

CSE 414: Section 4

Relational Algebra

Datalog

October 18th, 2018

RA Operators

\cap - Intersect

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

$$R1 \cap R2 = R1 \bowtie R2$$

Standard:

\cup - Union

$-$ - Diff.

σ - Select

π - Project

ρ - Rename

Joins:

\bowtie - Nat. Join

\Join - L.O. Join

\Join - R.O. Join

\Join - F.O. Join

\times - Cross Product

Extended:

δ - Duplicate Elim.

γ - Group/Agg.

τ - Sorting

A Few More SQL Keywords

Set operations on subqueries

(<sub>)\cap (<sub>)

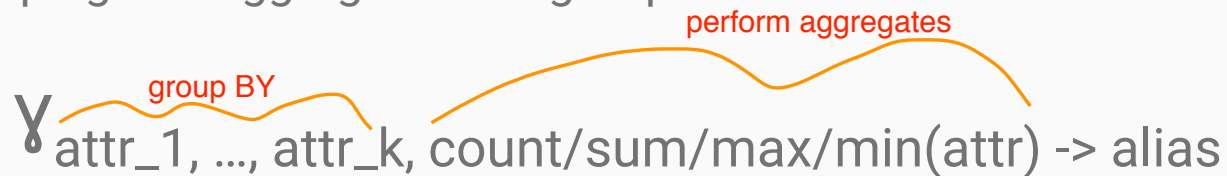
(<sub>)\cup (<sub>)

(<sub>)$-$ (<sub>)

Y Notation

Grouping and aggregation on group:

Y attr_1, ..., attr_k, count/sum/max/min(attr) -> alias



Aggregation on the entire table:

Y count/sum/max/min(attr) -> alias

without individual attributes name = NO GROUP BY

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
```

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

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?

NEVER use an `_` in a negated statement
(one `_` is not recognized as same as another, so even if theres an `_`
in a positive atom its still unsafe)

Find all of Alice's children without children:

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

y does not exist in a positive relational atom

Whats wrong with this?

*y is not bounded at all; can be literally any value
and dataLog will return pretty much everything?*

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.

after recursion reaches a point where $T(k) = T(k+1)$, the recursion will stop...

Stratified Datalog

Recursion might not work well with negation

E.g.

WRONG

$A(x) :- \text{Table}(x), \neg B(x)$

$B(x) :- \text{Table}(x), \neg A(x)$

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

Stratified Datalog Query

only use rules/relations defined above it

Stratified Datalog

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

