Introduction to Data Management CSE 344

Lecture 12: Relational Calculus

Announcements

- HW 3 due next Tuesday

Spent	Count
$0	22
$1	9
$2	8
$11	1
Not Signed Up!	1

Cost of Query Plans

Review: Physical Query

We only care about Disk I/O operations

Supplier (sid, sname ,scity, sstate)

Supply (pno, sid, quanty)

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

T(Supplier) = 1000
T(Supply) = 10,000

B(Supplier) = 100
B(Supply) = 100

# of distinct

V(Supplier,scity) = 20
V(Supplier,state) = 10
V(Supply,pno) = 2,500

M = 11

T(Supplier) = 1000
T(Supply) = 10,000

B(Supplier) = 100
B(Supply) = 100

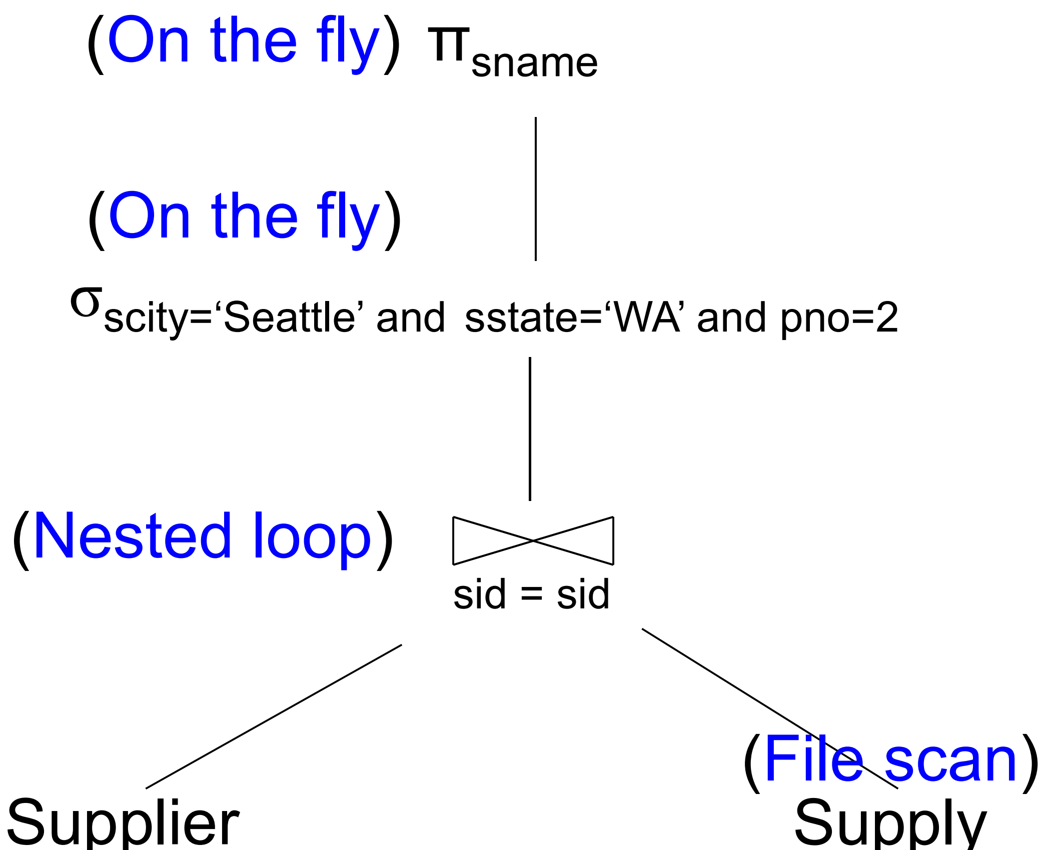
V(Supplier,scity) = 20
V(Supplier,state) = 10
V(Supply,pno) = 2,500

M = 11

Physical Query: Naive Plan

$$\pi_{\text{sname}}(\sigma_{\text{scity='Seattle' and sstate='WA' and pno=2}}(\text{Supplier} \bowtie \text{Supply})$$

