Introduction to Data Structure (Data Management) Lecture 7

Felipe P. Vista IV

INTRO TO DATA STRUCTURE

WRAPPING UP SQL

Wrapping-up SQL

- Combining everything we've learned so far
- Examples

Recap of last lecture

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

Complex Queries

- `likes(drinker,beer)`
- `frequent(drinker,bar)`
- `serves(bar,beer)`

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

Complex Queries

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

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

Complex Queries

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Example #1

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

```
SELECT DISTINCT F.drinker
FROM frequents F, serves S, likes L
WHERE (F.bar=S.bar) AND (L.beer=S.beer) AND
      (L.drinker=F.drinker)
```

Example #1

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

```
SELECT DISTINCT F.drinker
FROM frequents F, serves S, likes L
WHERE (F.bar=S.bar) AND (L.beer=S.beer) AND
      (L.drinker=F.drinker)
```

drinker + bar being frequented (visited) + beer served that they like
≥ drinker → is an answer

Example #1

likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)

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

```
SELECT DISTINCT F.drinker
FROM frequents F, serves S, likes L
WHERE (F.bar=S.bar) AND (L.beer=S.beer) AND
      (L.drinker=F.drinker)
```

drinker + bar being frequented (visited) + beer served that they like
≥ drinker → is an answer

We only need the drinker,
but we still need to know the rest to know that it is the correct answer.

Example #1

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

```
SELECT DISTINCT F.drinker
FROM frequents F, serves S, likes L
WHERE (F.bar=S.bar) AND (L.beer=S.beer) AND
      (L.drinker=F.drinker)
```

drinker + bar being frequented (visited) + beer served that they like
≥ drinker → is an answer

We only need the drinker,
but we still need to know the rest to know that it is the correct answer.

What will happen if we don't include DISTINCT?

Example #2

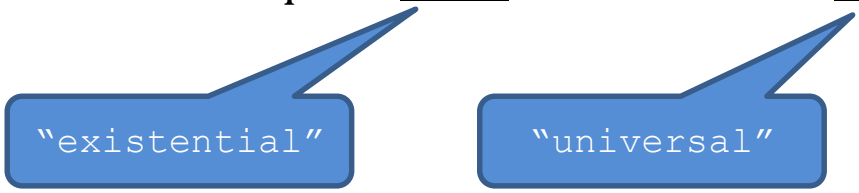
```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Example #2

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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



Example #2

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

"existential" "universal"

drinker

→ *bar that **only** serves beers that N do not like =*

→ *bar that does NOT serve **some** beer that N does like*

Example #2

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

"existential" "universal"

bar ⇒ some bars

all bars

*bar that **only** serves beers that N do not like =*
*bar that does NOT serve **some** beer that N does like*

some

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **some** beer they do like. →

Example #2

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

*bar that **only** serves beers that N do not like =*
*bar that does NOT serve **some** beer that N does like*

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **some** beer they do like.

This is the previous query (Example #1):

Example #2

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

*bar that **only** serves beers that N do not like =*
*bar that does NOT serve **some** beer that N does like*

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **some** beer they do like.

This is the previous query (Example #1):

```
SELECT DISTINCT F.drinker
FROM frequents F, serves S, likes L
WHERE (F.bar=S.bar) AND (L.beer=S.beer) AND
      (L.drinker=F.drinker)
```

Example #2

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **some** beer they do like.

This is the previous query (Example #1), write with a subquery:

```
SELECT DISTINCT F.drinker
FROM frequents F
WHERE EXISTS (SELECT *
              FROM serves S, likes L
              WHERE (F.bar=S.bar) AND
               L.beer=S.beer) AND (L.drinker=F.drinker)
```

Example #2

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **some** beer they do like.

*bar that **only** serves beers that N do not like =*
*bar that does NOT serve **some** beer that N does like*

Now let us **negate**:

