Proving database migrati correctness with category theory

Aaron Walter, fall 2020

CSci senior seminar

Talk outline

- Introduction
- Databases
- Data migration
- Graphs
- Representing databases with categories
- Functorial data migration
- Conclusions
- Questions

What are relational databases?

- Collection of tables
- Table rows are *records*
- Table columns are *attributes*
- Each record has a (globally unique) primary key
- Records contain foreign keys or raw data

Table 1: People					
id	FName	LName	BestFriend	Apartment	FavAnimal
p1	Joe	Schmoe	p2	a1	c1
p2	Jerry	Schmoe	p1	a1	c2
p3	Walter	Walter	p3	a2	c3
p4	Alice	Ames	p5	a3	c2
p5	Bob	Baker	p4	a3	c1

Table 2: Apartments	
id	Name
a1	Sunnyside
a2	High Oaks
a3	The Continental

Table 3: Animals	
id	Name
c1	Dog
c2	Cat
c3	Goat

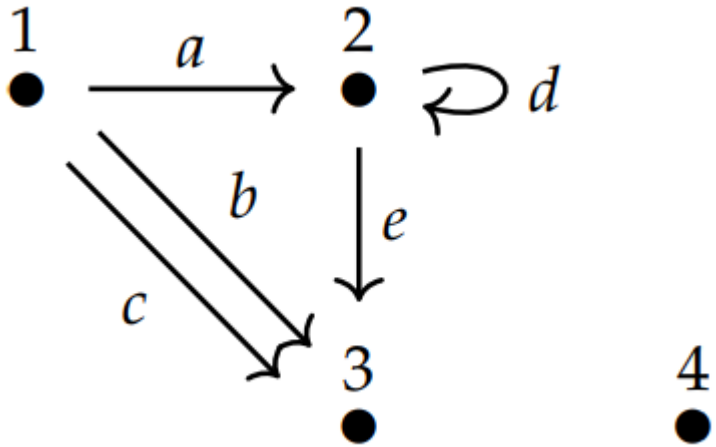
What is data migration?

- We want to move data between databases
- We will reinterpret data from one structure in the context of another structure
- What can go wrong?
  - Missing data
  - Mismatched data
  - Incorrect data

Graphs

A graph has…

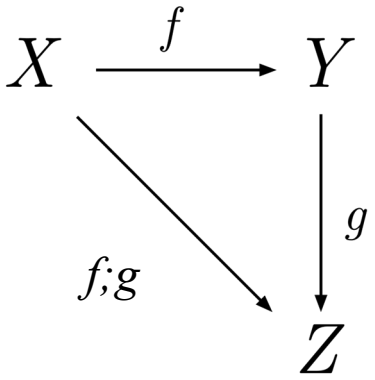
- A set of vertices (1, 2, 3, 4)
- A set of arrows (a, b, c, d, e), each of which has a start vertex and an end vertex



- We can describe paths in a graph by concatenating arrows.
- We also allow the trivial path of length 0 on each node.

[3]

What is category theory?



- Branch of mathematics
- About relationships and structures
- Important concepts: category, functor, natural transformation, adjoint

[4]

Representing relational databases categorically

Categorical representations of databases come in two forms: schemas and instances.

Beatle	Played
George	Lead guitar
John	Rhythm guitar
Paul	Bass guitar
Ringo	Drums

Rock-and-roll instrument
Bass guitar
Drums
Keyboard
Lead guitar
Rhythm guitar

[3]

A crash course in categories

Definition: a *category* $\mathbf{C}$ has...

- A collection of objects, called $\mathrm{Ob}(\mathbf{C})$.
- For every two objects c, d in $\mathrm{Ob}(\mathbf{C})$, a set $\mathbf{C}(c, d)$ of *morphisms* from c to d.
- For every object c in $\mathrm{Ob}(\mathbf{C})$, a morphism id_c in $\mathbf{C}(c, c)$ called the *identity morphism on c*.
- For every three objects c, d, e in $\mathrm{Ob}(\mathbf{C})$, and morphisms f in $\mathbf{C}(c, d)$ and g in $\mathbf{C}(d, e)$, we specify a morphism $f;g$ in $\mathbf{C}(c, e)$ called the *composite of f and g*.