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = ‘Seattle’
    and x.sstate = ‘WA’
```



T(Supplier) = 1000
T(Supply) = 10,000

B(Supplier) = 100
B(Supply) = ~~100~~ 80

V(Supplier,scity) = 20
V(Supplier,state) = 10
V(Supply,pno) = 2,500

M = 11

Physical Query: Naive Plan

(On the fly) π_{sname}

Selection and project on-the-fly
→ No additional cost.

(On the fly) $\sigma_{\text{scity='Seattle' and sstate='WA' and pno=2}}$

Total cost of plan is thus cost of join:
= B(Supplier)+B(Supplier)*B(Supply)
= 100 + 100 * ~~100~~ 80
= 10,100 I/Os

(Nested loop) $\bowtie_{\text{sid = sid}}$ 2

100 | 80

Supplier (File scan) | Supply (File scan)

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

T(Supplier) = 1000
T(Supply) = 10,000

B(Supplier) = 100
B(Supply) = 100

V(Supplier,scity) = 20
V(Supplier,state) = 10
V(Supply,pno) = 2,500

M = 11

Physical Query: Optimized

4. (On the fly) π_{sname}

3. (Sort-merge join) $\bowtie_{sid = sid}$

(Scan write to T1)

(Scan write to T2)

1. $\sigma_{scity=\text{'Seattle'} \text{ and } sstate=\text{'WA'}}$

Push down selection

2. $\sigma_{pno=2}$

Supplier (File scan)

Supply (File scan)

Total cost

(step 1) write T1

100 + $\boxed{100 * 1/20 * 1/10}$ ~= 100

Read

(step 2) write T2

100 + 100 * 1/2500 ~= 100

(step 3) 2

(step 4) 0

Total cost ≈ **204 I/Os**

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

T(Supplier) = 1000
T(Supply) = 10,000

B(Supplier) = 100
B(Supply) = 100

V(Supplier,scity) = 20
V(Supplier,state) = 10
V(Supply,pno) = 2,500

M = 11

Physical Query: Using Indexes

(On the fly) 4. π_{sname}

(On the fly)
3. $\sigma_{scity=\text{'Seattle'} \text{ and } sstate=\text{'WA'}}$

2. $\bowtie_{sid = sid}$ (Index nested loop)

1 or 2 reads

10000*1/2500
= 4 tuples

4 reads

(Use hash index) 1. $\sigma_{pno=2}$

Supply
(Index on pno)
Assume: clustered

Supplier
(Index on sid)
Clustering does not matter

Total cost
= 1 (or 2) (step 1.)
+ 4 (step 2.)
+ 0 (step 3.)
+ 0 (step 4.)
Total cost ≈ **5 I/Os (or 6)**

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = ‘Seattle’
    and x.sstate = ‘WA’
```

Query Optimizer Summary

- **Input**: A logical query plan
- **Output**: A good physical query plan
- **Basic query optimization algorithm**
  - Enumerate alternative plans (logical and physical)
  - Compute estimated cost of each plan
    - Compute number of I/Os
    - Optionally take into account other resources
  - Choose plan with lowest cost
  - This is called cost-based optimization

Big Picture

- Relational data model
  - Instance
  - Schema
  - Query language
    - SQL
    - Relational algebra
    - Relational calculus
    - Datalog
- Query processing
  - Logical & physical plans
  - Indexes
  - Cost estimation
  - Query optimization

Why bother with another QL?

- SQL and RA are good for query planning
  - They are not good for *formal reasoning*
  - How do you show that two SQL queries are equivalent / non-equivalent?
  - Two RA plans?
- RC was the first language proposed with the relational model (Codd)
- Influenced the design of datalog as we will see

Relational Calculus

- Aka *predicate calculus* or *first order logic*
  - 311 anyone?

- TRC = Tuple Relational Calculus
  - See book

- DRC = Domain Relational Calculus
  - We study only this one
  - Also see *Query Language Primer* on course website

Relational Calculus

Query Q:

This means: $(x_1, \ldots, x_k)$ is in Q if P is true

$$\boxed{Q(x_1, \ldots, x_k) = P}$$

Relational predicate P is a formula given by this grammar:

$$\boxed{P ::= \text{atom} \mid P \wedge P \mid P \vee P \mid P \Rightarrow P \mid \text{not}(P) \mid \forall x.P \mid \exists x.P}$$

Atomic predicate is either a relational or interpreted predicate:

$$\boxed{\text{atom} ::= R(x_1, \ldots, x_k) \mid x = y \mid x > c \mid \ldots}$$

R(x,y) means (x,y) is in R

Actor(pid,fName,lName)
Casts(pid,mid)
Movie(mid,title,year)

Relational Calculus

Example: find the first/last names of actors who acted in 1940

SQL

```
SELECT fname, lname FROM Actor a, Movie m, Casts c
WHERE a.pid = c.pid AND c.mid = m.mid AND m.year = 1940
```

Relational Algebra

$\pi_{fname,lname}(\sigma_{year=1940}(Actor \bowtie Casts \bowtie Movie))$

Relational Calculus

$Q(f,l) = \exists x.\ \exists y.\ \exists z.\ (Actor(z,f,l) \wedge Casts(z,x) \wedge Movie(x,y,1940))$

What does this query return ?

