CS422 Principles of Database Systems

Introduction to Datalog

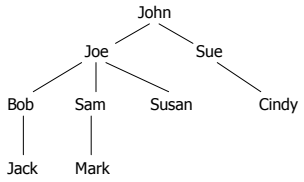
Chengyu Sun
California State University, Los Angeles

Limitation of Relational Algebra

Flight	Source	Destination
CA983	Beijing	LA
CA984	LA	Beijing
UA123	LA	New York
UA124	New York	LA
AA233	LA	Chicago
AA234	Chicago	LA
AA777	Boston	New York

- Is there a way to get from Boston to Beijing?

Another Example: Management Hierarchy



- How do we represent the management hierarchy with relations??
- How do we query about the number of employees supervised by Joe??

Datalog

- A query language inspired by Prolog
- Commonly used in *deductive* databases
- Nonrecursive datalog *rules* are equivalent to core relational algebra
- Recursive rules are used to provide recursive queries in SQL-99

Sample Schema

Bars

name	**address**
Joe's	Maple St.
Sue's	River Rd.

Sells

bar	**beer**	**price**
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

Datalog Example 1

- List the names of the bars.

Ans(name) Bars(name,address)

or

Ans(x) Bars(x,y)

or

Ans(x) Bars(x,_)

If a tuple <x,y> is in relation *Bars*, then the tuple <x> is in relation *Ans*.

Datalog Example 2

- List the names of the bars that sell Bud.

```
Ans(x)   Sells(x,y,_), y='Bud'
or
Ans(x)   Sells(x,'Bud',_)
```

Datalog Example 3

- List the names of the bars that sell Bud or Miller.

```
Ans(x)   Sells(x,y,_), y='Bud'
Ans(x)   Sells(x,y,_), y='Miller'
or
Ans(x)   Sells(x,'Bud',_)
Ans(x)   Sells(x,'Miller',_)
```

Datalog Query and Rules

- A datalog query is also called a *datalog program*, which consists of one or more *rules*.
- A rule is in the form: *head* *body*

Subgoal

- A rule head consists of one subgoal
- A rule body consists of the "AND" of one or more subgoals, e.g. *Sells(x,y,z),y='Bud'*
- There are *relational subgoals* and *arithmetic subgoal*.
  - A relational subgoal evaluates to *true* if the tuple is in the relation.
  - Arithmetic subgoals can be considered special cases of relational subgoals.
- Negated subgoal, e.g. *NOT Sells(x,y,z)*

From Relation Algebra to Datalog

Let R(A,B) and S(C,D) be two relations.

Operations	Relational Algebra	Datalog
Intersection	$R \cap S$	
Union	$R \cup S$	
Difference	$R - S$	
Selection	$\sigma_{A=10}(R)$	Ans(a) R(10,b)
Projection	$\pi_A(R)$	Ans(a) R(a,b)
Join	$R \triangleright\triangleleft_{A=C} S$	

Examples of Unsafe Rules

- S1(x) R(y,z)
- S2(x) R(y,z), x < 10
- S3(x) NOT R(x,y)

R

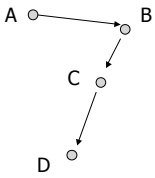
A	B
1	1
2	1
3	2
3	3
1	4

Safe Rules

- A rules is safe if
  - each variable that appears in a rule head
  - each variable in a arithmetic subgoal
  - each variable in a negated subgoal

  also appears in a *non-negated, relational subgoal*

Recursion Example



Edges

P1	P2
A	B
B	C
C	D

- Compute a relation *Paths(x,y)* – (x,y) is a tuple in *Paths* if there exists a path from x to y

Recursive Datalog Solution

- Paths(x,y) Edges(x,y)
- Paths(x,y) Paths(x,z),Edges(z,y)

Evaluation of Recursive Rules – Round 1

Edges

P1	P2
A	B
B	C
C	D

Paths

x	y

Paths(x,y) Edges(x,y)
Paths(x,y) Paths(x,z),Edges(z,y)

Evaluation of Recursive Rules – Round 1

Edges

P1	P2
A	B
B	C
C	D

Paths

x	y
A	B
B	C
C	D

Paths(x,y) Edges(x,y)
Paths(x,y) Paths(x,z),Edges(z,y)

Evaluation of Recursive Rules – Round 2

Edges

P1	P2
A	B
B	C
C	D

Paths

x	y
A	B
B	C
C	D

Paths(x,y) Edges(x,y)
Paths(x,y) Paths(x,z),Edges(z,y)

Evaluation of Recursive Rules – Round 2

Edges

P1	P2
A	B
B	C
C	D

Paths

x	y
A	B
B	C
C	D
A	C
B	D

Paths(x,y) Edges(x,y)
Paths(x,y) Paths(x,z),Edges(z,y)

Evaluation of Recursive Rules – Round 3

Edges

P1	P2
A	B
B	C
C	D

Paths

x	y
A	B
B	C
C	D
A	C
B	D

Paths(x,y) Edges(x,y)
Paths(x,y) Paths(x,z),Edges(z,y)

Evaluation of Recursive Rules – Round 3

Edges

P1	P2
A	B
B	C
C	D

Paths

x	y
A	B
B	C
C	D
A	C
B	D
A	D

Paths(x,y) Edges(x,y)
Paths(x,y) Paths(x,z),Edges(z,y)

Evaluation of Recursive Rules – Done

- No more tuples can be added to Paths – we have reaches a fixed-point.

Dependency Graph

- Form a *dependency graph* whose nodes = *computed relations* (called *IDB* in the textbook).
- Arc $X \rightarrow Y$ if and only if there is a rule with X in the head and Y in the body.
- Cycle = recursion; no cycle = no recursion.

More Complex Recursive Examples

- *Cousins* in Ullman's notes

Recursion + Negation = Trouble

- Example 1 — R: { 1 }

  P(x) R(x), NOT Q(x)
  Q(x) R(x), NOT P(x)

- Example 2

  P(x) R(x), NOT P(x)

Stratified Recursion

- In a dependency graph, label arc X Y with a minus sign if Y appears in a negated subgoal.
- If a cycle in the dependency graph has no negative arc, it's a *stratified recursion.*
- We restrict ourselves to only recursions that are *stratified*.

Un-stratified recursion:

Recursive Query in Oracle

- Not SQL-99 compliant
- A.K.A. *Hierarchical Queries*
- E.g. find the nodes reachable from A.

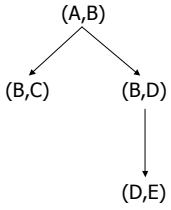
Edges

P1	P2
A	B
B	C
B	D
D	E

```
SELECT P2 FROM Edges
      START WITH P1='A'
      CONNECT BY P1 = PRIOR P2;
```

How CONNECT BY Works

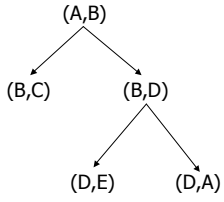
1. Find the root row(s) using the START WITH condition
2. For each parent row, find the child row(s) using the CONNECT by condition
3. Repeat (2) until no more rows can be added



Cycles in Data

Edges

P1	P2
A	B
B	C
B	D
D	E
D	A




```
SELECT P2 FROM Edges
      START WITH P1='A'
      CONNECT BY NOCYCLE P1 = PRIOR P2;
```

Hierarchical Query Pseudocolumns and Functions

- Pseudcolumns
  - CONNECT_BY_ISCYCLE
  - CONNECT_BY_ISLEAF
  - LEVEL
- Functions
  - SYS_CONNECT_BY_PATH