Active Record Makeover

Marty Haught
@mghaught
http://martyhaught.com
HAUGHT CODEWORKS

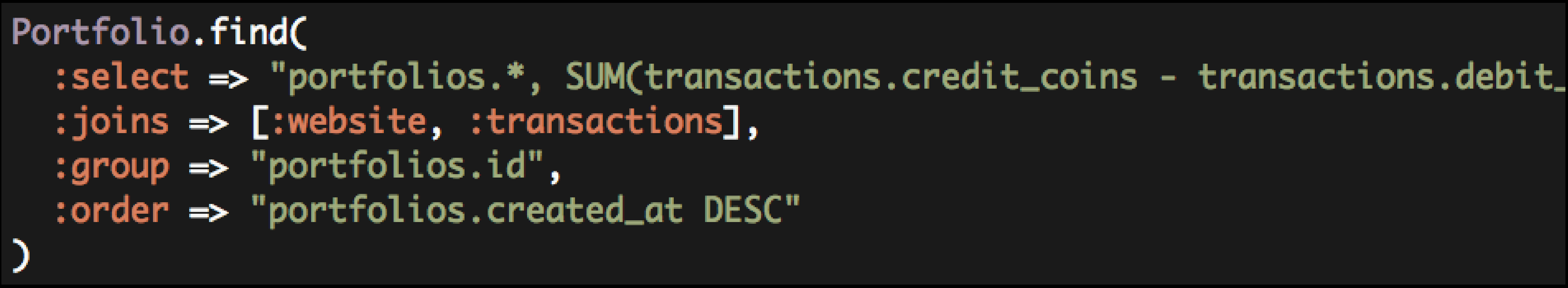
Boulder Ruby

mountain.rb 2010

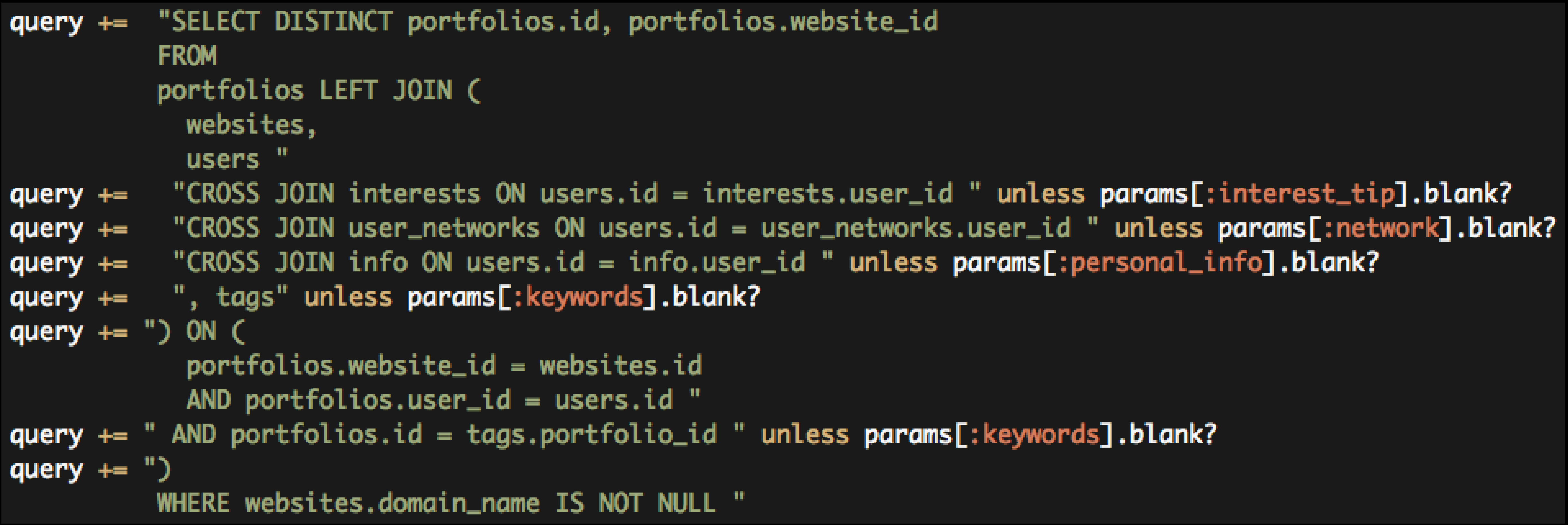
HAUGHT CODEWORKS

ActiveRecord feels like

Growing tired of this



```
Portfolio.find(
  :select => "portfolios.*, SUM(transactions.credit_coins - transactions.debit_
  :joins => [:website, :transactions],
  :group => "portfolios.id",
  :order => "portfolios.created_at DESC"
)
```



```
query +=  "SELECT DISTINCT portfolios.id, portfolios.website_id
          FROM
          portfolios LEFT JOIN (
            websites,
            users "
query +=   "CROSS JOIN interests ON users.id = interests.user_id " unless params[:interest_tip].blank?
query +=   "CROSS JOIN user_networks ON users.id = user_networks.user_id " unless params[:network].blank?
query +=   "CROSS JOIN info ON users.id = info.user_id " unless params[:personal_info].blank?
query +=   ", tags" unless params[:keywords].blank?
query += ") ON (
            portfolios.website_id = websites.id
            AND portfolios.user_id = users.id "
query += " AND portfolios.id = tags.portfolio_id " unless params[:keywords].blank?
query += ")
          WHERE websites.domain_name IS NOT NULL "
```

Wished I was driving...