Only acted in 1940

$Q(f,l) = \exists z.\ (Actor(z,f,l) \wedge \forall x.(Casts(z,x) \Rightarrow \exists y.Movie(x,y,1940)))$

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer), happy-hour

311 Review: Implication

Find all bars that serve all beers that Fred likes

$$A(x) = \forall y.\ \text{Likes}(\text{"Fred"}, y) \Rightarrow \text{Serves}(x,y)$$

- Note: $P \Rightarrow Q$ (read P implies Q) is the same as (not P) $\vee$ Q

  In this query: If Fred likes a beer the bar must serve it ($P \Rightarrow Q$) In other words: Either Fred does not like the beer (not P) OR the bar serves that beer (Q).

$$A(x) = \forall y.\ \text{not}(\text{Likes}(\text{"Fred"}, y)) \vee \text{Serves}(x,y)$$

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

311 Review: Meaning of ∀

Find all bars that serve all beers that Fred likes

$$A(x) = \forall y.\ Likes(\text{"Fred"}, y) \Rightarrow Serves(x,y)$$

- We want to find x's such that the formula on the RHS is true
- For a given bar x, we need to check whether the implication holds **for all values of y**
  - Not enough to just check one value of y!

$$A(x) = \forall y.\ not(Likes(\text{"Fred"}, y)) \lor Serves(x,y)$$

y1, y2, y3....

- Likewise, given a bar x, we need to iterate over **all values of y** and check whether Serves(x,y) is true!

311 Review: Remember your logical equivalences!

- $A \Rightarrow B$ = not(A) $\vee$ B
- not(A $\wedge$ B) = not(A) $\vee$ not(B)
- not(A $\vee$ B) = not(A) $\wedge$ not(B)
- $\forall x.$ P(x) = not($\exists x.$ not(P(x)))

- Example:
  - $\forall z.$ Serves(y,z) $\Rightarrow$ Likes(x,z)
  - $\forall z.$ not(Serves(y,z)) $\vee$ Likes(x,z)
  - not ($\exists z.$ Serves(y,z) $\wedge$ not(Likes(x,z))

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

More Examples

Average Joe

Find drinkers that frequent <u>some</u> bar that serves <u>some</u> beer they like.

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

More Examples

Average Joe

Find drinkers that frequent some bar that serves some beer they like.

$$Q(x) = \exists y.\ \exists z.\ \text{Frequents}(x, y) \land \text{Serves}(y,z) \land \text{Likes}(x,z)$$

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

More Examples

Average Joe

Find drinkers that frequent some bar that serves some beer they like.

$$Q(x) = \exists y.\ \exists z.\ \text{Frequents}(x, y) \wedge \text{Serves}(y,z) \wedge \text{Likes}(x,z)$$

Prudent Peter

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

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

More Examples

Average Joe

Find drinkers that frequent <u>some</u> bar that serves <u>some</u> beer they like.

$$Q(x) = \exists y.\ \exists z.\ \text{Frequents}(x, y) \land \text{Serves}(y,z) \land \text{Likes}(x,z)$$

Prudent Peter

Find drinkers that frequent <u>only</u> bars that serves <u>some</u> beer they like.

$$Q(x) = \forall y.\ \text{Frequents}(x, y) \Rightarrow (\exists z.\ \text{Serves}(y,z) \land \text{Likes}(x,z))$$

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

More Examples

Average Joe

Find drinkers that frequent <u>some</u> bar that serves <u>some</u> beer they like.

$$Q(x) = \exists y.\ \exists z.\ \text{Frequents}(x, y) \wedge \text{Serves}(y,z) \wedge \text{Likes}(x,z)$$

Prudent Peter

Find drinkers that frequent <u>only</u> bars that serves <u>some</u> beer they like.

$$Q(x) = \forall y.\ \text{Frequents}(x, y) \Rightarrow (\exists z.\ \text{Serves}(y,z) \wedge \text{Likes}(x,z))$$

Cautious Carl

Find drinkers that frequent <u>some</u> bar that serves <u>only</u> beers they like.

