Mixin' it Up: Datastore Edition

Chris Cahoon
Eric Oestrich

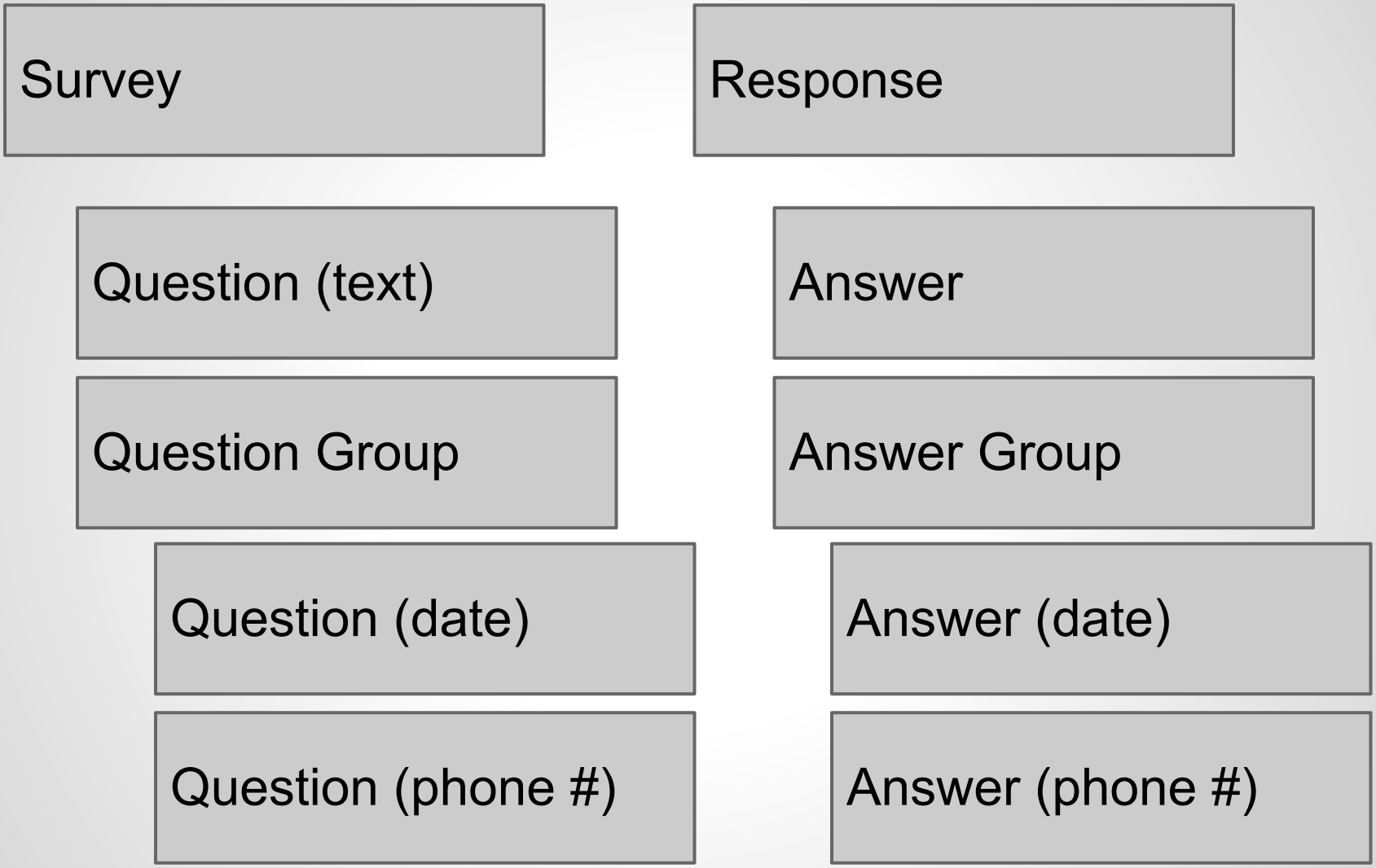

smartl • gic

Using only an object store on a largeish app

and then remembering you can use more than one database in a system

The (Central) Problem

Hierarchical forms & responses


Survey
Response
Question (text)
Answer
Question Group
Answer Group
Question (date)
Answer (date)
Question (phone #)
Answer (phone #)

NUMBERS

Lots of questions
Lots of forms
Lots of users
Lots of responses
= Lots of queries

