CSE 414: Section 4
Relational Algebra
Datalog

October 18th, 2018

RA Operators

∩ - Intersect

$R1 \cap R2 = R1 - (R1 - R2)$

$R1 \cap R2 = R1 \bowtie R2$

Standard:

∪ - Union

- - Diff.

σ - Select

π - Project

ρ - Rename

Joins:

⋈ - Nat. Join

⟕ - L.O. Join

⟖ - R.O. Join

⟗ - F.O. Join

× - Cross Product

Extended:

δ - Duplicate Elim.

γ - Group/Agg.

τ - Sorting

A Few More SQL Keywords

```
(<sub>) INTERSECT (<sub>)

(<sub>) UNION (<sub>)

(<sub>) EXCEPT (<sub>)
```

ɣ Notation

Grouping and aggregation on group:

$$\gamma_{\text{attr\_1, …, attr\_k, count/sum/max/min(attr) -> alias}}$$

Aggregation on the entire table:

$$\gamma_{\text{count/sum/max/min(attr) -> alias}}$$

Query Plans (Example SQL -> RA)

Select-Join-Project structure

Make this SQL query into RA (remember FWGHOS):

```
SELECT R.b, T.c, max(T.a) AS T_max
  FROM Table_R AS R, Table_T AS T
 WHERE R.b = T.b
 GROUP BY R.b, T.c
HAVING max(T.a) > 99
```

Query Plans (Example SQL -> RA)

Select-Join-Project structure

Make this SQL query into RA (remember FWGHOS):

```
SELECT R.b, T.c, max(T.a) AS T_max
  FROM Table_R AS R, Table_T AS T
 WHERE R.b = T.b
 GROUP BY R.b, T.c
HAVING max(T.a) > 99
```

$\pi_{R.b,\ T.c,\ T\_max}(\sigma_{T\_max>99}(\gamma_{R.b,\ T.c,\ max(T.a)->T\_max}(R \bowtie_{R.b=T.b} T)))$

Datalog

Datalog Terminology

Head - Body - Atom/Subgoal/Relational predicate

Base Relations (EDB) vs Derived Relations (IDB)

- Negation + Aggregate

Wildcard

```
Helper(a,b):-Base1(a,b,_)
NonAns(j):-Base2(j,k),!Base3(k)
Ans(x):-Helper(x,y),!NonAns(y)
```

Query Safety

Need a positive relational atom of every variable

What’s wrong with this query?

Find all of Alice’s children without children:

```
U(x) :- ParentChild(“Alice”,x), !ParentChild(x,y)
```

Query Safety

```
U(x) :- ParentChild(“Alice”,x), !ParentChild(x,y)
```

It is domain dependent! Unsafe!

Double negation to the rescue. Why does this work?

```
NonAns(x) :- ParentChild(“Alice”,x), ParentChild(x,y)
# All of Alice’s children with children
U(x) :- ParentChild(“Alice”,x), !NonAns(x)
# All of Alice’s children without children (safe!)
```

But we can do better...

Query Safety

But we can do better...

```
hasChild(x) :- ParentChild(x,_)
# People with children
U(x) :- ParentChild(“Alice”,x), !hasChild(x)
# All of Alice’s children without children (safe!)
```

Datalog with Recursion

Able to write complicated queries in a few lines

Graph analysis

Done with query once output does not change.

Stratified Datalog

Recursion might not work well with negation

E.g.

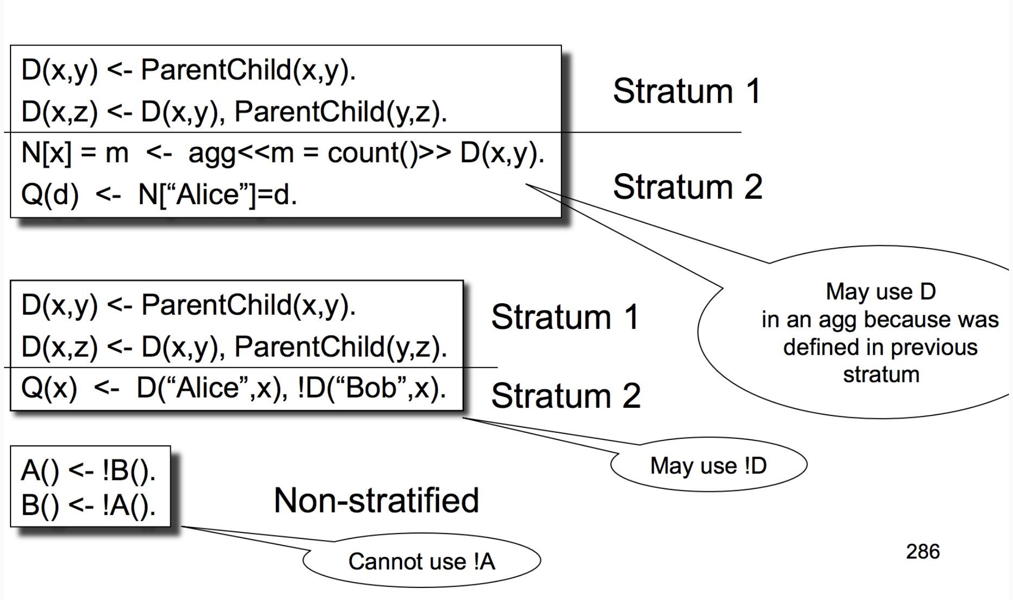
```
A(x):- Table(x), !B(x)
B(x):- Table(x), !A(x)
```

Solution: Don’t negate or aggregate on an IDB predicate until it is defined

Stratified Datalog Query

Stratified Datalog

Only IDB predicates defined in strata 1, 2, ..., n may appear under ! or agg in stratum n+1



```
D(x,y) <- ParentChild(x,y).
D(x,z) <- D(x,y), ParentChild(y,z).
```

Stratum 1

```
N[x] = m  <-  agg<<m = count()>> D(x,y).
Q(d)  <-  N["Alice"]=d.
```

Stratum 2

May use D in an agg because was defined in previous stratum

```
D(x,y) <- ParentChild(x,y).
D(x,z) <- D(x,y), ParentChild(y,z).
```

Stratum 1

```
Q(x)  <-  D("Alice",x), !D("Bob",x).
```

Stratum 2

May use !D

```
A() <- !B().
B() <- !A().
```

Non-stratified

Cannot use !A

286