Categorical Informatics At Scale

Ryan Wisnesky

Conexus AI

SemWebPro 2020

$$\Sigma \dashv \Delta \dashv \Pi$$

Outline

The

- Who
- What
- When
- Where
- Why
- How

of Conexus and what it means to you, a semantic web user.

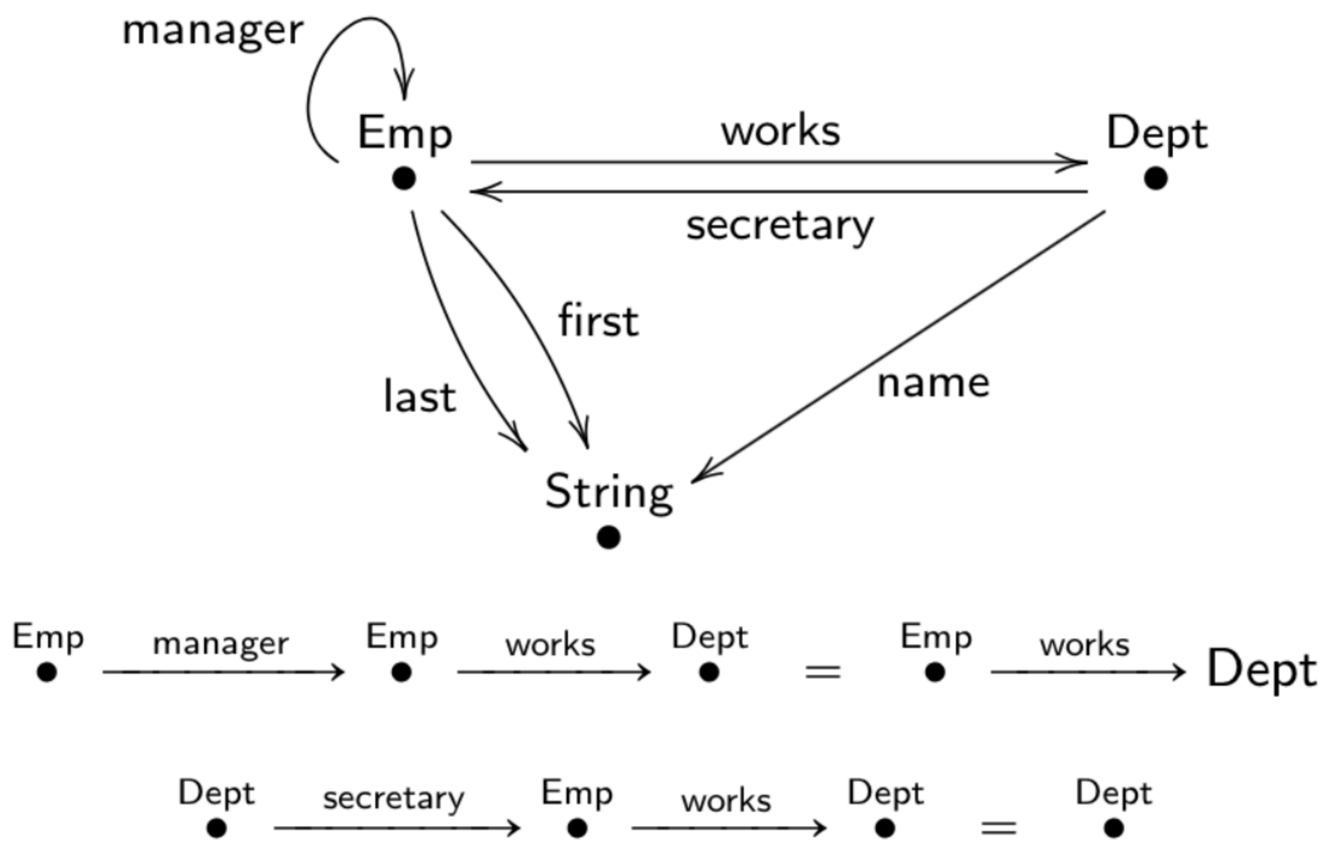
Conexus Overview & Relevance

- Who: David Spivak (math), myself (cs), Eric Daimler (business), et al.
- What: solving problems in ETL, data integration, IT interoperability, etc. (data)
- How: Kan extensions, limits and co-limits, etc. (functorial data migration)
- Why: because other technologies cannot solve these problems. (fun and profit)
- Where: San Francisco, Boston, Munich (Daniel Filonik) (also friends at NIST/DC).
- When: 2015-2018 Grant funded; 2018-present Seed stage VC funded.

- Relevance to you:
  - Use and/or contribute to the open source CQL project to do semantic web: CQL is a better OWL than OWL.
