Lecture 10: Relational Algebra (continued), Datalog

Monday, October 16, 2000

Outline

- Really finish Relational Algebra (4.1)
- Datalog (4.2-4.4)

Relational Algebra

- Five basic operators, many derived
- Combine operators in order to construct queries: relational algebra expressions, usually shown as trees

Complex Queries

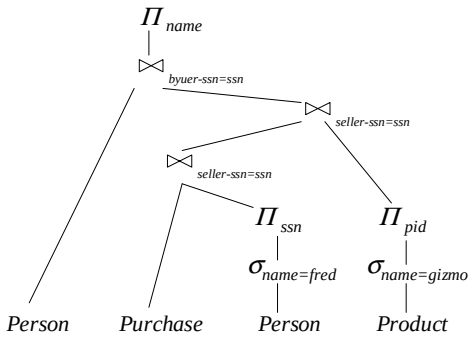
Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

Note:
• in Purchase: buyer-ssn, seller-ssn are **foreign keys** in Person, pid is **foreign key** in Product;
• in Product maker-cid is a **foreign key** in Company

Find phone numbers of people who bought gizmos from Fred.

Find telephony products that somebody bought

Expression Tree



Exercises

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

Ex #1: Find people who bought telephony products.
Ex #2: Find names of people who bought American products
Ex #3: Find names of people who bought American products and did not buy French products
Ex #4: Find names of people who bought American products and they live in Seattle.
Ex #5: Find people who bought stuff from Joe or bought products from a company whose stock prices is more than $50.

Operations on Bags (and why we care)

- Union: {a,b,b,c} U {a,b,b,b,e,f,f} = {a,a,b,b,b,b,b,c,e,f,f}
  - *add* the number of occurrences
- Difference: {a,b,b,b,c,c} – {b,c,c,c,d} = {a,b,b,d}
  - subtract the number of occurrences
- Intersection: {a,b,b,b,c,c} {b,b,c,c,c,c,d} = {b,b,c,c}
  - minimum of the two numbers of occurrences
- Selection: preserve the number of occurrences
- Projection: preserve the number of occurrences (no duplicate elimination)
- Cartesian product, join: no duplicate elimination

Reading assignment: 4.6.2 - 4.6.6

Rationale of Relational Algebra

- Why bother ? Can write any RA expression directly in C++/Java, seems easy.
- Two reasons:
  - Each operator admits sophisticated implementations (think of $\bowtie$, σ_C)
  - Expressions in relational algebra can be rewritten: optimized

Glimpse Ahead: Different Implementations of Operators

- $\sigma_{(\textbf{age} >= 30 \text{ AND } \textbf{age} <= 35)}(\textbf{Employees})$
  - Method 1: scan the file, test each employee
  - Method 2: use an index on **age**
  - Which one is better ? Depends a lot…
- **Employees** $\bowtie$ **Relatives**
  - Many implementation methods (some researchers built careers out of that)

Glimpse Ahead: Join-order Optimization

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Person(ssn, name, phone number, city)

- Find people living in Seattle, buying>$100:
  $\sigma_{price>100}(\text{Product}) \bowtie (\text{Purchase} \bowtie \sigma_{city=sea}\text{Person})$
  $(\sigma_{price>100}(\text{Product}) \bowtie \text{Purchase}) \bowtie \sigma_{city=sea}\text{Person}$
- Which is better ? Depends…

Finally: RA has Limitations !

- Cannot compute "transitive closure"

Name1	Name2	Relationship
Fred	Mary	Father
Mary	Joe	Cousin
Mary	Bill	Spouse
Nancy	Lou	Sister

- Find all direct and indirect relatives of Fred
  - Answer: Mary, Joe, Bill
- Cannot express in RA !!! Need to write C program

Datalog

- RA is good because it can express different implementations
- RA is unfriendlier for writing queries
- Logic is friendlier for writing queries
- First Order Logic: $\wedge, \vee, \neg, \forall, \exists, \in$
- Datalog: a subset, friendlier but as powerful
- SQL is based on logic (following lectures)

Datalog Example 1

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

- Find all products over $99.99:

S(i,n,p,c,m) ← Product(i,n,p,c,m) AND p>99.99

- a selection: $\sigma_{price>99.99}(Product)$

Datalog Example 2

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

- Find the names of all products over $99.99:

S(n) ← Product(i,n,p,c,m) AND p>99.99

a selection-projection: $\Pi_{name}\sigma_{price>99.99}(Product)$

Datalog Example 3

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

- Find store names where Fred bought:

S(n) ← Person(s,"Fred",t,c) AND Purchase(s,l,n,p)

a selection-projection-join:
$\Pi_{store}\sigma_{name="Fred"}(Person) \bowtie_{ssn=buyer\text{-}ssn} Purchase$

- Datalog is really friendly
- Let's see the formal definitions, then more examples

Datalog Definitions

- A predicate P is a relation name
  - E.g. Product
  - E.g. Company
- An atom is P(x,y,z,…), with P a predicate and x,y,z variables or constants
  - E.g. Product(i,n,p,c,m)
  - E.g. Product(i,"gizmo",p,"electronics","gizmoWorks")
  - Given a database instance, an atom is true or false
- Arithmetic atoms
  - E.g. x > 5
  - E.g. y <= z
  - Are true or false independent on the database instance