...but I can't

- Existing system
- Project already committed

or doesn’t make sense

- Not the best fit for project
- No significant gain
- rdbms solves the problem

So where's the love?

- Where's the new hotness
- Envious of the nosql gems
- Want a better way to model data

Rails 3’s ActiveRecord

Unpimp your auto

Engine’s been rebuilt

- ActiveRelation
- arel gem - relational algebra
- Consistency under the hood

Easier to model with

- Simpler api
- Fully chainable
- Lazy loaded

find all

```ruby
Score.all(:conditions =>
  ["player_id = ? AND game_mode = ?", player, "run"])
```

Rails 2x

Rails 3

```ruby
Score.where(:player_id => player, :game_mode => "run")
```

first, last, count

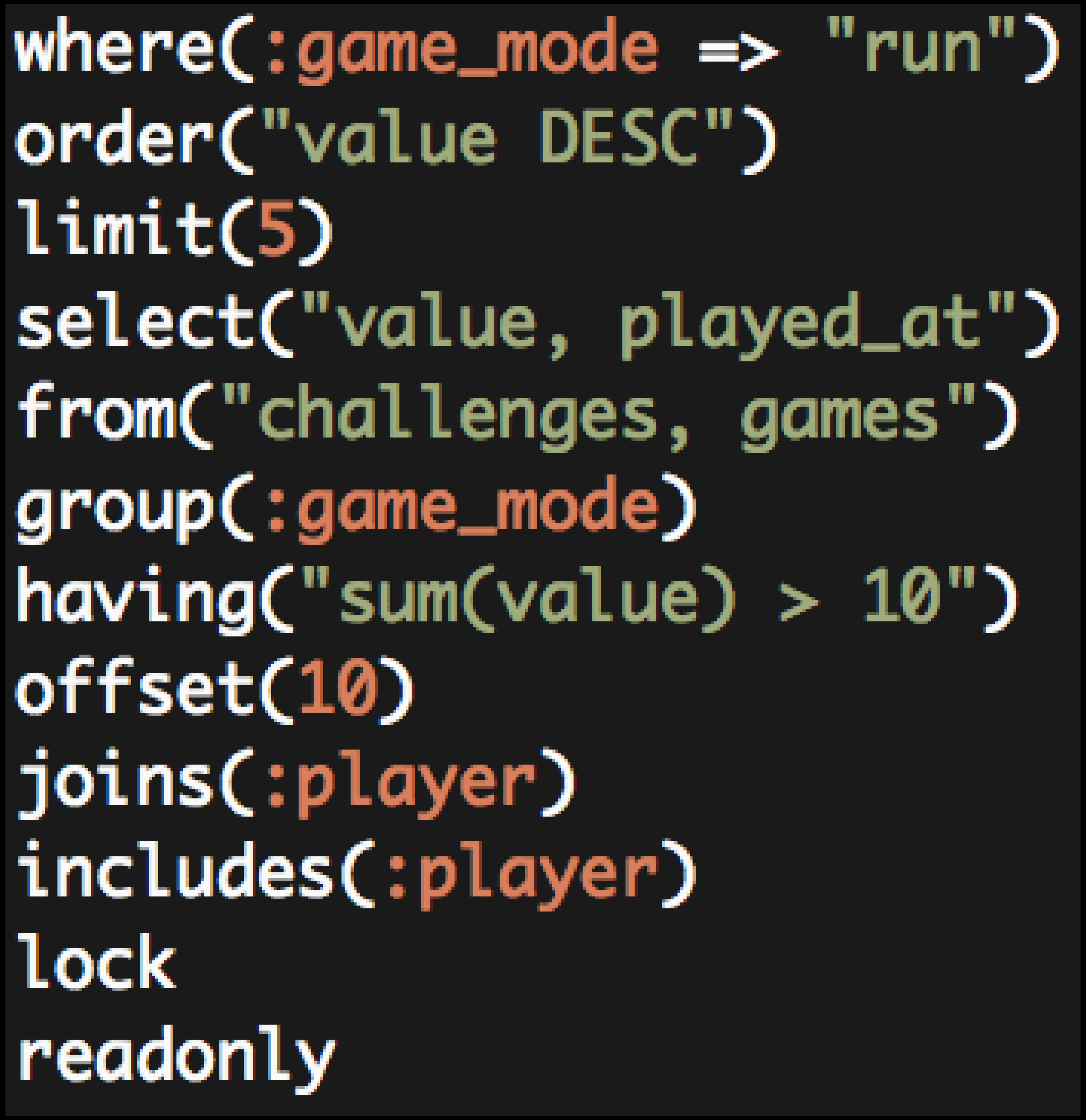
```
Score.first(:order => "value DESC")
Score.last(:order => "value DESC")
Score.count(:conditions => ["played_at > ?", 1.days.ago])
```