  - Use the free and/or commercial CQL IDE and/or our team to manipulate data “using real math”
    - Really hard due to non-computability issues often ignored for pragmatic reasons by mathematicians.
  - Sometimes we hire!
  - We collaborate with entrepreneurs, academics, programmers, domains experts, and more.
    - Data wrangling: it’s dirty job, but someone’s got to do it

CQL Overview

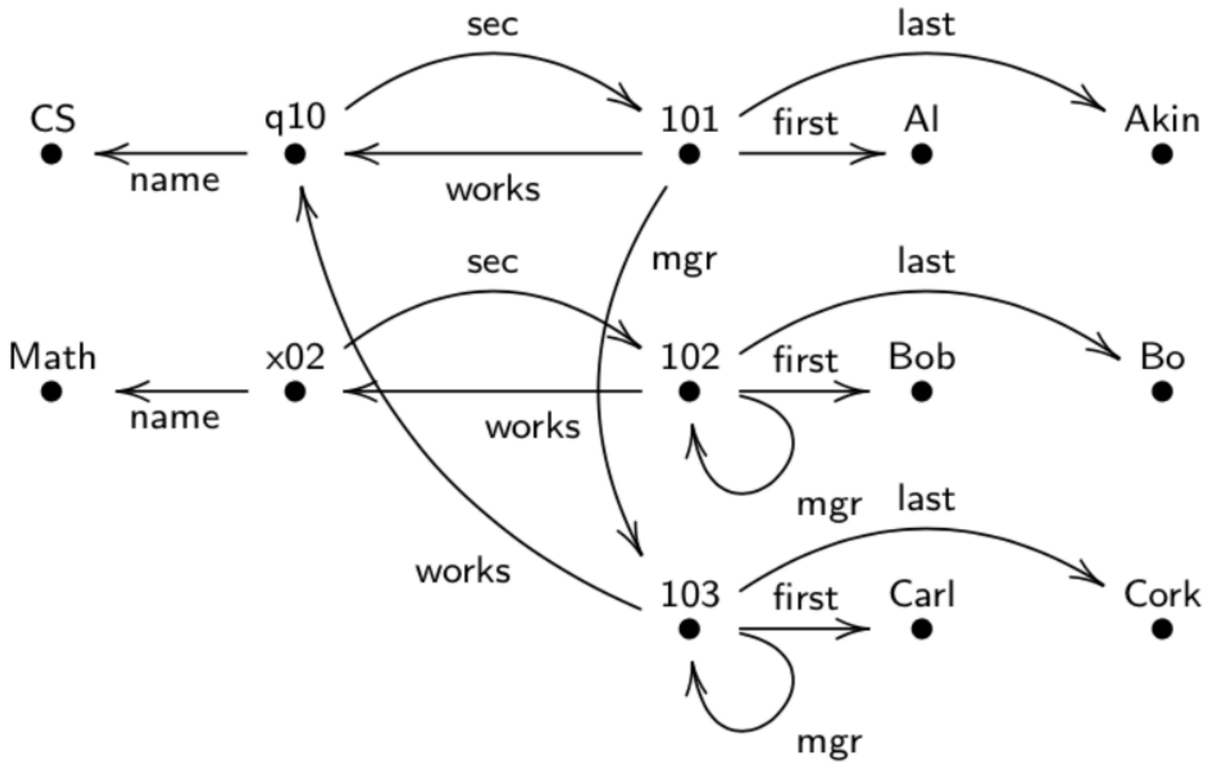
- Category theory was designed to migrate theorems from one area of mathematics to another, so it is a very natural language with which to describe how to migrate data from one schema to another.
- Community site: http://categoricaldata.net
- Projects:
  - NIST - several projects.
  - DARPA BRASS project.
  - Empower Retirement.
  - Stanford Chemistry Department.
  - Uber/Apache Tinkerpop (joint paper: ‘algebraic property graphs’)