```
SELECT DISTINCT F.drinker
FROM frequents F
WHERE NOT EXISTS (SELECT *
                  FROM serves S, likes L
                  WHERE (F.bar=S.bar) AND
                   L.beer=S.beer) AND (L.drinker=F.drinker)
```

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

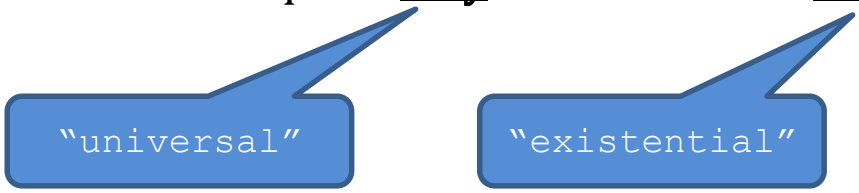
1: s B s B
   s B o B X

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

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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



Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

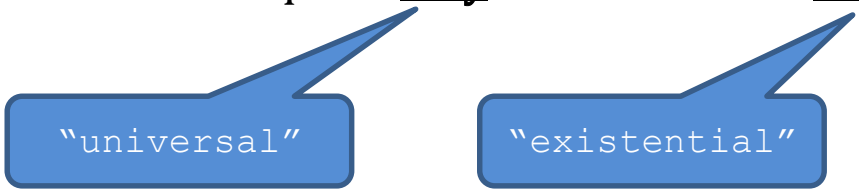
"universal" "existential"

*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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



*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

This is the previous query (Example #2):

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

This is the previous query (Example #2):

```
SELECT DISTINCT F.drinker
FROM frequents F
WHERE NOT EXISTS (SELECT *
                  FROM serves S, likes L
                  WHERE (F.bar=S.bar) AND
                   L.beer=S.beer) AND (L.drinker=F.drinker)
```

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

This is the previous query (Example #2), write with a subquery:

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

This is the previous query (Example #2), write with a subquery:

```
SELECT DISTINCT U.drinker
FROM frequents U
WHERE U.drinker IN
  (SELECT DISTINCT F.drinker
  FROM frequents F
  WHERE NOT EXISTS (SELECT *
              FROM serves S, likes L
              WHERE (F.bar=S.bar) AND
               L.beer=S.beer) AND (L.drinker=F.drinker))
```

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Now let us negate:

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Now let us **negate**:

```
SELECT DISTINCT U.drinker
FROM frequents U
WHERE U.drinker NOT IN
  (SELECT DISTINCT F.drinker
  FROM frequents F
  WHERE NOT EXISTS (SELECT *
              FROM serves S, likes L
              WHERE (F.bar=S.bar) AND
               L.beer=S.beer) AND (L.drinker=F.drinker))
```

Example #3

```
likes(drinker,beer)
frequent(drinker,bar)
serves(bar,beer)
```

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

Let us find the other members of the crew (remove the NOT):
Drinkers that frequent **some** bars that serves **only** beer they don't like.

*N frequents **only** bars that serves **some** beers that N likes =*
*N do not frequent **some** bar that serves **only** beers that N do not like*

Now let us negate:

```
SELECT DISTINCT U.drinker
FROM frequents U
WHERE U.drinker NOT IN
  (SELECT DISTINCT F.drinker
  FROM frequents F
  WHERE NOT EXISTS (SELECT *
              FROM serves S, likes L
              WHERE (F.bar=S.bar) AND
               L.beer=S.beer) AND (L.drinker=F.drinker))
```

And now, we need three nested queries!

Unnesting Aggregates

```
product(pname,price,cid)
company(cid,cname,city)
```

Find the number of companies in each city

Unnesting Aggregates

product(pname,price,cid)
company(cid,cname,city)

Find the number of companies in each city

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

Unnesting Aggregates

```
product(pname,price,cid)
company(cid,cname,city)
```

Find the number of companies in each city

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

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

Unnesting Aggregates

```
product(pname,price,cid)
company(cid,cname,city)
```

Find the number of companies in each city

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

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

Note:
No need for DISTINCT
(DISTINCT is same as GROUP BY)

Unnesting Aggregates

```
product(pname,price,cid)
company(cid,cname,city)
```

Find the number of companies in each city

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

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

Note:
No need for DISTINCT
(DISTINCT is same as GROUP BY)

Equivalent Queries

Are they?

Assignment!
? Due: 8th Oct

Grouping vs nested Queries

- purchase(pid,product,quantity,price,cid)

```
SELECT product, sum(quantity) AS totalSales
FROM purchase
WHERE price > 1
GROUP BY product
```

Grouping vs nested Queries

purchase(pid,product,quantity,price,cid)

```
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 Y.price > 1 AS totalSales

