Introduction to Data Management CSE 344

Lecture 13: Datalog

Guest lecturer: Laurel Orr

Midterm

- Monday, February 8th in class
- Content
  - Lectures 1 through 13
  - Homework 1 through 4 (due Feb 10)
  - Webquiz 1 through 4 (due Feb 6)
- Closed book. No computers, phones, watches, etc.!
- Can bring one letter-sized piece of paper with notes
  - Can write on both sides

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

Datalog

- Book: 5.3, 5.4
- Query Language primer on CSE344 web

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
- Increase in datalog interest due to recursive analytics
- Interesting: relational algebra, non-recursive datalog with negation, and relational calculus all have the same expressive power!

Why Do We Learn Datalog?

Datalog, Relational Algebra, and Relational Calculus are of fundamental importance in DBMSs because

1. Sufficiently expressive to be useful in practice yet
2. Sufficiently simple to be efficiently implementable

```
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 = N'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.
```


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

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

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

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

No need for ∃x ∃z

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

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

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

No need for ∃x ∃z

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

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

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

No need for ∃x ∃z

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

head body

atom atom atom (aka subgoal)

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

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

Book writes:
Q(args) :- R1(args) AND R2(args) AND ....

- $R_i(args_i)$ is called an atom, or a relational predicate
- $R_i(args_i)$ evaluates to true when relation R_i contains the tuple described by $args_i$.
  - Example: Actor(344759,'Douglas', 'Fowley') is true
- In addition to relational predicates, we can also have arithmetic predicates
  - Example: z='1940'.

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

Semantics

- Meaning of a datalog rule = a logical statement !

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

- Means:
  - ∀x. ∀y. ∀z. [(Movie(x,y,z) and z='1940') ⇒ Q1(y)]
  - and Q1 is the smallest relation that has this property
- Note: logically equivalent to:
  - ∀ y. [(∃x.∃ z. Movie(x,y,z) and z='1940') ⇒ Q1(y)]
  - That's why vars not in head are called "existential variables".

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

Datalog program

A datalog program is a collection of one or more rules
Each rule expresses the idea that from certain combinations of tuples in certain relations, we may infer that some other tuple must be in some other relation or in the query answer
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)
```

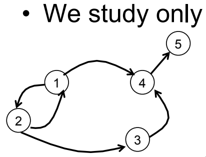
Note: Q4 means the *union* of B0 and B2
We actually don't need Q4(x) :- B0(x)

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

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

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)
```

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

Safe Datalog Rules

Here are *unsafe* datalog rules. What's "unsafe" about them ?

U1(x,y) :- Movie(x,z,1994), y>1910

U2(x) :- Movie(x,z,1994), not Casts(u,x)

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

Simpler than in relational calculus

Datalog v.s. Relational Algebra

- Every expression in the basic relational algebra can be expressed as a Datalog query
- But operations in the extended relational algebra (grouping, aggregation, and sorting) have no corresponding features in the version of datalog that we discussed today
- Similarly, datalog can express recursion, which relational algebra cannot

RA to Datalog by Examples

Schema for our examples
R(A,B,C)
S(D,E,F)
T(G,H)

RA to Datalog by Examples

Union R(A,B,C) ∪ 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) ∩ S(D,E,F)

I(x,y,z) :- R(x,y,z), S(x,y,z)

RA to Datalog by Examples

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

Selection x>100 **or** y='some string'
L(x,y,z) :- R(x,y,z), x > 100
L(x,y,z) :- R(x,y,z), y='some string'

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)

RA to Datalog by Examples

Projection

P(x) :- R(x,y,z)

RA to Datalog by Examples

To express difference, we add negation

D(x,y,z) :- R(x,y,z), 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", a variable that appears only once.

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,_)

Friend(name1, name2)
Enemy(name1, name2)

More Examples

Find Joe's friends, and Joe's friends of friends.

A(x) :- Friend('Joe', x)
A(x) :- Friend('Joe', z), Friend(z, x)

Friend(name1, name2)
Enemy(name1, name2)

More Examples

Find all of Joe's friends who do not have any friends except for Joe:

```
JoeFriends(x) :- Friend('Joe',x)
NonAns(x) :- JoeFriends(x), Friend(x,y), y != 'Joe'
A(x) :- JoeFriends(x), NOT NonAns(x)
```


Friend(name1, name2)
Enemy(name1, name2)

More Examples

Find all people such that all their enemies' enemies are their friends

- Q: if someone doesn't have any enemies nor friends, do we want them in the answer?
- A: Yes!

```
Everyone(x) :- Friend(x,y)
Everyone(x) :- Friend(y,x)
Everyone(x) :- Enemy(x,y)
Everyone(x) :- Enemy(y,x)
NonAns(x) :- Enemy(x,y),Enemy(y,z), NOT Friend(x,z)
A(x) :- Everyone(x), NOT NonAns(x)
```


Friend(name1, name2)
Enemy(name1, name2)

More Examples

Find all persons x that have a friend all of whose enemies are x's enemies.

```
Everyone(x) :- Friend(x,y)
NonAns(x) :- Friend(x,y) Enemy(y,z), NOT Enemy(x,z)
A(x) :- Everyone(x), NOT NonAns(x)
```


Datalog Summary

- EDB (base relations) and IDB (derived relations)
- Datalog program = set of rules
- Datalog is recursive
  - But we learn non-recursive datalog
- Pure datalog does not have negation; if we want negation we say “datalog+negation”
- Multiple atoms in a rule mean join (or intersection)
  - Variables with the same name are join variables
- Multiple rules with same head mean union