KONGREGATE

Modeling Denormalization

Duncan Beevers 2008

What is denormalization?

Duplicating facts

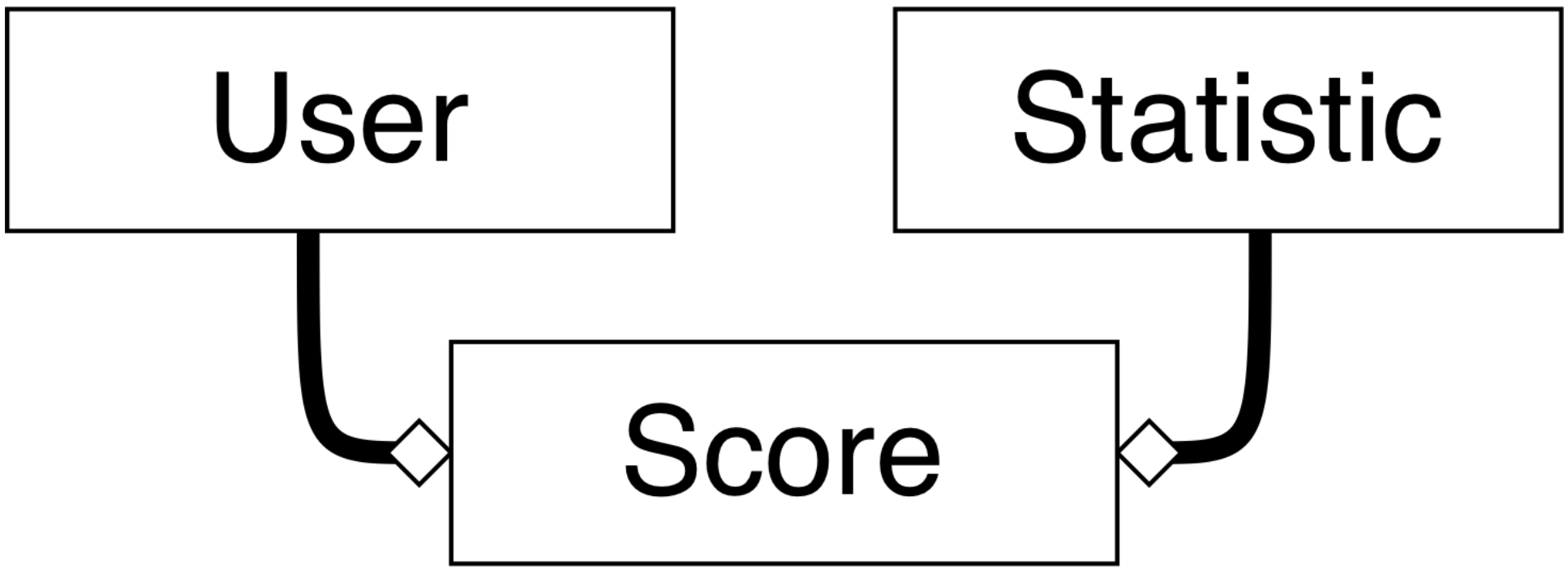
Duplicating facts

Modeling

```
class Game < ActiveRecord::Base
  has_many :plays
  has_one :lifetime_plays_count
end
```

