# Logical Query Languages

1

## Datalog

- Logical query language for the relational model
- Consists of "if-then" rules made up of *atoms*:

  - *relational* : predicates corresponding to relations
    >EDB extensional database (stored relations)
    >IDB intensional database (relations defined by rules)

  - *arithmetic*

2

## Datalog example

Example:

*database schema:*
Movie(title, year, length, inColor, studionName, producerC#)
Contracts(starName, studioName, title, year, salary)

*relational atom:* Movie (t, y, l, c, s, p)

*arithmetic atom:* l > 100

3

## Datalog Rules

**Rule:** head ← body

**Head:** a relational atom (no EDB predicates!)

**Body:** one or more atoms called *subgoals*

Example:
*datalog rule:* LongMovie(t, y) ← Movie(t,y,l,c,s,p) AND l >=10
Relational Algebra....
Relational Calculus...

4

## Interpreting Datalog Rules

**Variables:** - distinguished – appear in the head
- nondistinguished –appear in the body

**Interpreting rules**
the head is true of the *distinguished variables* if there exist values of the *non-distinguished variables* that make all subgoals of the body true.

5

## Safe Datalog Rules

A rule is *safe* if each distinguished and nondistinguished variable appears in at least one nonnegated relational atom.

**Note:** only safe rules are allowed

6

## Unsafe Datalog Rules

Example:

E(w) ⟵ NOT Movies(t, y, l, c, s, p)

Years(w) ⟵ Movies(t, y, l, c, s, p) AND w < y

**Note:** in each case an infinity of w's can satisfy the rule, even though Movies is a finite relation.

7

## Algorithms for Evaluating Datalog Rules

**Variable-based:** Consider all possible assignments to the variable of the body. If the assignment makes the body true, add the tuple for the head to the result.

**Tuple-based:** Consider all assignments of tuples from the nonnegated relational subgoals. If the assignment makes the body true, add the tuple for the head to the result.

8

## Variable-based Evaluation

Example:

Database: Edge(from, to)

| from | to |
|---|---|
| 1 | 2 |
| 2 | 3 |

Datalog rule: NotTranzitive(x,z) ⟵ Edge(x, y) AND Edge(y, z) AND NOT Edge(x, z)

Assignment x = 1, y = 2 , z = 3
Edge(1, 2) AND Edge(2, 3) AND NOT Edge(1, 3) is true, make (1, 3) a tuple of the answer
Assignment x = 1, y = 2 , z = ?
Edge(1, 2) AND Edge(2, ?) AND NOT Edge(1, ?) no z makes the body true
Assignment x = 2, y = 3 , z = ?
Edge(2, 3) AND Edge(3, ?) AND NOT Edge(2, ?) no z makes the body true

Note: No other assignment for x and y makes Edge(x, y) true. Stop searching.

9

## Tuple-based Evaluation

Example:

Database: Edge(from, to)

| from | to |
|---|---|
| 1 | 2 |
| 2 | 3 |

Datalog rule: NotTranzitive(x,z) ⟵ Edge(x, y) AND Edge(y, z) AND NOT Edge(x, z)

Assignment (x, y) = (1, 2) , (y, z) = (2, 3), consistent assignment
Edge(1, 2) AND Edge(2, 3) AND NOT Edge(1, 3) is true, make (1, 3) a tuple of the answer

Assignment (x, y) = (1, 2) , (y, z) = (1, 2), inconsistent assignment

Assignment (x, y) = (2, 3), (y, z)=(1, 2), inconsistent assignment

Assignment (x, y) = (2, 3), (y, z)=(2, 3), inconsistent assignment

Note: No other assignment for( x, y) makes Edge(x, y) true. Stop searching.

10

## Datalog Programs

A Datalog Program is a collection of rules

Example:
"Find actors who starred in the color movies made in the 1950"

MoviesColor50 (t,y)⟵ Movie(t,y,l,c,s,p) AND y = "1950" AND c = "y"
Answer(star) ⟵ Movies90(t,y) AND Contracts(star, studio, t, y, salary)

11

## Datalog Programs Evaluation

**Non-recursive programs:**
- pick an order to evaluate the rules (the IDB predicates) so that all the predicates in the body have already been evaluated.

- if an IDB predicate has more than one rule, each contributes tuples to its relation (union).

