Introduction to Data Management
CSE 344

Lecture 14: Datalog
(guest lecturer Dan Suciu)
Announcements

• WQ 4 and HW 4 due on Thursday

• Midterm next Monday in class

• This week:
  – Today: RC + Datalog
  – Wed: NoSQL (a taste of non relational data model)
  – Thurs: Midterm review
Query $Q$:

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

Relational predicate $P$ is a formula given by this grammar:

$$P ::= \text{atom} | P \land P | P \lor P | P \Rightarrow P | \text{not}(P) | \forall x.P | \exists x.P$$

Atomic predicate is either a relational or interpreted predicate:

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

$R(x,y)$ means $(x,y)$ is in $R$
Review: Domain independence

• Consider $Q(x) = \neg R(x)$  
  $R(x)$ contains $\{1,2\}$
  – If domain of $x$ is $\{1,2,3,4\}$ then $Q(x) = ?$
  – What if domain of $x$ is $\{1,2,3,4,5\}$?

• Definition: a query is **domain independent** if it returns the same result regardless of its variables’ domain

• How to fix? Bound the range of $x$!
  – $Q(x) = \neg R(x) \land S(x)$  (many other possibilities)
Review: Domain independence

• Let’s be a bit more formal:
  – Active domain of Q consists of all tuples from the relation instances referenced in Q (along with any constants in Q)
  – Ex: \( AD(Q) = \{1,2\} \) for query on previous slide

• If \( \text{eval}(Q, AD(Q)) = \text{eval}(Q, D) \), where D is a domain that is larger than or equal to \( AD(Q) \), then Q is domain independent (dependent otherwise)

• We will see an analog of this in Datalog

• Make sure you understand the examples from last lecture!
Datalog
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
- Or IRIS
  http://www.iris-reasoner.org/demo
- Or pydatalog
  (https://sites.google.com/site/pydatalog/home)
- Or DrRacket
  http://www.racket-lang.org/

```prolog
parent(william, john).
president(john, james).
president(james, bill).
president(sue, bill).
president(james, carol).
president(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

Rules = queries

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

Actor(344759, ‘Douglas’, ‘Fowley’).
Casts(344759, 29851).
Casts(355713, 29000).
Movie(29445, ‘Ave Maria’, 1940).
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’.
Datalog: Facts and Rules

Facts = tuples in the database

```
Actor(344759, 'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :- Movie(x, y, z), z='1940'.
```

Find Movies made in 1940
Datalog: Facts and Rules

Facts = tuples in the database

Actor(344759, 'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).

Rules = queries

Q1(y) :- Movie(x, y, z), z='1940'.

Find Movies made in 1940
Datalog: Facts and Rules

**Facts** = tuples in the database

- Actor(344759, ‘Douglas’, ‘Fowley’).
- Casts(344759, 29851).
- Casts(355713, 29000).

**Rules** = queries

- Q1(y) :- Movie(x, y, z), z=‘1940’.
- Q2(f, l) :- Actor(z, f, l), Casts(z, x), Movie(x, y, ’1940’).
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’).

Find Actors who acted in Movies made in 1940
Datalog: Facts and Rules

Facts = tuples in the database

Actor(344759, 'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
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)
Datalog: Facts and Rules

**Facts** = tuples in the database

- Actor(344759, ‘Douglas’, ‘Fowley’).
- Casts(344759, 29851).
- Casts(355713, 29000).

**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).

Find Actors who acted in a Movie in 1940 and in one in 1910.
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
Datalog: Terminology

Q2(f, l) :- Actor(z,f,l), Casts(z,x), 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: $z=\text{‘1940’}$.
Semantics

- Meaning of a datalog rule = a logical statement!

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

- 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".
A datalog program is a collection of one or more rules. Each rule tells how to infer its 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) :- B1(x)
Q4(x) :- B2(x)

Note: Q4 means the **union** of B0 and B2. We actually don’t need Q4(x) :- B0(x) and Q4(x) :- B1(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 $\geq 2$

\[
\begin{align*}
B0(x) & : \text{Actor(x,'Kevin', 'Bacon')} \\
B1(x) & : \text{Actor(x,f,l), Casts(x,z), Casts(y,z), B0(y)} \\
Q6(x) & : \text{Actor(x,f,l), not B1(x), not B0(x)}
\end{align*}
\]
Here are *unsafe* datalog rules. What’s “unsafe” about them?

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

\[ U2(x) :\neg \text{Movie}(x,z,1994), \neg \text{Casts}(u,x) \]

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

Simpler than in relational calculus.
RA to Datalog by Examples

Union \( R(A,B,C) \cup S(D,E,F) \)

\( U(x,y,z) \) :- \( R(x,y,z) \)
\( U(x,y,z) \) :- \( S(x,y,z) \)
RA to Datalog by Examples

Intersection $R(A,B,C) \cap S(D,E,F)$

$I(x,y,z) :- R(x,y,z), S(x,y,z)$
RA to Datalog by Examples

Selection: $\sigma_{x>100 \text{ and } y='foo'} (R)$
$L(x,y,z) :\sim R(x,y,z), x > 100, y='foo'$

Selection $\sigma_{x>100 \text{ or } y='foo'} (R)$
$L(x,y,z) :\sim R(x,y,z), x > 100$
$L(x,y,z) :\sim R(x,y,z), y='foo'$
RA to Datalog by Examples

Equi-join: \( R \bowtie_{R.A=S.D \text{ and } R.B=S.E} S \)

\[
J(x,y,z,q) :: R(x,y,z), \ S(x,y,q)
\]
Projection

\[ P(x) \text{ :- } R(x,y,z) \]
RA to Datalog by Examples

To express difference, we add negation

\[ D(x, y, z) :- R(x, y, z), \text{ NOT } S(x, y, z) \]
Examples

R(A,B,C)
S(D,E,F)
T(G,H)

Translate: $\Pi_A(\sigma_{B=3}(R))$

A(a) :- R(a,3,_)  
Underscore used to denote an "anonymous variable"
Each such variable is unique
Examples

R(A,B,C)
S(D,E,F)
T(G,H)

Translate: $\Pi_A(\sigma_{B=3} (R) \bowtie_{R.A=S.D} \sigma_{E=5} (S) )$

A(a) :- R(a,3,_), S(a,5,_)
Datalog Summary

- EDB (base relations) and IDB (derived relations)
- Datalog program = set of rules
- Datalog is recursive
  - But we only focused on non-recursive datalog

- Some reminders about semantics:
  - Multiple atoms in a rule mean join (or intersection)
  - Variables with the same name are join variables
  - Multiple rules with same head mean union
Using what we have learned

How to write a complex SQL query:
• Write it in RC
• Translate RC to datalog
• Translate datalog to SQL

Take shortcuts when you know what you’re doing