Building Database Management on top of Category Theory in Coq

Jason Gross — jgross@mit.edu

POPL 2013

This document is available at http://web.mit.edu/jgross/Public/POPL/jgross-student-talk.pdf.
My category theory library is available at https://bitbucket.org/JasonGross/catdb.

Outline

Introduction — Databases and Category Theory
- Categories
- Relational Databases
- Relational Database Schema = Category
- Usefulness

Outline

Introduction — Databases and Category Theory
- Categories
- Relational Databases
- Relational Database Schema = Category
- Usefulness

Category Theory in Coq
- Universe Levels
- Limits and Colimits

Categories

A category is:

- a collection of objects,

Categories

A category is:

- a collection of objects, together with
- arrows between those objects,

Categories

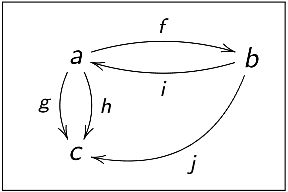
A category is:

- a collection of objects, together with
- arrows between those objects, together with
- a composition law for the arrows satisfying coherence conditions:
  - existence of identity
  - associativity

Categories

A category is:

- a collection of objects, together with
- arrows between those objects, together with
- a composition law for the arrows satisfying coherence conditions:
  - existence of identity
  - associativity



Relational Databases

A database schema for a relational database can be modeled as a

- collection of **tables**,

Relational Databases

A database schema for a relational database can be modeled as a

- collection of **tables**, together with
- a collection of **attributes** or column-labels for each table,

Relational Databases

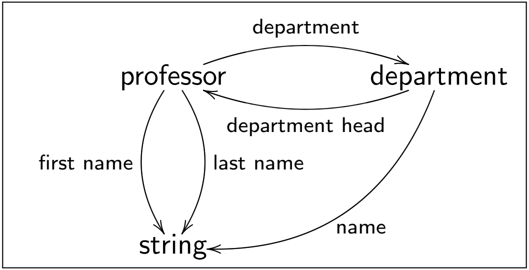
A database schema for a relational database can be modeled as a

- collection of **tables**, together with
- a collection of **attributes** or column-labels for each table, together with
- integrity constraints

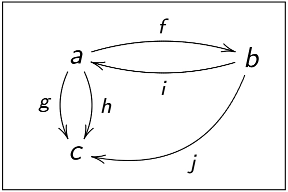
Relational Databases

A database schema for a relational database can be modeled as a

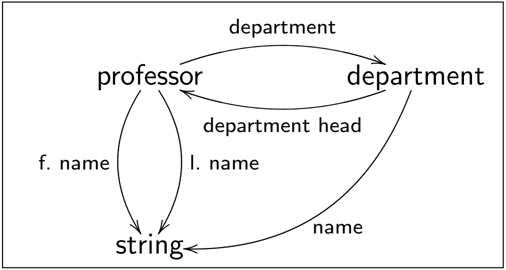
- collection of **tables**, together with
- a collection of **attributes** or column-labels for each table, together with
- integrity constraints



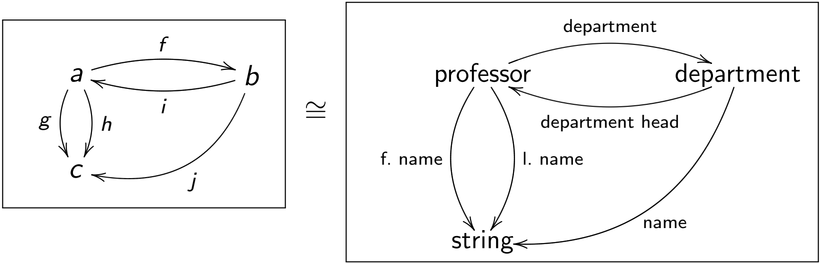
Relational Database Schema = Category



Relational Database Schema = Category



Relational Database Schema = Category



The diagrams are “the same”.

Usefulness of Categorical Databases

- Built in notion of path equivalence (multiple equivalent paths of foreign keys can be a pain in typical database management).

Usefulness of Categorical Databases

- Built in notion of path equivalence (multiple equivalent paths of foreign keys can be a pain in typical database management).
- Provides a rigorous language for data migration between databases (another hard task in standard database management).

Category Theory in Coq

- Many people learn a proof assistant by coding up category theory.

Category Theory in Coq

- Many people learn a proof assistant by coding up category theory.
- Category theory is relatively simple to code up.

Category Theory in Coq

- Many people learn a proof assistant by coding up category theory.
- Category theory is relatively simple to code up.
  - Standard rigorous formulation of concepts exists in the literature.

Category Theory in Coq

- Many people learn a proof assistant by coding up category theory.
- Category theory is relatively simple to code up.
  - Standard rigorous formulation of concepts exists in the literature.
  - It's rare to get caught up in minute details of proofs.

Category Theory in Coq

- Many people learn a proof assistant by coding up category theory.
- Category theory is relatively simple to code up.
    - Standard rigorous formulation of concepts exists in the literature.
    - It's rare to get caught up in minute details of proofs.
    - If you can define something categorically, it's probably interesting.

Universe Levels (Russel's Paradox)

- Consider, naïvely, the set of all sets.

Universe Levels (Russel's Paradox)

- Consider, naïvely, the set of all sets. Does it contain itself?

Universe Levels (Russel's Paradox)

- Consider, naïvely, the set of all sets. Does it contain itself?
  - It's a set, and it contains all sets, so it must.

