Updated 1/29/2020

T1: Data models and query languages
L7: Datalog evaluation

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp20)

https://northeastern-datalab.github.io/cs7240/sp20/

Version 1/28/2020

The issue of associativity

null appears in a join column. No matter what choice is taken, $\overset{+}{\bowtie}$ is not associative. Consider the relations

$q(A\ B)$: 1 2 $\quad$ $r(B\ C)$: 2 3 $\quad$ $s(A\ C)$: 1 4

Computing $(q \overset{+}{\bowtie} r) \overset{+}{\bowtie} s$ we get

$q'(A\ B\ C)$:

A	B	C
1	2	3
1	⊥	4

while $q \overset{+}{\bowtie} (r \overset{+}{\bowtie} s)$ gives

$q''(A\ B\ C)$:

A	B	C
1	2	4
⊥	2	3

$$\{a^-\} \vec{\cup}_a (\{a\} \vec{\cup}_a \{b\}) = \{a^-\}$$

$$(\{a^-\} \vec{\cup}_a \{a\}) \vec{\cup}_a \{b\} = \{b\}$$

left outer join example from p392 in "Maier. The theory of relational databases, 1983."

right preferred union example from "Gatterbauer, Suciu. Conflict resolution using trust mapping. SIGMOD 2010.

Outline: Datalog

- Datalog
  - Datalog rules
  - Recursion
  - Semantics
  - Datalog$^{\neg}$: Negation, stratification
  - Datalog$^{\pm}$
  - Stable model semantics (Answer set programming)
  - Datalog vs. RA
  - Naive and Semi-naive evaluation

RA to Datalog by examples: Union

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$R(A,B,C) \cup S(D,E,F)$

Datalog:

?

RA to Datalog by examples: Union

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$R(A,B,C) \cup S(D,E,F)$

Datalog:

U(x,y,z) :- R(x,y,z)
U(x,y,z) :- S(x,y,z)

RA to Datalog by examples: Intersection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$R(A,B,C) \cap S(D,E,F)$

Datalog:

?

RA to Datalog by examples: Intersection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$R(A,B,C) \cap S(D,E,F)$

Datalog:

I(x,y,z) :- R(x,y,z), S(x,y,z)

RA to Datalog by examples: Selection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\sigma_{\text{x>100 and y='foo'}}(R)$

Datalog:

?

RA to Datalog by examples: Selection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\sigma_{x>100 \text{ and } y='foo'}(R)$

Datalog:

S(x,y,z) :- R(x,y,z), x > 100, y='foo'

$\sigma_{x>100 \text{ or } y='foo'}(R)$

?

RA to Datalog by examples: Selection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\sigma_{x>100 \text{ and } y='foo'}(R)$

Datalog:

S(x,y,z) :- R(x,y,z), x > 100, y='foo'

$\sigma_{x>100 \text{ or } y='foo'}(R)$

S(x,y,z) :- R(x,y,z), x > 100
S(x,y,z) :- R(x,y,z), y='foo'

RA to Datalog by examples: Equi-join

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$$R \bowtie_{R.A=S.D \text{ and } R.B=S.E} S$$

Datalog:

?

RA to Datalog by examples: Equi-join

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$$R \bowtie_{R.A=S.D \text{ and } R.B=S.E} S$$

Datalog:

J(x,y,z,q) :- R(x,y,z), S(x,y,q)

RA to Datalog by examples: Projection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\pi_A(R)$

Datalog:

?

RA to Datalog by examples: Projection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\pi_A(R)$

Datalog:

P(x) :- R(x,y,z)

RA to Datalog by examples: Difference

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

R-S

Datalog:

?

RA to Datalog by examples: Difference

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

R-S

Datalog¬: (we need to add negation)

D(x,y,z) :- R(x,y,z), not S(x,y,z)

Outline: Datalog

- Datalog
  - Datalog rules
  - Recursion
  - Semantics
  - Datalog$^{\neg}$: Negation, stratification
  - Datalog$^{\pm}$
  - Stable model semantics (Answer set programming)
  - Datalog vs. RA
  - Naive and Semi-naive evaluation

Datalog Evaluation Algorithms

- Needs to preserve the efficiency of query optimizers, while extending them to recursion
- Two general strategies:
  - Naive datalog evaluation
  - Semi-naive datalog evaluation
- Some more powerful optimizations:
  - Magic sets (not covered)

Naive Datalog evaluation

Path(x,y) :- Edge(x,y)
Path(x,z) :- Edge(x,y), Path(y,z)