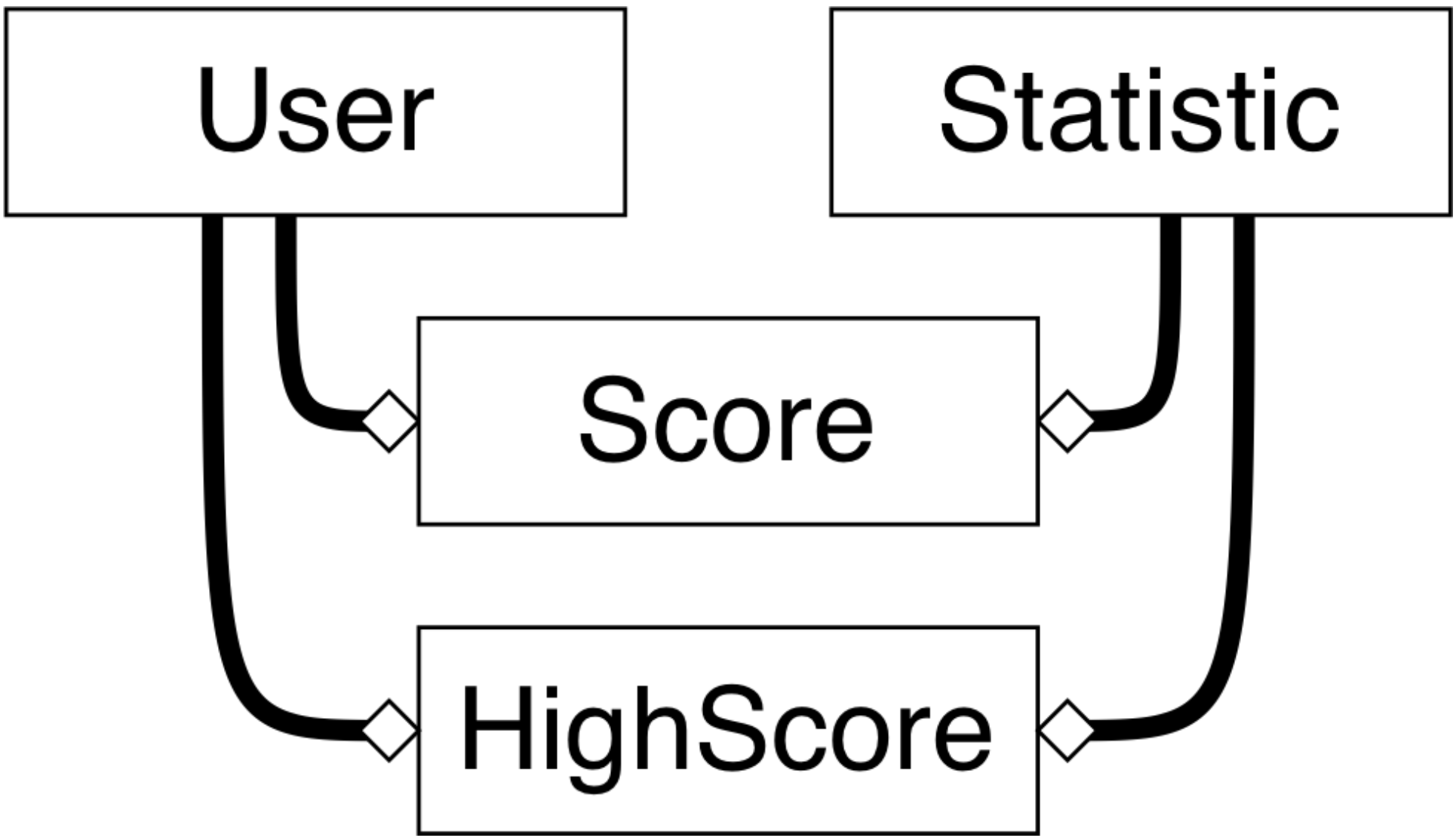
Piles of things are things

Where we started



Fully normalized data

First denormalization



Why denormalize?

- Large number of rows
- Frequent expensive queries
- Complex business rules about data

Complex Business Rules

They make simple Business Rules?

