Introduction to Database Systems
CSE 414

Lecture 10: Relational Algebra

Recap: Datalog

- Facts and Rules
- Selection, projection, join
- Recursive rules
- Grouping, aggregates
- Negation
- Safe vs unsafe rules
- Stratification

Class Overview

- Unit 1: Intro
- Unit 2: Relational Data Models and Query Languages
  - Data models, SQL, Datalog, Relational Algebra
- Unit 3: Non-relational data
- Unit 4: RDMBS internals and query optimization
- Unit 5: Parallel query processing
- Unit 6: DBMS usability, conceptual design
- Unit 7: Transactions

Relational Algebra

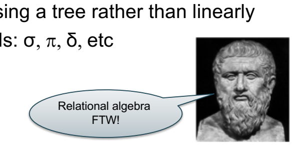
- Set-based algebra that manipulates relations
  - We will extend it to multisets / bags
- In SQL & Datalog we say *what* we want
- In RA we can express *how* to get it
- Every DBMS implementations converts a SQL query to RA in order to execute it
- An RA expression is called a *query plan*

Why study *yet* another relational query language?

- RA is how SQL is implemented in DBMS
  - We will see more of this in a few weeks
- RA opens up opportunities for *query optimization*

Basics

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



Relational Algebra Operators

- Union ∪, ~~intersection ∩~~, difference -
- Selection σ
- Projection π
- Cartesian product X, join ⋈
- (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

A ∪ B

$\boxed{\begin{array}{l} R1 \cup R2 \\ R1 - R2 \end{array}}$

Only make sense if R1, R2 have the same schema

What do they mean over bags ?

What about Intersection ?

- Derived operator using minus

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

- Derived using join (as we will see later)

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

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:
  - $\pi_{\text{SSN, Name}}$ (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

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

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

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

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

$\pi_{\text{zip,disease}}(\sigma_{\text{disease='heart'}}$(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:
  - Given Employee(Name, SSN)
  - $\rho_{N,\,S}$(Employee) → Answer(N, S)

Natural Join

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

- Meaning: $R1 \bowtie R2 = \Pi_A(\sigma_\theta(R1 \times R2))$
- Where:
  - Selection σ_θ checks equality of all common attributes (i.e., attributes with same ~~names~~)
  - Projection Π_A 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
Alice	54	98125
Bob	20	98120

P ⋈ V

age	zip	disease	name
54	98125	heart	Alice
20	98120	flu	Bob

σ_θ = P.zip = V.zip AND P.age = V.age

Π_A = zip, age, ...

Natural Join

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?
- Given R(A, B, C), S(D, E), what is R ⋈ S?
- Given R(A, B), S(A, B), what is R ⋈ 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:

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

Equijoin

- A theta join where θ is an equality predicate

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

- By far the most used variant of join in practice
- What is the relationship with natural join?

Equijoin Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

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

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

Join Summary

- **Theta-join**: R $\bowtie_\theta$ S = σ_θ (R × S)
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ
  - No projection
- **Equijoin**: R $\bowtie_\theta$ S = σ_θ (R × S)
  - Join condition θ consists only of equalities
  - No projection
- **Natural join**: R ⋈ S = π_A (σ_θ (R × S))
  - Equality on **all** fields with same name in R and in S
  - Projection π_A drops all redundant attributes

So Which Join Is It ?

When we write R ⋈ S we usually mean an equijoin, but we often omit the equality predicate when it is clear from the context

More Joins

- **Outer join**
  - Include tuples with no matches in the output
  - Use NULL values for missing attributes
  - Does not eliminate duplicate columns
- Variants
  - Left outer join
  - Right outer join
  - Full outer join

Outer Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu
33	98120	lung

AnnonJob J

job	age	zip
lawyer	54	98125
cashier	20	98120

P ⟕ J

P.age	P.zip	P.disease	J.job	J.age	J.zip
54	98125	heart	lawyer	54	98125
20	98120	flu	cashier	20	98120
33	98120	lung	null	null	null

⟖ ROJ

⟗ FOJ

⟕ LOJ