

CS589 Principles of DB Systems

Fall 2008

Lecture 4a: Introduction to Datalog

Lois Delcambre

imd@cs.pdx.edu

503 725-2405



Goals for today

- Introduce you to Datalog queries
- Briefly introduce the various versions of Datalog
- Explain how Datalog queries are evaluated



Datalog – a query language based on definite Horn logic clauses

- Datalog is a query language
- 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; the query answer will be the same.

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

Choosing variables for answer (project)

Query 1:

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

join

Write an equivalent query in domain calculus:

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

Write an equivalent query in relational algebra:

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

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 $(x, y) :- \text{Student}(x, y, \text{"CS"}, f, a).$

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

Write an equivalent query in domain calculus:

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

Write an equivalent query in relational algebra:

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



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 calculus:

Write an equivalent query in relational algebra:



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 $(a,b,c) :- \text{Student}(a,b,c,x,y).$

This is a definite Horn clause that says

“If $\text{Student}(a,b,c,x,y)$ is true, then $\text{Result}(a,b,c)$ 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?

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.



Exercise

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

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

Write a Datalog query that is equivalent to the following
Relational Algebra query:

$$\Pi_{s\text{-name}}(\sigma_{s\text{-age}=21}\text{Student}) \cup \Pi_{f\text{-name}}(\sigma_{\text{rank}=\text{"Professor"}}\text{Faculty})$$



Datalog Example – for Union

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

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

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

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

This is a Datalog program consisting of two rules. They both fire and produce **Ans** tuples.

This query is equivalent to the following:

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



Exercise

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

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

Write a Datalog query that is equivalent to:

```
SELECT      *  
FROM        Student, Faculty;
```



Exercise

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

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

Write a Datalog query that is equivalent to:
 $(\text{Grad-course}) \cap (\text{Undergrad-course})$



Exercise

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

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

Write a Datalog query that is equivalent to:

Grad-course – Undergrad-course



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), \neg Student(c,d,e,x,f).

Show 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. $(\pi_{\text{name} \neq \text{age}=21} \text{Student}) \cup \pi_{\text{name}} \text{Faculty}$
2. Student |X| Faculty
3. $(\pi_{\text{name} \neq \text{age}=21} \text{Student}) - (\pi_{\text{name} \neq \text{major}=\text{"CS"}} \text{Student})$



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)

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(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?

Fire all rules (from right to left) until you don't produce any new tuples in Academic-descendant.

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.



Several Datalog languages

- Datalog – one rule, no negation, no recursion.
- Datalog – multiple rules, no negation, no recursion.
- Datalog – multiple rules, no negation, with recursion.
- Datalog – multiple rules, with negation, no recursion.
- Datalog – multiple rules, with negation, with recursion.

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 with recursion but some queries are ambiguous!**



How is Datalog evaluated?

- Single rule (non-recursive) Datalog without negation
- (Non-recursive) Datalog without negation
- (Non-recursive) Datalog with negation
- Recursive Datalog without negation (see next slide)
- Recursive Datalog with negation ... a whole different topic