

Introduction to Data Management

CSE 344

Lecture 6: Nested Queries in SQL

Announcements

- Webquiz 2 was due on Tuesday
 - Webquiz 3 is out
- Homework 2 is due on Wednesday
 - Homework 3 is out
 - We are using Microsoft Azure Cloud services!
(no more sqlite!)
 - Use the promotion code that you received in email
 - Will cover materials this week and next

What have we learned so far

- Data models
- Relational data model
 - Instance: relations
 - Schema:
 - Language: SQL

What have we learned so far

- SQL features
 - Projections
 - Selections
 - Joins (inner and outer)
 - Aggregates
 - Group by
 - Inserts, updates, and deletes
- Make sure you read the textbook!

```
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase ON
Product.name = Purchase.prodName
```

join pred.

Product

Purchase

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

```
SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON
Product.name = Purchase.prodName
```

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

```
SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON
Product.name = Purchase.prodName
```

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

```
SELECT Product.name, Purchase.store
FROM Product FULL OUTER JOIN Purchase ON
Product.name = Purchase.prodName
```

Product

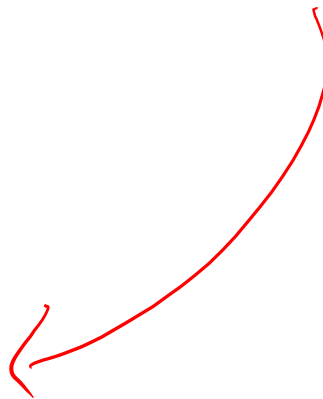
Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
Phone	Foo

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL
NULL	Foo




```

SELECT Product.name, COUNT(*)
FROM Product JOIN Purchase ON
      Product.name = Purchase.prodName
GROUP BY Product.name

```

Pro

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

~~Output~~

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

1 | Gizmo

 2 | Camera

~~0 | OneClick~~

Projecting Columns with Grouping

```
SELECT product,  
       max(quantity)  
FROM   Purchase  
GROUP BY product
```

```
SELECT product, quantity  
FROM   Purchase  
GROUP BY product
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

```
SELECT product, quantity
FROM Purchase
GROUP BY product
```

Projecting Columns with Grouping

```
SELECT product
FROM Purchase
GROUP BY product
```

```
SELECT quantity
FROM Purchase
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

Product
Bagel
Banana

+

???

Quantity
20
20
50
10
10

```
SELECT product, quantity
FROM Purchase
GROUP BY product
```



Can't project a non-grouped / non-aggregated column!

Lecture Goals

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

- 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**
 - But sometimes it's impossible, as we will see

Subqueries...

- Can appear as computed values in a `SELECT` clause
- Can appear in `FROM` clauses and aliased using a **tuple variable** that represents the tuples in the result of the subquery
- Can return a single constant to be compared with another value in a `WHERE` clause
- Can return relations to be used in `WHERE` clauses

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

(and SQLite simply ignores the extra values...)

Product (pname, price, cid)

Company (cid, cname, city)

1. Subqueries in SELECT

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

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 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

```
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
```

Side note: This is not a correlated subquery. (why?)

Try 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.

Product (pname, price, cid)

Company (cid, cname, city)

3. Subqueries in WHERE

Find all companies that make some products with price < 200

Product (pname, price, cid)

Company (cid, cname, city)

3. Subqueries in WHERE

Find all companies that make some products with price < 200

Existential quantifiers

Product (pname, price, cid)

Company (cid, cname, city)

3. Subqueries in WHERE

Find all companies that make some 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 some 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 some 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 (pname, price, cid)

Company (cid, cname, city)

3. Subqueries in WHERE

Find all companies that make some 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)
```

Not supported
in sqlite

Product (pname, price, cid)

Company (cid, cname, city)

3. Subqueries in WHERE

Find all companies that make some 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
```

Product (pname, price, cid)

Company (cid, cname, city)

3. Subqueries in WHERE

Find all companies that make some 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. all their products have price < 200

same as:

Find all companies that make only products with 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

same as:

Find all companies that make only products with price < 200

Universal quantifiers

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

3. Subqueries in WHERE

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

same as:

Find all companies that make only 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 that make some product ≥ 200

```
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 s.t. all their products have price < 200

1. Find *the other* companies that make 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)
```