Emp				
ID	**mgr**	**works**	**first**	**last**
101	103	q10	Al	Akin
102	102	x02	Bob	Bo
103	103	q10	Carl	Cork

Dept		
ID	**sec**	**name**
q10	101	CS
x02	102	Math

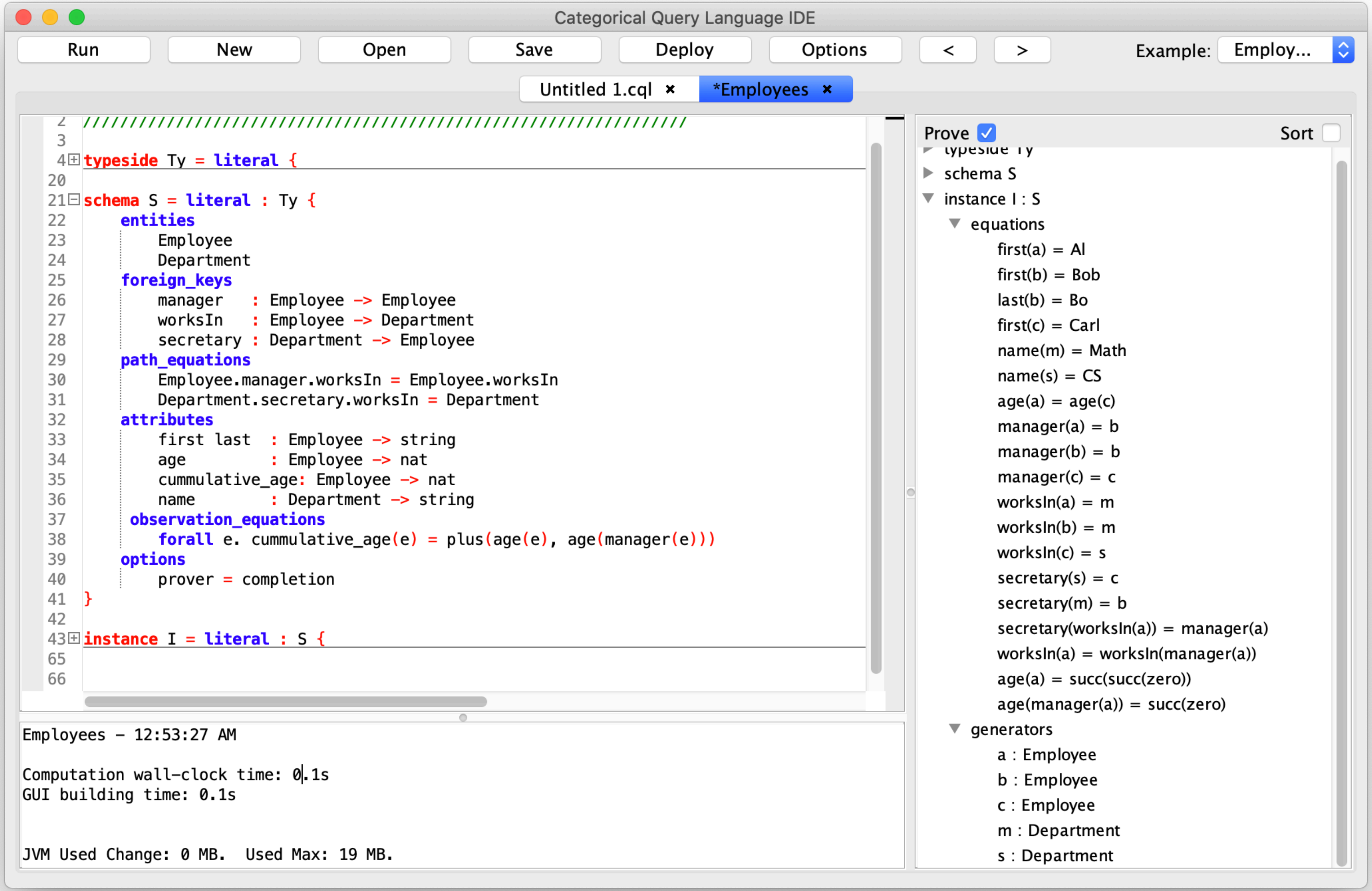
String
ID
Al
Bob
...