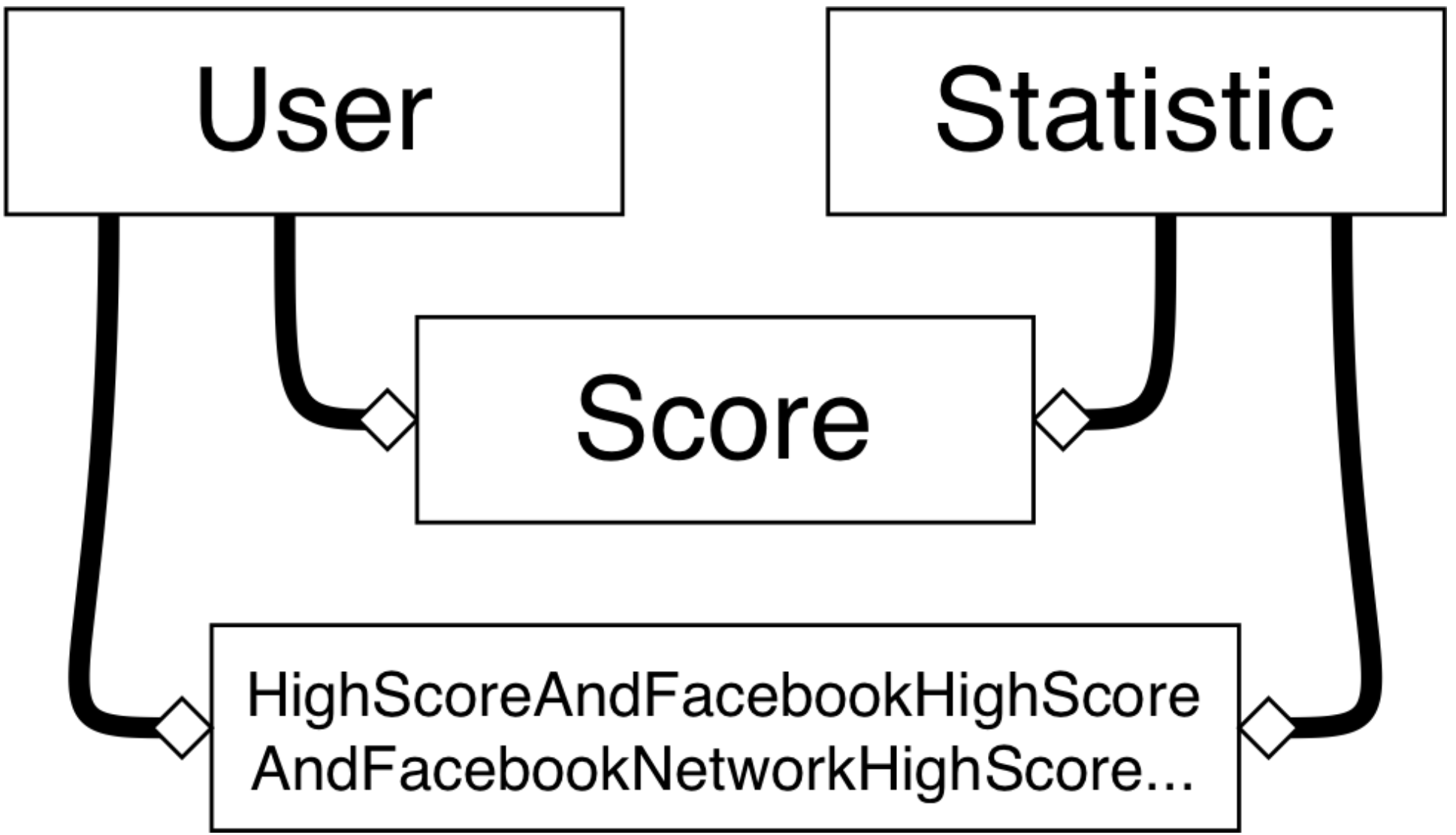
Model the requirement

Just get it working first

```ruby
class DailyHighScore < ActiveRecord::Base
  set_table_name 'scores'

  default_scope :select =>
    "#{quoted_table_name}.*, MAX(value) AS value"

  named_scope :for_date, lambda { |date|
    { :conditions => {
      :created_at => date.beginning_of_day ..
        date.end_of_day
    } }
  }
end

class User < ActiveRecord::Base
  has_many :daily_high_scores
end

user.daily_high_scores.
  for_date(Date.today).first
```

```ruby
class ActiveRecord::Base
  def self.default_scope options
    self.scoped_methods <<
      { :find => options }
  end
end
```