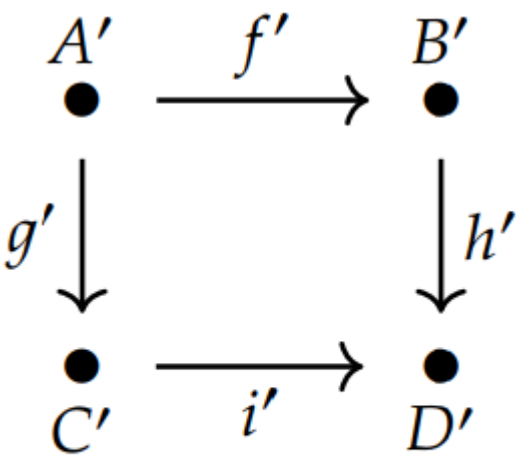
A crash course in categories, continued

The components of a category must satisfy these conditions:

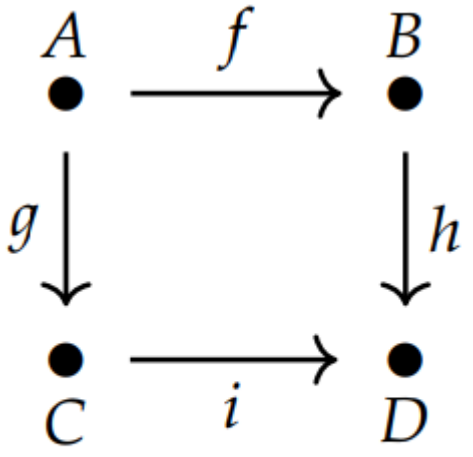
1. *unitality* : for any morphism $f : c \to d$, composing with the identities at c or d does nothing, i.e. $id_c ; f = f ; id_d = f$.
2. *associativity* : for any three morphisms $f : c_0 \to c_1$, $g : c_1 \to c_2$, and $h : c_2 \to c_3$, $f ; (g ; h)$ and $(f ; g) ; h$ are equal. We write this composite as $f ; g ; h$.

For our purposes, we can think of categories as graphs where we can declare paths to be equal.

Looking at path equivalences: the free and commutative



no equations



$f \fatsemi h = g \fatsemi i$

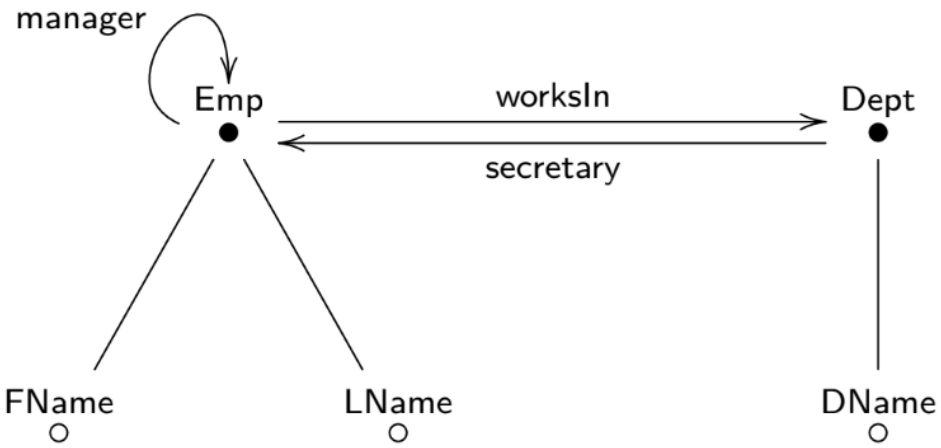
[3]

Two special categories

- A *discrete category* has no morphisms other than identity morphisms
- **Set**, the category of sets
  - Objects: sets
  - Morphisms: functions between sets
- We will use these ideas to integrate data types into our database schemas

Database schemas as categories

- In a schema **S**...
  - Objects are table names
  - Morphisms are column names that correspond to foreign keys
  - Attributes with data types are open circles with lines



