Lecture 24

Thursday, November 12, 2020 3:59 PM

Complexity of Evaluating Relational Queries

Given a basic graph pattern Q

& a database I

checking whether Q matches I is NP-complete.

For basic graph patterns

Data complexity

- Poly(|I|).

Query complexity

- NP-complete

Question: Given a database I

& a RA query Q

what is the complexity of computing Q(I)?

- Depends on the query. ← "schema"

"Query complexity".

Some queries are easy to compute

Other queries ("joins") require a lot more work.

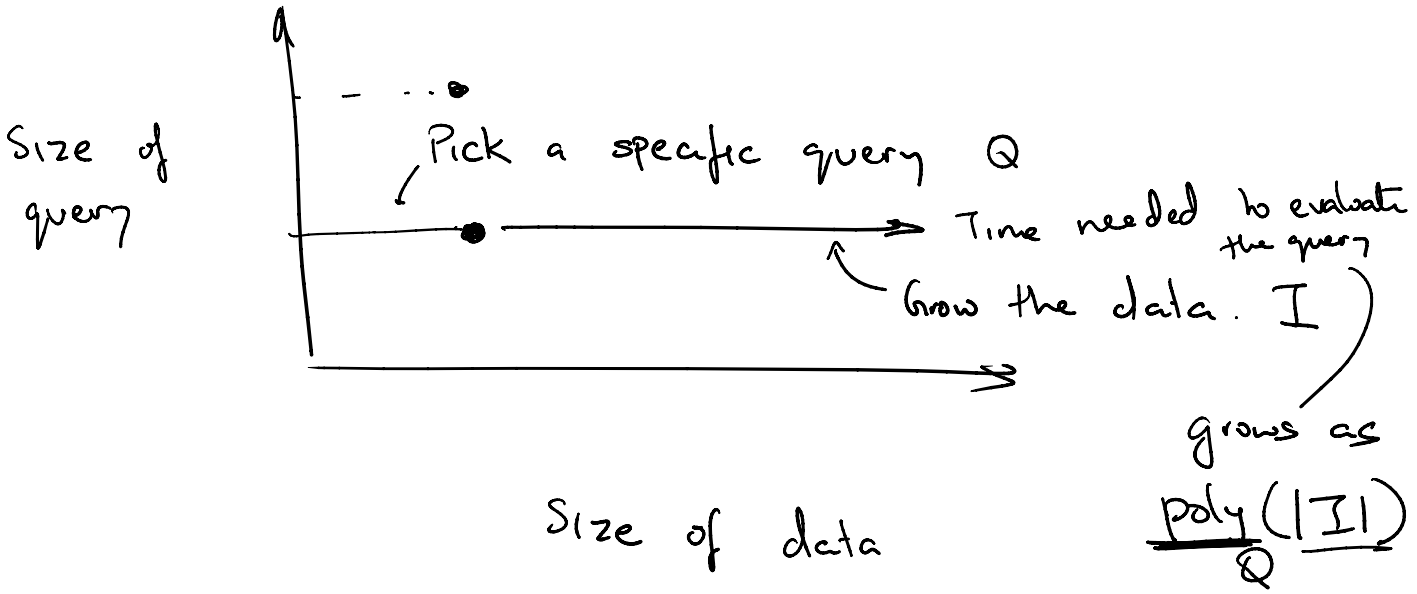
- Depends on the data. ("schema") : "Data" complexity