Rails 2x

Rails 3

```
Score.order("value DESC").first
Score.order("value DESC").last
Score.where("played_at > ?", 1.days.ago).count
```

Finder methods



```
where(:game_mode => "run")
order("value DESC")
limit(5)
select("value, played_at")
from("challenges, games")
group(:game_mode)
having("sum(value) > 10")
offset(10)
joins(:player)
includes(:player)
lock
readonly
```

Chainable

```ruby
Score.find(:all, :conditions => {:game_mode => "super_bops"},
  :order => "value DESC", :limit => 5, :include => :player)
```

Rails 2x

Rails 3

```ruby
# all inline
Score.where(:game_mode => "super_bops").
  order("value DESC").limit(5).includes(:player)

# applied separate
sb_scores = Score.where(:game_mode => "super_bops").order("value DESC")
todays_sb_scores = sb_scores.where("played_at >= ?", 1.day.ago)
top_5_today = todays_sb_scores.limit(5)
```

What about scopes?

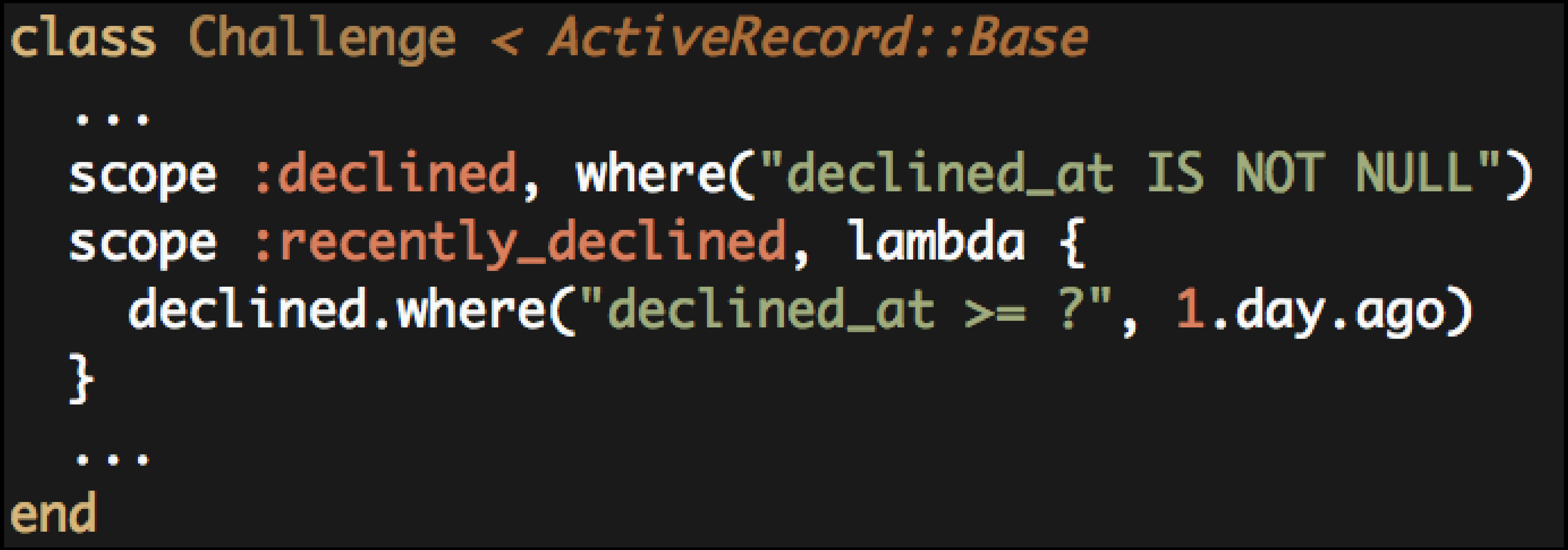
```ruby
class Score < ActiveRecord::Base
  default_scope :order => "value DESC"
  named_scope :top_runs, :conditions => {:game_mode => "run"},
    :limit => 10
end
```

