



CS589 Principles of DB Systems Spring 2016 Lecture 1-3: Introduction to Datalog

David Maier



Goals for this lecture

- Introduce you to Datalog queries
- Briefly introduce the various versions of Datalog
- Explain how Datalog queries are interpreted
We will consider *efficient* interpretation later

Datalog – a query language based on definite Horn logic clauses

- Datalog is a query language
Related to domain calculus
- A Datalog program consists of one or more clauses (also called rules)
- Datalog syntax is the same as Prolog but without functions and without the extralogical features such as Cut and Fail.
- The order of clauses does not matter in Datalog; the order of literals in the body of a rule does not matter.

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

3

Example Datalog Query

Student(s-id, s-name, major, f-id, age)
Faculty(f-id, f-name, rank, dept)

Literals separated by commas are "ANDed" together

Query 1:
Result (x, y) :- Student(s, x, m, f, a), Faculty(f, y, r, d).

Choosing variables for answer (project)

"if"

join

An equivalent query in domain calculus:

$\{x, y \mid \exists s(\exists m(\exists f(\exists a(\exists d(\text{Student}(s, x, m, f, a) \wedge \text{Faculty}(f, y, r, d))))))\}$

An equivalent query in relational algebra:

$\pi_{s\text{-name}, f\text{-name}}(\text{Student} \bowtie \text{Faculty})$

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

4



Example Datalog Query 2

Student(s-id, s-name, major, f-id, age)
Faculty(f-id, f-name, rank, dept)

Choosing variables for answer (project)
Query 2:
Result (x,y) :- Student(x,y,"CS",f,a).

Constant of "CS" selects students with major = "CS"

An equivalent query in domain calculus:

$\{x, y \mid \exists f(\exists a(\text{Student}(x,y,\text{"CS"},f,a)))\}$

An equivalent query in relational algebra:

$\pi_{s\text{-id}, s\text{-name}}(\sigma_{\text{major}=\text{"CS"}}(\text{Student}))$

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

5



Exercise

Student(s-id, s-name, major, f-id, age)
Faculty(f-id, f-name, rank, dept)

Query 3:
Answer(x) :- Student(a,x,b,c,21),Faculty(c,d,e,"CS").

Write an equivalent query in domain or tuple calculus:

Write an equivalent query in relational algebra:

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

6



Comparing domain calculus & Datalog

Student(s-id, s-name, major, f-id, age)

$\{ a, b, c \mid \exists x(\exists y(\text{Student}(a,b,c,x,y))) \}$

This expression defines a set (query answer). The tuple $\langle a,b,c \rangle$ is in this set provided there exists an x and a y where the tuple $\langle a,b,c,x,y \rangle$ is in relation **student** (with relation schema **Student**).

Result(a,b,c) :- Student(a,b,c,x,y).

This is a definite Horn clause that says

"**Result(a,b,c)** is true if **Student(a,b,c,x,y)** is true."

In a Horn clause, every variable is universally quantified.

This clause is the same as:

$(\forall a)(\forall b)(\forall c)(\forall x)(\forall y)(\text{Student}(a,b,c,x,y) \rightarrow \text{Result}(a,b,c))$

So ... are these expressions defining the same query?

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

7



Implication (quick reminder)

Any logical connector can be defined using a truth table.

Here we show the truth table for $\wedge$ (and) and for $\rightarrow$ (implication).

p	q	$p \wedge q$
true	true	true
true	false	false
false	true	false
false	false	false

p	q	$p \rightarrow q$
true	true	true
true	false	false
false	true	true
false	false	true

$(\forall a)(\forall b)(\forall c)(\forall x)(\forall y)(\text{Student}(a,b,c,x,y) \rightarrow \text{Result}(a,b,c))$

If ever the left part is true, then the right part must be true.

Result(a,b,c) :- Student(a,b,c,x,y).

If the body (the right hand side) is true, then the head (left hand side) must be true.

Evaluation of Datalog actively looks for tuples that satisfy the body.

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

8



Datalog Example – for Union

Student(s-id, name, major, f-id, age)
Faculty(f-id, name, rank, dept)

Result(x,y) :- Student(x,y,a,b,c).

Result(x,y) :- Faculty(x,y,d,e).

This is a Datalog program consisting of two rules. They both produce **Result** tuples.

This query is equivalent to the following:

$\pi_{s-id, name}(Student) \cup \pi_{f-id, name}(Faculty)$

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

9



Exercise

Grad-course (c-num, title, credits)

Undergrad-course (c-num, title, credits)

Write a Datalog query that is equivalent to:

$(Grad-course) \cap (Undergrad-course)$

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

10



Datalog with negation

Student(s-id, name, major, f-id, age)

Faculty(f-id, name, rank, dept)

No-advisees(x,y) :- Faculty(x,y,a,b), ¬Student(c,d,e,x,f).