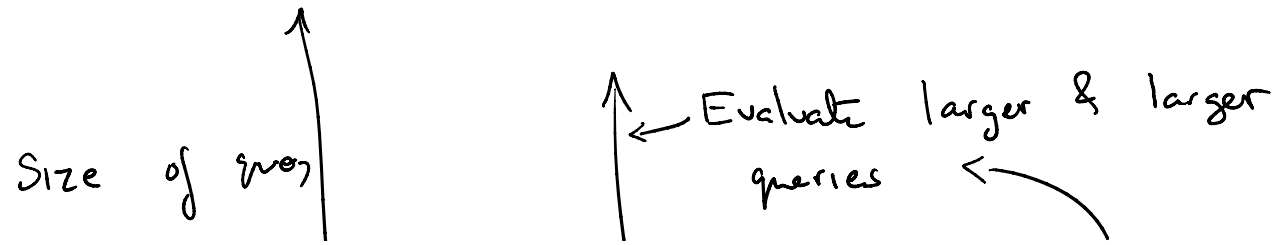
Query complexity of RA. NP-complete

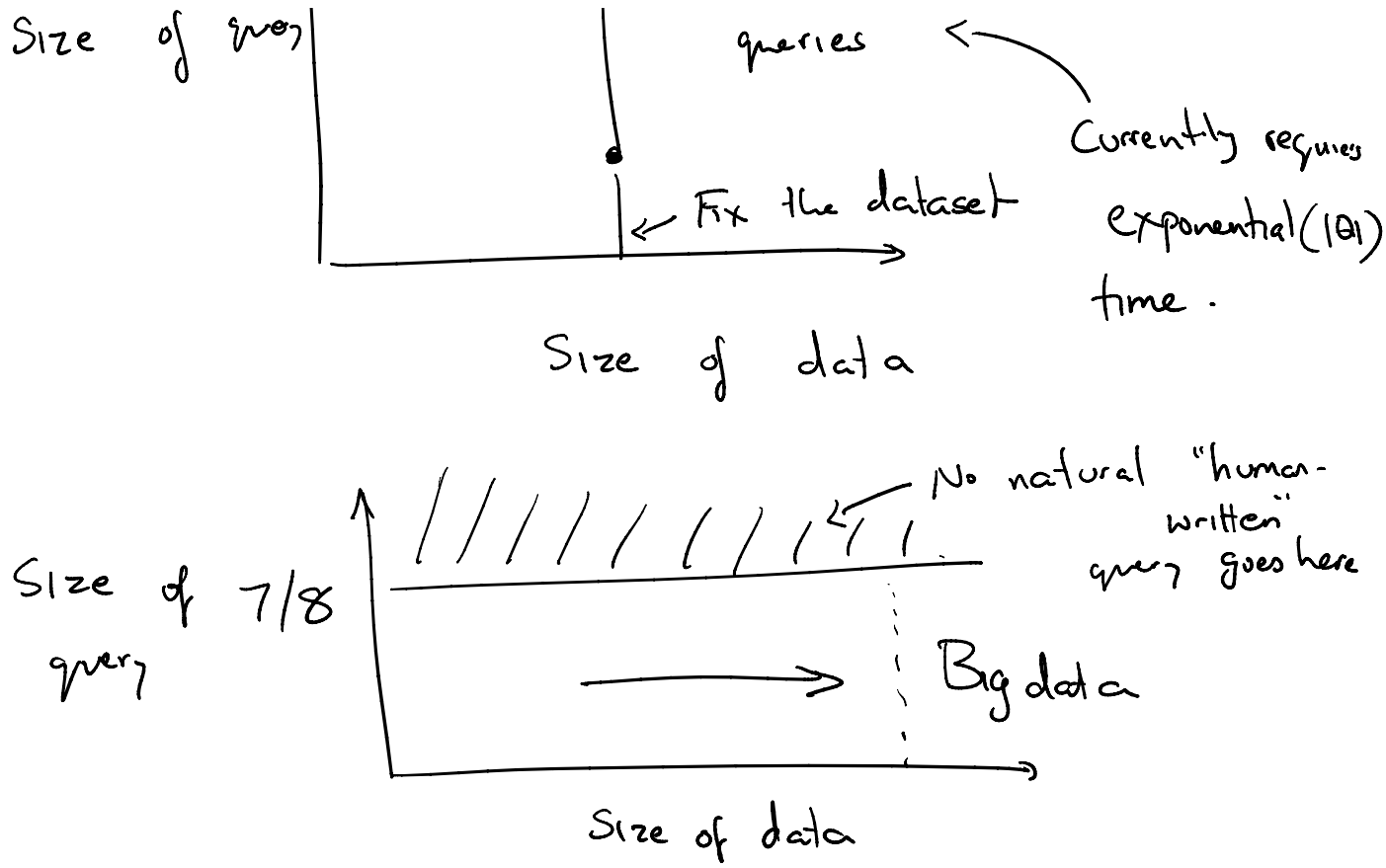
"Is Q(I) non-empty?"

