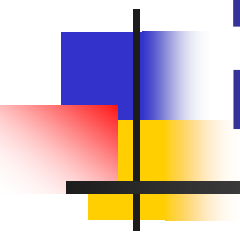
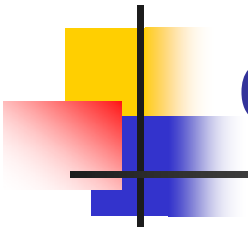


CS589 Principles of DB Systems
Fall 2008
Lecture 4a: Introduction to Datalog

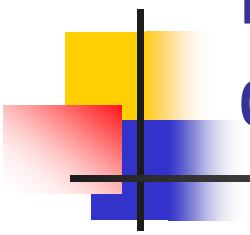
Lois Delcambre
lmd@cs.pdx.edu
503 725-2405



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

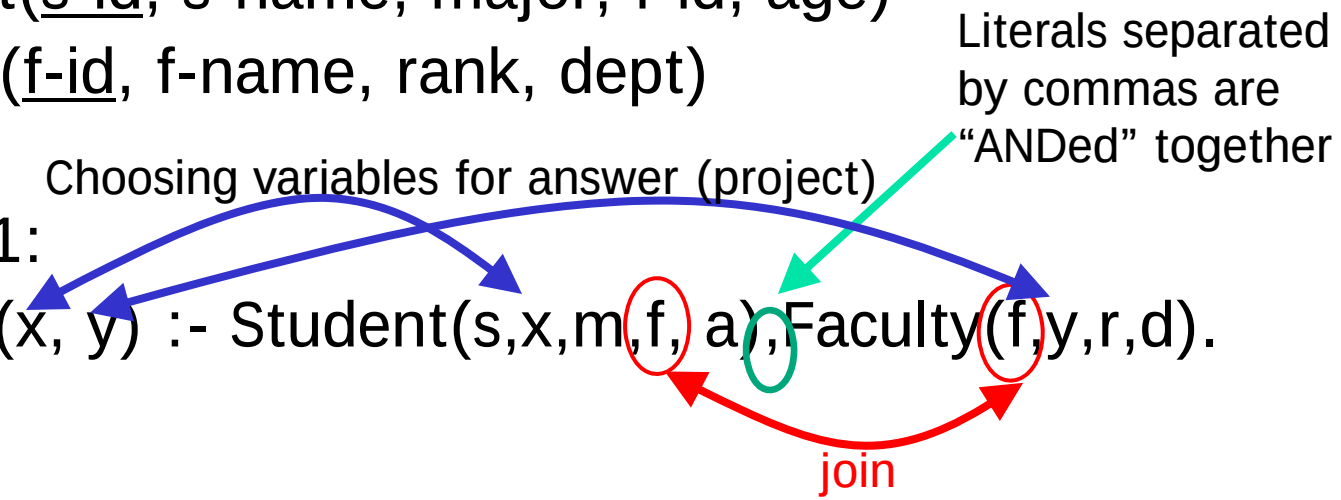


- Datalog is a query language
- A Datalog program consists of one or more clauses (also called rules)
- Datalog syntax is the same as Prolog but without functions and without the extralogical features such as Cut and Fail.
- The order of clauses does not matter in Datalog; the order of literals in the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



Write an equivalent query in domain calculus:

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

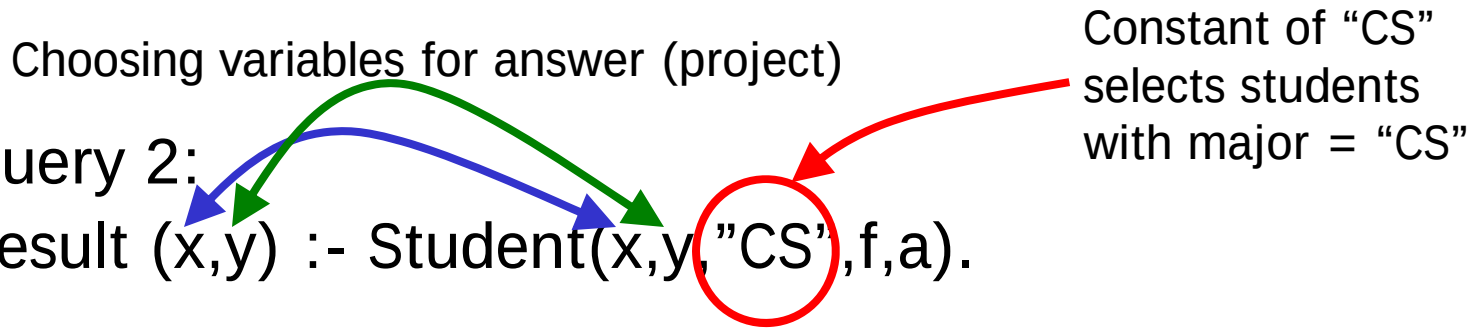
Write an equivalent query in relational algebra:

