Database Systems
CSE 414
Lecture 9-10: Datalog
(Ch 5.3–5.4)

Announcements
• HW2 is due today 11pm
• WQ2 is due tomorrow 11pm
• WQ3 is due Thursday 11pm
• HW4 is posted and due on Nov. 9, 11pm

What is Datalog?
• Another query language for relational model
  – Simple and elegant
  – Initially designed for recursive queries
  – 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

SQL Query vs. Datalog
(DirectReports(eid, level+1) :-
  DirectReports(mid, level),
  Manages(mid, eid))

Why Do We Learn Datalog?
• Datalog can be translated to SQL
  – Helps to express complex queries

Datalog
• See book: 5.3 – 5.4
• See also: Query Language primer
  – article by Dan Suciu
  – covers relational calculus as well
**Why Do We Learn Datalog?**

- Datalog can be translated to SQL
  - Helps to express complex queries
- Increase in Datalog interest due to recursive analytics
- A query language that is closest to mathematical logic
  - Good language to reason about query properties
  - Can show that:
    1. Non-recursive Datalog & RA have equivalent power
    2. Recursive Datalog is strictly more powerful than RA
    3. Extended RA & SQL92 is strictly more powerful than Datalog

**Datalog**

We won’t run Datalog in 414. Try out on your own:
- Download DLV (http://www.dlvsystem.com/dlv/)
- Run DLV on this file
- Can also try IRIS

**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).

**Find Actors who acted in a Movie in 1940 and in one in 1910**

Q2(f, l) :- Actor(z, f, l), Casts(z, x1), Movie(x1, y1, 1910), Casts(z, x2), Movie(x2, y2, 1940).

**Find Movies made in 1940**

Q1(y) :- Movie(x, y, 1940).

**Some History**

Early database history:
- 60s: network data models
- 70s: relational DBMSs
- 80s: OO-DBMSs

Ullman (1988) predicts KBMSs will replace DBMSs as they replaced what came before
- KBMS: knowledge-base
- combines data & logic (inferences)

**Actually… relational DBMSs still dominate**

CSE 414 - Spring 2017
Datalog: Facts and Rules

Facts = tuples in the database
Rules = queries

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).

Datalog: Terminology

head
atom
atom (a.k.a. subgoal)

Q1(y) :- Movie(x, y, 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

More Datalog Terminology

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

- R(args) is called an atom, or a relational predicate
- R(args) evaluates to true when relation R contains the tuple described by args,
  - 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) :- Movie(x, y, z), z=1940.

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

- Note: logically equivalent to:
  - $\forall y. (\exists x. \exists z. \text{Movie}(x, y, z) \land z=1940) \Rightarrow \text{Q1}(y)$
  - 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 $\leq 2$

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, B1, & B2

Recursive Datalog

- In Datalog, rules can be recursive
  Path(x, y) :- Edge(x, y).
  Path(x, y) :- Path(x, z), Edge (z, y).
- We’ll focus 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

\[
\begin{align*}
B_0(x) & : \text{Actor}(x, 'Kevin', 'Bacon') \\
B_1(x) & : \text{Actor}(x, f, l), \text{Casts}(x, z), \text{Casts}(y, z), B_0(y) \\
Q_6(x) & : \text{Actor}(x, f, l), \text{not } B_1(x), \text{not } B_0(x)
\end{align*}
\]

Safe Datalog Rules

Here are unsafe Datalog rules. What’s “unsafe” about them?

\[
\begin{align*}
U_1(x, y) & : \text{Movie}(x, z, 1994), y > 1910 \\
U_2(x) & : \text{Movie}(x, z, 1994), \text{not } \text{Casts}(u, x)
\end{align*}
\]

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

Datalog vs. Relational Algebra

- Every expression in standard 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:

\[
\begin{align*}
R(A, B, C) \\
S(D, E, F) \\
T(G, H)
\end{align*}
\]

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

\[
\begin{align*}
U(x, y, z) & : R(x, y, z) \\
U(x, y, z) & : S(x, y, z)
\end{align*}
\]
RA to Datalog by Examples

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

$L(x, y, z) :- R(x, y, z), S(x, y, z)$

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

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

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

RA to Datalog by Examples

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

$J(x, y, z, u, v, w) :- R(x, y, z), S(u, v, w), x=u, y=v$

$J(x, y, z, w) :- R(x, y, z), S(x, y, w)$

RA to Datalog by Examples

Projection $\pi_x (R)$

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

Examples

To express set difference $R - S$, we add negation

$D(x, y, z) :- R(x, y, z), \neg S(x, y, z)$
Examples

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

Translate: \( \Pi \{ \sigma_{B=3} (R) \} \)
A(a) :- R(a, 3, _)

Underscore used to denote an "anonymous variable", a variable that appears only once.

More Examples

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

\[
\begin{align*}
A(x) & \iff \text{Friend('Joe', x)} \\
A(x) & \iff \text{Friend('Joe', y)}, \text{Friend(z, x)}
\end{align*}
\]

More Examples

Find all of Joe's friends who do not have any friends except for Joe:
- NonAns(x): all people (of Joe's friends) who have some friends who are not Joe

\[
\begin{align*}
\text{JoeFriends}(x) & \iff \text{Friend('Joe', x)} \\
\text{NonAns}(x) & \iff \text{Friend(y, x), y \neq 'Joe'} \\
A(x) & \iff \text{JoeFriends}(x), \neg \text{NonAns}(x)
\end{align*}
\]

More Examples

Find all people such that all their enemies' enemies are their friends
- NonAns(x): all people such that some of their enemies' enemies are not their friends

\[
\begin{align*}
\text{NonAns}(x) & \iff \text{Enemy(x, y), Enemy(y, z), not Friend(x, z)} \\
A(x) & \iff \text{Everyone(x), not NonAns(x)}
\end{align*}
\]

More Examples

Find all people x who have only friends all of whose enemies are x's enemies.
- NonAns(x): all people x who have some friends some of whose enemies are not x's enemies

\[
\begin{align*}
\text{NonAns}(x) & \iff \text{Friend(x, y), Enemy(y, z), not Enemy(x, z)} \\
A(x) & \iff \neg \text{NonAns(x)}
\end{align*}
\]
Datalog Summary

• facts (extensional relations) and rules (intensional relations)
  – rules can use relations, arithmetic, union, intersect, …

• As with SQL, existential quantifiers are easier
  – use negation to handle universal

• Everything expressible in RA is expressible in non-recursive Datalog and vice versa
  – recursive Datalog can express more than (extended) RA
  – extended RA can express more than recursive Datalog