Introduction to Data Management
CSE 344

Lecture 7: Nested Queries in SQL

Lecture Goals

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

• Reminder: Book chapters associated with lectures are listed on the calendar page of the course website

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

Today’s Examples in SQLite

create table Product(pname varchar(10), price int, cid int);
create table Company(cid int, cname varchar(10), city varchar(10));
insert into Product values ('gizmo', 100, 1);
insert into Product values('powergizmo', 200, 1);
insert into Product values('iStuff', 500, 2);
insert into Company values(1,'GizmoWorks','San Jose');
insert into Company values(1,'BigCompany','Boston');
insert into Company values(1,'PowerWorks','Seattle');

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)

Whenever possible, don’t use a nested queries:

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

We have “unnested” the query
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
```

Are these really equivalent?

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

No! Different results if a company has no products

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

2. Subqueries in FROM

Find all products whose prices is > 20 and < 500

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

Unnest this query!

3. Subqueries in WHERE

Find all companies that make some products with price < 200

Using EXISTS:

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

Existential quantifiers
3. Subqueries in WHERE

**Existential quantifiers**

Find all companies that make some products with price < 200

Using **ANY**:

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

Existential quantifiers are easy !

**Universal quantifiers**

Find all companies that make only products with price < 200 same as:

Find all companies whose products all have price < 200

Universal quantifiers are hard !

Find all companies that make only products with price < 200

Using **EXISTS**:

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

Find all companies that make only products with price < 200

Using **ALL**:

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

Now let's unnest it:

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

Magda Balazinska - CSE 344, Fall 2011
Question for Database Fans and their Friends

• Can we unnest the universal quantifier query?

Monotone Queries

• A query Q is monotone if:
  – Whenever we add tuples to one or more of the tables...
  – ...the answer to the query cannot contain fewer tuples

• Fact: all unnested queries are monotone
  – Proof: using the “nested for loops” semantics

• Fact: Query with universal quantifier is not monotone

• Consequence: we cannot unnest a query with a universal quantifier

Queries that must be nested

• Queries with universal quantifiers or with negation
• The drinkers-bars-beers example next
• This is a famous example from textbook on databases by Ullman

The drinkers-bars-beers example

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)
Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serves some beer they like.

Find drinkers that frequent only bars that serves only beer they like.

x : ∃ y. ∃ z. Frequents(x, y) ∧ Serves(y, z) ∧ Likes(x, z)

x : ∀ y. Frequents(x, y) ⇒ (∃ z. Serves(y, z) ∧ Likes(x, z))

x : ∀ y. Frequents(x, y) ⇒ ∀ z. (Serves(y, z) ⇒ Likes(x, z))

Challenge: write these in SQL

GROUP BY v.s. Nested Queries

SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product

SELECT DISTINCT x.product, (SELECT Sum(y.quantity)
FROM Purchase y
WHERE x.product = y.product
AND price > 1)
AS TotalSales
FROM Purchase x
WHERE price > 1

Unnesting Aggregates

Product (pname, price, company)
Company(cname, city)
Find the number of companies in each city

SELECT DISTINCT city, COUNT(*)
FROM Company Y
WHERE X.city = Y.city

SELECT DISTINCT city,
(SELECT COUNT(*)
FROM Company Y
WHERE X.city = Y.city)

Note: no need for DISTINCT (DISTINCT is the same as GROUP BY)
Unnesting Aggregates

Find the number of products made in each city:

\[
\begin{align*}
&\text{SELECT DISTINCT } X.\text{city}, \left( \text{SELECT count(*) FROM Product Y, Company Z WHERE Z.cname=Y.company AND Z.city = X.city} \right) \\
&\text{FROM Company X}
\end{align*}
\]

They are NOT equivalent! (WHY?)

More Unnesting

• Find authors who wrote ≥ 10 documents:
• Attempt 1: with nested queries

\[
\begin{align*}
&\text{SELECT DISTINCT Author.name FROM Author}
\end{align*}
\]

This is SQL by a novice

Finding Witnesses

For each store, find its most expensive products

\[
\begin{align*}
&\text{SELECT Store.sname, max(Product.price) FROM Store, Product,}
\end{align*}
\]

To find the witnesses, compute the maximum price in a subquery

\[
\begin{align*}
&\text{SELECT Store.sname, Product.pname FROM Store, Product,}
\end{align*}
\]

But we need the witnesses, i.e. the products with max price

\[
\begin{align*}
&\text{SELECT Store.sid, max(Product.price) FROM Store, Product,}
\end{align*}
\]

Finding Witnesses

For each store, find its most expensive products

\[
\begin{align*}
&\text{SELECT Store.sid AS sid, max(Product.price) AS p FROM Store,Product,}
\end{align*}
\]

They are NOT equivalent! (WHY?)

More Unnesting

Author(login, name)
Wrote(login, url)

• Find authors who wrote ≥ 10 documents:
• Attempt 1: with nested queries

\[
\begin{align*}
&\text{SELECT DISTINCT Author.name FROM Author}
\end{align*}
\]

This is SQL by a novice

Finding Witnesses

For each store, find its most expensive products

\[
\begin{align*}
&\text{SELECT Store.sname, Product.pname FROM Store, Product,}
\end{align*}
\]

To find the witnesses, compute the maximum price in a subquery

\[
\begin{align*}
&\text{SELECT Store.sid AS sid, max(Product.price) AS p FROM Store,Product,}
\end{align*}
\]
Finding Witnesses

There is a more concise solution here:

```
SELECT Store.sname, x.pname
FROM   Store, Product x
WHERE  Store.sid = x.sid and
        x.price >= ALL (SELECT y.price
                         FROM Product y
                         WHERE Store.sid = y.sid)
```