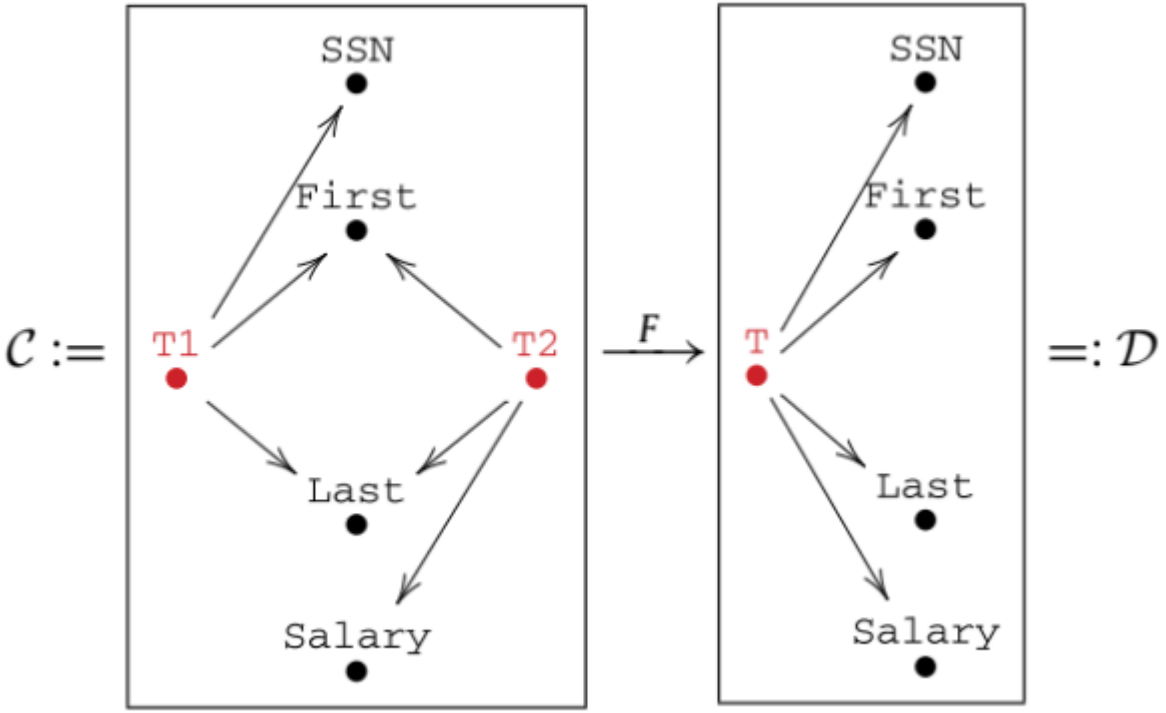
$$\text{Emp.manager.worksIn} = \text{Emp.worksIn}$$

$$\text{Dept.secretary.worksIn} = \text{Dept}$$

Functors: how we translate between schemas

A functor maps objects in **C** to objects in **D**, and morphisms in **C** to morphisms in **D**.

A functor must respect identity, and it should produce the same results whether we apply the functor in the source or target category.



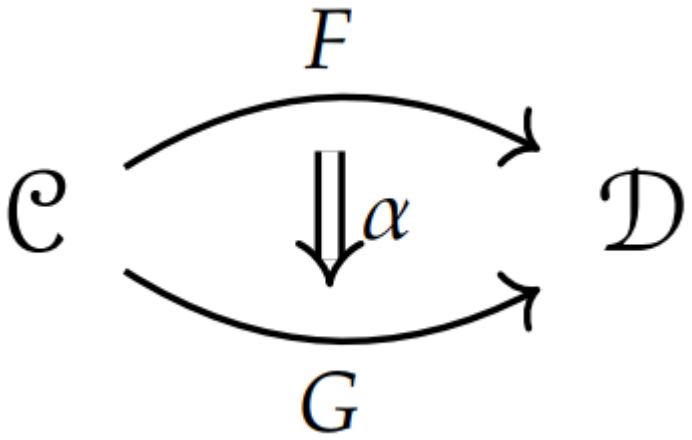
Database instances as functors

- An *instance* on a schema **S** is a functor from **S** to **Set**
  - Table names (objects) in **S** are mapped to sets in **Set** that hold that table's records
  - Column names (morphisms) in **S** are mapped to functions in **Set** that map records in a table to records in another table
- All the possible instances on a schema **S** form a category, **S-Inst**. (Its morphisms are natural transformations.)