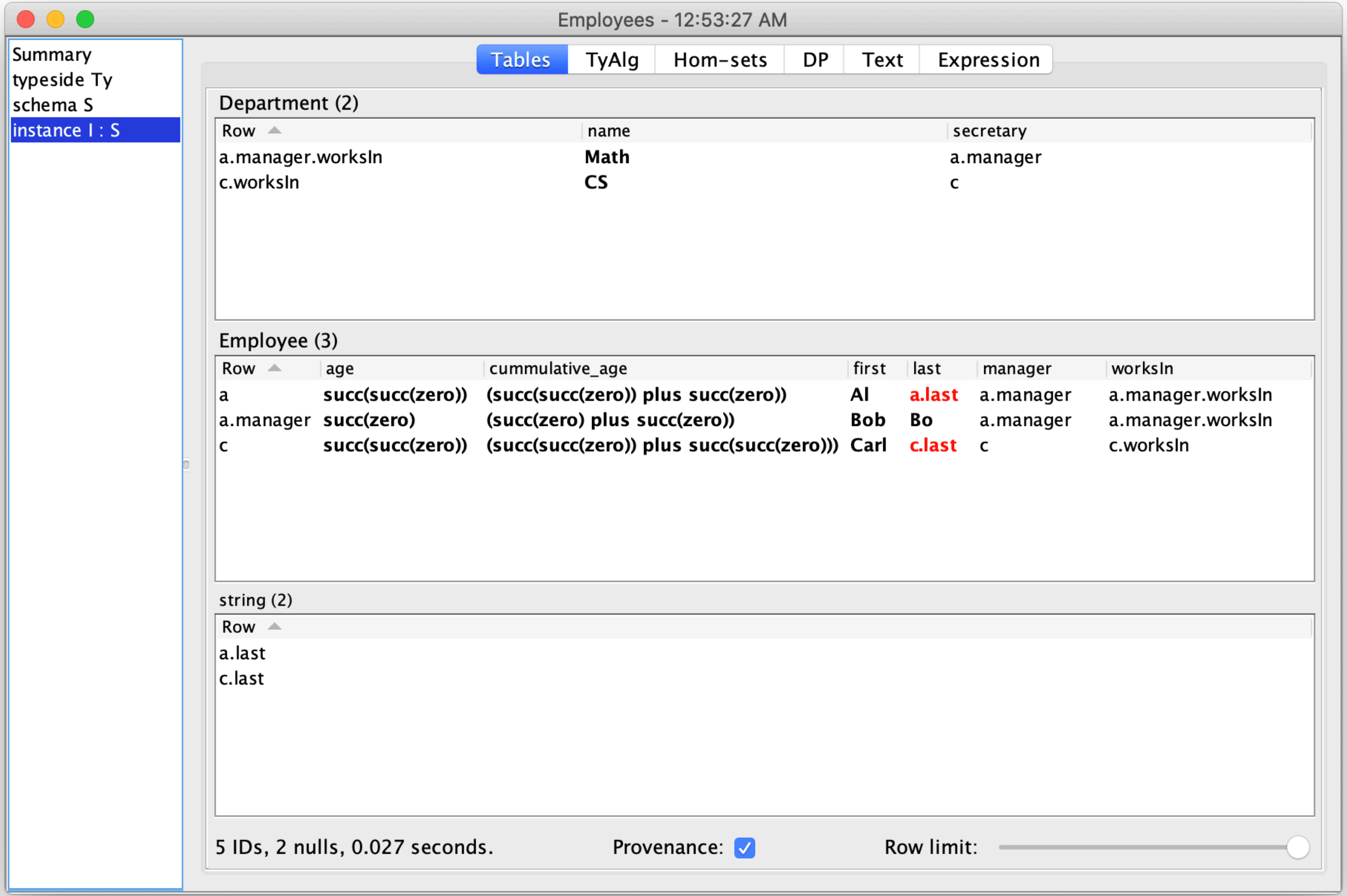
Emp				
ID	**mgr**	**works**	**first**	**last**
101	103	q10	Al	Akin
102	102	x02	Bob	Bo
103	103	q10	Carl	Cork

