Introduction to Data Management
CSE 344

Lecture 14: Datalog
Announcements

• WQ 4 and HW 4 are out
  – Both due next week

• Midterm on 11/7 in class
  – Previous exams on course webpage

• Midterm review next Fri (11/4) in class
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
Review

Query Q:

\[ Q(x_1, \ldots, x_k) = P \]

Relational predicate P is a formula given by this grammar:

\[ P ::= \text{atom} \mid P \land P \mid P \lor P \mid P \Rightarrow P \mid \text{not}(P) \mid \forall x.P \mid \exists x.P \]

Atomic predicate is either a relational or interpreted predicate:

\[ \text{atom ::= } R(x_1, \ldots, x_k) \mid x = y \mid x > k \mid \ldots \]

R(x,y) means (x,y) is in R
Find all bars that serve all beers that Fred likes

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

- 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!

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

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

- The **active domain** of a RC formula $P$ includes all constants that occur in $P$:
  - $y > 3$, then $\text{AD}(P) = 3$
  - $R(x,y)$ then $\text{AD}(P) = \text{none}$ (R is a predicate)
  - $\forall y. R(x,2,y) \Rightarrow S(x,y)$, then $\text{AD}(P) = 2$
    (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 $\text{AD}(P) \cup \text{AD}(I) \subseteq D$, then $P_D(I) = P_{\text{AD}(P) \cup \text{AD}(I)}(I)$

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

- \( Q(x) = \forall y. \text{Likes}(x,y) \) is domain dependent
  - Suppose Likes = \{ (d1,b1), (d1,b2) \}
  - What if we evaluate y over \{ b1, b2 \}?
  - What about \{ b1, b2, b3 \}?

- \( Q(x) = \exists y. \text{Likes}(x,y) \) is domain independent
  - What if we evaluate y over \{ b1, b2 \}?
  - What about \{ b1, b2, b3 \}?

- \( Q(x) = \text{IsBar}(x) \land \forall y. \text{Serves}(x,y) \Rightarrow \text{IsBeer}(y) \) 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 \}?
Domain Independence

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

A2(x,y) = Likes("Fred", x) ∨ Serves("Bar", y)

A3(x) = ∇ y. Serves(x,y)

Lesson: make sure your RC queries are domain independent
Datalog

• Book: 5.3, 5.4
• Query Language primer on website
What is Datalog?

• Another query language for relational model
  – Simple and elegant
  – Initially designed for recursive queries
• Today:
  – Some companies use datalog for data analytics, e.g., LogicBlox
  – Increased interest due to recursive analytics
• We discuss only recursion-free or non-recursive datalog and add negation
Why Do We Learn Datalog?

- A query language that is closest to mathematical logic
  - Good language to reason about query properties
- Datalog can be translated to SQL (practice at home!)
  - Helps to express complex SQL as we will see next lecture
  - Can also translate back and forth between datalog and RA

- Fact: relational algebra, non-recursive datalog with negation, and relational calculus all have the same expressive power!
USE AdventureWorks2008R2;
GO
WITH DirectReports (ManagerID, EmployeeID, Title, DeptID, Level) AS
(
    -- Anchor member definition
    SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID, 0 AS Level
    FROM dbo.MyEmployees AS e
    INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh
        ON e.EmployeeID = edh.BusinessEntityID AND edh.EndDate IS NULL
    WHERE ManagerID IS NULL
    UNION ALL
    -- Recursive member definition
    SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID, Level + 1
    FROM dbo.MyEmployees AS e
    INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh
        ON e.EmployeeID = edh.BusinessEntityID AND edh.EndDate IS NULL
    INNER JOIN DirectReports AS d
        ON e.ManagerID = d.EmployeeID
)
-- Statement that executes the CTE
SELECT ManagerID, EmployeeID, Title, DeptID, Level
FROM DirectReports
INNER JOIN HumanResources.Department AS dp
    ON DirectReports.DeptID = dp.DepartmentID
WHERE dp.GroupName = 'Sales and Marketing' OR Level = 0;
GO

DirectReports(eid, 0) :-
    Employee(eid),
    not Manages(_, eid)
DirectReports(eid, level+1) :-
    DirectReports(mid, level),
    Manages(mid, eid)

SQL Query vs Datalog
(which would you rather write?)
Datalog

We do not run datalog in 344; to try out on you own:

