From Graphs to Categories

And Algebraic Property Graphs Too

Ryan Wisnesky
Conexus AI

November 10, 2020

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

Introduction

- These slides describe new ways to formalize databases and migrate data based on category theory.
- 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 migrating data / knowledge from one schema / ontology to another.
- Research has culminated in an open-source language, CQL, available at categoricaldata.net, being commercialized by Conexus AI, conexus.com.
- Categories are graphs with extra structure, and so category theory has deep connections to property graphs (joint work with Joshua Shinavier), also described in these slides.

Category Theory

A category $\mathcal{C}$ consists of

- *objects* A, B, C ... and *arrows* (also called *morphisms*) f, g, h ... such that:
- For every arrow f there is an object $\mathsf{src}(f)$ called the *source* of f and an object $\mathsf{tgt}(f)$ called the *target* of f. When $S = \mathsf{src}(f)$ and $T = \mathsf{tgt}(f)$, we may write $f : S \to T$. Visually:

$$S \xrightarrow{f} T$$

- For every arrow $f : A \to B$ and arrow $g : B \to C$ there is an arrow $g \circ f : A \to C$ called the *composite* of f and g:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[rr, bend right, "g \circ f"'] & B \arrow[r, "g"] & C \end{tikzcd}$$

- Composition is *associative*, i.e. $h \circ (g \circ f) = (h \circ g) \circ f$ for arbitrary f, g, and h.
- For every object A there is an *identity arrow* $\mathsf{id}_A : A \to A$:

$$\begin{tikzcd} A \arrow[loop above, "\mathsf{id}_A"] \end{tikzcd}$$

- Furthermore, for any arrow $f : A \to B$, $f \circ \mathsf{id}_A = f = \mathsf{id}_B \circ f$.

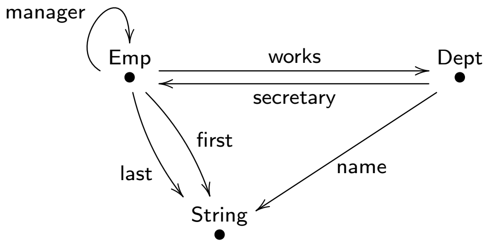
A *functor* $F : \mathcal{C} \to \mathcal{D}$ is a function from $\mathcal{C}$'s objects to $\mathcal{D}$'s objects and $\mathcal{C}$'s arrows to $\mathcal{D}$'s arrows that preserves composition and identity:

$$F(\mathsf{id}_c) = \mathsf{id}_{F(c)} \qquad F(f \circ g) = F(f) \circ F(g).$$

Categorical Schemas and Databases

- A *schema* S is a directed multi-graph and a set of paths through the graph called "equivalent".
- A schema S denotes a category $[\![S]\!]$:
  - The objects of $[\![S]\!]$ are the nodes of S.
  - The arrows of $[\![S]\!]$ are the paths through S, modulo the path equivalences in S.
- An S-instance (database on schema S) is a collection of sets, one per node in S, and a collection of (unary) functions, one per edge in S, satisfying the path equivalences in S.
- For example, these sets and functions may be represented as a collection of SQL tables, one per node in S, each with columns for edges out of that node.
- An S-instance denotes a functor $[\![S]\!] \to \mathbf{Set}$, where $\mathbf{Set}$, the category of sets, has for objects all sets and for arrows all (unary) functions.

Example Categorical Schema and Database



$$\text{Emp} \xrightarrow{\text{manager}} \text{Emp} \xrightarrow{\text{works}} \text{Dept} \;=\; \text{Emp} \xrightarrow{\text{works}} \text{Dept}$$

$$\text{Dept} \xrightarrow{\text{secretary}} \text{Emp} \xrightarrow{\text{works}} \text{Dept} \;=\; \text{Dept}$$

Emp

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

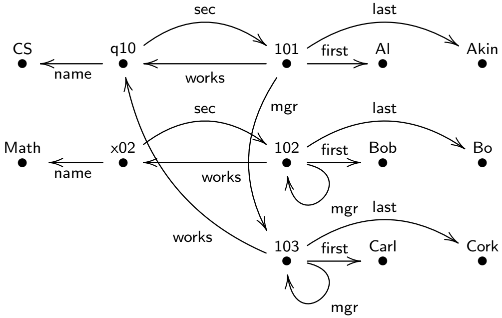
Dept

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

String

ID
Al
Bob
...