Natural transformations: how we translate between f

- For every object c in **C**, we choose a morphism α_c in **D**
- Our choices must obey the naturality condition:
- For every morphism $f: x \to y$ in **C**, we have

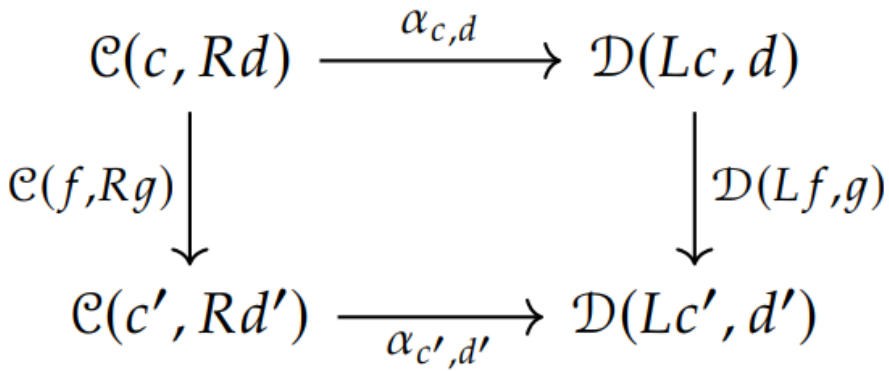
$$F(f); \alpha_y = \alpha_x; G(f)$$



[3]

Adjoints: "duals" or "shadows" of functors

- Given categories C, D and functors $L: C \to D$, $R: D \to C$, we say L is the left adjoint of R if...
- The sets $C(c, R(d))$ and $D(L(c), d)$ have the same number of elements, and these α-mappings make the diagram commute for all morphisms $f: c' \to c$ and $g: d \to d'$.

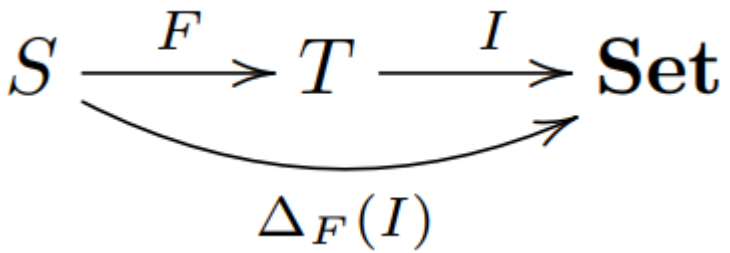


[3]

The data migration functors: how we construct queri

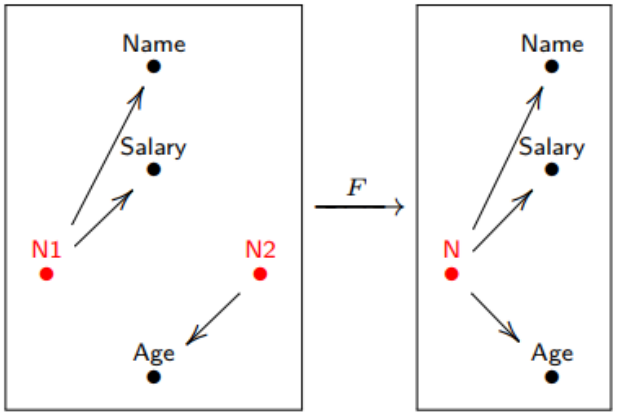
Given a functor F from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, we define:

- The pullback functor: Δ_F: $\mathbf{T}$-Inst $\to$ $\mathbf{S}$-Inst to be $\Delta_F(I) = F; I$, which is a functor from $\mathbf{S} \to \mathbf{Set}$



- The left push-forward functor: Σ_F: $\mathbf{S}$-Inst $\to$ $\mathbf{T}$-Inst to be the left adjoint of Δ_F
- The right push-forward functor: Π_F: $\mathbf{S}$-Inst $\to$ $\mathbf{T}$-Inst to be the right adjoint of Δ_F