Rails 2x

Rails 3

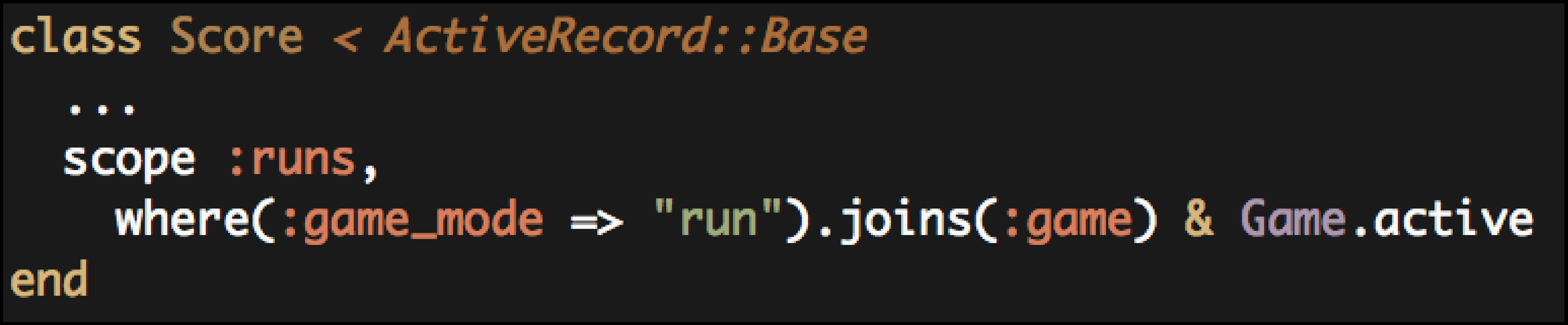
```ruby
class Score < ActiveRecord::Base
  default_scope order("value DESC")
  scope :top_runs, where(:game_mode => "run").limit(10)
end
```

Reusing scopes



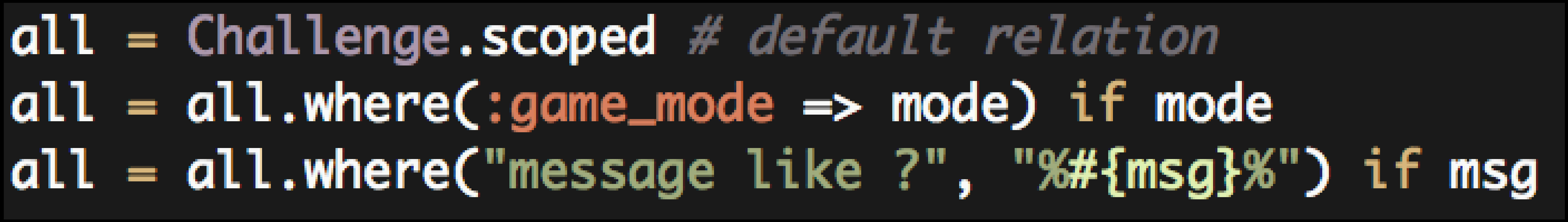
```ruby
class Challenge < ActiveRecord::Base
  ...
  scope :declined, where("declined_at IS NOT NULL")
  scope :recently_declined, lambda {
    declined.where("declined_at >= ?", 1.day.ago)
  }
  ...
end
```

Merging scopes



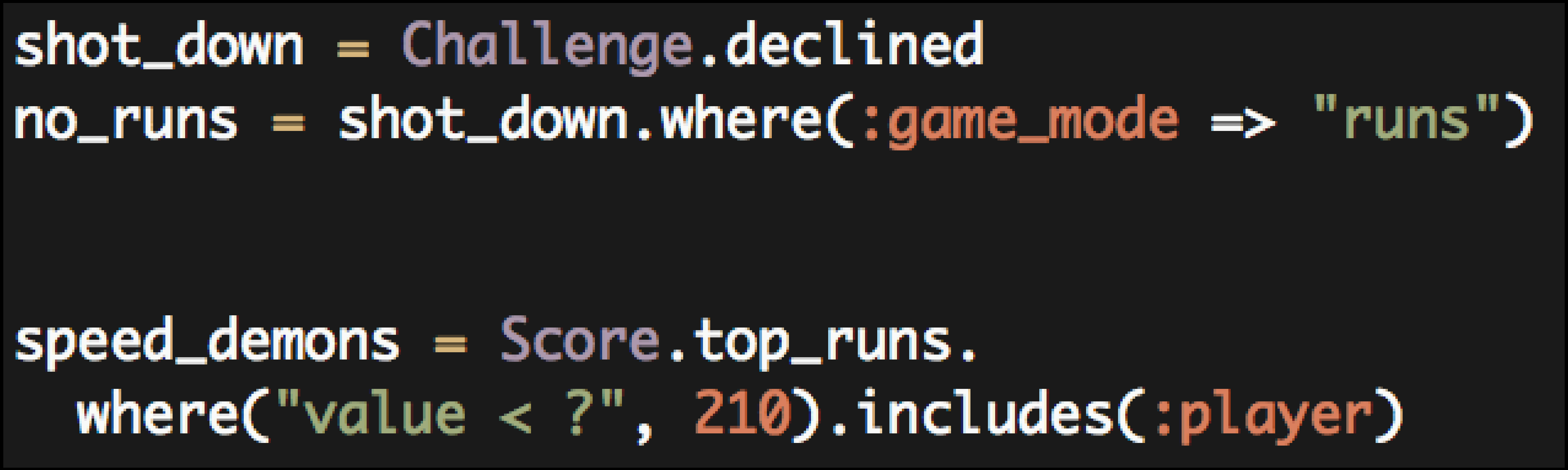
```
class Score < ActiveRecord::Base
  ...
  scope :runs,
    where(:game_mode => "run").joins(:game) & Game.active
end
```

Using scoped



```
all = Challenge.scoped # default relation
all = all.where(:game_mode => mode) if mode
all = all.where("message like ?", "%#{msg}%") if msg
```