Find the f-id and name for any faculty tuple for which there does not exist a Student tuple advised by this faculty member.

What is an equivalent relational algebra query?



Exercise

Write each of these queries in Datalog.

Student(s-id, s-name, major, f-id, age)

Faculty(f-id, f-name, rank, dept)

1. $(\pi_{\text{name}}(\sigma_{\text{age}=21}\text{Student})) \cup \pi_{\text{name}}\text{Faculty}$
2. $\text{Student} \bowtie \text{Faculty}$
3. $(\pi_{\text{name}}(\sigma_{\text{age}=21}\text{Student})) - (\pi_{\text{name}}(\sigma_{\text{major}=\text{"CS"}}\text{Student}))$



Datalog Facts

- An empty body is interpreted as true. So
Student(126,"Hector Ng", "CS", 146, 23) :- .
means
true $\rightarrow$ Student(126,"Hector Ng", "CS", 146, 23)
that is, Student is true for these values
- Abbreviated to
Student(126,"Hector Ng", "CS", 146, 23).
Called a *fact* (or *ground fact*, if no variables)



Datalog Program

- A Datalog program is a collection of rules (some could be facts)
- Will usually have a special relation name (e.g., Result, Answer) we are interested in
Result(d) :- Course("CS",n,"Winter"),
Prereq(d,m,"CS",n),
Course(d,m,"Fall").
Course("CS",311,"Winter").
Course("CS",312,"Spring").
Course("Math",119,"Fall").
Prereq("Math",119,"CS",311).
Prereq("CS",311,"CS",312).



Database Perspective

Can think of ground facts as stored database.

Extensional DB

Can think of rules as view over stored data (and other views)

Intensional DB



Interpreting a Datalog Program

Treat a rule as representing all its *ground instances*

substitute a value for each variable symbol

```
Result("Math") :- Course("CS",311,"Winter"),  
                  Prereq("Math",119,"CS",311),  
                  Course("Math",119,"Fall").
```

```
Result("CS") :- Course("CS",312,"Winter"),  
                Prereq("CS",311,"CS",312),  
                Course("CS",311,"Fall").
```

```
Result("Acorn") :- Course("CS",96557,"Winter"),  
                  Prereq("Acorn",2,"CS",96557),  
                  Course("Acorn",2,"Fall").
```

Generally restrict to a *safe* substitution: Only values in program



Derived Database

Start with a Datalog program P

- Start with Der = all ground facts from P
- Add any tuple to Der that is the head of a ground instance of a rule in P where all predicates in the body are already in Der .

Will return to Datalog later to talk about efficient ways to compute the derived database of a program.



Datalog syntax

Atomic formulas:

$R(y_1, y_2, \dots, y_k)$ – a predicate formula

$x = y$ (Note: this is syntactic sugar for $\text{equal}(x, y)$.)

$R(v_1, v_2, \dots, v_k)$ – a ground atomic formula, where v_i are values

Literal:

an atomic formula (positive literal) or

the negation of an atomic formula (negative literal): $\neg A$

Clause (Datalog rule):

$L :- L_1, L_2, \dots, L_n.$

where L is a predicate formula and $L_i, i = 1, \dots, n$, is a literal.

(Some versions of Datalog require all literals to be positive literals.) The comma means “and”.



Datalog with recursion (more about this in a future lecture)

Parent-child(p-id, ch-id)

Ancestor(x,y) :- Parent-child(x, y).

Ancestor(x,z) :- Ancestor(x,y), Parent-child(y,z).

How does this Datalog program get evaluated?

Keep building the derived database until no new tuples get added to Ancestor.

The book describes the meaning of a program using the "immediate consequence" of a program.

Note each Datalog rule is independent. The variable names in separate rules have no connection.

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

19



Expressive power of Datalog languages (compared to relational algebra)

- Datalog – one rule, no negation, no recursion.
Conjunctive queries SPJ
- Datalog – multiple rules, no negation, no recursion. SPJU
- Datalog – multiple rules, no negation, with recursion. SPJU+ recursion but NOT relationally complete
- Datalog – multiple rules, with negation, no recursion. SPJU- relationally complete but no recursion
- Datalog – multiple rules, with negation, with recursion. Relationally complete plus recursion, but some queries are ambiguous!

CS 589 Princ. of DB Systems, Spring 2016 © David Maier

20