Categorical Databases to Triples (Graphs) and Back

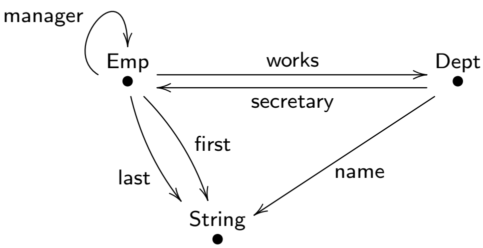


Emp				
ID	**mgr**	**works**	**first**	**last**
101	103	q10	Al	Akin
102	102	x02	Bob	Bo
103	103	q10	Carl	Cork

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

String
ID
Al
Bob
...

Categorical Select-From-Where/For-Where-Return Syntax

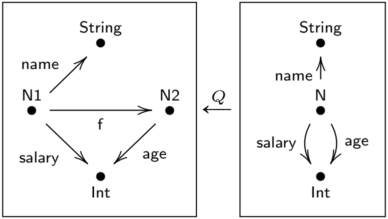


Find the name of every manager's department:

```
CQL
select e.manager.works.name
from Emp as e
```

```
SQL
select d.name
from Emp as e1, Emp as e2, Dept as d
where e1.manager = e2.ID and
      e2.works = d.ID
```

Query Evaluation and Co-evaluation



N1			
ID	**name**	**salary**	**f**
1	Alice	$100	4
2	Bob	$250	5
3	Sue	$300	6

N2	
ID	**age**
4	20
5	20
6	30

$\xleftarrow{eval_Q}$

$\xrightarrow{coeval_Q}$

N			
ID	**name**	**salary**	**age**
a	Alice	$100	20
b	Bob	$250	20
c	Sue	$300	30

Example Round Trip (column f removed)

N1

ID	Name	Salary
1	Alice	\$100
2	Bob	\$250
3	Sue	\$300

N2

ID	Age
4	20
5	20
6	30

$\xrightarrow{coeval_Q}$

N

ID	Name	Salary	Age
a	Alice	\$100	$null_1$
b	Bob	\$250	$null_2$
c	Sue	\$300	$null_3$
d	$null_4$	$null_5$	20
e	$null_6$	$null_7$	20
f	$null_8$	$null_9$	30

$\downarrow \eta$ $\qquad eval_Q \swarrow$

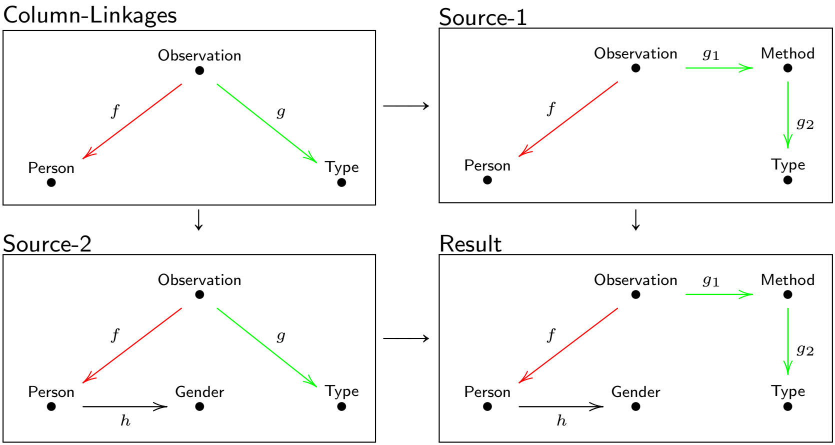
N1

ID	Name	Salary
a	Alice	\$100
b	Bob	\$250
c	Sue	\$300
d	$null_4$	$null_5$
e	$null_6$	$null_7$
f	$null_8$	$null_9$