Universe Levels (Russel's Paradox)

- Consider, naïvely, the set of all sets. Does it contain itself?
  - It's a set, and it contains all sets, so it must.
- Consider the set of all sets that do not contain themselves. Does it contain itself?

Universe Levels (Russel's Paradox)

- Consider, naïvely, the set of all sets. Does it contain itself?
  - It's a set, and it contains all sets, so it must.
- Consider the set of all sets that do not contain themselves. Does it contain itself?
  - If it contains itself, then it is not a set that doesn't contain itself, and so it cannot be a member of itself; contradiction. Thus it cannot contain itself.

Universe Levels (Russel's Paradox)

- Consider, naïvely, the set of all sets. Does it contain itself?
  - It's a set, and it contains all sets, so it must.
- Consider the set of all sets that do not contain themselves. Does it contain itself?
  - If it contains itself, then it is not a set that doesn't contain itself, and so it cannot be a member of itself; contradiction. Thus it cannot contain itself.
  - If it does not contain itself, then it is a set that does not contain itself, and thus must be a member of itself; contradiction. Thus it cannot fail to contain itself.

Universe Levels (Russel's Paradox)

- Consider, naïvely, the set of all sets. Does it contain itself?
  - It's a set, and it contains all sets, so it must.
- Consider the set of all sets that do not contain themselves. Does it contain itself?
  - If it contains itself, then it is not a set that doesn't contain itself, and so it cannot be a member of itself; contradiction. Thus it cannot contain itself.
  - If it does not contain itself, then it is a set that does not contain itself, and thus must be a member of itself; contradiction. Thus it cannot fail to contain itself.
- This is the **paradox of naïve set theory**.

Universe Levels

- This is the **paradox of naïve set theory**.
- Solution: universe levels

Universe Levels

- This is the **paradox of naïve set theory**.
- Solution: universe levels
  - `Set` or `Type`(0) is the collection of all sets, `Type`(1) is the collection of all `Type`(0)s, . . . , `Type`($i+1$) is the collection of all Type(i)s
  - The **universe level** of an object of type `Type`(i) is i

Universe Levels

- This is the **paradox of naïve set theory**.
- Solution: universe levels
  - `Set` or `Type`(0) is the collection of all sets, `Type`(1) is the collection of all `Type`(0)s, ..., `Type`($i+1$) is the collection of all Type(i)s
  - The **universe level** of an object of type `Type`(i) is i
- In some cases, Coq can infer the universe level of an inductive type from the universe levels of its parameters; when this happens, the inductive type is polymorphic over universe levels.

Universe Levels

- This is the **paradox of naïve set theory**.
- Solution: universe levels
  - `Set` or `Type`(0) is the collection of all sets, `Type`(1) is the collection of all `Type`(0)s, ..., `Type`($i+1$) is the collection of all Type(i)s
  - The **universe level** of an object of type `Type`(i) is i
- In some cases, Coq can infer the universe level of an inductive type from the universe levels of its parameters; when this happens, the inductive type is polymorphic over universe levels.
- It's useful to talk about "a category whose objects are of type `T`" rather than just "a category".

Limits and Colimits

- Categorical **limits** are like **Cartesian products**, subject to constraints about equality of components

Limits and Colimits

- Categorical **limits** are like **Cartesian products**, subject to constraints about equality of components
- Categorical **colimits** are like **disjoint unions**, modulo equivalence relations

Coq Category

- Coq has all limits
  - Product types provide products (function types, e.g, `forall a : A, f a` is the product $\prod_{a \in A} f(a)$)
  - Sigma types provide constraints about equality of components (e.g., `{ f : A → B | f a = f b }`)

Coq Category

- Coq has all limits
  - Product types provide products (function types, e.g, `forall a : A, f a` is the product $\prod_{a \in A} f(a)$)
  - Sigma types provide constraints about equality of components (e.g., `{ f : A → B | f a = f b }`)
- Coq has some colimits
  - Sigma types provide disjoint unions (e.g., `{ j : J & f j }` is the disjoint union $\coprod_{j \in J} f(j)$)
  - Quotients are ... hard

Quotients

- Quotients can be defined via axioms

Quotients

- Quotients can be defined via axioms
  - `proof_irrelevance`; `(A ↔ B) → A = B` for propositions; either decidable existence, or a way of turning proofs of existence into objects (`constructive_indefinite_description : (exists x, P x) → { x | P x }`)
  - Not computational

Quotients

- Quotients can be defined via axioms
  - `proof_irrelevance`; `(A ↔ B) → A = B` for propositions; either decidable existence, or a way of turning proofs of existence into objects (`constructive_indefinite_description : (exists x, P x) → { x | P x }`)
  - Not computational
- Quotients can be defined via setoids

Quotients

- Quotients can be defined via axioms
  - `proof_irrelevance`; `(A ↔ B) → A = B` for propositions; either decidable existence, or a way of turning proofs of existence into objects (`constructive_indefinite_description : (exists x, P x) → { x | P x }`)
  - Not computational
- Quotients can be defined via setoids
  - All objects carry around extra information of what the equivalence relation is
  - This is somewhat clunky
  - Not first-class quotients

Limits and Colimits (High-Level Summary)

- There are two categorical constructions (limits and colimits) that are "dual"

Limits and Colimits (High-Level Summary)

- There are two categorical constructions (limits and colimits) that are "dual"
- Coq's type-system fully implements only one of these (limits)

Limits and Colimits (High-Level Summary)

- There are two categorical constructions (limits and colimits) that are "dual"
- Coq's type-system fully implements only one of these (limits)
- It's harder to define colimits inside of Coq than limits, in general, even for the ones that Coq does support

Thank You!