Announcements

- HW1 is due today 11pm
- WQ1 is due tomorrow 11pm
  - no late days
- WQ3 is posted and due on Oct. 19, 11pm

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 nested queries when possible; keep in mind that sometimes it’s impossible
  - (though use in FROM is often not as bad)

Subqueries…

- Can return a single constant and this constant can be compared with another value in a WHERE clause
- Can return relations that can be used in various ways in WHERE clauses
- Can appear in FROM clauses, followed by a tuple variable that represents the tuples in the result of the subquery
- Can appear as computed values in a SELECT clause

1. Subqueries in SELECT

For each product, return the city where it is manufactured

```sql
SELECT X.pname, (SELECT Y.city FROM Company Y WHERE Y.cid=X.cid) as City
FROM Product X
```

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

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
```

What happens if the subquery returns more than one city?
- We get a runtime error
  - (SQLite simply ignores the extra values)

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
```

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
```

But are these really equivalent?
- Yes, they are equivalent.

```
SELECT C.cname, count(pname)
FROM Company C LEFT OUTER JOIN Product P
ON C.cid=P.cid
GROUP BY C.cname
```

1. 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
```

Unnest this query:

```
SELECT pname
FROM Product
WHERE price > 20 AND price < 500
```

2. Subqueries in FROM

- We will see that sometimes we really need a subquery
  - will see most compelling examples next lecture
  - in that case, we can put it in the FROM clause
3. Subqueries in WHERE

Find all companies that make some products with price < 100

**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 < 100
)
```

**CSE 414 - Fall 2017**

13

Using IN:

```
SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid IN (SELECT P.cid
    FROM Product P
    WHERE P.price < 100)
```

**CSE 414 - Fall 2017**

14

Using ANY:

```
SELECT DISTINCT C.cname
FROM Company C
WHERE 100 > ANY (SELECT price
    FROM Product P
    WHERE P.cid = C.cid)
```

**CSE 414 - Fall 2017**

15

Not supported in sqlite

Now let's unnest it:

```
SELECT DISTINCT C.cname
FROM Company C, Product P
WHERE C.cid = P.cid AND P.price < 100
```

**CSE 414 - Fall 2017**

16

**Existential quantifiers are easy ! 😊**

Universal quantifiers are hard ! 😐

**CSE 414 - Fall 2017**

17

Find all companies where all their products have price < 100

same as:

Find all companies that make only products with price < 100

**Universal quantifiers**

```
SELECT DISTINCT C.cname
FROM Company C
WHERE 100 > ALL (SELECT price
    FROM Product P
    WHERE P.cid = C.cid)
```

**CSE 414 - Fall 2017**

18

Find all companies where all their products have price < 100

1. Find the other companies with some product having price ≥ 100

```
SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid NOT IN (SELECT P.cid
    FROM Product P
    WHERE P.price >= 100)
```

**CSE 414 - Fall 2017**

18

2. Find all companies where all their products have price < 100

```
SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid NOT IN (SELECT P.cid
    FROM Product P
    WHERE P.price > 100)
```

**CSE 414 - Fall 2017**

18
3. Subqueries in WHERE

Find all companies where all their products have price < 100

Using EXISTS:

```
SELECT DISTINCT C.cname
FROM Company C
WHERE NOT EXISTS (SELECT * FROM Product P
WHERE P.cid = C.cid AND P.price >= 100)
```

Using ALL:

```
SELECT DISTINCT C.cname
FROM Company C
WHERE 100 >= 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?  
  – No

Monotone Queries

• Theorem: If Q is a SELECT-FROM-WHERE query that does not have subqueries, and no aggregates, then it is monotone.

• Proof.  We use the nested loop semantics: if we insert a tuple in a relation Rᵢ, this will not remove any tuples from the answer

```
SELECT aᵥ₁, aᵥ₂, ..., aᵢ
FROM Rᵥ₁ AS xᵥ₁, Rᵥ₂ AS xᵥ₂, ..., Rᵢ AS xᵢ
WHERE Conditions
```

Monotone Queries

• Consequence: we cannot write it as a SELECT-FROM-WHERE query without nested subqueries

Not supported in sqlite
Queries that must be nested
(that is, cannot be SPW queries)

• Queries with universal quantifiers or negation

• Queries that use aggregates in usual ways
  are not monotone
  – Note: sum(...) etc. are NOT monotone
  – select count(*) from R is not monotone!