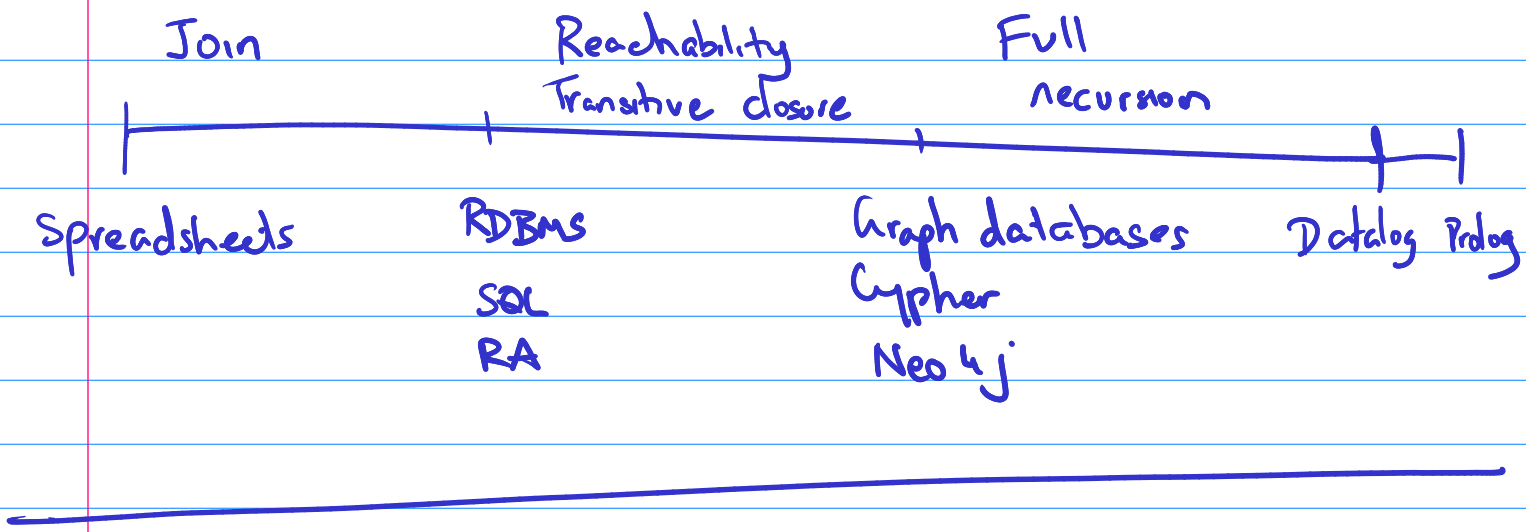


Question: How do we express more complex forms of recursion?

Example (Non-recursive)

Cousins: Two people whose parents are siblings.

Two people who share the same grandparent.