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

More Examples

Average Joe

Find drinkers that frequent <u>some</u> bar that serves <u>some</u> beer they like.

$$Q(x) = \exists y.\ \exists z.\ Frequents(x, y) \wedge Serves(y,z) \wedge Likes(x,z)$$

Prudent Peter

Find drinkers that frequent <u>only</u> bars that serves <u>some</u> beer they like.

$$Q(x) = \forall y.\ Frequents(x, y) \Rightarrow (\exists z.\ Serves(y,z) \wedge Likes(x,z))$$

Cautious Carl

Find drinkers that frequent <u>some</u> bar that serves <u>only</u> beers they like.

$$Q(x) = \exists y.\ Frequents(x, y) \wedge \forall z.(Serves(y,z) \Rightarrow Likes(x,z))$$

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

More Examples

Average Joe

Find drinkers that frequent <u>some</u> bar that serves <u>some</u> beer they like.

$$Q(x) = \exists y.\ \exists z.\ Frequents(x, y) \wedge Serves(y,z) \wedge Likes(x,z)$$

Prudent Peter

Find drinkers that frequent <u>only</u> bars that serves <u>some</u> beer they like.

$$Q(x) = \forall y.\ Frequents(x, y) \Rightarrow (\exists z.\ Serves(y,z) \wedge Likes(x,z))$$

Cautious Carl

Find drinkers that frequent <u>some</u> bar that serves <u>only</u> beers they like.

$$Q(x) = \exists y.\ Frequents(x, y) \wedge \forall z.(Serves(y,z) \Rightarrow Likes(x,z))$$

Paranoid Paul

Find drinkers that frequent <u>only</u> bars that serves <u>only</u> beer they like.

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

More Examples

Average Joe

Find drinkers that frequent <u>some</u> bar that serves <u>some</u> beer they like.

$$Q(x) = \exists y.\ \exists z.\ Frequents(x, y) \land Serves(y,z) \land Likes(x,z)$$

Prudent Peter

Find drinkers that frequent <u>only</u> bars that serves <u>some</u> beer they like.

$$Q(x) = \forall y.\ Frequents(x, y) \Rightarrow (\exists z.\ Serves(y,z) \land Likes(x,z))$$

Cautious Carl

Find drinkers that frequent <u>some</u> bar that serves <u>only</u> beers they like.

$$Q(x) = \exists y.\ Frequents(x, y) \land \forall z.(Serves(y,z) \Rightarrow Likes(x,z))$$

Paranoid Paul

Find drinkers that frequent <u>only</u> bars that serves <u>only</u> beer they like.

$$Q(x) = \forall y.\ Frequents(x, y) \Rightarrow \forall z.(Serves(y,z) \Rightarrow Likes(x,z))$$

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

Domain Independent Relational Calculus

- An *unsafe* RC query, aka *domain dependent,* returns an answer that does not depend just on the relations, but on the entire domain of possible values

$\boxed{A1(x) = \text{not Likes}(\text{"Fred"}, x)}$

$A1(x) = \exists y \text{ Serves}(y,x) \wedge \text{not Likes}(\text{"Fred"}, x)$

Make sure x ~~is a beer~~ is served

- There are lots of objects that could be liked.
- Make sure to limit to those that are served.

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

Domain Independent Relational Calculus

- An *unsafe* RC query, aka *domain dependent,* returns an answer that does not depend just on the relations, but on the entire domain of possible values

Make sure x is a beer

$\boxed{A1(x) = \text{not } Likes(\text{"Fred"}, x)}$

$A1(x) = \exists y\ Serves(y,x) \wedge \text{not } Likes(\text{"Fred"}, x)$

$\boxed{A2(x,y) = Likes(\text{"Fred"}, x) \vee Serves(\text{"Bar"}, y)}$

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

Domain Independent Relational Calculus

- An *unsafe* RC query, aka *domain dependent,* returns an answer that does not depend just on the relations, but on the entire domain of possible values

Make sure x is a beer

$\boxed{\text{A1(x) = not Likes("Fred", x)}}$

A1(x) = ∃y Serves(y,x) ∧ not Likes("Fred", x)

$\boxed{\text{A2(x,y) = Likes("Fred", x) ∨ Serves("Bar", y)}}$

Limit to things that are served.

A2(x,y) = ∃u Serves(u,x) ∧ ∃w Serves(w,y) ∧ [Likes("Fred", x) ∨ Serves("Bar", y)]

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

Domain Independent Relational Calculus

- An *unsafe* RC query, aka *domain dependent,* returns an answer that does not depend just on the relations, but on the entire domain of possible values

Make sure x is a beer

$A1(x) = \text{not Likes}(\text{"Fred"}, x)$

$A1(x) = \exists y \text{ Serves}(y,x) \land \text{not Likes}(\text{"Fred"}, x)$

$A2(x,y) = \text{Likes}(\text{"Fred"}, x) \lor \text{Serves}(\text{"Bar"}, y)$

Limit to things that are served.