12

## From Relational Algebra to Datalog -1

**Intersection:** R(x, y) ∩ T(x, y)
I(x, y) ←R(x, y) AND T(x, y)

**Union:** R(x, y) U T(x, y)
U(x, y) ←R(x, y)
U(x, y) ←T(x, y)

**Differece:** R(x, y) –T(x, y)
D(x, y) ←R(x, y) AND NOT T(x, y)

13

## From Relational Algebra to Datalog -2

**Projection:** $\pi_x$(R)
P(x) ←R(x,y)

**Selection:** $\sigma_{x>10}$(R)
S(x, y) ←R(x, y) AND x>10

**Product:** R X T
P(x, y, z, w) ←R(x,y ) AND T(z, w)

14

## From Relational Algebra to Datalog -3

**Natural Join** R $\bowtie$ T
J(x, y, z) ← R(x, y) AND T(y, z)

**Theta Join** R $\bowtie_{R.x > T.y}$ T
J(x, y, z, w) ← R(x, y) AND T(z, w) AND x > y

15

## Datalog Queries

**Datalog Query:** a datalog program.

**Expressive Power:**
- without recursion, Datalog has the same power as Core Relational Algebra and Relational Calculus

- with recursion: much more, but not Turing-complete

16

## Recursivity

Example:
*Database:* SequelOf(movie, sequel)

*Query:* "What are the sequels of sequels of movies in the database?"

$$\pi_{first,second}(\rho_{first,second}(SequelOf) \bowtie \rho_{second,third}(SequelOf))$$

"What are the sequels of the sequels of the sequels?"

Infinite unions?

17

## Recursive Rules

Example:
FollowOn(x, y) ← SequelOf(x, y)
FollowOn(x, y) ← SequelOf(x, z) AND FollowOn(z, y)

**Dependency Graph** (of a program)
- nodes: the IDB predicates
- edges: from node1(predicate1) to node(predicate2) if and only if there is a rule with predicate1 in the head and predicate2 in the body.

A datalog program is recursive iff its dependency graph has a cycle.

18

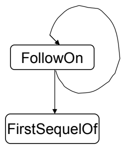
## Dependency graph

Example:

FirstSequelOf(x, y) ⟵ SequelOf(x, y)

FollowOn(x, y) ⟵ FirstSequelOf(x, y)

FollowOn(x, y) ⟵ FirstSequelOf(x, z) AND FollowOn(z, y)




Cyclic graph → recursive datalog program

19

## Evaluating Recursive Rules without Negation

**Naive algorithm**

1. Begin by assuming all IDB relations are empty
2. Repeatedly evaluate the rules using the EDB and the previous IDB to get a new IDB
3. End when there is no change to IDB

20

Example:

*Database:* SequelOf := {(t1,t2), (t2,t3), (t3,t4),(t5,t6),(t6,t7), (t7,t8)}

SequelOf

| movie | sequel |
|---|---|
| t1 | t2 |
| t2 | t3 |
| t3 | t4 |
| t5 | t6 |
| t6 | t7 |
| t7 | t8 |

FirstSequelOf(x, y) ⟵ SequelOf(x, y)

FollowOn(x, y) ⟵ FirstSequelOf(x, y)

FollowOn(x, y) ⟵ FirstSequelOf(x, z) AND FollowOn(z, y)

We will proceed in rounds to infer FirstSequel facts and then FollowOn facts.

Initial FirstSequelof:={}, FollowOn := {}

Round1 FirstSequelOf := {(t1,t2), (t2,t3), (t3,t4),(t5,t6),(t6,t7), (t7,t8)}, FollowOn ={}

Round2 FollowOn := {(t1,t2), (t2,t3), (t3,t4),(t5,t6),(t6,t7), (t7,t8)}

Round3 FollowOn := {(t1,t2), (t2,t3), (t3,t4),(t5,t6),(t6,t7), (t7,t8)} U {(t1,t3), (t2,t4), (t5,t7),(t6,t8)}

Round4 FollowOn := {(t1,t2), (t2,t3), (t3,t4),(t5,t6),(t6,t7), (t7,t8), (t1,t3), (t2,t4), (t5,t7),(t6,t8)} U {(t1,t4), (t5,t8)}

Round 5 no change in FollowOn. STOP

21

## Negation in recursive rules

- Naive evaluation does not work when there are negated subgoals.
- Arguably negation wrapped in a recursion makes little or no sense in general
- Even when negation and recursion are separate there is ambiguity sbout the “correct” IDB relations

22

Example:

EDB predicate R= {(0)}

P(x) ← R(x) AND NOT Q(x)

Q(x) ← R(x) AND NOT P(x)

2 solutions

P={(0)}, Q = Φ

P = Φ,   Q = {(0)}

Which one to choose?

23

Example:

EDB predicate S = {(1)}

R(x) ← S(x) AND NOT R(x)

Initial R := {}

Round 1 R = {(1)}

Round 2 R = {}

Round 3 R = {(1)}, etc.

24

## Stratified Negation

- Constraint imposed on recursive Datalog programs

- Rules out negation wrapped in recursion

- The maximum number of negations that can be applied to an IDB predicate used in evaluating an IDB predicate must be finite.



## Stratum Graph

Labeled dependency graph

- nodes: the IDB predicates

- edges: from node1(predicate1) to node(predicate2) if and only if there is a rule with predicate1 in the head and predicate2 in the body. If predicate2 appears negated, label the edge with “-”.