Datalog Definitions

- A datalog rule:

$atom \leftarrow atom_1$ AND … AND $atom_n$

head, body, Subgoals: may be preceded by NOT

- E.g.:
  S(n) ← Person(s, n, p, t, c) AND Purchase (b, s, "Gizmo Store", i)
- Datalog program = a collection of rules (later)

Anonymous Variables

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

Find names of people who bought from "Gizmo Store"

- E.g.:

S(n) ← Person(s, n, _, _, _) AND Purchase (_, s, "Gizmo Store", _)

Each _ means a fresh, new variable
Very useful: makes Datalog even easier to read

Anonymous Variables Continued

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

Find phone numbers of people who bought gizmos from Fred.

A(p) ← Person(x,"Fred",_,_) AND Purchase(y,x,_,pid) AND
Product(pid,"gizmo",_,_,_) AND Person(y,_,p,_)

Safe Datalog Rules

- A datalog rule is safe if:
  - Each variable in the rule appears at least in one non-negated atom in the body
- Examples of unsafe rules:

S(x,w) ← Product(x,y,z,u,v)
S(x) ← Product(x,y,z,u,v) AND z > w
S(x) ← Product(x,y,z,u,v) AND NOT Purchase(s,t,w,v)

Meaning of a Safe Datalog Rule

$atom \longleftarrow atom_1$ AND … AND $atom_n$

- Recall head and body
- Let {x1,…,xn} be all variables in the rule
- Assign values to x1,…,xn in all possible ways
- For each assignment, if the body is true, add the head to the answer
- Alternative meaning: map tuples rather than variables
  - reading assignment section 4.2.4

Multiple Datalog Rules

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

- Find names of buyers and sellers:

A(n) ← Person(s,n,_,_), Purchase(s,_,_,_)
A(n) ← Person(s,n,_,_), Purchase(_,s,_,_)

- Multiple rules correspond to union

Multiple Datalog Rules

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

- Find Seattle residents who bought products over $100:

E(s) ← Product(i,_,p,_,_) AND Purchase(s,_,_,i) AND p>100
A(n) ← Person(s,n,_,"Seattle") AND E(s)

- Multiple rules correspond to sequential computation
- Same as substituting E's body in the second rule

Negation in Datalog

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

- Find all "bad pid's" in Purchase (I.e. which don't occur in Product)

  P(p) ← Product(p,_,_,_)
  BadP(p) ← Purchase(_,_,_,p) AND NOT P(p)

- Wrong solution why ?
  BadPWrong(p) ← Purchase(_,_,_,p) AND NOT Product(p,_,_,_)

Negation in Datalog (continued)

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, stock price, country)
Person(ssn, name, phone number, city)

- Find products that were never sold:

  Sold(p) ← Purchase(_,_,_,p) AND Product(p,_,_,_,_)
  NeverSold(p) ← Product(p,_,_,_) AND NOT Sold(p)

Relational Algebra and Datalog

- Datalog:
  - Friendly
  - Says nothing about how to evaluate
- Relational Algebra
  - Unfriendly
  - Can say in which order to evaluate
- Good news: relational algebra is equivalent to (non-recursive) datalog !

From Relational Algebra to Datalog

- Union R1 U R2:
  S(x,y,z) ← R1(x,y,z)
  S(x,y,z) ← R2(x,y,z)
- Difference R1 - R2
  S(x,y,z) ← R1(x,y,z) AND NOT R2(x,y,z)
- Cartesian product R1 x R2
  S(x,y,z,u,w) ← R1(x,y,z) AND R2(u,w)

From RA to Datalog (cont'd)

- Selection $\sigma_{z > 35}(R)$

  S(x,y,z,u) ← R(x,y,z,u) AND z > 35

- Projection $\Pi_{x,z}(R)$

  S(x,z) ← R(x,y,z,u)

From (non-recursive) Datalog to RA

- Let's take an example: R(A,B,C), S(D,E,F,G), T(H,I)
  S(x,y) ← R(x,y,z) AND S(y,y,w,x) AND T(z,55)
- First make all variables distinct, add arithmetic atoms:
  S(x,y) ← R(x,y,z) AND S(y1,y2,w,x3) AND T(z4,c5)
  AND y=y1 AND y1=y2 AND x=x3
  AND z=z4 AND c5=55
- In RA: a select-project-join expression:
  $\Pi_{A, B}(\sigma_{B=D \text{ AND } D=E \text{ AND } A=G \text{ AND } C=H \text{ AND } I=55}(R \times S \times T))$

From (non-recursive) Datalog to RA

- Exercises:
  - Translate a rule with negation to RA (hint: use difference)
  - Translated multiple rules to RA (hint: use union and/or substitutions; remember that rules are non-recursive)

Recursive Datalog Programs

- Recall:
  - Find Fred's relatives

Name1	Name2	Relation ship
Fred	Mary	Father
Mary	Joe	Cousin
Mary	Bill	Spouse
Nancy	Lou	Sister

Relative(x) ← R("Fred",x,_)
Relative(y) ← Relative(x) AND R(x,y,_)

Recommended reading: 4.4