Dept		
ID	**sec**	**name**
q10	101	CS
x02	102	Math

String
ID
Al
Bob
...



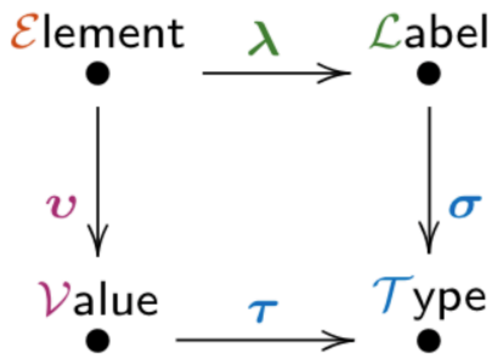
Categorical Query Language IDE
Run
New
Open
Save
Deploy
Options
<
>
Example: Employ...
Untitled 1.cql
*Employees
//
typeside Ty = literal {
schema S = literal : Ty {
entities
Employee
Department
foreign_keys
manager : Employee -> Employee
worksIn : Employee -> Department
secretary : Department -> Employee
path_equations
Employee.manager.worksIn = Employee.worksIn
Department.secretary.worksIn = Department
attributes
first last : Employee -> string
age : Employee -> nat
cummulative_age: Employee -> nat
name : Department -> string
observation_equations
forall e. cummulative_age(e) = plus(age(e), age(manager(e)))
options
prover = completion
}
instance I = literal : S {
Employees – 12:53:27 AM
Computation wall-clock time: 0.1s
GUI building time: 0.1s
JVM Used Change: 0 MB. Used Max: 19 MB.
Prove
Sort
typeside Ty
schema S
instance I : S
equations
first(a) = Al
first(b) = Bob
last(b) = Bo
first(c) = Carl
name(m) = Math
name(s) = CS
age(a) = age(c)
manager(a) = b
manager(b) = b
manager(c) = c
worksIn(a) = m
worksIn(b) = m
worksIn(c) = s
secretary(s) = c
secretary(m) = b
secretary(worksIn(a)) = manager(a)
worksIn(a) = worksIn(manager(a))
age(a) = succ(succ(zero))
age(manager(a)) = succ(zero)
generators
a : Employee
b : Employee
c : Employee
m : Department
s : Department