Mix and match



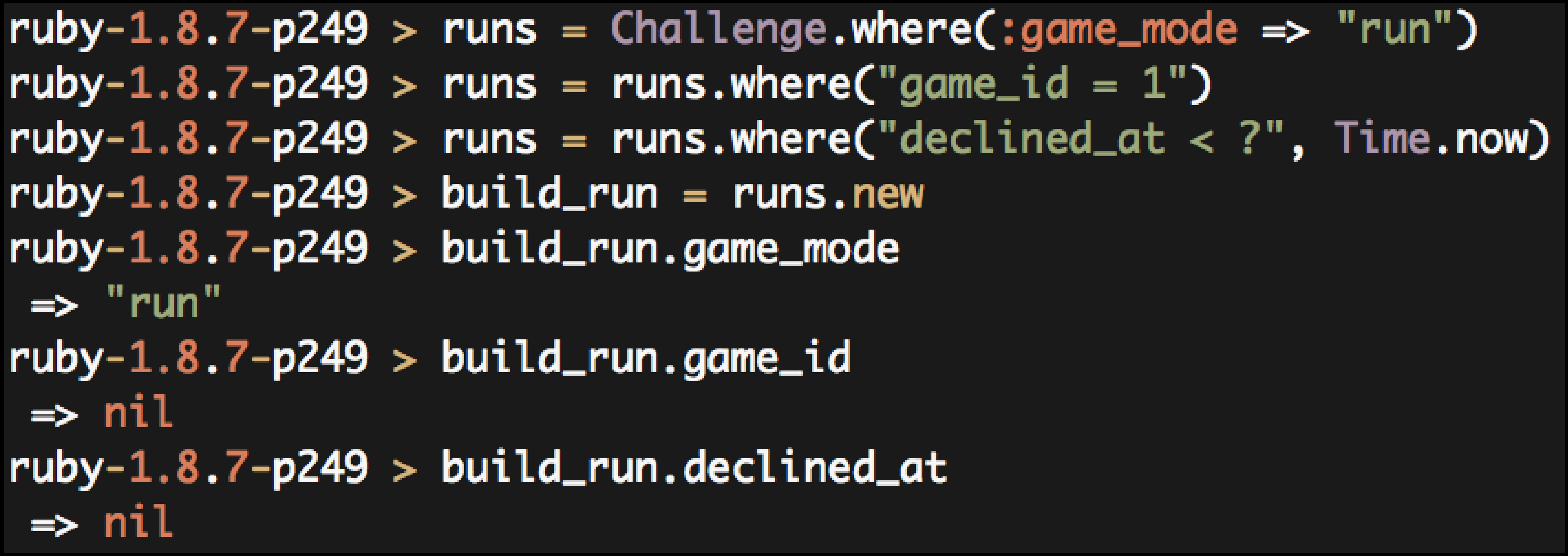
```
shot_down = Challenge.declined
no_runs = shot_down.where(:game_mode => "runs")


speed_demons = Score.top_runs.
  where("value < ?", 210).includes(:player)
```

Behaves like the model

new
create
create!
find
destroy
destroy_all

delete
delete_all
update
update_all
exists?

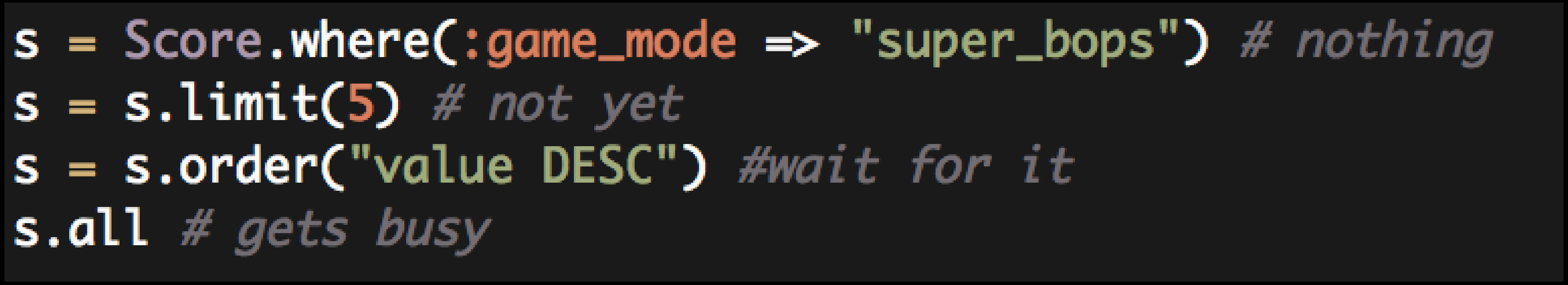
Building with where values



```
ruby-1.8.7-p249 > runs = Challenge.where(:game_mode => "run")
ruby-1.8.7-p249 > runs = runs.where("game_id = 1")
ruby-1.8.7-p249 > runs = runs.where("declined_at < ?", Time.now)
ruby-1.8.7-p249 > build_run = runs.new
ruby-1.8.7-p249 > build_run.game_mode
 => "run"
ruby-1.8.7-p249 > build_run.game_id
 => nil
ruby-1.8.7-p249 > build_run.declined_at
 => nil
```