• Download DLV (http://www.dbai.tuwien.ac.at/proj/dlv/)
• Run DLV on this file
• Can also try IRIS
  (http://www.iris-reasoner.org/demo)

parent(william, john).
parent(john, james).
parent(james, bill).
parent(sue, bill).
parent(james, carol).
parent(sue, carol).
male(john).
male(james).
female(sue).
male(bill).
female(carol).

grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
father(X, Y) :- parent(X, Y), male(X).
mother(X, Y) :- parent(X, Y), female(X).
brother(X, Y) :- parent(P, X), parent(P, Y), male(X), X != Y.
sister(X, Y) :- parent(P, X), parent(P, Y), female(X), X != Y.
Datalog: Facts and Rules

**Facts** = tuples in the database

Actor(344759, ‘Douglas’, ‘Fowley’).
Casts(344759, 29851).
Casts(355713, 29000).
Movie(29445, ‘Ave Maria’, 1940).

**Rules** = queries

Q1(y) :- Movie(x,y,z), z='1940'.
Q2(f, l) :- Actor(z,f,l), Casts(z,x), Movie(x,y,’1940’).
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
        Casts(z,x2), Movie(x2,y2,1940)

**Extensional Database Predicates** = EDB = Actor, Casts, Movie
**Intensional Database Predicates** = IDB = Q1, Q2, Q3

No need for ∃x ∃z
Datalog: Terminology

\[ Q2(f, l) :\quad \text{Actor}(z, f, l), \text{Casts}(z, x), \text{Movie}(x, y, '1940'). \]

- **f, l** = head variables
- **x, y, z** = existential variables
More Datalog Terminology

• $R_i(\text{args}_i)$ is called an atom, or a relational predicate
• $R_i(\text{args}_i)$ evaluates to true when relation $R_i$ contains the tuple described by $\text{args}_i$.
  – Example: $\text{Actor}(344759, \text{‘Douglas’}, \text{‘Fowley’})$ is true

• In addition to relational predicates, we can also have arithmetic predicates
  – Example: $z=\text{‘1940’}$

Q(args) :- R1(args), R2(args), ....

Your book uses:
Q(args) :- R1(args) AND R2(args) AND ....
Semantics

- Meaning of a datalog rule = a logical statement!

\[
Q1(y) :- \text{Movie}(x,y,z), z='1940'.
\]

- Means:
  - \(\forall x. \forall y. \forall z. [(\text{Movie}(x,y,z) \text{ and } z='1940') \Rightarrow Q1(y)]\)
  - and Q1 is the smallest relation that has this property

- Note: logically equivalent to:
  - \(\forall y. [(\exists x. \exists z. \text{Movie}(x,y,z) \text{ and } z='1940') \Rightarrow Q1(y)]\)
  - That's why vars not in head are called "existential variables".
Datalog program

A datalog program is a collection of one or more rules
Each rule tells us how to infer the contents of relations from others

Example: Find all actors with Bacon number ≤ 2

B0(x) :- Actor(x,'Kevin', 'Bacon')
B1(x) :- Actor(x,f,l), Casts(x,z), Casts(y,z), B0(y)
B2(x) :- Actor(x,f,l), Casts(x,z), Casts(y,z), B1(y)
Q4(x) :- B0(x)
Q4(x) :- B2(x)

Note: Q4 means the union of B0 and B2
We actually don’t need Q4(x) :- B0(x)
Recursive Datalog

• In datalog, rules can be recursive

Path(x, y) :- Edge(x, y).
Path(x, y) :- Path(x, z), Edge (z, y).

• We study only on non-recursive datalog

Edge encodes a graph
Path finds all paths
Datalog with negation

Find all actors who do not have a Bacon number < 2

B0(x) :- Actor(x,'Kevin', 'Bacon')
B1(x) :- Actor(x,f,l), Casts(x,z), Casts(y,z), B0(y)
Q6(x) :- Actor(x,f,l), not B1(x), not B0(x)
Here are *unsafe* datalog rules. What’s “unsafe” about them?

\[
U1(x,y) :- \text{Movie}(x,z,1994), y > 1910
\]

\[
U2(x) :- \text{Movie}(x,z,1994), \text{not} \text{Casts}(u,x)
\]

A datalog rule is *safe* if every variable appears in some positive relational atom.

Simpler than in relational calculus.