

- Proof. We use the nested loop semantics: if we insert a tuple in a relation R_i, this will not remove any tuples from the answer

```
SELECT  a_1, a_2, ..., a_k
FROM    R_1 AS x_1, R_2 AS x_2, ..., R_n AS x_n
WHERE   Conditions
```

Monotone Queries

- <u>Theorem</u>: If Q is a SELECT-FROM-WHERE query that does not have subqueries, and no aggregates, then it is monotone.

- Proof. We use the nested loop semantics: if we insert a tuple in a relation R_i, this will not remove any tuples from the answer

```
SELECT  a₁, a₂, …, aₖ
FROM    R₁ AS x₁, R₂ AS x₂, …, Rₙ AS xₙ
WHERE   Conditions
```

```
for x₁ in R₁ do
    for x₂ in R₂ do
        …..
            for xₙ in Rₙ do
                if Conditions
                    output (a₁,…,aₖ)
```

Product (pname, price, cid)
Company(cid, cname, city)

Monotone Queries

- The query:

Find all companies s.t. __all__ their products have price < 200

is not monotone

pname	price	cid
Gizmo	19.99	c001

cid	cname	city
c001	Sunworks	Bonn

⇒

cname
Sunworks

pname	price	cid
Gizmo	19.99	c001
Gadget	999.99	c001

cid	cname	city
c001	Sunworks	Bonn

⇒

cname

- <u>Consequence</u>: we cannot write it as a SELECT-FROM-WHERE query without nested subqueries

Queries that must be nested

- Queries with universal quantifiers or with negation

- Queries that use aggregates in funny ways
    - Note: sum(..) and count(*) are NOT monotone, because they do not satisfy set containment
    - **select count(*) from R** is not monotone!