Introduction to Data Management CSE 344

Lecture 13: Relational Calculus

Announcements

- WQ 4, HW 4 are out
- Midterm review session in class next Thursday
- Section attendance:
  - Checking in for absentees etc is an academic dishonesty
  - We hope we don't need to but will pursue such students if needed

Cost of Query Plans

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 Plan 1

(On the fly) π_{sname}

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

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

Supplier (File scan)

Supply (File scan)

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

Total cost of plan is thus cost of join:
= B(Supplier)+B(Supplier)*B(Supply)
= 100 + 100 * 100
= 10,100 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 Plan 2

4. (On the fly) π_{sname}

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

(Scan write to T1)

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

Supplier (File scan)

(Scan write to T2)

2. $\sigma_{pno=2}$

Supply (File scan)

Total cost

$= \underbrace{100}_{\text{read Supplier}} + \underbrace{100 * 1/20 * 1/10}_{\text{write } T_1}$ (step 1)

$+ \underbrace{100}_{\text{read Supply}} + \underbrace{100 * 1/2500}_{\text{write } T_2}$ (step 2)

$+ \underbrace{2}_{\text{read } T_1, T_2}$ (step 3)

$+ 0$ (step 4)

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 Plan 3

(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)

10000*1/2500
= 4 tuples

(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:

$Q(x_1, \ldots, x_k) = P$

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

Relational predicate P is a formula given by this grammar:

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

Atomic predicate is either a relational or interpreted predicate:

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

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

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

Relational Calculus

Query Q:

$Q(x_1, \ldots, x_k) = P$

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

Relational predicate P is a formula given by this grammar:

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

Atomic predicate is either a relational or interpreted predicate:

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

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

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

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

What does this query return ?

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

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

Important Observation

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)

More Examples

Average Joe

Find drinkers that frequent some bar that serves some 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.

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

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.

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

Prudent Peter

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

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

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.

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

Cautious Carl

Find drinkers that frequent some bar that serves only beers 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.

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

Prudent Peter

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

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

Cautious Carl

Find drinkers that frequent some bar that serves only beers they like.

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

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.

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

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 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.

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

Cautious Carl

Find drinkers that frequent some bar that serves only beers they like.

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

Paranoid Paul

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

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

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)

311 Review: Meaning of $\forall$

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!

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

$= not(L(\text{"F"}, y_1)) \vee S(x, y_1) \wedge$
$not(L(\text{"F"}, y_2)) \vee S(x, y_2) \wedge \ldots$

for all values of y

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

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

A1(x) = not Likes("Fred", x)

Make sure x is a beer

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

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

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

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

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

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

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

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

Same here

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

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

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

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

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

Same here

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

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

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

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

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

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

Same here

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

Likewise

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

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

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

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 }?

- $Q(x) = \exists y.\ Likes(x,y)$ 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

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

Same here

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

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

Likewise

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

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

Lesson: make sure your RC queries are domain independent