Introduction to Data Management CSE 344

Lecture 7: SQL Wrap-up

Relational Algebra

Announcements

- Webquiz 3 is open, due next Tuesday
- Homework 3 will be posted, due on Wednesday, 10/26
  - We are using Microsoft Azure Cloud services! (no more sqlite!)
  - Use the promotion code that you will receive in email
  - Will cover materials this week and next
- Check piazza for assignment updates
  - Watch out for “instructor note” emails

What We Learned Last Time

- Subqueries can occur in every clause:
  - `SELECT`
  - `FROM`
  - `WHERE`
- Monotone queries: SELECT-FROM-WHERE
  - Existential quantifier
- Non-monotone queries
  - Universal quantifier
  - Aggregation

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

Monotone Queries

- The query:

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

is not monotone

pname	price	cid
Gizmo	19.99	c001

cid	cname	city
c001	Sunworks	Bonn

⇒

cname
Sunworks

pname	price	cid
Gizmo	19.99	c001
Gadget	999.99	c001

cid	cname	city
c001	Sunworks	Bonn

⇒

cname

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

Queries that must be nested

- Queries with universal quantifiers or with negation
  - These are non-monotonic queries

Queries that must be nested

- Queries with universal quantifiers or with negation
  - These are non-monotonic queries

- Queries that use aggregates in certain ways
  - `sum(..)` and `count(*)` are NOT monotone, because they do not satisfy set containment
  - `select count(*) from R` is not monotone!

```
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

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

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

Unnesting Aggregates

Find the number of products made in each city

```
SELECT DISTINCT X.city, (SELECT count(*)
                         FROM Product Y, Company Z
                         WHERE Z.cid=Y.cid
                           AND Z.city = X.city)
FROM  Company X
```

```
SELECT X.city, count(*)
FROM Company X, Product Y
WHERE X.cid=Y.cid
GROUP BY X.city
```

NOT equivalent !
You should know why!

Purchase(pid, product, quantity, price)

GROUP BY v.s. 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 ?

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

More Unnesting

Find authors who wrote ≥ 10 documents:

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

More Unnesting

Find authors who wrote ≥ 10 documents:

Attempt 1: with nested queries

This is SQL by a novice

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

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

This is SQL by an expert

`Product (pname, price, cid)`

Review: Counting and NULL

- `count(*)`: counts number of rows regardless of NULL
- `count(A)`: counts number of non NULL values of attribute A

Product

pname	price	cid
Gizmo	null	c001
Gadget	null	c004
null	null	c003

```
SELECT count(*)
FROM Product
```
⇒ 3

```
SELECT count(pname)
FROM Product
```
⇒ 2

```
SELECT count(price)
FROM Product
```
⇒ 0

Try this in sqlite

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

1. Subqueries in SELECT

```
SELECT DISTINCT C.cname, (SELECT count(*)
                          FROM Product P
                          WHERE P.cid=C.cid)
FROM  Company C
```

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

cid	cname	city
c002	Sunworks	Bonn
c001	DB Inc.	Lyon
c003	Builder	Lodtz

pname	price	cid
Gizmo	19.99	c001
Gadget	999.99	c004
Camera	149.99	c003

cid	cname	city	pname	price	cid
c002	Sunworks	Bonn	null	null	null
c001	DB Inc.	Lyon	GIzmo	19.99	c001
c003	Builder	Lodtz	null	null	null

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

In-class Exercise

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

```
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

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

Finding Witnesses

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;
```

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

Finding Witnesses

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);
```

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

Finding Witnesses

There is a more concise solution here:

```
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)
```