Parent	
John	Mary
Mary	Mike
John	Nate
Nate	Susan

→

Cousin	
Mike	Susan
Susan	Mike
Mike	Mike
Susan	Susan

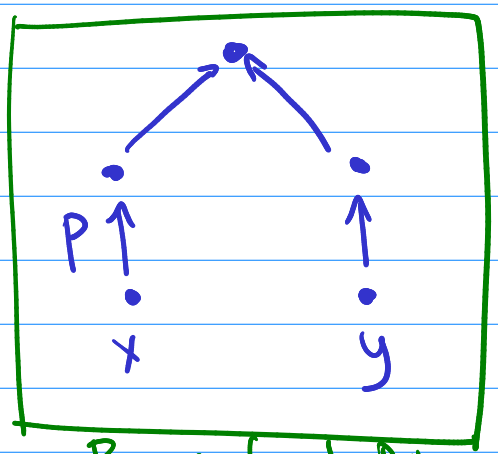
Computing Cousins using RA/SQL

GP = select $P_1 \cdot c_1, P_2 \cdot c_2$

from Parent as P_1

Parent as P_2

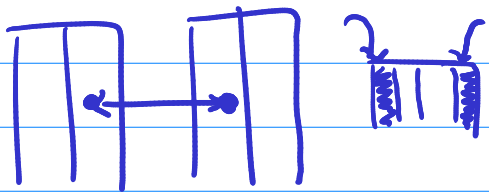
where $P_1 \cdot c_2 = P_2 \cdot c_1$



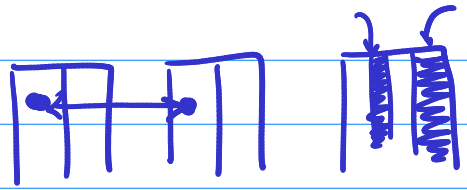
Basic Graph Pattern

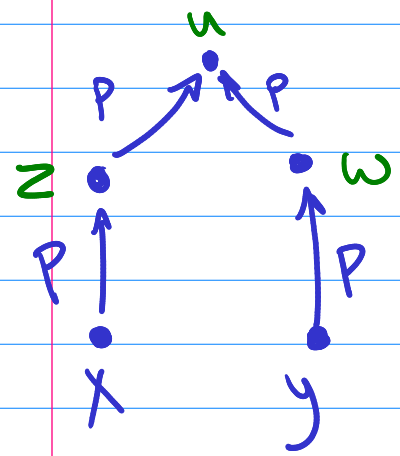
∴ views

$$GP = \pi_{c_1, c_4}\left(P \underset{c_2 = c_1'}{\bowtie} P\right)$$



$$Cousins = \pi_{c_2, c_4}\left(GP \underset{c_1 = c_1'}{\bowtie} GP\right)$$





For all people x y z w u,

If we have

Parent(u z) ∧ P(z x) ∧

P(u w) ∧ P(w y)

then, we have Cousins(x y)

$\forall x y z u w$ then the head holds too. | if all the literals in the body hold

Cousins(x y) :- P(u z) P(z x) P(u w) P(w y)

literals

Head "Output" | Body "Input"

"Rule", "Clause", "Query", "Program"

This is our first Datalog query!

Question: How can we write recursive Datalog queries?

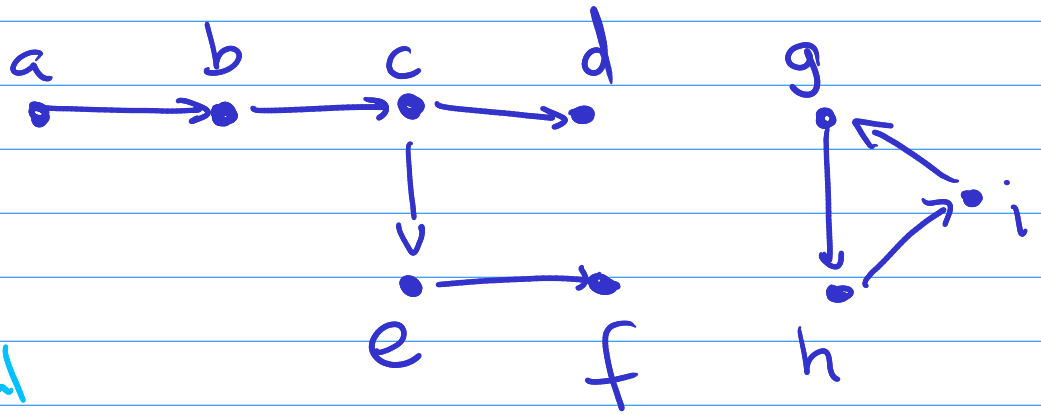
Just use the same relation in both the head & body!

```
path(x, y) :- edge(x, y).
path(x, z) :- edge(x, y), path(y, z).
```

"or" disjunction

"... then, if ..." — ":-"

"and" conjunctions — ","



Positive Relational Datalog

"UCQ"

Theorem: Positive non-recursive Datalog

= Positive relational algebra

Example: Simulating a DFA (Finite state machine)