Applying the data migration functors



N1		
ID	Name	Salary
1	Alice	$100
2	Bob	$250
3	Sue	$300

N2	
ID	Age
4	20
5	20
6	30

$\xleftarrow{\Delta_F}$

N			
ID	Name	Salary	Age
a	Alice	$100	20
b	Bob	$250	20
c	Sue	$300	30

$\downarrow \Sigma_F$

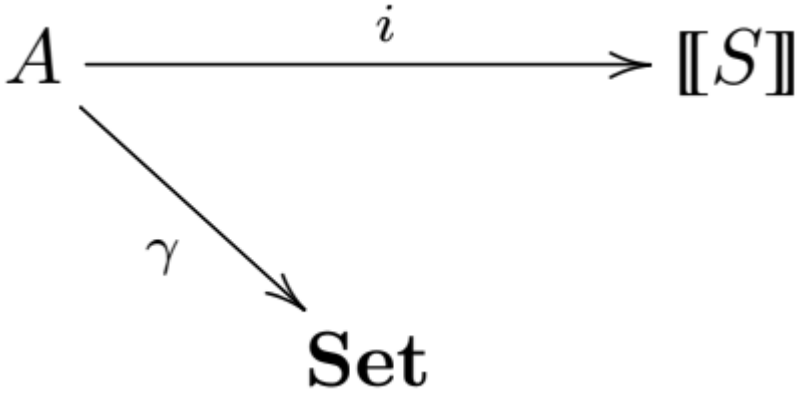
$\searrow \Pi_F$

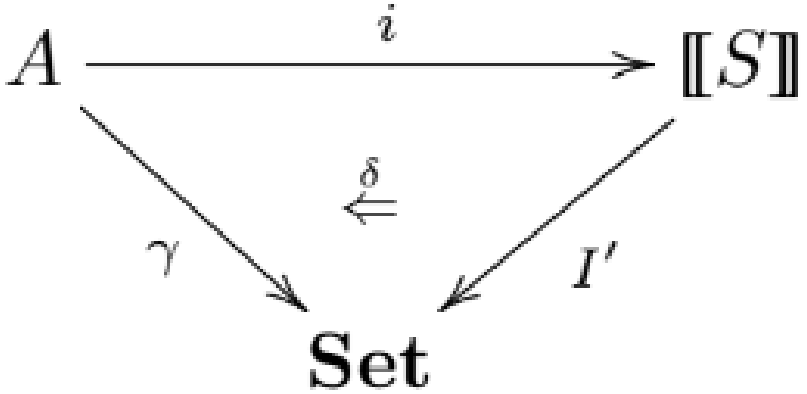
N			
ID	Name	Salary	Age
a	Alice	$100	$null_1$
b	Bob	$250	$null_2$
c	Sue	$300	$null_3$
d	$null_4$	$null_5$	20
e	$null_6$	$null_7$	20
f	$null_8$	$null_9$	30

N			
ID	Name	Salary	Age
a	Alice	$100	20
b	Alice	$100	20
c	Alice	$100	30
d	Bob	$250	20
e	Bob	$250	20
f	Bob	$250	30
g	Sue	$300	20
h	Sue	$300	20
i	Sue	$300	30

[2]

How do we deal with raw data?





A: a discrete category with the attributes/columns of **S** as objects

[2]

Does it really work?

- Categorical Query Language (CQL): categoricaldata.net
- Conexus AI: conexus.com
  - "Unlock Business Opportunities That Require Full Semantic Integration Of Your Data and Complex Constraint Adherence"
  - "Automate Your SQL ETL Processes with Mathematically Provable Data Integrity and Save Your Project Before It's Too Late"

conexus

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

Conclusions

- Data migration is a challenge database systems should address
- Data migration is the act of moving data between schemas
- We've reinterpreted this act in terms of category theory
- The structure-preserving properties of category-theoretic constructs ensure correctness of data migration

References

[1] D. I. Spivak. Functorial data migration. Information and Computation, 217:31 – 51, 2012.

[2] D. I. Spivak and R. Wisnesky. Relational foundations for functorial data migration. InProceedings of the 15th Symposium on Database Programming Languages,DBPL 2015, page 21 –28, New York, NY, USA, 2015. Association for Computing Machinery.

[3] B. Fong and D. I. Spivak.An invitation to applied category theory: seven sketches in compositionality. Cambridge University Press, 2019.

[4] https://en.wikipedia.org/wiki/Category_theory