

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
- They are needed in Homework 3
- 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

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
```

“correlated
subquery”

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 (pname, price, cid)

Company(cid, cname, city)

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
```

=

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

We have
“unnested”
the query

1. Subqueries in SELECT

Product (pname, price, cid)

Company(cid, cname, city)

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
```

1. Subqueries in SELECT

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
```

2. Subqueries in FROM

Product (pname, price, cid)

Company(cid, cname, city)

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”).

3. Subqueries in WHERE

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

Existential quantifiers

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 C.cid = P.cid and P.price < 200)
```

3. Subqueries in WHERE

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

Existential quantifiers

Find all companies that make some products with price < 200

Using IN

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

3. Subqueries in WHERE

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

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)
```

3. Subqueries in WHERE

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

Existential quantifiers

Find all companies that make some products with price < 200

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 ! 😊

3. Subqueries in WHERE

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

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 ! 😞

3. Subqueries in WHERE

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)
```

3. Subqueries in WHERE

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

Universal quantifiers

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 P.cid = C.cid and P.price >= 200)
```

3. Subqueries in WHERE

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

Universal quantifiers

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 P.cid = C.cid)
```

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)

Challenge: write these in SQL

Find drinkers that frequent some bar that serves some beer they like.

x: $\exists y. \exists z. \text{Frequents}(x, y) \wedge \text{Serves}(y, z) \wedge \text{Likes}(x, z)$

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

x: $\forall y. \text{Frequents}(x, y) \Rightarrow (\exists z. \text{Serves}(y, z) \wedge \text{Likes}(x, z))$

Find drinkers that frequent some bar that serves only beers they like.

x: $\exists y. \text{Frequents}(x, y) \wedge \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z))$

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

x: $\forall y. \text{Frequents}(x, y) \Rightarrow \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z))$

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
```

Why twice ?

Unnesting Aggregates

Product (pname, price, cid)

Company(cid, cname, city)

Find the number of companies in each city

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

```
SELECT city, count(*)  
  FROM Company  
 GROUP BY city
```

Equivalent queries

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

Unnesting Aggregates

Product (pname, price, cid)

Company(cid, cname, city)

Find the number of products made in each city

```
SELECT DISTINCT X.city, (SELECT count(*)  
                           FROM Product Y, Company Z  
                          WHERE Z.cid=Y.cid  
                            AND Z.city = X.city)  
      FROM Company X
```

```
SELECT X.city, count(*)  
      FROM Company X, Product Y  
     WHERE X.cid=Y.cid  
   GROUP BY X.city
```

What if there
are no products
for a city?

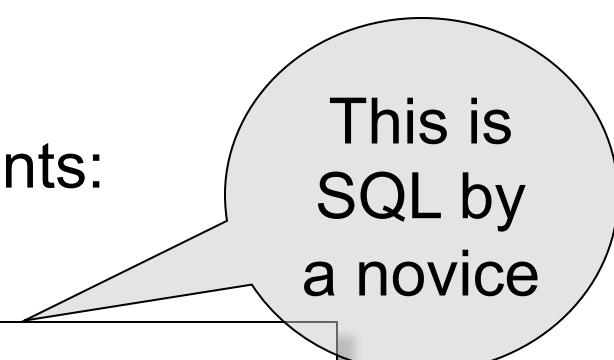
They are NOT
equivalent !
(WHY?)

More Unnesting

Author(login,name)

Wrote(login,url)

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



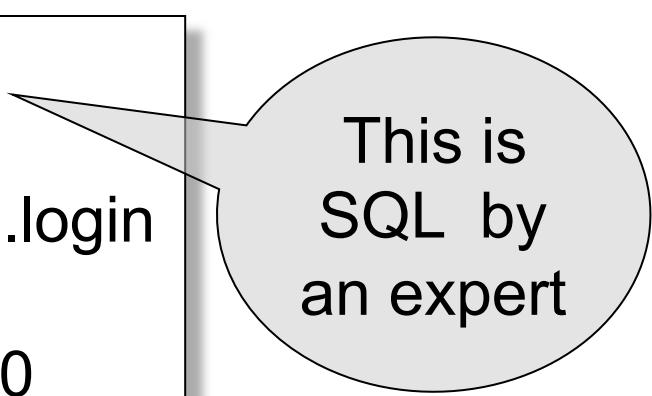
This is
SQL by
a novice

```
SELECT DISTINCT Author.name
FROM      Author
WHERE      (SELECT count(Wrote.url)
            FROM Wrote
            WHERE Author.login=Wrote.login)
                  > 10
```

More Unnesting

- Find all authors who wrote at least 10 documents:
- Attempt 2: SQL style (with GROUP BY)

```
SELECT      Author.name  
FROM        Author, Wrote  
WHERE       Author.login=Wrote.login  
GROUP BY    Author.name  
HAVING     count(wrote.url) > 10
```



This is
SQL by
an expert

Finding Witnesses

Product (pname, price, cid)

Company(cid, cname, city)

For each city, find the most expensive product made in that city

Finding Witnesses

Product (pname, price, cid)

Company(cid, cname, city)

For each city, find the most expensive product made in that city

Finding the maximum price is easy...

```
SELECT x.city, max(y.price)
FROM Company x, Product y
WHERE x.cid = y.cid
GROUP BY x.city;
```

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

Finding Witnesses

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

```
SELECT DISTINCT u.city, v.pname, v.price
FROM Company u, Product v,
(SELECT x.city, max(y.price) as maxprice
 FROM Company x, Product y
 WHERE x.cid = y.cid
 GROUP BY x.city) w
WHERE u.cid = v.cid
    and u.city = w.city
    and v.price=w.maxprice;
```

Finding Witnesses

There is a more concise solution here:

```
SELECT u.city, v.pname, v.price  
FROM Company u, Product v, Company x, Product y  
WHERE u.cid = v.cid and u.city = x.city and x.cid = y.cid  
GROUP BY u.city, v.pname, v.price  
HAVING v.price = max(y.price);
```

Finding Witnesses

And another one:

```
SELECT u.city, v.pname, v.price
FROM Company u, Product v
WHERE u.cid = v.cid
and v.price >= ALL (SELECT y.price
                     FROM Company x, Product y
                     WHERE u.city=x.city
                     and x.cid=y.cid);
```