FROM purchase Y
WHERE X.price > 1
```

Grouping vs nested Queries

purchase(pid,product,quantity,price,cid)

```
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 Y.price > 1 AS totalSales

FROM purchase Y
WHERE X.price > 1
```

Why twice?

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10 books

- Attempt # 1: Using Nested Queries

More Unnesting

```
author(login,name)
wrote(login,url)
```

- Find authors who wrote ≥ 10

Attempt # 1: Using Nested Queries

```
SELECT DISTINCT name
FROM author
WHERE 10 ≤ (SELECT count(url)
            FROM wrote
            WHERE author.login = wrote.login)
```

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10

Attempt # 1: Using Nested Queries

```
SELECT DISTINCT author.name
FROM author
WHERE 10≤ (SELECT count(url)
           FROM wrote
           WHERE author.login=wrote.login
```

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10

Attempt # 2: Using GROUP BY and HAVING

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10

Attempt # 2: Using GROUP BY and HAVING

```
SELECT name
FROM author, wrote
WHERE author.login = wrote.login
GROUP BY name
HAVING count(url) ≥ 10
```

Group BY ≒ DISTINCT

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10

Attempt # 2: Using GROUP BY and HAVING

```
SELECT name
FROM author, wrote
WHERE author.login=wrote.login
GROUP BY name
HAVING count(url)≥ 10
```

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10

Attempt # 1: Using Nested Queries

```
SELECT DISTINCT author.name
FROM author
WHERE 10≤ (SELECT count(url)
           FROM wrote
           WHERE author.login=wrote.login
```

Attempt # 2: Using GROUP BY and HAVING

```
SELECT name
FROM author, wrote
WHERE author.login=wrote.login
GROUP BY name
HAVING count(url)≥ 10
```

More Unnesting

```
author(login,name)
wrote(login,url)
```

Find authors who wrote ≥ 10

Attempt # 1: Using Nested Queries

```
SELECT DISTINCT author.name
FROM author
WHERE 10≤ (SELECT count(url)
           FROM wrote
           WHERE author.login=wrote.login
```

Beginners' SQL statement

Attempt # 2: Using GROUP BY and HAVING

```
SELECT name
FROM author, wrote
WHERE author.login=wrote.login
GROUP BY name
HAVING count(url)≥ 10
```

A Pros' SQL statement

Find the "witnesses*"

- `product(pname,price,cid)`
- `company(cid,cname,city)`

* witnesses – products with the maximum price

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

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

Finding the maximum price is easy….

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

Finding the maximum price is easy....

```
SELECT X.city, max(price)
FROM company X, product Y
WHERE X.cid = Y.cid
GROUP BY X.city
```

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

Finding the maximum price is easy....

```
SELECT X.city, max(price)
FROM company X, product Y
WHERE X.cid = Y.cid
GROUP BY X.city
```

But what we need is the ***witnesses***...

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

To find the witnesses:

1. Compute the maximum price in a subquery

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

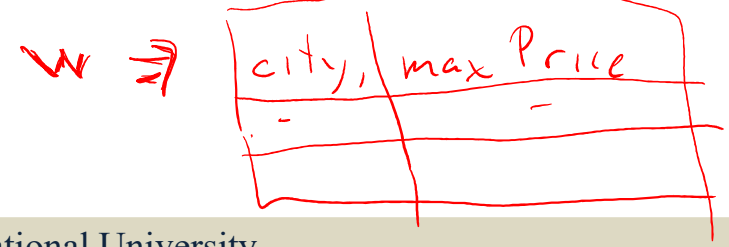
* witnesses – products with the maximum price

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

To find the witnesses:

1. 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
```



Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

To find the witnesses:

1. 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
```

Good solution

Find the “witnesses*”

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

To find the witnesses:

2. Using subquery in WHERE clause

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

To find the witnesses:

2. Using 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)
```

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

To find the witnesses:

3. And a more concise solution

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

To find the witnesses:

3. And a more concise solution

```
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)
```

Find the "witnesses*"

```
product(pname,price,cid)
company(cid,cname,city)
```

* witnesses – products with the maximum price

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

To find the witnesses:

3. And a more concise solution

```
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)
```

The IDEA?

1. Product JOIN Product ON "made in the same city"
2. Group by first product
3. Check that first product has equal or higher price than each of the second products in the group

Thank you.