Data complexity of RA poly(|I|)

AC^0







Normalization / Denormalization.

- For performance
- Architectural reasons

1NF
2NF
⋮
BCNF
⋮

- Moving on from graph databases,

Can we express richer forms of recursion?

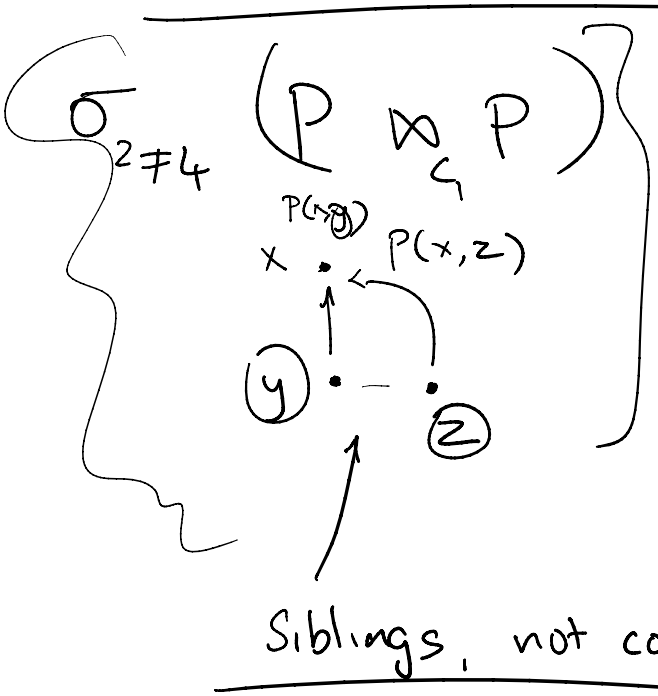
— Cousins: Two people whose parents are siblings

— Two people who share the same grandparent.

Parent : x is a parent of y.

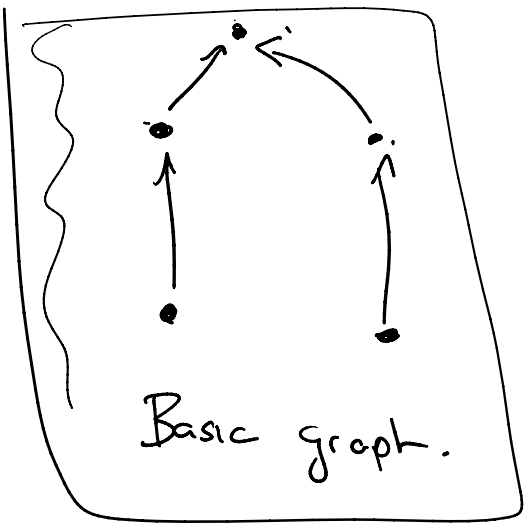
c_1	c_2
x	y

How to compute cousins using RA/SQL?



Siblings, not cousins.

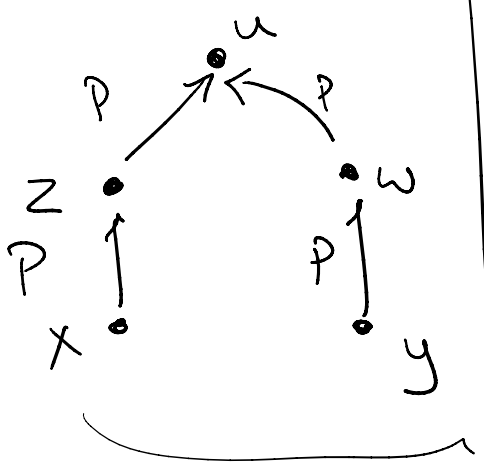
Cousins.