$A2(x,y) = \exists u \text{ Serves}(u,x) \land \exists w \text{ Serves}(w,y) \land [\text{Likes}(\text{"Fred"}, x) \lor \text{Serves}(\text{"Bar"}, y)]$

$A3(x) = \forall y. \text{ Serves}(x,y)$

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

Domain Independent Relational Calculus

- An *unsafe* RC query, aka *domain dependent,* returns an answer that does not depend just on the relations, but on the entire domain of possible values

Make sure x is a beer

$A1(x) = \text{not Likes}(\text{"Fred"}, x)$

$A1(x) = \exists y\ \text{Serves}(y,x) \wedge \text{not Likes}(\text{"Fred"}, x)$

$A2(x,y) = \text{Likes}(\text{"Fred"}, x) \vee \text{Serves}(\text{"Bar"}, y)$

Limit to things that are served.

$A2(x,y) = \exists u\ \text{Serves}(u,x) \wedge \exists w\ \text{Serves}(w,y) \wedge [\text{Likes}(\text{"Fred"}, x) \vee \text{Serves}(\text{"Bar"}, y)]$

Likewise

$A3(x) = \forall y.\ \text{Serves}(x,y)$

$A3(x) = \exists u.\text{Serves}(x,u) \wedge \forall y.\exists z.\text{Serves}(z,y) \rightarrow \text{Serves}(x,y)$

Domain of variables

- The **active domain** of a RC formula P includes all constants that occur in P:
  - $y > 3$, then AD(P) = 3
  - pred(x,y) then AD(P) = none (pred = Bool. predicate)
  - $\forall y.\ R(x,2,y) \Rightarrow S(x,y)$, then AD(P) = 2
    (R, S are predicates)
- Active domain of a database instance includes all values that occurs in it

Making RC queries safe requires limiting to the Active Domain

Domain independence

- A RC formula P is **domain independent** if for every database instance I and every domain D such that $AD(P) \cup AD(I) \subseteq D$, then $P_D(I) = P_{AD(P) \cup AD(I)}(I)$
  – Note: P has to be evaluated in at least $AD(P) \cup AD(I)$

- In other words, evaluating P on a larger domain than $AD(P) \cup AD(I)$ does not affect the query results
  – This is a desirable property!

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

Domain independence

- $Q(x) = \forall y.\ Likes(x,y)$ is domain dependent
  - Suppose Likes = { (d1,b1), (d1,b2) }
  - What if we evaluate y over { b1, b2 }?
  - What about { b1, b2, b3 }? → Nothing
    ↳ not liked
- $Q(x) = \exists y.\ Likes(x,y)$ is domain independent
  - What if we evaluate y over { b1, b2 }?
  - What about { b1, b2, b3 }?
- $Q(x) = IsBar(x) \wedge \forall y.\ Serves(x,y) \Rightarrow IsBeer(y)$ is domain independent
  - Let IsBeer = { b1, b2 }, IsBar = { bar1 }, and Serves = { (bar1, b1), (bar1, b2) }
  - What if we evaluate y over { b1, b2 }? { b1, b2, b3 }?

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

Domain Independence

Make sure x is a beer

$\boxed{A1(x) = \text{not Likes("Fred", x)}}$

$A1(x) = \exists y\ \text{Serves}(y,x) \wedge \text{not Likes("Fred", x)}$

can be anything

$\boxed{A2(x,y) = \text{Likes("Fred", x)} \vee \text{Serves("Bar", y)}}$

Same here

$A2(x,y) = \exists u\ \text{Serves}(u,x) \wedge \exists w\ \text{Serves}(w,y) \wedge [\text{Likes("Fred", x)} \vee \text{Serves("Bar", y)}]$

$\boxed{A3(x) = \forall y.\ \text{Serves}(x,y)}$

Likewise

$A3(x) = \exists u.\text{Serves}(x,u) \wedge \forall y.\exists z.\text{Serves}(z,y) \rightarrow \text{Serves}(x,y)$

Lesson: make sure your RC queries are domain independent

Big Picture

- Relational data model
  - Instance
  - Schema
  - Query language
    - SQL
    - Relational algebra
    - Relational calculus
    - Datalog
- Query processing
  - Logical & physical plans
  - Indexes
  - Cost estimation
  - Query optimization

Datalog vs Relational Algebra