Employees - 12:53:27 AM
Summary
typeside Ty
schema S
instance I : S
Tables
TyAlg
Hom-sets
DP
Text
Expression
Department (2)
Row | name | secretary
a.manager.worksIn | Math | a.manager
c.worksIn | CS | c
Employee (3)
Row | age | cummulative_age | first | last | manager | worksIn
a | succ(succ(zero)) | (succ(succ(zero)) plus succ(zero)) | Al | a.last | a.manager | a.manager.worksIn
a.manager | succ(zero) | (succ(zero) plus succ(zero)) | Bob | Bo | a.manager | a.manager.worksIn
c | succ(succ(zero)) | (succ(succ(zero)) plus succ(succ(zero))) | Carl | c.last | c | c.worksIn
string (2)
Row
a.last
c.last
5 IDs, 2 nulls, 0.027 seconds.
Provenance:
Row limit:

Friend of a friend

```
constraints works_at_determined1 = literal : FOAF {
forall s:salary ->
exists w:works_at
where s.salary1 = w.works_at1
      s.salary2 = w.works_at2
}

constraints works_at_determined2 = literal : FOAF {
forall w:works_at ->
exists s:salary
where s.salary1 = w.works_at1
      s.salary2 = w.works_at2
}

constraints frenemy_determined = literal : FOAF {
forall fr : frenemy_of ->
exists f : friend_of
       e : enemy_of
where fr.frenemy_of1 = f.friend_of1
      fr.frenemy_of2 = f.friend_of2
      fr.frenemy_of1 = e.enemy_of1
      fr.frenemy_of2 = e.enemy_of2
}
```

```
typeside Ty = literal {
    types
        Number
        String
}

schema FOAF = literal : Ty {
    entities
        //entities
        Person
        Organization
        //spans
        knows
        friend_of
        works_at
        salary
        enemy_of
        frenemy_of
    foreign_keys //total functions
        knows1 : knows -> Person
        knows2 : knows -> Person
        friend_of1 : friend_of -> Person
        friend_of2 : friend_of -> Person
        works_at1 : works_at -> Person
        works_at2 : works_at -> Organization
        salary1 : salary -> Person
        salary2 : salary -> Organization
        enemy_of1 : enemy_of -> Person
        enemy_of2 : enemy_of -> Person
        frenemy_of1 : frenemy_of -> Person
        frenemy_of2 : frenemy_of -> Person
    attributes //total functions
        family_name : Person -> String
        age : Person -> Number
        given_name : Person -> String
        salary3 : salary -> Number
}

constraints knows_symmetric = literal : FOAF {
forall k1:knows ->
exists k2:knows
where k1.knows1 = k2.knows2
      k1.knows2 = k2.knows1
}
```



$$\mathcal{E} \xrightarrow{\upsilon} \mathcal{V} \xrightarrow{\tau} \mathcal{T} \quad = \quad \mathcal{E} \xrightarrow{\lambda} \mathcal{L} \xrightarrow{\sigma} \mathcal{T}$$

$\mathcal{E}$lement		
ID	λ	υ
t_1	Trip	(u_1, u_2)
t_2	Trip	(u_1, u_3)
u_1	User	Alice
u_2	User	Bob
u_3	User	Chaz

$\mathcal{V}$alue	
ID	τ
Alice	String
Bob	String
Chaz	String
(u_1, u_2)	User $\times$ User
(u_1, u_3)	User $\times$ User

$\mathcal{L}$abel	
ID	σ
User	String
Trip	User $\times$ User

$\mathcal{T}$ype
ID
String
User $\times$ User

CQL: Further Capabilities

- CQL has over 100 keywords
- Categories, functors, and natural transformations (schema, data)
- Left and right Kan extensions (migration of data)
- Limits and colimits (integration of schema, data)
- Grothendieck construction (data <-> schema)

- Patented implementation techniques: automated theorem proving, custom "chase" algorithms, reduction to SQL.

Thanks

- Me: ryan@conexus.com
- http://Categoricaldata.net
- http://conexus.com

- Collaborators welcome!