$Path^{(0)} := \emptyset$, t:=0
Repeat {
    inc(t)
    $Path^{(t)}(x,y) := Edge(x,y) \cup \Pi_{xy}(Edge(x,z) \bowtie Path^{(t-1)}(z,y))$
until $Path^{(t)} = Path^{(t-1)}$}

"immediate consequence operator"

- The same facts are discovered over and over again
- Next: The semi-naive algorithm tries to reduce the number of facts discovered multiple times

Background: Incremental View Maintenace

Let V be a view computed by one datalog rule (no recursion)

View :- body

If (some of) the relations are updated:

$$R_1 \longleftarrow R_1 \cup \Delta R_1,\ R_2 \longleftarrow R_2 \cup \Delta R_2,\ \ldots$$

Then the view is also modified as follows:

$$V \longleftarrow V \cup \Delta V$$

Incremental view maintenance:
Compute **ΔV** without having to recompute **V**

Background: Incremental View Maintenace

Example 1:

V(x,y) :- R(x,z), S(z,y)

If R ⟵ R∪ΔR,
then what is ΔV(x,y) ?

?

Background: Incremental View Maintenace

Example 1:

$V(x,y) :- R(x,z), S(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), S(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is $\Delta V(x,y)$?

Algebra:

$$V = R \bowtie S$$

$$V \cup \Delta V = (R \cup \Delta R) \bowtie S$$

?

Background: Incremental View Maintenace

Example 1:

$V(x,y) :- R(x,z), S(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), S(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is $\Delta V(x,y)$?

$z = x \cdot y$

$z+\Delta z = (x+\Delta x) \cdot y$

?

Algebra:

$$V = R \bowtie S$$

$$V \cup \Delta V = (R \cup \Delta R) \bowtie S$$

$$V \cup \Delta V = (R \bowtie S) \cup (\Delta R \bowtie S)$$

$$V \cup \Delta V = V \cup (\Delta R \bowtie S)$$

$$\Delta V = \Delta R \bowtie S$$

Union distributes over join

Background: Incremental View Maintenace

Example 1:

$V(x,y) :- R(x,z), S(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), S(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is $\Delta V(x,y)$?

$\otimes$ *distributes over* $\oplus$

$$z = x \cdot y$$

$$z+\Delta z = (x+\Delta x) \cdot y$$

...

$$\Delta z = \Delta x \cdot y$$

Algebra:

$$V = R \bowtie S$$

$$V \cup \Delta V = (R \cup \Delta R) \bowtie S$$

$$V \cup \Delta V = (R \bowtie S) \cup (\Delta R \bowtie S)$$

$$V \cup \Delta V = V \cup (\Delta R \bowtie S)$$

$$\Delta V = \Delta R \bowtie S$$

Union distributes over join

Background: Incremental View Maintenace

Example 2:

V(x,y) :- R(x,z), S(z,y)

If R ⟵ R∪ΔR, and S ⟵ S∪ΔS,
then what is ΔV(x,y) ?

?

Background: Incremental View Maintenace

Example 2:

$V(x,y) :- R(x,z), S(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), S(z,y)$

$\Delta V(x,y) :- R(x,z), \Delta S(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), \Delta S(z,y)$

If $R \longleftarrow R \cup \Delta R$, and $S \longleftarrow S \cup \Delta S$, then what is $\Delta V(x,y)$?

$\otimes$ *distributes over* $\oplus$

$$z = x \cdot y$$

$$z+\Delta z = (x+\Delta x)\cdot(y+\Delta y)$$

?

Algebra:

$$V = R \bowtie S$$

$$V \cup \Delta V = (R \cup \Delta R) \bowtie (S \cup \Delta S)$$

?

Union distributes over join

Background: Incremental View Maintenace

Example 2:

$V(x,y) :- R(x,z), S(z,y)$

If $R \longleftarrow R \cup \Delta R$, and $S \longleftarrow S \cup \Delta S$,
then what is $\Delta V(x,y)$?