Lazy loaded

- won't interact with the db until accessed
- such as each, map, etc.
- force load - all, first, last

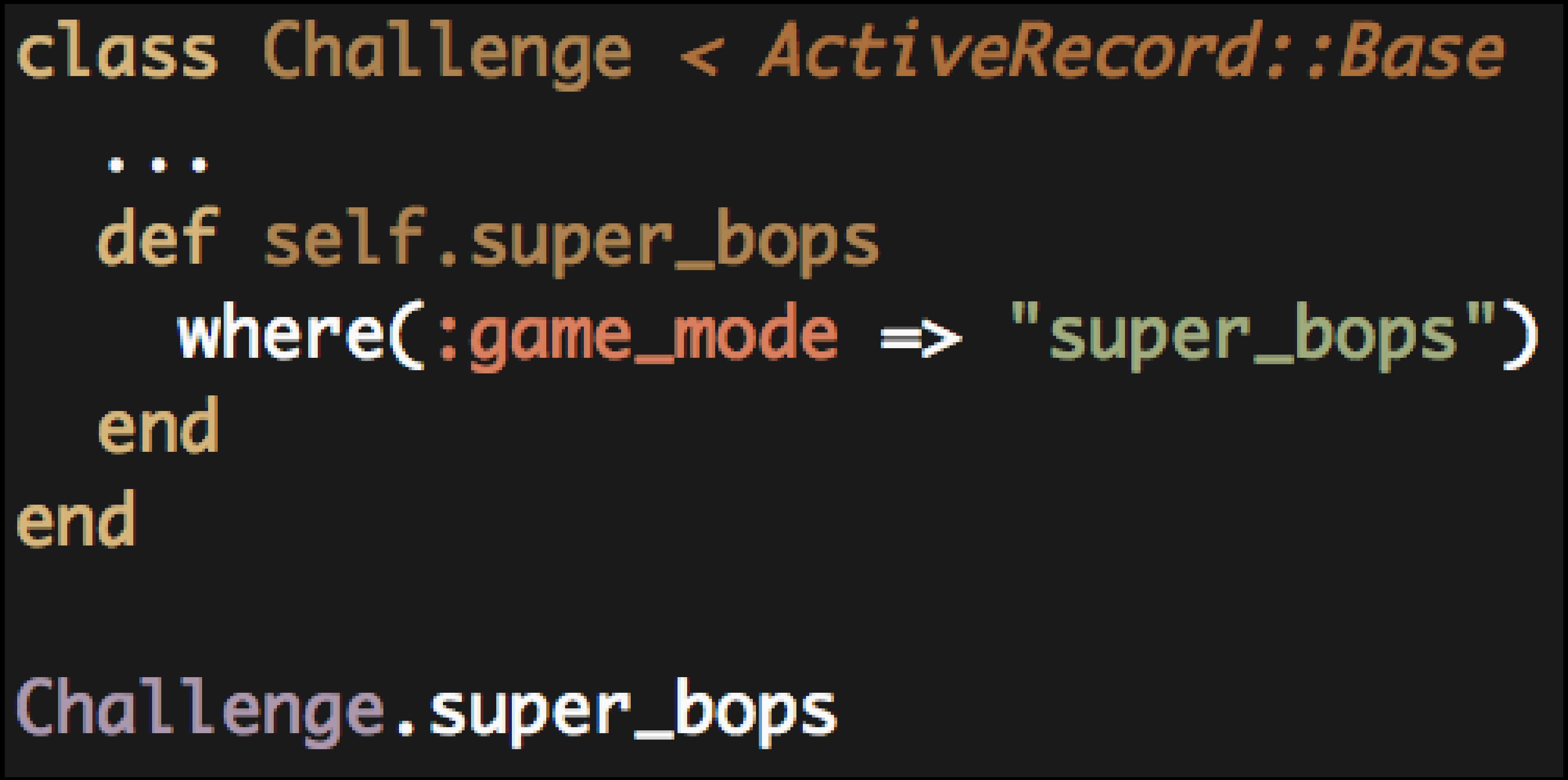


```ruby
s = Score.where(:game_mode => "super_bops") # nothing
s = s.limit(5) # not yet
s = s.order("value DESC") #wait for it
s.all # gets busy
```