$$GP = \pi_{14}(P \bowtie_{2=3} P)$$

$$Cousins = \pi_{24}(GP \bowtie_{1=3} GP)$$

4-way join in SQL/RA



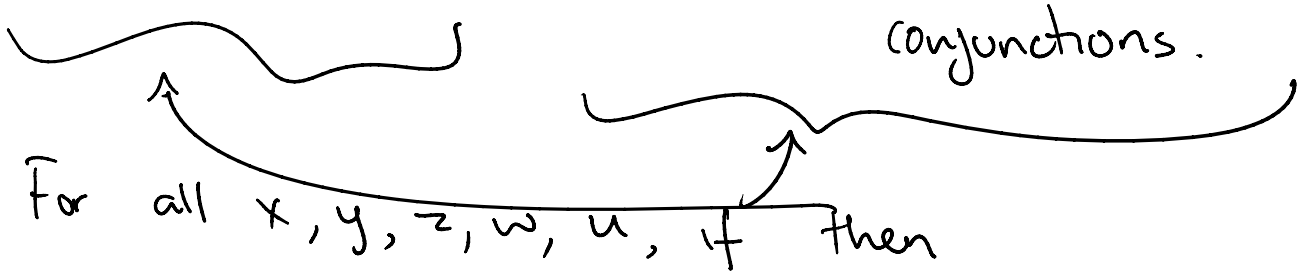
For all people x y z w u,
if we have
Parent(x z) and P(z u)
and P(y w) and P(w u)
then we have
Cousins(x y).

(A) Cousins(x y) :- P(x z), P(z, u), P(y w), P(w, u).

Our first Datalog query

These commas really mean "and"

These commas really mean "and".



Datalog "rules"

Head — Body

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

If everything in body, then head.

From body, produce head.

What might recursion be?

The body itself has head.

What about disjunctions / ors?

Just write 2/3/4/... rules.

Three "approaches" to evaluating Datalog programs

- Top down ←
- Bottom-up: Naive, semi-naive
- Magic sets ← "Query-guided evaluation"

$\forall x, y$ (If x then y) $\iff$ $\forall x, y$ (not x) or y

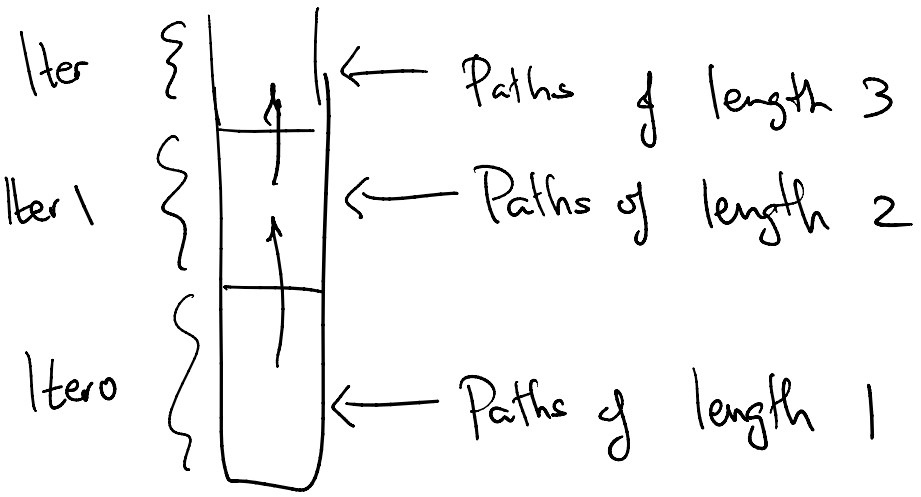
x: Contravariant; y: Covariant

$\forall y$ ((not $\exists x$) or y)

Theorem: Non recursive Datalog $=$ Relation Algebra.

Equal in expressive

Equal in expressive power.



But if there are n vertices in the graph, at most n^2 paths to be discovered.

"Fixed point"

It is eventually going to run out of tuples to discover