N2

ID	Age
a	$null_1$
b	$null_2$
c	$null_3$
d	20
e	20
f	30

Schema Integration



Data Integration

Observation

ID	f	g

Person

ID
p

Type

ID
BP
Wt

→

Method

ID	g2
m_1	BP
m_2	BP
m_3	Wt
m_4	Wt

Type

ID
BP
Wt

Observation

ID	f	g1
o_1	Pete	m_1
o_2	Pete	m_2
o_3	Jane	m_3
o_4	Jane	m_1

Person

ID
Jane
Pete

↓

Gender

ID
F
M

Type

ID
BP
Wt
HR

Observation

ID	f	g
o_5	Peter	BP
o_6	Paul	HR
o_7	Peter	Wt

Person

ID	h
Paul	M
Peter	M

→

Method

ID	g2
$null_1$	BP
$null_2$	Wt
$null_3$	HR
m_1	BP
m_2	BP
m_3	Wt
m_4	Wt

Observation

ID	f	g1
o_1	Peter	m_1
o_2	Peter	m_2
o_3	Jane	m_3
o_4	Jane	m_1
o_5	Peter	$null_1$
o_6	Paul	$null_2$
o_7	Peter	$null_3$

Gender

ID
F
M
$null_4$

Type

ID
BP
Wt
HR

Person

ID	h
Jane	$null_4$
Paul	M
Peter	M

An Overly-General Schema for Algebraic Property Graphs

$$
\begin{tikzcd}
\mathcal{E}\text{lement} \arrow[r, "\lambda"] \arrow[d, "\upsilon"'] & \mathcal{L}\text{abel} \arrow[d, "\sigma"] \\
\mathcal{V}\text{alue} \arrow[r, "\tau"] & \mathcal{T}\text{ype}
\end{tikzcd}
$$

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

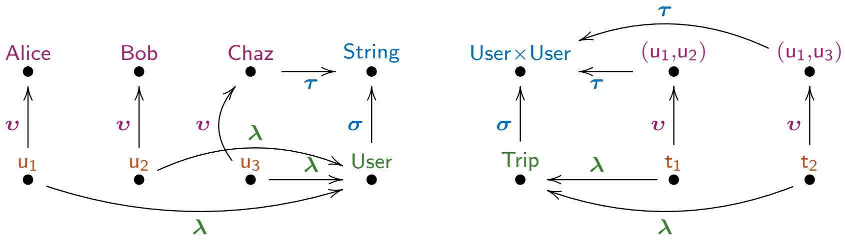
$\mathcal{E}$lement		
ID	λ	υ
t_1	Trip	(u_1, u_2)
t_2	Trip	(u_1, u_3)
u_1	User	Alice
u_2	User	Bob
u_3	User	Chaz

$\mathcal{V}$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

Algebraic Property Graphs as 4-sorted Triples



$\mathcal{E}$lement		
ID	$\boldsymbol{\lambda}$	$\boldsymbol{\upsilon}$
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	$\boldsymbol{\tau}$
Alice	String
Bob	String
Chaz	String
(u_1, u_2)	User × User
(u_1, u_3)	User × User

$\mathcal{L}$abel	
ID	$\boldsymbol{\sigma}$
User	String
Trip	User × User

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

Algebraic Property Graphs with Product Schemas

$$
\begin{tikzcd}
\text{String} & \text{Trip} \arrow[d, "\phi_{\text{Trip}}"] \\
\text{User} \arrow[u, "\phi_{\text{User}}"] & \text{User} \times \text{User} \arrow[l, bend right, "\text{fst}"'] \arrow[l, bend left, "\text{snd}"]
\end{tikzcd}
$$

