Introduction to Data Management
CSE 344

Lectures 8: Relational Algebra

Announcements

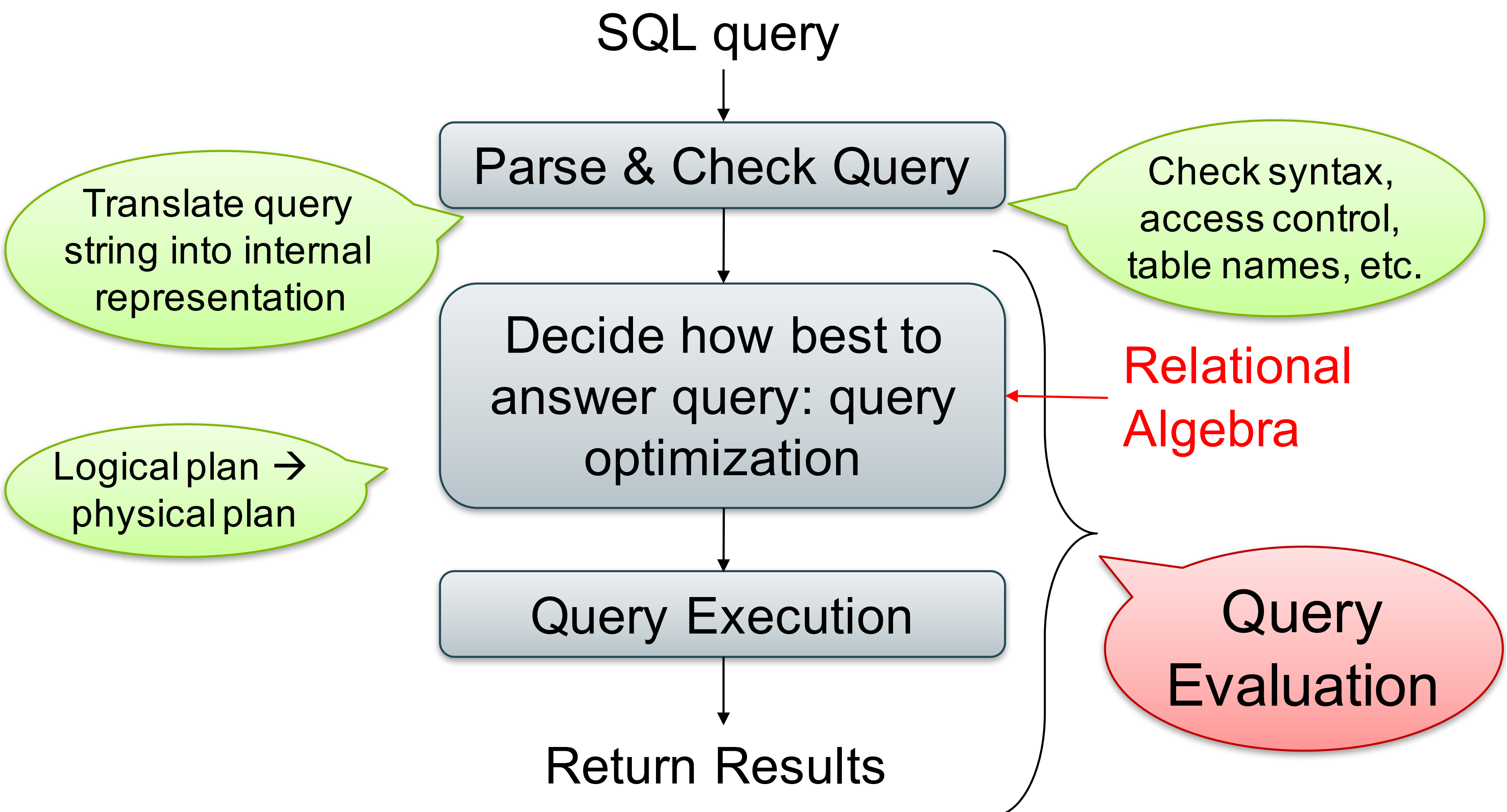
- Homework 3 is posted
  - Microsoft Azure Cloud services!
  - Use the promotion code you received
  - Due in two weeks

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

Query Evaluation Steps



The WHAT and the HOW

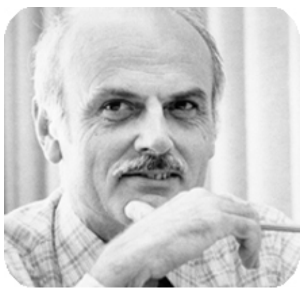
- SQL = WHAT we want to get from the data
- Relational Algebra = HOW to get the data we want
- The passage from WHAT to HOW is called query optimization
  - SQL → Relational Algebra → Physical Plan
  - Relational Algebra = Logical Plan

Relational Algebra

Turing Awards in Data Management

Charles Bachman, 1973

IDS and CODASYL



Ted Codd, 1981

Relational model

Michael Stonebraker, 2014

INGRES and Postgres

Sets v.s. Bags

- Sets: {a,b,c}, {a,d,e,f}, { }, . . .
- Bags: {a, a, b, c}, {b, b, b, b, b}, . . .

Relational Algebra has two semantics:

- Set semantics = standard Relational Algebra
- Bag semantics = extended Relational Algebra

DB systems implement bag semantics (Why?)