Where We Are

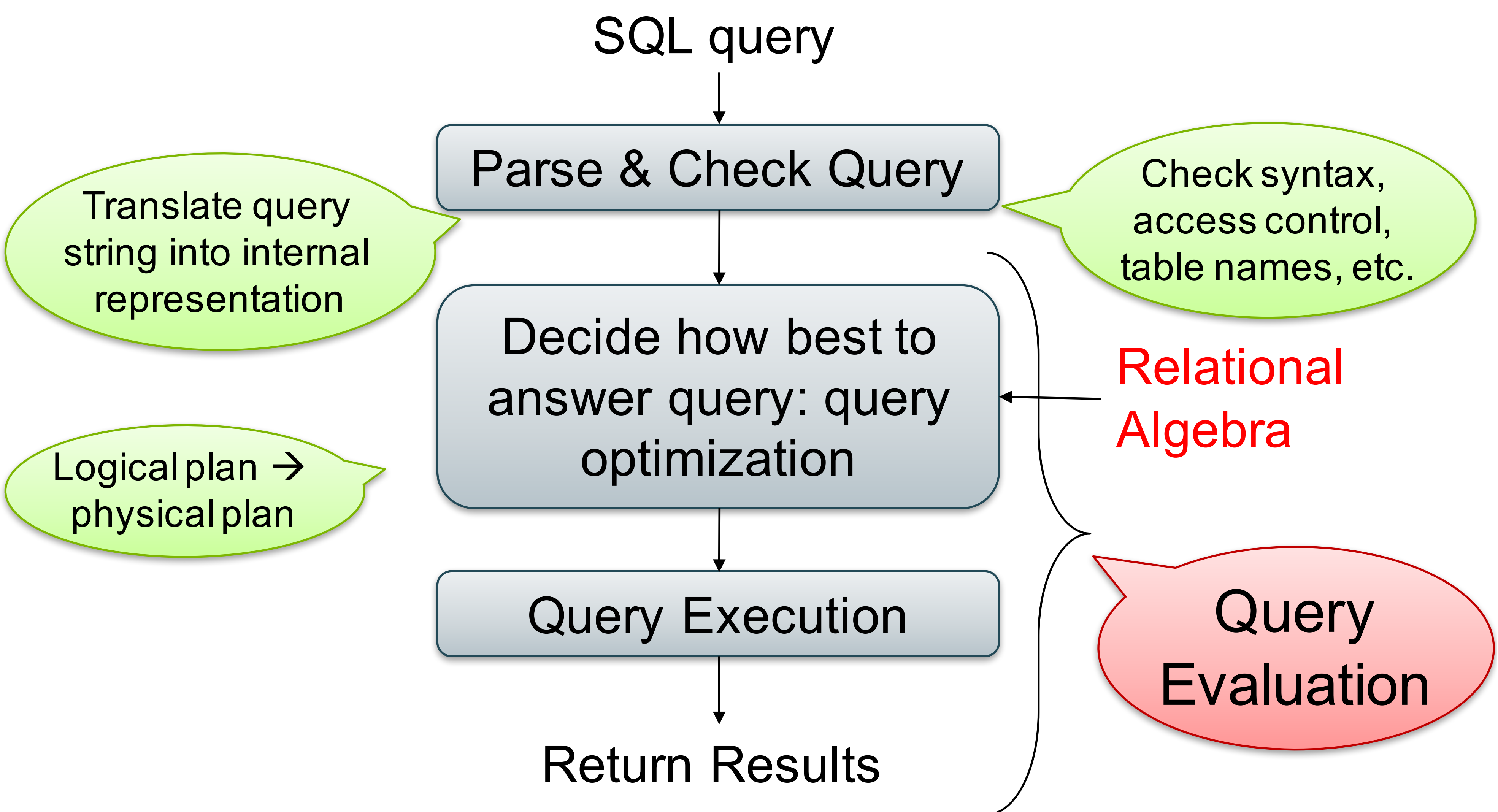
- Data models
- SQL, SQL, SQL
  - Declaring the schema for our data (CREATE TABLE)
  - Inserting data one row at a time or in bulk (INSERT/.import)
  - Querying the data (SELECT)
  - Modifying the schema and updating the data (ALTER/UPDATE)

- Next step: More knowledge of how DBMSs work
  - Relational algebra, query execution, and physical tuning
  - Client-server architecture

The Relational Model

- Instance
  - Organized as “table” or “relation”
- Schema
  - tables and columns / relations and attributes
- Query languages
  - SQL
  - Relational algebra (RA)
- We will learn RA by studying the internals of DBMS

Query Evaluation Steps



The WHAT and the HOW

- SQL = WHAT we want to get form the data
- Relational Algebra = HOW to get the data we want
- The passage from WHAT to HOW is called query optimization
  - SQL → Relational Algebra → Physical Plan
  - Relational Algebra = Logical Plan

Overview: SQL = WHAT

```
Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)
```

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid
and x.price > 100 and z.city = 'Seattle'
```

It's clear WHAT we want, unclear HOW to get it

Relation Algebra

- Relations and attributes
- Functions that are applied to relations
  - Return relations
  - Can be composed together
  - Often displayed using a tree rather than linearly
  - Uses Greek symbols: σ, π, δ, etc

- Language for describing query plans

Overview: Relational Algebra = HOW

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid
and y.cid = z.cid
and x.price > 100
and z.city = 'Seattle'
```

Overview: Relational Algebra = HOW

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid
and y.cid = z.cid
and x.price > 100
and z.city = ‘Seattle’
```