Haphazard denormalization

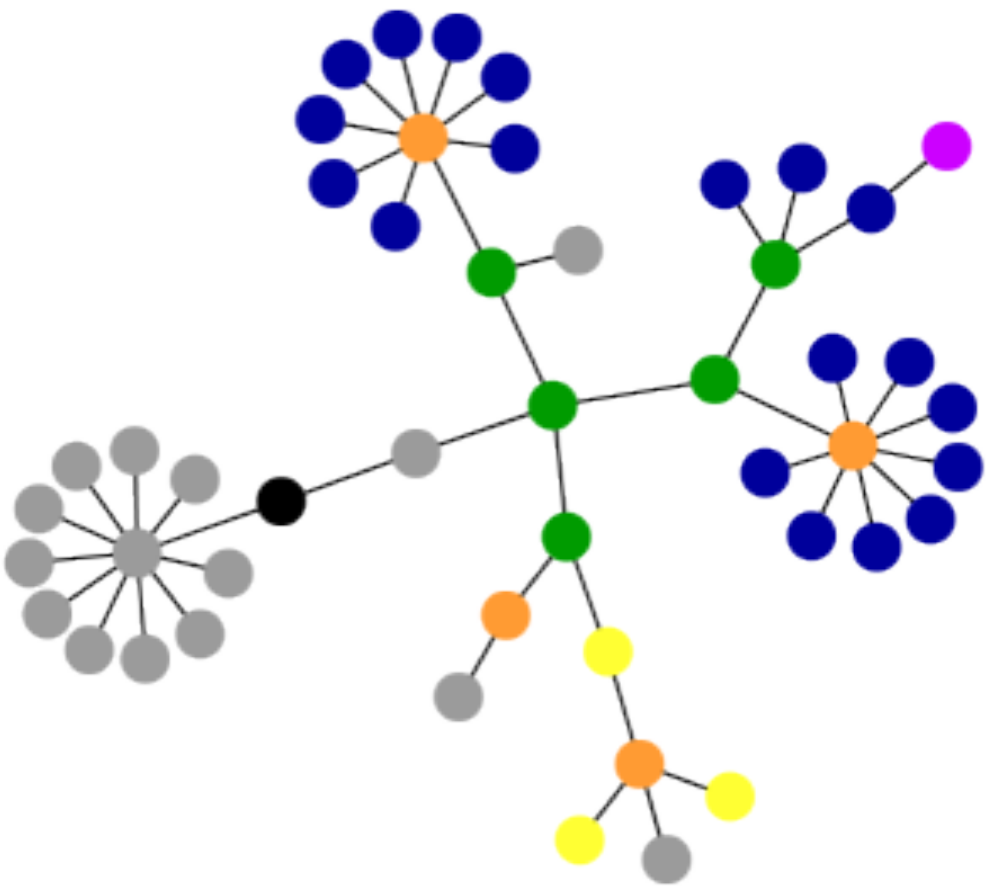


Minimize Indices-per-table

more indices = slower writes

Minimize complex queries

Make sure you're using your indices!



Why use real tables instead of triggers or views?

Eventual Consistency

Do it later

Batch processing can save you writes and lighten replication load

Take a picture, it'll last longer

Denormalized models capture history

```
class DailyHighScore
  uniqueness :user_id,
    :statistic_id, :date

  takes :user_id, :statistic_id,
    :from => Score

  takes :value, :from => Score,
    :if => :new_score_better?

  def new_score_better? new_score
    new_score.better_than? self.value
  end
end
```

Like Observers, but different

```
class LifetimeHighScore
end

class DailyGameplaysCount
end

class MonthlyPaymentRun
end
```

Use Descriptive Names

VeryLongDescriptiveNamesThatProgrammingPairsThinkProvideGoodDescriptions

Avoiding our mistakes

Updates to models triggered external API calls

Single Table Inheritance

Use a queue

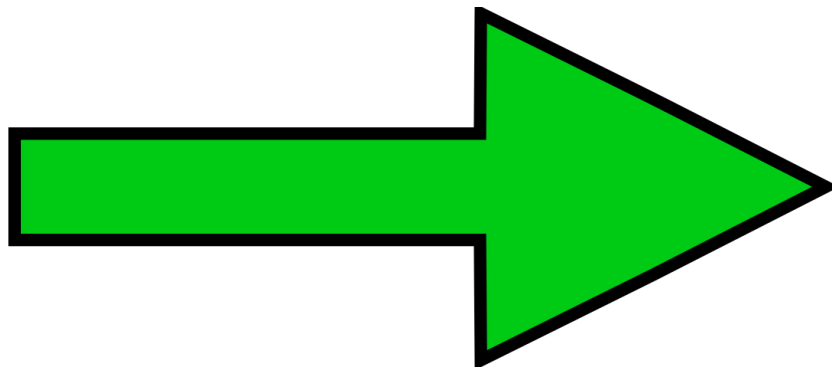
Lots of choices out there, or just roll your own

Ye Olde Cleaver

http://github.com/stopdropandrew/single-table-inheritance-cleaver

Pulling up stakes

Moving to the new scheme



Interim strategies

Disabling features and turning tables into placeholders

```
CREATE TABLE dummy_#{table_name}
LIKE #{table_name}
ENGINE=BLACKHOLE
```

ewww SQL

- Hash rewrite http://www.dweebd.com/ruby/hash-key-rewrite/

```
{ :username => 'johnnyfive' }.
  rewrite(:username => :login)

=> { :login => 'johnnyfive' }
```