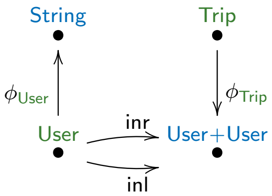
User × User		
ID	fst	snd
(u_1, u_1)	u_1	u_1
(u_1, u_2)	u_1	u_2
(u_1, u_3)	u_1	u_3
(u_2, u_1)	u_2	u_1
(u_2, u_2)	u_2	u_2
(u_2, u_3)	u_2	u_3
(u_3, u_1)	u_3	u_1
(u_3, u_2)	u_3	u_2
(u_3, u_3)	u_3	u_3

User	
ID	ϕ_{User}
u_1	Alice
u_2	Bob
u_3	Chaz

Trip	
ID	ϕ_{Trip}
t_1	(u_1, u_2)
t_2	(u_1, u_3)

String
ID
Alice
Bob
Chaz

Algebraic Property Graphs with Sum Schemas



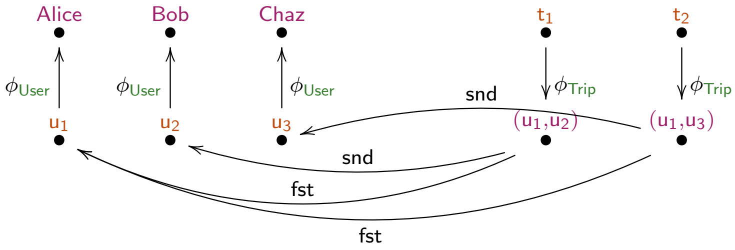
User + User
ID
inl(u_1)
inl(u_2)
inl(u_3)
inr(u_1)
inr(u_2)
inr(u_3)

User			
ID	ϕ_{User}	inl	inr
u_1	Alice	inl(u_1)	inr(u_1)
u_2	Bob	inl(u_2)	inr(u_2)
u_3	Chaz	inl(u_3)	inr(u_3)

Trip	
ID	ϕ_{Trip}
t_1	inl(u_1)
t_2	inr(u_2)

String
ID
Alice
Bob
Chaz

Algebraic Property Graphs as Typed-sorted Triples



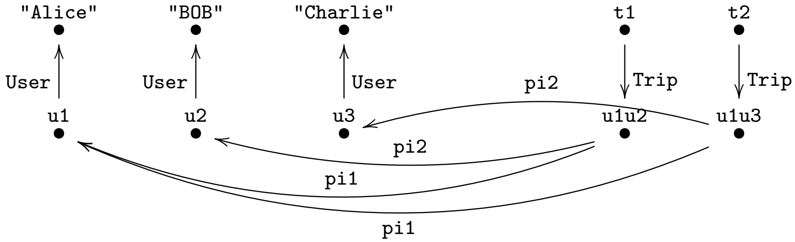
User × User		
ID	fst	snd
(u_1, u_2)	u_1	u_2
(u_1, u_3)	u_1	u_3

User	
ID	ϕ_{User}
u_1	Alice
u_2	Bob
u_3	Chaz

Trip	
ID	ϕ_{Trip}
t_1	(u_1, u_2)
t_2	(u_1, u_3)

String
ID
Alice
Bob
Chaz

Algebraic Property Graphs as RDF



User × User			
ID	fst	snd	Resource
(u_1, u_2)	u_1	u_2	u1u2
(u_1, u_3)	u_1	u_3	u1u3

User		
ID	ϕ_{User}	Resource
u_1	Alice	u1
u_2	Bob	u2
u_3	Charlie	u3

Trip		
ID	ϕ_{Trip}	Resource
t_1	(u_1, u_2)	t1
t_2	(u_1, u_3)	t2

String	
ID	Resource
Alice	"Alice"
Bob	"BOB"
Chaz	"Charlie"