Announcements

- HW3 will be posted tomorrow and due on Nov. 7, 11pm

Recap from last lecture

- Subqueries can occur in many clauses:
  - SELECT
  - FROM
  - WHERE
- Monotone queries: SELECT-FROM-WHERE
  - Existential quantifier
- Non-monotone queries
  - Universal quantifier
  - Aggregation

Examples of Complex Queries

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

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

```
SELECT DISTINCT X.drinker
FROM Frequents X, Serves Y, Likes Z
WHERE X.bar = Y.bar AND Y.beer = Z.beer AND X.drinker = Z.drinker
```

2. Find drinkers that frequent some bar that serves only beers they don’t like.

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

Example 1

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

```
SELECT DISTINCT X.drinker
FROM Frequents X, Serves Y, Likes Z
WHERE X.bar = Y.bar AND Y.beer = Z.beer AND X.drinker = Z.drinker
```

(drinker + bar they frequent + beer served that they like
=> drinker is an answer
(even though we only want the drinker, we need the rest to know it’s an answer.)

What happens if we didn’t write DISTINCT?

Example 1

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

```
SELECT DISTINCT X.drinker
FROM Frequents X, Serves Y, Likes Z
WHERE X.bar = Y.bar AND Y.beer = Z.beer AND X.drinker = Z.drinker
```

What happens if we didn’t write DISTINCT?
Example 2

Find drinkers that frequent some bar that serves only beers they don't like

Like(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

Existential
Universal

Let's find the others (drop the NOT):
Drinkers that frequent some bars that serve some beer they like.

That's the previous query...

That's the previous query... Let's write it with a subquery:

```
SELECT DISTINCT X.drinker
FROM Frequents X, Serves Y, Likes Z
WHERE X.bar = Y.bar AND
  Y.beer = Z.beer AND
  X.drinker = Z.drinker
```

Example 2

Find drinkers that frequent some bar that serves only beers they don't like

Let's find the others (drop the NOT):
Drinkers that frequent some bars that serve some beer they like.

That's the previous query...

That's the previous query... Let's write it with a subquery:

```
SELECT DISTINCT X.drinker
FROM Frequents X, Serves Y, Likes Z
WHERE EXISTS (SELECT * FROM Serves Y, Likes Z
WHERE X.bar = Y.bar AND
  Y.beer = Z.beer AND
  X.drinker = Z.drinker)
```

Example 3

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

Like(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

Existential
Universal
Example 3

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

Let's find the others (drop the NOT):
Drinkers that frequent some bar that serves only beer they don't like.

That's the previous query!

SELECT DISTINCT X.drinker
FROM Frequents X
WHERE NOT EXISTS (SELECT *
FROM Serves Y, Likes Z
WHERE X.bar=Y.bar AND
X.drinker=Z.drinker AND
Y.beer = Z.beer)

Unnesting Aggregates

Find the number of companies in each city

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

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

Find the number of companies in each city

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

**Equivalent queries**

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

Wait... are they equivalent?

Grouping vs. Nested Queries

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

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

Why twice?

More Unnesting

Find authors who wrote ≥ 10 documents:

**Attempt 1:** with nested queries

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

This is SQL by a novice

**Attempt 2:** using GROUP BY and HAVING

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

This is SQL by an expert

Finding Witnesses

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

Finding the maximum price is easy...

```sql
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
For each city, find the most expensive product made in that city
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
For each city, find the most expensive product made in that city
Or we can use a subquery in where clause

```
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);
```

Finding Witnesses
For each city, find the most expensive product made in that city
There is a more concise solution here:

```
Idea: Product JOIN Product ON "made in the same city"
Then group by first product.
Then check that first product has equal or higher price than
each of the second products in the group.

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);
```