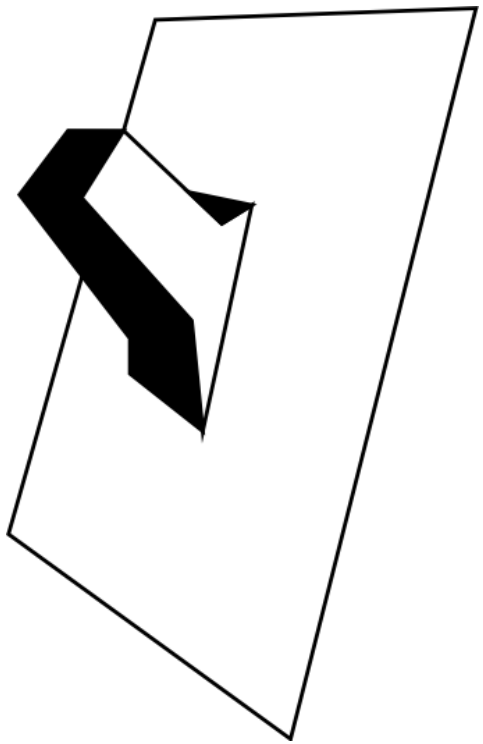
- Bulk insert http://gist.github.com/7681

```
Gameplay.insert( [
  { :user_id => 1, :game_id => 1 },
  { :user_id => 2, :game_id => 1 }
] )
```

Toolbox

Catch-up

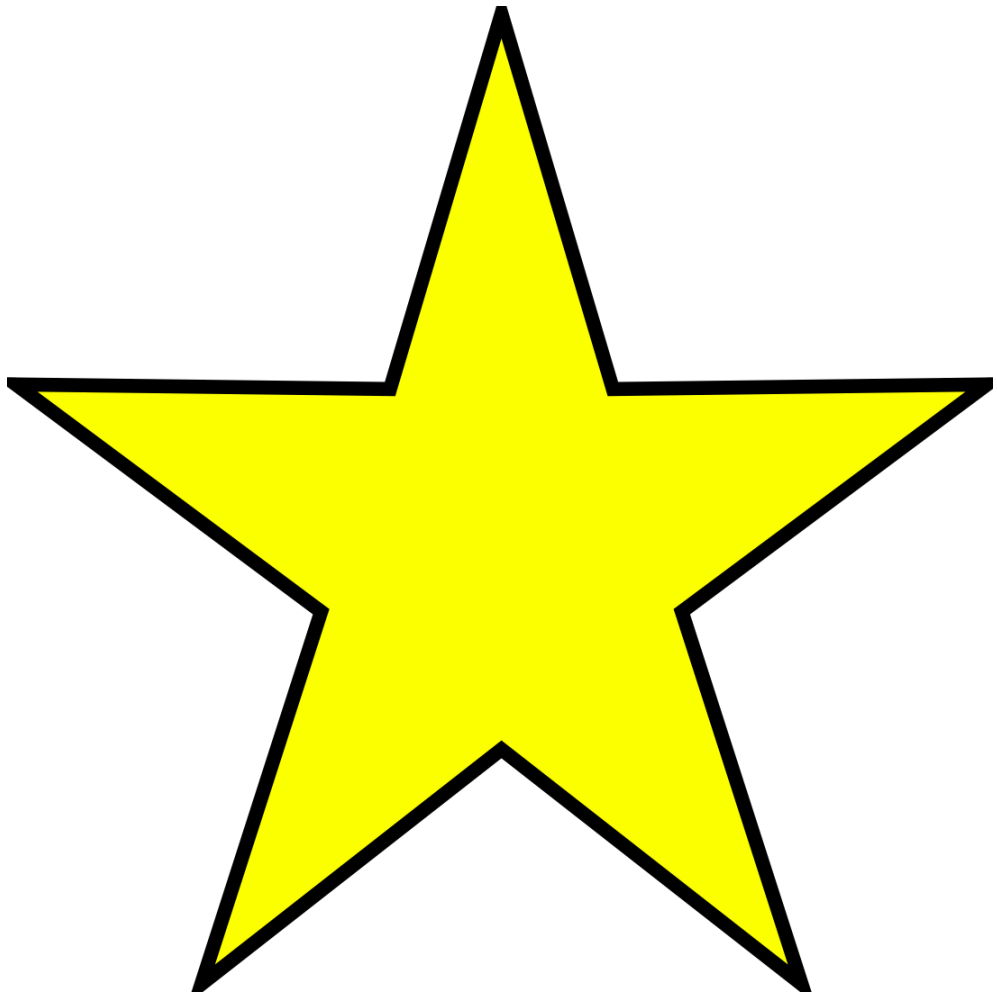
Creating denormalized data should be idempotent



There are lots of ways to speed up an application

Pick the easy ones first

Make models judiciously



Over 3 million badges awarded in August

 ~1.5 badges per second