Relational Algebra Operators

- Union $\cup$, intersection $\cap$, difference -
- Selection σ
- Projection π
- Cartesian product X, join $\bowtie$
- Rename ρ
- Duplicate elimination δ
- Grouping and aggregation γ
- Sorting τ

RA

Extended RA

All operators take in 1 or more relations as inputs and return another relation

Union and Difference

$R1 \cup R2$

$R1 - R2$

What do they mean over bags ?

What about Intersection ?

- Derived operator using minus

$$\boxed{R1 \cap R2 = R1 - (R1 - R2)}$$

- Derived using join

$$\boxed{R1 \cap R2 = R1 \bowtie R2}$$

  – Only makes sense if R1 and R2 have the same schema

Selection

- Returns all tuples which satisfy a condition

$$\boxed{\sigma_c(R)}$$

- Examples
  - $\sigma_{\text{Salary > 40000}}$ (Employee)
  - $\sigma_{\text{name = "Smith"}}$ (Employee)
- The condition c can be =, <, <=, >, >=, <> combined with AND, OR, NOT

Employee

SSN	Name	Salary
1234545	John	20000
5423341	Smith	60000
4352342	Fred	50000

$\sigma_{\text{Salary} > 40000}$ (Employee)

SSN	Name	Salary
5423341	Smith	60000
4352342	Fred	50000

Projection

- Eliminates columns

$$\boxed{\pi_{A1,\ldots,An}(R)}$$

- Example: project social-security number and names:
  - $\pi_{SSN,\ Name}$ (Employee)
  - Answer(SSN, Name)

Different semantics over sets or bags! Why?

Employee

SSN	Name	Salary
1234545	John	20000
5423341	John	60000
4352342	John	20000

$\pi_{\text{Name,Salary}}$ (Employee)

Name	Salary
John	20000
John	60000
John	20000

Bag semantics

Name	Salary
John	20000
John	60000

Set semantics

Which is more efficient?

Composing RA Operators

Patient

no	name	zip	disease
1	p1	98125	flu
2	p2	98125	heart
3	p3	98120	lung
4	p4	98120	heart

$\pi_{\text{zip,disease}}(\text{Patient})$

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

$\sigma_{\text{disease='heart'}}(\text{Patient})$

no	name	zip	disease
2	p2	98125	heart
4	p4	98120	heart

$\pi_{\text{zip,disease}}(\sigma_{\text{disease='heart'}}(\text{Patient}))$

zip	disease
98125	heart
98120	heart

Cartesian Product

- Each tuple in R1 with each tuple in R2

$$\boxed{R1 \times R2}$$

- Rare in practice; mainly used to express joins

Cross-Product Example

Employee

Name	SSN
John	999999999
Tony	777777777

Dependent

EmpSSN	DepName
999999999	Emily
777777777	Joe

Employee X Dependent

Name	SSN	EmpSSN	DepName
John	999999999	999999999	Emily
John	999999999	777777777	Joe
Tony	777777777	999999999	Emily
Tony	777777777	777777777	Joe

Renaming

- Changes the schema, not the instance

$$\boxed{\rho_{B1,\ldots,Bn}\,(R)}$$

- Example:
  - $\rho_{N,\,S}$(Employee) → Answer(N, S)

Not really used by systems, but needed on paper

Natural Join

$$\boxed{R1 \bowtie R2}$$

- Meaning: $R1 \bowtie R2 = \Pi_A(\sigma_\theta (R1 \times R2))$

- Where:
  - Selection checks equality of all common attributes (i.e., attributes with same names)
  - Projection eliminates duplicate common attributes

Natural Join Example

R

A	B
X	Y
X	Z
Y	Z
Z	V

S

B	C
Z	U
V	W
Z	V

R ⋈ S =

$\Pi_{ABC}(\sigma_{R.B=S.B}(R \times S))$

A	B	C
X	Z	U
X	Z	V
Y	Z	U
Y	Z	V
Z	V	W

Natural Join Example 2

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie V$

age	zip	disease	name
54	98125	heart	p1
20	98120	flu	p2

Natural Join

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R $\bowtie$ S ?
- Given R(A, B, C), S(D, E), what is R $\bowtie$ S?
- Given R(A, B), S(A, B), what is R $\bowtie$ S?

AnonPatient (age, zip, disease)
Voters (name, age, zip)

Theta Join

- A join that involves a predicate

$$\boxed{R1 \bowtie_\theta R2 \ = \ \sigma_\theta (R1 \times R2)}$$

- Here θ can be any condition
- No projection in this case!
- For our voters/patients example:

$$P \bowtie_{P.zip = V.zip \text{ and } P.age >= V.age -1 \text{ and } P.age <= V.age +1} V$$

Equijoin

- A theta join where θ is an equality predicate
- Projection drops all redundant attributes

$$\boxed{R1 \bowtie_{\theta} R2 \quad = \pi_{A}(\sigma_{\theta} (R1 \times R2))}$$

- By far the most used variant of join in practice

Equijoin Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.age=V.age} V$

age	P.zip	disease	name	V.zip
54	98125	heart	p1	98125
20	98120	flu	p2	98120

Join Summary

- **Theta-join**: $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ
- **Equijoin**: $R \bowtie_\theta S = \pi_A (\sigma_\theta (R \times S))$
  - Join condition θ consists only of equalities
  - Projection π_A drops all redundant attributes
- **Natural join**: $R \bowtie S = \pi_A (\sigma_\theta (R \times S))$
  - Equijoin
  - Equality on **all** fields with same name in R and in S
  - Projection π_A drops all redundant attributes