Introduction to Data Management
CSE 344

Lecture 11: Datalog
Announcements

• Homework 3 due next Monday, 11 pm
  – Don’t wait until the last minute – you don’t want to be part of the experiment to see if the server bogs down when everyone tries to use it at once (!)

• Homework 4 due next Thursday, 11pm
  – Fun with datalog!

• We might be moving to another room
  – Check website over weekend
Midterm!

• 2/9 Monday, in-class
• 1 sheet of letter-size notes
  – Can write on both sides
• Review session: 2/8 Sunday, 3-5pm, EEB 037
• Previous quarters’ exams are online
Today

• Datalog

• Before that: join review
## Fun with Joins

<table>
<thead>
<tr>
<th>Type</th>
<th>Predicate</th>
<th>Remove duplicate columns?</th>
<th>RA expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theta</td>
<td>Anything</td>
<td>No</td>
<td>$\sigma_\theta(R \times S)$</td>
</tr>
<tr>
<td>Equi</td>
<td>Any column, must use $=$</td>
<td>Yes</td>
<td>$\pi_A(\sigma_\theta(R \times S))$</td>
</tr>
<tr>
<td>Natural</td>
<td>Common columns, must use $=$</td>
<td>Yes</td>
<td>$\pi_A(\sigma_\theta(R \times S))$</td>
</tr>
<tr>
<td>Outer (3 flavors)</td>
<td>Anything</td>
<td>Yes</td>
<td>$R \text{\Join} S$</td>
</tr>
</tbody>
</table>


Datalog

• Book: 5.3, 5.4
• Query Language primer on CSE344 web
• Initially designed for recursive queries
• Today:
  – Some companies use datalog for data analytics, e.g. LogicBlox
  – Popular notation for many CS problems
• We discuss only recursion-free or non-recursive datalog, and add negation
Datalog

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

• Download DLV from http://www.dbai.tuwien.ac.at/proj/dlv/

• Run DLV on this file:

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

male(john).
male(james).
female(sue).
males(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).

**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'.
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

<table>
<thead>
<tr>
<th>Actor(id, fname, lname)</th>
<th>Casts(pid, mid)</th>
<th>Movie(id, name, year)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(344759, 'Douglas', 'Fowley')</td>
<td>Casts(344759, 29851)</td>
<td>Movie(7909, 'A Night in Armour', 1910)</td>
</tr>
<tr>
<td>Casts(344759, 29851)</td>
<td>Casts(355713, 29000)</td>
<td>Movie(29000, 'Arizona', 1940)</td>
</tr>
<tr>
<td>Movie(7909, 'A Night in Armour', 1910)</td>
<td>Movie(29445, 'Ave Maria', 1940)</td>
<td></td>
</tr>
</tbody>
</table>

**Rules** = queries

\[ Q1(y) :- \text{Movie}(x,y,z), z='1940' \]
\[ Q2(f, l) :- \text{Actor}(z,f,l), \text{Casts}(z,x), \text{Movie}(x,y,'1940') \]
\[ Q3(f,l) :- \text{Actor}(z,f,l), \text{Casts}(z,x1), \text{Movie}(x1,y1,1910), \text{Casts}(z,x2), \text{Movie}(x2,y2,1940) \]

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).

**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

**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'**.
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 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 is the union of B0, B1, and B2
Non-recursive Datalog

- In datalog, rules can be recursive

\[
\text{Path}(x, y) :- \text{Edge}(x, y).
\]
\[
\text{Path}(x, y) :- \text{Path}(x, z), \text{Edge}(z, y).
\]

- We focus 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 \textit{unsafe} datalog rules. What’s “unsafe” about them?

\begin{align*}
\text{U1}(x,y) & : - \text{Movie}(x,z,1994), y>1910 \\
\text{U2}(x) & : - \text{Movie}(x,z,1994), \text{not} \text{Casts}(u,x)
\end{align*}

A datalog rule is \textit{safe} if every variable appears in some positive relational atom.
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) \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: $\sigma_{x>100 \text{ and } y='some string'}(R)$

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

Selection $x>100$ \textbf{or} $y='some string'$

$L(x,y,z) :- R(x,y,z), x > 100$

$L(x,y,z) :- R(x,y,z), y='some string'$