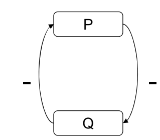
## Strata

- The *stratum* of a node (predicate) is the maximum number of “-” labeled edges on a path leading from that node .

- A Datalog program is *stratified* if al its IDB predicates have finite strata.

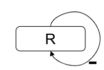


Example:




P(x) ← R(x) AND NOT Q(x)
Q(x) ← R(x) AND NOT P(x)

R(x) ← S(x) AND NOT R(x)






## Stratified Datalog Evaluation

Algorithm:

1. Evaluate IDB predicates lowest-stratum-first

2. Once evaluated, treat them as “EDB” for the IDB predicates with higher strata.



## SQL Recursion

- Datalog recursion has inspired the introduction of recursion in the SQL-99 standard.

- More difficult: SQL allows grouping and aggregation → requires a more complex notion of stratification

## SQL Recursive Queries
## Syntax

WITH
<Datalog-like rules>

<a core SQL query using the predicates in the rules >

31

- The keyword **WITH**
- One or more definitions, separated by comas, of the form:
  - the optional keyword **RECURSIVE**
  - the name of the relation being defined
  - the keyword **AS**
  - the query that defines the relation
- A query which may refer to any of the prior definitions, and forms the result of the WITH statement.

32

**Example:** "Find all Rocky's sequels"

```
WITH
FirstSequelOf(x,y) AS SELECT * FROM  SequelOf;
RECURSIVE FollowOn(x, y) AS
(SELECT * FROM FirstSequelOf)
UNION
(SELECT FirstSequelOf.x, FollowOn.y
FROM FirstSequelOf, FollowOn
WHERE FirstSequelOf.y = FollowOn.x )

SELECT y FROM FollowOn WHERE x="Rocky"
```

33

## Monotonicity

- If a relation P is a function of a relation Q, we say P is *monotone* in Q if inserting tuples into Q cannot cause any tuples to be deleted from P.

**Example:**

P = Q  UNION  R

P = SELECT * FROM Q

34

## Nonmonotonicity

Example:

Let P be the result relation of the query SELECT AVG(x) FROM Q

P is not monotone in Q: inserting a new tuple in Q may change the average and thus delete the old average.

35

## SQL Stratum Graph

Nodes - IDB relations declared in WITH clause
- Subqueries in the body of the rules (at any level of nesting)

Edges P→Q if:
- P is a rule head and Q is a relation in the FROM clause or an immediate subquery
- P is a subquery and Q is a relation in its FROM clause or an immediate subquery.

Label with "-" an edge if P is not monotone in Q

36

## Stratified SQL

Stratified SQL = finite number of “-”’s on the paths of the stratum graph

Example:

FirstSequelOf = ... ← Subquery S1

FollowOn = (... FROM FirstSequelOf)

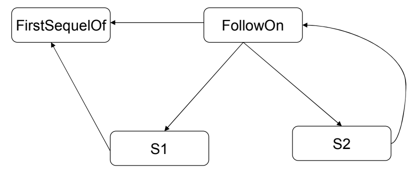
UNION ← Subquery S2

(... FROM FollowOn)



## The stratum graph

No “-” → stratified






## Nonmonotone Example

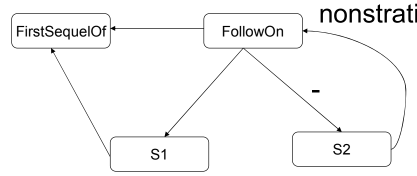
WITH
FirstSequelOf(x,y) AS SELECT * FROM SequelOf;
RECURSIVE FollowOn(x, y) AS ← Subquery S1
(SELECT * FROM FirstSequelOf)
EXCEPT ← Subquery S2
(SELECT FirstSequelOf.x, FollowOn.y
FROM FirstSequelOf, FollowOn
WHERE FirstSequelOf.y = FollowOn.x )
SELECT * FROM FollowOn

- Note: inserting a tuple intoS2 can delete a tuple from Follow on



## The Graph

cyclic graph →
nonstratified query






## Not and Nonmonotonicity

- Not every NOT means that the query is not monotone.

Example:

SELECT * FROM Q is monotone in Q

SELECT * FROM Q WHERE NOT(Q.x >10) is also monotone in Q

Note: All selections are monotone



## Example

WITH
FirstSequelOf(x,y) AS SELECT * FROM SequelOf;
RECURSIVE FollowOn(x, y) AS
(SELECT * FROM FirstSequelOf)
UNION
(SELECT FirstSequelOf.x, FollowOn.y
FROM FirstSequelOf, FollowOn
WHERE FirstSequelOf.y = FollowOn.x AND
NOT ( FirstSequelOf.x = FollowOn.y) )
SELECT * FROM FollowOn