

CS589 Principles of DB Systems
Winter 2011
Unit 4: Recursive Query Processing

David Maier



Goals for this Unit

- Study recursive query processing using Datalog
 - There are other data languages with recursion: QBE, G-Whiz, SQL:1999, LDL
- Models of a Datalog program
- Evaluation methods
- Negation and recursion
- Applications
- (If time) Constraints and Typing

Example Recursive Program

- Three predicates, that talk about a distributed algebra expression:
 - $Op1(Id, In, Loc)$: unary operation Id with input In is performed at location Loc
 - $Op2(Id, In1, In2, Loc)$: binary operation Id with inputs $In1$ and $In2$ is performed at location Loc
 - $Local(Id, Loc)$: expression with root Id can be evaluated completely at location Loc

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 3


Clauses

```

Local(r, node1).
Local(s, node1).
Local(q, node1).
Local(u, node2).
Local(Id, Loc) :- Op1(Id, In, Loc), Local(In, Loc).
Local(Id, Loc) :- Op2(Id, In1, In2, Loc),
                  Local(In1, Loc), Local(In2, Loc).
Local(Id, Loc) :- LocalOp(Id, Loc).
Op1(s1, r, node1).
Op1(s1 u, node2).

Op2(j1, s1, u1, node1).
Op2(u1, s, q, node1).
Op2(j2, j1, s2, node1).
    
```

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 4



Facts versus Rules


It will be useful if we can separate predicates into

- Extensional: defined by facts
- Intensional: defined by rules

Notice that in our program the `Local` predicate has both facts and rules.

How could we modify the program to get this separation?

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 5



Conventions

- Assume predicates are extensional or intensional
- If I write just the intensional rules, we will assume that predicates without rules are extensional predicates.


Can think of them as the database

```

Local(Id, Loc) :- Local0(Id, Loc)
Local(Id, Loc) :- Op1(Id, In, Loc), Local(In, Loc).
Local(Id, Loc) :- Op2(Id, In1, In2, Loc),
                  Local(In1, Loc), Local(In2, Loc).
    
```

Extensional predicates

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 6



Computed Predicates


Bancilhon and Ramakrishnan allow for computed predicates

$\text{Sum}(X, Y, Z)$: true if $X + Y = Z$ as integers

Can think of Sum as an infinite extensional predicate:

$\text{Sum}(1, 1, 2), \text{Sum}(1, 2, 3), \text{Sum}(1, 3, 4), \dots$
 $\text{Sum}(2, 1, 3), \text{Sum}(2, 2, 4), \text{Sum}(2, 3, 5), \dots$
 $\text{Sum}(3, 1, 4), \dots$
 \dots

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 7



Safety and Computed Predicates

Uses of computed predicates have required binding patterns so we get finite answers

$\text{Sum}(1, 7, 8)$ OK
 $\text{Sum}(2, 7, 10)$ OK
 $\text{Sum}(2, 7, Z)$ OK
 $\text{Sum}(X, 7, 9)$ OK
 $\text{Sum}(2, Y, 9)$ OK
 $\text{Sum}(X, 7, Z)$ Not OK
 $\text{Sum}(X, Y, Z)$ Not OK


$\text{ans}(X, Y) :- \text{sum}(X, Y, Z), \text{age}(\text{john}, X).$
b f b

$\text{ans}(X) :- \text{sum}(X, 5, Z), \text{age}(\text{john}, X).$
age(john, 12).

Note that other literals in a rule body can provide these bindings

Each computed predicate has a *natural interpretation*

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 8



Example with Computed Predicate

Local(Id, Loc, Num): expression with root Id has Num elements and can be evaluated completely at location Loc

```

Local(Id, Loc, 1) :- Local0(Id, Loc).

Local(Id, Loc, K) :- Op1(Id, In, Loc),
  Local(In, Loc, M), Sum(M, 1, K).

Local(Id, Loc, K) :- Op2(Id, In1, In2, Loc),
  Local(In1, Loc, M), Local(In2, Loc, N),
  Sum(M, N, J), Sum(J, 1, K).

