

Database Systems CSE 414

Lecture 7: SQL Wrap-up

CSE 414 - Fall 2017

1

Announcements

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

CSE 414 - Fall 2017

2

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

CSE 414 - Fall 2017

3

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.
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.

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

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.)

5

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

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?

6

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

Example 2

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

Existential Universal

7

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

Example 2

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

bar serves only beers that X does not like =
bar that does NOT serve some beer that X does like

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

8

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

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 serves some beer they like.

That's the previous query...

```
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
```

9

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

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 serves some beer they like.

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

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

10

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

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 serves some beer they like.

bar serves only beers that X does not like =
bar that does NOT serve some beer that X does like

Now **negate!**

```
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)
```

11

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

Example 3

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

Universal Existential

12

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

Example 3

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

X frequents only bars that serve some beer X likes =
X does NOT frequent some bar that serves only beer X doesn't like

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

13

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

Example 3

Find drinkers that frequent only bars that serves 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!

14

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

Example 3

Find drinkers that frequent only bars that serves 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)
```

15

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

Example 3

Find drinkers that frequent only bars that serves 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! But write it as a nested query:

```
SELECT DISTINCT U.drinker
FROM Frequents U
WHERE U.drinker IN
      (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))
```

16

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

Example 3

Find drinkers that frequent only bars that serves 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.

X frequents only bars that serve some beer X likes =
X does NOT frequent some bar that serves only beer X doesn't like

Now negate!

```
SELECT DISTINCT U.drinker
FROM Frequents U
WHERE U.drinker NOT IN
      (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))
```

Now need three nested queries

17

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

Unnesting Aggregates

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
```

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

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

CSE 414 - Fall 2017

18

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

Unnesting Aggregates

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
```

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

Equivalent queries

CSE 414 - Fall 2017 19

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

Unnesting Aggregates

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
```

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

Wait... are they equivalent?

CSE 414 - Fall 2017 20

Purchase(pid, product, quantity, price)

Grouping vs. 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 y.price > 1)
AS TotalSales
FROM Purchase x
WHERE x.price > 1
```

Why twice ?

CSE 414 - Fall 2017 21

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

More Unnesting

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

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

This is SQL by a novice

CSE 414 - Fall 2017 22

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

More Unnesting

Find authors who wrote ≥ 10 documents:
Attempt 1: with nested queries
Attempt 2: using GROUP BY and HAVING

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

This is SQL by an expert

CSE 414 - Fall 2017 23

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

Finding Witnesses

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

CSE 414 - Fall 2017 24

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

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;
```

Not a bad
solution...

CSE 414 - Fall 2017

25

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

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);
```

CSE 414 - Fall 2017

26

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

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);
```

CSE 414 - Fall 2017

27