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

Example Datalog Query 2

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

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



Query 2:

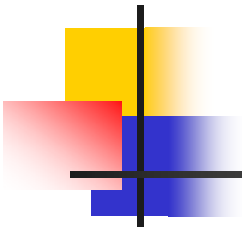
Result (x,y) :- Student(x,y,"CS",f,a).

Write an equivalent query in domain calculus:

{x, y | Student(x, y,"CS",f, a) }

Write an equivalent query in relational algebra:

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



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 | ∃x ∃y Student(a,b,c,x,y) }

This expression defines a set (query answer). The tuple <a,b,c> is in this set provided there exists an x and a y where the tuple <a,b,c,x,y> 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

"If Student(a,b,c,x,y) is true, then 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)

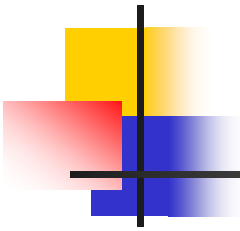
Any logical connector can be defined using a truth table.
Here we show the truth table for ^ (and) and for ⇒ (implication).

p	q	p^q
true	true	true
true	false	false
false	true	false
false	false	false

p	q	p⇒q
true	true	true
true	false	false
false	true	true
false	false	true

$(\forall a)(\forall b)(\forall c)(\forall x)(\forall y)(Student(a,b,c,x,y) \rightarrow Result(a,b,c))$
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:

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

Datalog Example – for Union

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

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

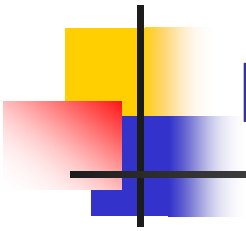
Ans(x,y) :- Student(x,y,a,b,c).

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:

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



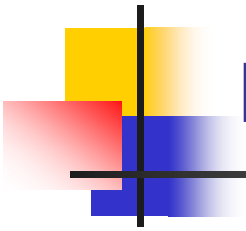
Exercise

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;
```



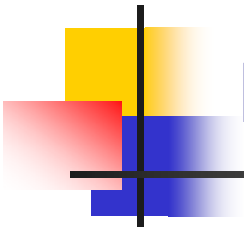
Exercise

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

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

Write a Datalog query that is equivalent to:

(Grad-course) $\cap$ (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),¬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. $(\pi_{name} \not\forall_{age=21} Student)) \cup \pi_{name} Faculty$
2. Student |X| Faculty
3. $(\pi_{name} \not\forall_{age=21} Student)) - (\pi_{name} \not\forall_{major=\text{“CS”}} Student))$

Datalog syntax

Atomic formulas:

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

$x = y$ (Note: this is syntatic sugar for equal(x,y).)

$R(v_1, v_2, ..., v_k)$ – a ground atomic formula

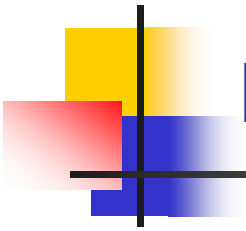
Literal:

an atomic formula (positive literal) or the negation of an atomic formula (negative literal)

Clause (Datalog rule):

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

where L is a predicate formula and Li, i = 1, ..., n, is a literal. (Some versions of Datalog require all literals to be positive literals.) The comma means “and”.



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 ....

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).

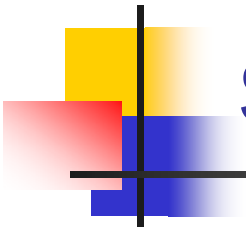
How does this Datalog program get evaluated?

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!