```


$K = M + N + 1$

```

:- Local(j1, L, C).
   Local(j1, node1, 6).

```

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 9



Ground Instance of a Clause

Given a Datalog clause C, we get a *ground instance* of C by replacing all variables with constant *s from the program*.

Clause:

```

Local(Id, Loc, K) :- Op1(Id, In, Loc),
  Local(In, Loc, M), Sum(M, 1, K).

```

Ground instances:


```

Local(s2, node2, 2) :- Op1(s2, u, node2),
  Local(u, node2, 1), Sum(1, 1, 2).

Local(s2, node2, 2) :- Op1(s2, r, node2),
  Local(r, node2, 1), Sum(1, 1, 2).

```

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 10



Model of a Datalog Program


A *database (aka interpretation)* M for a program P is a set of facts over the predicates in P .

A database M is a model for program P if

1. It assigns each computable predicate its natural interpretation
2. For any ground instance of a clause $H :- L_1, L_2, \dots, L_n$.
if $\{L_1, L_2, \dots, L_n\} \subseteq M$ then $H \in M$.

Note that all facts of P must be in M .

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 11



A Program Can Have Many Models

```

Local(Id, Loc, 1) :- Local0(Id, Loc)
Local(Id, Loc, K) :- Op1(Id, In, Loc),
                    Local(In, Loc, M), Sum(M, 1, K).
Local(Id, Loc, K) :- Op2(Id, In1, In2, Loc),
                    Local(In1, Loc, M), Local(In2, Loc, N),
                    Sum(M, N, J), Sum(J, 1, K).
Local0(r, node1).
Local0(s, node1).
Local0(q, node1).
Local0(u, node2).
Op1(s1, r, node1).
Op1(s2, u, node2).
Op2(j1, s1, u1, node1).
Op2(u1, s, q, node1).
Op2(j2, j1, s2, node1).
Local(r, node1, 1).
Local(r, node2, 1).
Local(s, node1, 1).
Local(q, node1, 1).
Local(u, node2, 1).
Local(s1, node1, 2).
Local(u1, node1, 3).
Local(j1, node1, 6).
Local(s2, node2, 2).
Local0(r, node1).
Local0(r, node2).
Local0(s, node1).
Local0(q, node1).
Local0(u, node2).
Op1(s1, r, node1).
Op1(s2, u, node2).
Op2(j1, s1, u1, node1).
Op2(u1, s, q, node1).
Op2(j2, j1, s2, node1).
    
```

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 12

A Second Model

Program	Model
<pre> Local(Id, Loc, 1) :- Local0(Id, Loc) Local(Id, Loc, K) :- Op1(Id, In, Loc), Local(In, Loc, M), Sum(M, 1, K). Local(Id, Loc, K) :- Op2(Id, In1, In2, Loc), Local(In1, Loc, M), Local(In2, Loc, N), Sum(M, N, J), Sum(J, 1, K). Local0(r, node1). Local0(s, node1). Local0(q, node1). Local0(u, node2). Op1(s1, r, node1). Op1(s1, u, node2). Op2(j1, s1, u1, node1). Op2(u1, s, q, node1). Op2(j2, j1, s2, node1). </pre>	<pre> Local(r, node1, 1). Local(s, node1, 1). Local(q, node1, 1). Local(u, node2, 1). Local(s1, node1, 2). Local(u1, node1, 3). Local(j1, node1, 6). Local(s2, node2, 2). Local(j2, node2, 9). Local0(r, node1). Local0(s, node1). Local0(q, node1). Local0(u, node2). Op1(s1, r, node1). Op1(s1, u, node2). Op2(j1, s1, u1, node1). Op2(u1, s, q, node1). Op2(j2, j1, s2, node1). </pre>

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 13

Closure Under Intersection

Lemma: If M1 and M2 are both models for Datalog program P, then so is $M1 \cap M2$.

Proof: Let $M = M1 \cap M2$. Both M1 and M2 give any computed predicate its natural interpretation, so part 1. of model definition is covered.

How could M fail to be a model of P?

Must have a ground instance of a clause:

$H :- L1, L2, \dots, Ln.$

Where $\{L1, L2, \dots, Ln\} \subseteq M$ but $H \notin M$.