Plays nice with caching

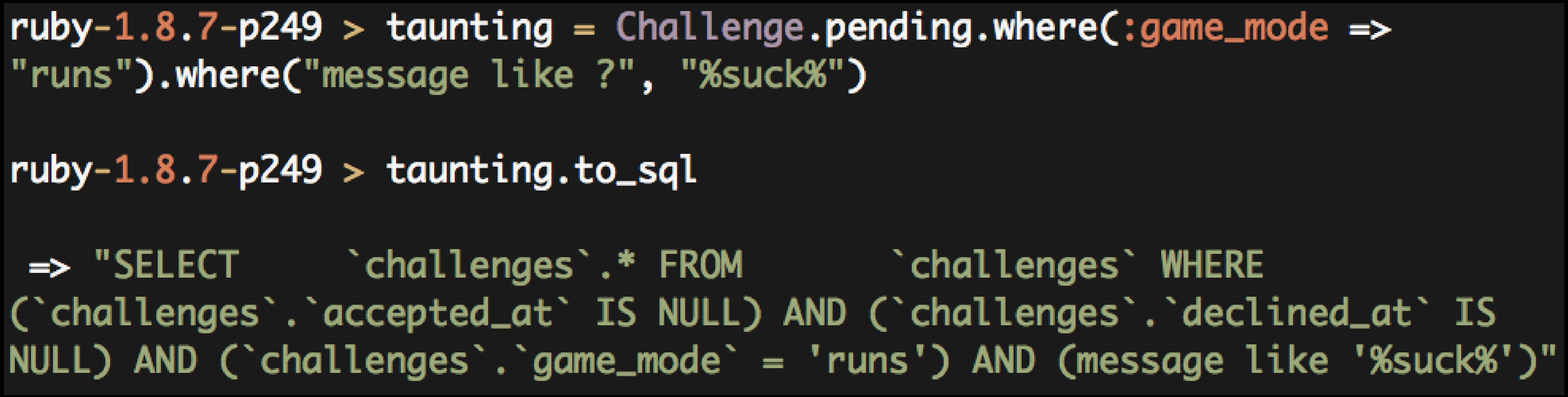
```
# in controller
  def index
    @todays_scores = Score.limit(10).where("played_at >= ?", 1.day.ago)
  end

# in view
  <% cache('today_scores') do %>
    <% @todays_scores.each do |score| %>
      ...
    <% end %>
  <% end %>
```

Modeling with class



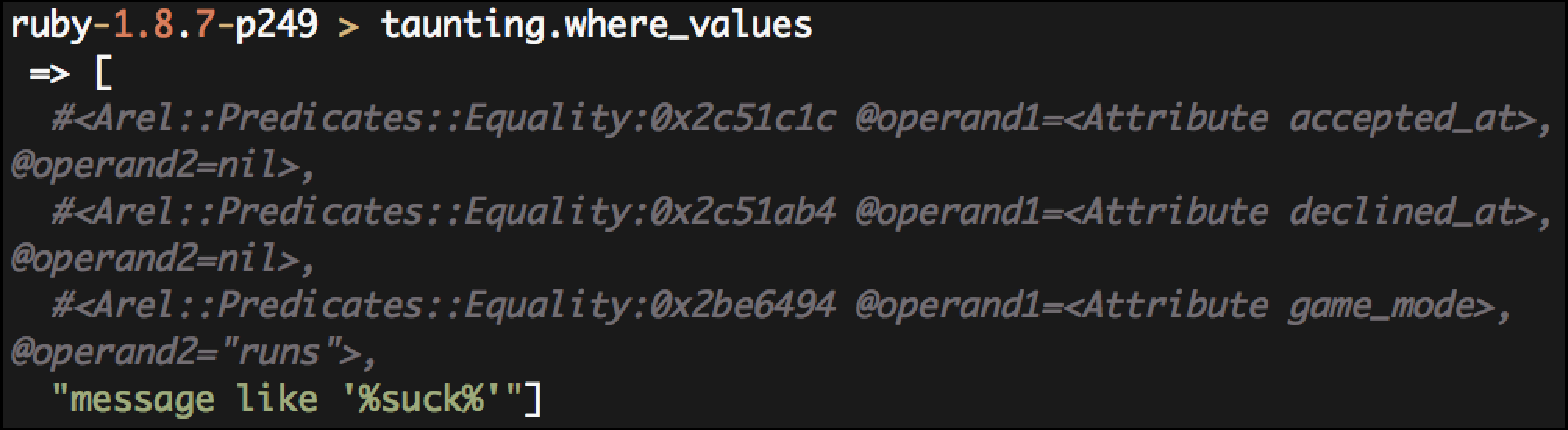
```
class Challenge < ActiveRecord::Base
  ...
  def self.super_bops
    where(:game_mode => "super_bops")
  end
end

Challenge.super_bops
```

to_sql



```
ruby-1.8.7-p249 > taunting = Challenge.pending.where(:game_mode =>
"runs").where("message like ?", "%suck%")

ruby-1.8.7-p249 > taunting.to_sql

 => "SELECT     `challenges`.* FROM       `challenges` WHERE
(`challenges`.`accepted_at` IS NULL) AND (`challenges`.`declined_at` IS
NULL) AND (`challenges`.`game_mode` = 'runs') AND (message like '%suck%')"
```