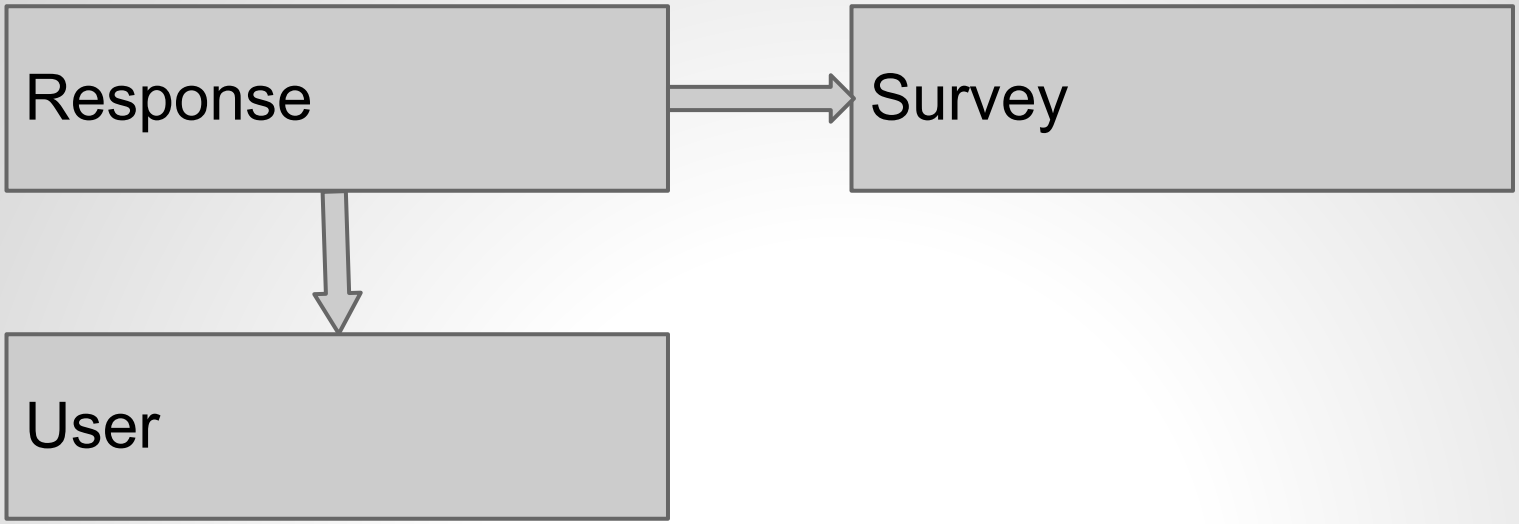
"We need an object store!"

MONGOID!

