CSE 344: Section 4 Relational Algebra, Datalog

October 19th, 2017

Administrivia

- HW3 due **TOMORROW**, Oct. 20th @ 11:00pm
- WQ4 due **Tuesday**, Oct. 24th @ 11:59pm

RA Operators

∩ - Intersect

R1∩R2 = R1−(R1−R2)

R1∩R2 = R1⋈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.

ɣ - Group/Agg.

τ - Sorting

γ 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

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 R, Table_T T
 WHERE R.b = T.b
 GROUP BY R.b, T.c
HAVING max(T.a) > 99
```

Query Plans

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 R, Table_T T
 WHERE R.b = T.b
 GROUP BY R.b, T.c
HAVING max(T.a) > 99
```

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

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.

VERY similar idea to context-free grammars (CSE 311)

Stratified Datalog

Recursion might not work well with negation

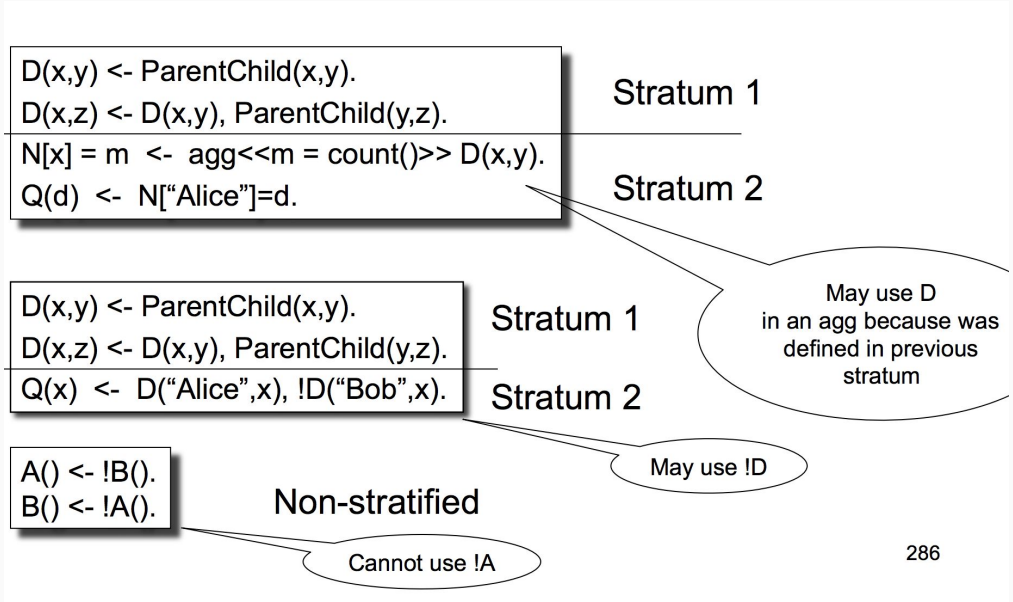
E.g.

```
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

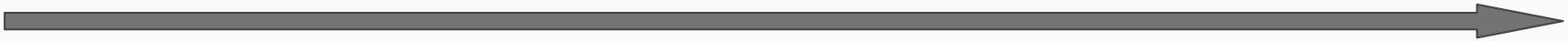
Only IDB predicates defined in strata 1, 2, ..., n may appear under ! or agg in stratum n+1



Expressive Capability

Nothing can do everything.

Forms of RA and Datalog can express things the other cannot.



	Positive Relations	**Negation**	**Aggregates**
Recursive	Pure Datalog	Stratified Datalog	Stratified Datalog + agg.
Non-recursive	Non-recursive Datalog Positive RA	 RA	 Extended RA