$\Delta V(x,y) :- \Delta R(x,z), S(z,y)$

$\Delta V(x,y) :- R(x,z), \Delta S(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), \Delta S(z,y)$

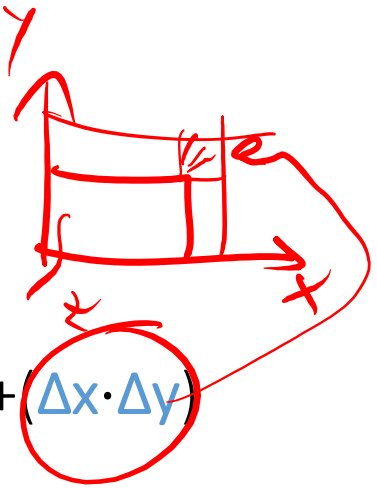
$\otimes$ *distributes over* $\oplus$

$$z = x \cdot y$$

$$z+\Delta z = (x+\Delta x)\cdot(y+\Delta y)$$

...

$$\Delta z = (\Delta x \cdot y)+(x \cdot \Delta y)+(\Delta x \cdot \Delta y)$$



Algebra:

$$V = R \bowtie S$$

$$V \cup \Delta V = (R \cup \Delta R) \bowtie (S \cup \Delta S)$$

$$V \cup \Delta V = (R \bowtie S) \cup (\Delta R \bowtie S) \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie \Delta S)$$

$$V \cup \Delta V = V \cup (\Delta R \bowtie S) \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie \Delta S)$$

$$\Delta V = (\Delta R \bowtie S) \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie \Delta S)$$

Union distributes over join

Background: Incremental View Maintenace

Example 3:

V(x,y) :- R(x,z), R(z,y)

If R ⟵ R∪ΔR,
then what is ΔV(x,y) ?

?

Background: Incremental View Maintenace

Example 3:

$V(x,y) :- R(x,z), R(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), R(z,y)$

$\Delta V(x,y) :- R(x,z), \Delta R(z,y)$

$\Delta V(x,y) :- \Delta R(x,z), \Delta R(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is $\Delta V(x,y)$?

$\otimes$ *distribute over* $\oplus$

$$z = x^2$$

$$z+\Delta z = (x+\Delta x)^2$$

...

$$\Delta z = 2x\Delta x+\Delta x^2$$

Algebra:

$$V = R \bowtie R$$

$$V \cup \Delta V = (R \cup \Delta R) \bowtie (S \cup \Delta R)$$

$$V \cup \Delta V = (R \bowtie R) \cup (\Delta R \bowtie R) \cup (R \bowtie \Delta R) \cup (\Delta R \bowtie \Delta R)$$

$$V \cup \Delta V = V \cup (\Delta R \bowtie R) \cup (R \bowtie \Delta R) \cup (\Delta R \bowtie \Delta R)$$

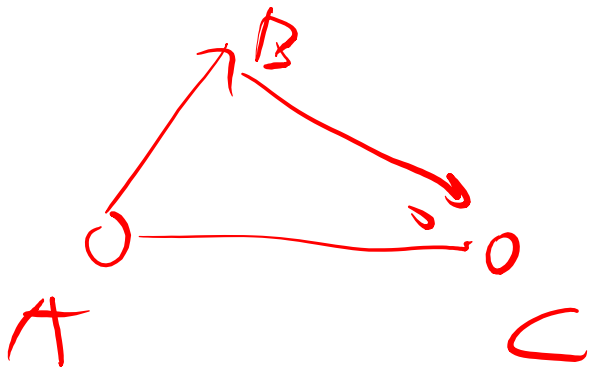
$$\Delta V = (\Delta R \bowtie R) \cup (R \bowtie \Delta R) \cup (\Delta R \bowtie \Delta R)$$

Union distributes over join

Semi-Naive Datalog evaluation

Path(x,y) :- Edge(x,y)
Path(x,z) :- Edge(x,y), Path(y,z)

Path := Edge(x,z); ΔPath := Edge(x,z)
Repeat {
    $\Delta Path(x,y) := \Pi_{xy}(Edge(x,z) \bowtie \Delta Path(z,y)) - Path(x,y)$
    Path := Path ∪ ΔPath
until ΔPath = ∅}



Datalog Summary

- EDB (base relations) and IDB (derived relations)
- Datalog program = set of rules
- Datalog is recursive

- Some reminders about semantics:
  - Multiple atoms in a rule mean join (or intersection)
  - Variables with the same name are join variables
  - Multiple rules with same head mean union

Datalog and SQL

- Stratified data (w/ recursion, w/o +,*,...): expresses precisely* queries in PTIME
  - Cannot find a Hamiltonian cycle (why?)
- SQL has also been extended to express recursive queries:
  - Use a recursive "with" clause, also CTE (Common Table Expression)
  - Often with bizarre restrictions...
  - ... Just use datalog

* needs to use the < predicate