feels like ActiveRecord inside MongoDB

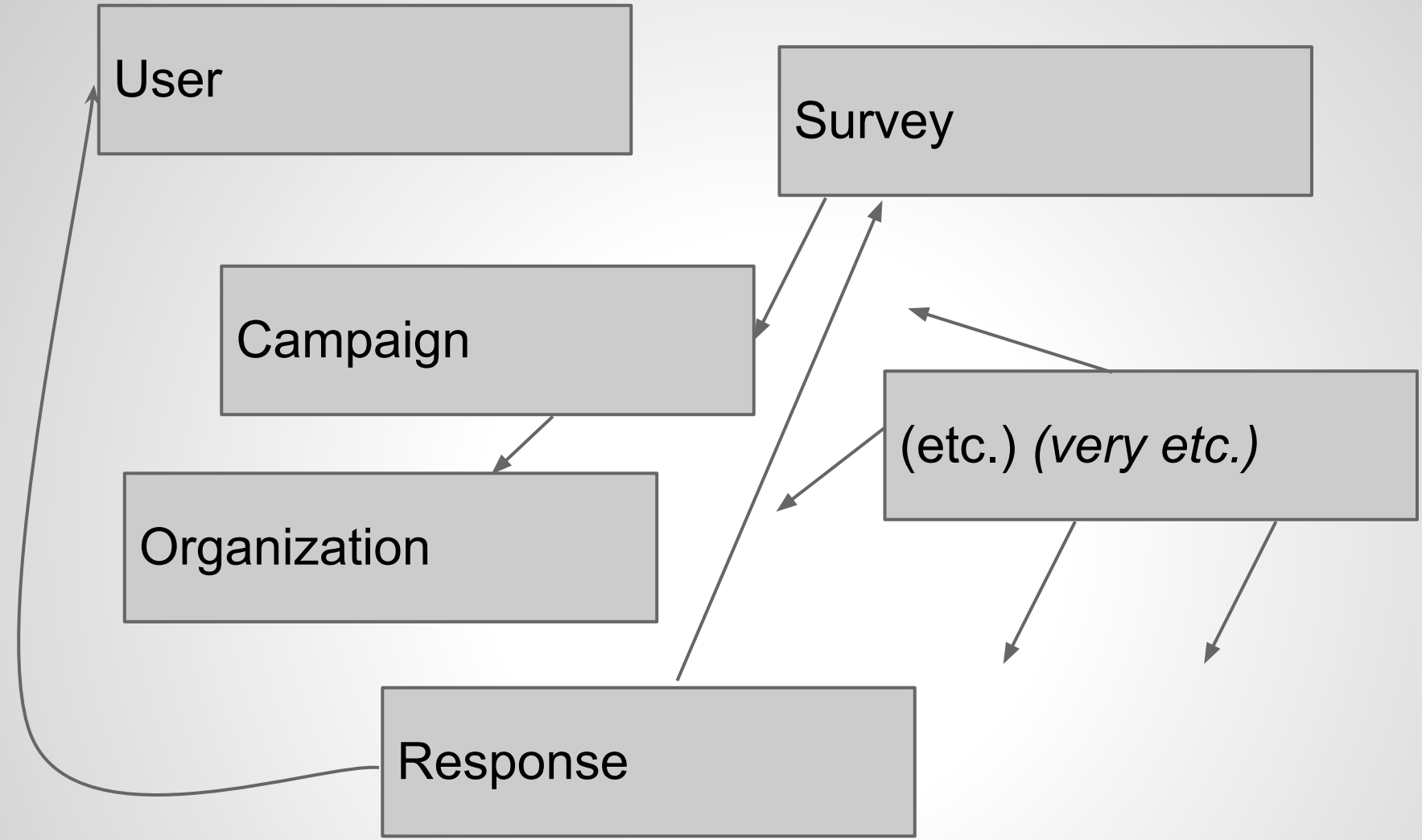
"This form belongs_to user!"

"Mongoid has helpers for relations?!"



Not bad.

(months pass...)



Oh.

Mongoid works perfectly for those two models

...but everything else is relational!

There are problems with object stores

... used like a relational db

(there are things we missed from relational DBs)

Relations

Can only query one collection at a time

Denormalization

No foreign keys

I accidentally the object

Can I have my cake?

and eat it too?

Just Mongoid

Querying

```
# 2 queries
surveys = Survey.where(:campaign_id.in =>
     @user.campaigns.map(&:id))

survey_ids = surveys.map(&:id)

# + 1 query
Response.where(:survey_id.in => survey_ids,
   :complete => true).first.answers

# = 3 total queries for search, not
chainable
```

Interlude

Mongoid & ActiveRecord

Bridging the gap

```ruby
class Response < ActiveRecord::Base
  belongs_to :survey
  has_one :campaign, :through => :survey

  before_create :create_answer_collection

  def answer_collection
    AnswerCollection.find(self.bson_id)
  end

  private
  def create_answer_collection
    self.bson_id = AnswerCollection.create.id
  end
end
```

Search with SQL, store with Mongo

```
# 1 query
Response.
   joins(:survey, :campaign).
   where('complete' => true,
      "campaigns.user_id" => @user.id). first.
   answer_collection

# = 1 total query for search, chainable
```

```ruby
class Response < ActiveRecord::Base
  include Mongoid::ActiveRecordBridge

  belongs_to :survey
  has_one :campaign, :through => :survey

  attach_mongoid_document :answer_collection, :create => true
end
```

```ruby
module Mongoid::ActiveRecordBridge
  extend ActiveSupport::Concern

  module ClassMethods
    def attach_mongoid_document(klass_sym, options = {})
      mongo_document_finder(klass_sym)

      mongo_document_creater(klass_sym) if options[:create]
    end

    private
    def mongo_document_finder(klass_sym)
      define_method(klass_sym) {
        constant = klass_sym.to_s.camelize.constantize
        constant.find(self.bson_id)
      }
    end

    def mongo_document_creater(klass_sym)
      before_create :"create_#{klass_sym}"
      define_method("create_#{klass_sym}") {
        constant = klass_sym.to_s.camelize.constantize
        self.bson_id = constant.create.id
      }
    end
  end
end
```

Should we gemify this?

GitHub:

https://github.com/oestrich/mixin_it_up_datastore

(look at the branches)

Thanks