So H must be missing from M1 or M2 (or both).

Suppose $H \notin M1$. $\{L1, L2, \dots, Ln\} \subseteq M1$
 So M1 violates that the ground instance

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 14

Example Intersection

M1

```

Local(r, node1, 1).
Local(s, node1, 1).
Local(q, node1, 1).
Local(u, node2, 1).
Local(s1, node1, 2).
Local(u1, node1, 3).
Local(j1, node1, 6).
Local(s2, node2, 2).
- Local(j2, node2, 9).
Local0(r, node1).
Local0(s, node1).
Local0(q, node1).
Local0(u, node2).
Op1(s1, r, node1).
Op1(s2, u, node2).
Op2(j1, s1, u1, node1).
Op2(u1, s, q, node1).
Op2(j2, j1, s2, node1).
                
```

M₁ ∩ M₂

M2

```

Local(r, node1, 1).
Local(r, node2, 1).
Local(s, node1, 1).
Local(q, node1, 1).
Local(u, node2, 1).
Local(s1, node1, 2).
Local(u1, node1, 3).
Local(j1, node1, 6).
Local(s2, node2, 2).
Local0(r, node1).
Local0(r, node2).
Local0(s, node1).
Local0(q, node1).
Local0(u, node2).
Op1(s1, r, node1).
Op1(s2, u, node2).
Op2(j1, s1, u1, node1).
Op2(u1, s, q, node1).
Op2(j2, j1, s2, node1).
                
```

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier

15

Minimal Model

A model M for program P is *minimal* if no proper subset of M is also a model.

Every Datalog program P (without negation) has a unique minimal model.


Consider two different minimal models N1 and N2 for P. Then $N = N1 \cap N2$ is also a model, and is a proper subset of at least one of them.

Minimum model of P: Unique minimal model.

Every Datalog program (w/o negation) has a model, so it has a minimum model

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier

16




Minimum Model and Queries

When we answer queries against Datalog program P , we want to answer them against the minimal model. (*Closed-World Assumption*)

`:- Local(j1, L, C).`

We want all `Local` facts in the minimum model where the first component is `j1`.


CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 17



Evaluation Methods

- Oblivious: Compute the minimum model, then look up query answers.
- Directed: Use information from the query to reduce the portion of the minimum model computed to answer the query.


CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 18



Naïve Evaluation of P

A “bottom-up” or “right-to-left” method.
 Have a current set of facts f (a partial model).
 Use clauses in P in a right-to-left manner to
 derive additional facts that must also be in a
 minimum model for P .

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 19



Example Right-to-Left Derivation

Consider:

```
Local(Id, Loc, K) :- Op1(Id, In, Loc),
                    Local(In, Loc, M), Sum(M, 1, K).
```

Suppose we have in f :

```
Op1(a6, p5, node2). Local(p5, node2, 2).
Op1(s4, u3, node1). Local(u2, node2, 3).
Op1(a7, r, node1). Local(r, node1, 1).
Op1(s9, a7, node1). ...
```

...

Can derive: *Local(a6, node2, 3).*
Local(a7, node1, 2).
Local(s9, node1, 3) ? ← takes 2 uses of rule

CS 589 Princ of DB Systems, Winter 2011 ©Lois Delcambre, David Maier 20



Immediate Consequence Operator

If P is a Datalog program, the *immediate consequence operator* $I_P(f)$ computes all facts derivable from the set of facts f by a single application of the clauses in P .

Note: If f is a subset of the minimum model of P , then so is $I_P(f)$, $I_P(I_P(f))$, $I_P(I_P(I_P(f)))$, ...

So start the process with $f = \emptyset$, because we know we have a subset of the minimum model.

Inflationary semantics of program P : Start with empty set of facts, apply I_P until no change.



Naïve Evaluation

Naïve(P , q)

$f_{old} = \emptyset$

$f = I_P(f_{old})$

while $f \neq f_{old}$ **do**

$f_{old} = f$

$f = I_P(f)$

return match(q , f)

Naïve always halts on programs that don't use computed predicates, with f as the minimum model.