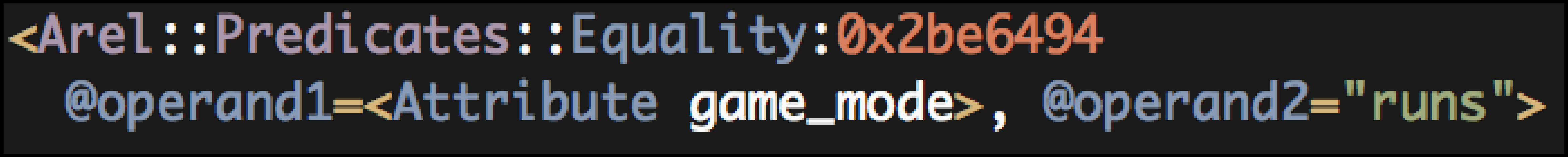
where_values



```
ruby-1.8.7-p249 > taunting.where_values
 => [
  #<Arel::Predicates::Equality:0x2c51c1c @operand1=<Attribute accepted_at>,
@operand2=nil>,
  #<Arel::Predicates::Equality:0x2c51ab4 @operand1=<Attribute declined_at>,
@operand2=nil>,
  #<Arel::Predicates::Equality:0x2be6494 @operand1=<Attribute game_mode>,
@operand2="runs">,
  "message like '%suck%'"]
```

Extend the possibilities

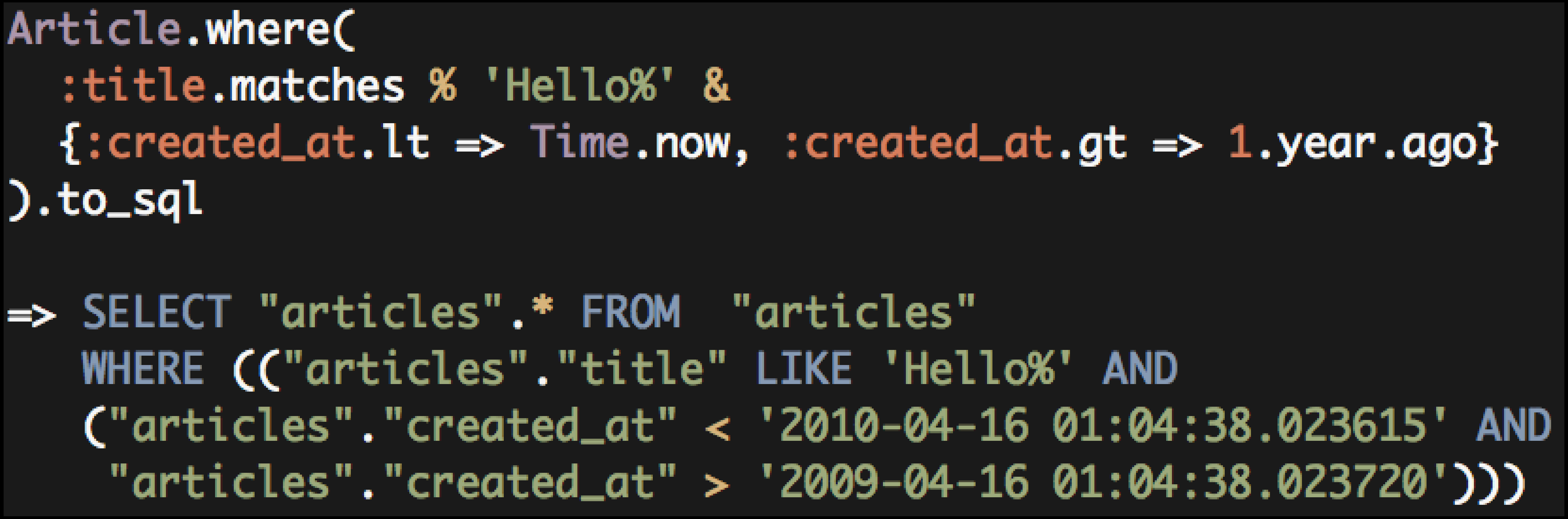
- ActiveRecord::QueryMethods opens a door
- invokes to_sql on where_values unless a String
- pass in your own objects that create sql



```
<Arel::Predicates::Equality:0x2be6494
  @operand1=<Attribute game_mode>, @operand2="runs">
```

MetaSearch & MetaWhere

Ernie Miller



```
Article.where(
  :title.matches % 'Hello%' &
  {:created_at.lt => Time.now, :created_at.gt => 1.year.ago}
).to_sql

=> SELECT "articles".* FROM  "articles"
   WHERE (("articles"."title" LIKE 'Hello%' AND
   ("articles"."created_at" < '2010-04-16 01:04:38.023615' AND
    "articles"."created_at" > '2009-04-16 01:04:38.023720')))
```

http://github.com/ernie/meta_where

http://github.com/ernie/meta_search

Got some hotness

- new engine
- concise and readable
- simpler to model
- flexible
- extendable

Go have fun, get dirty

Thank you

Marty Haught
@mghaught
mghaught@gmail.com
http://martyhaught.com

HAUGHT CODEWORKS

Image Credits

wagon - http://www.flickr.com/photos/halfy/3410561333
vw car - http://www.youtube.com/watch?v=qgEvy60bZYI
white race car - http://www.flickr.com/photos/33031095@N03/4538638400/
yellow sports car - http://www.flickr.com/photos/retinafunk/57699167/
red sports car - http://www.flickr.com/photos/freefoto/3844246045/
gray sports car - http://www.flickr.com/photos/doramosnoopy/2479741916/
blue sports car - http://www.flickr.com/photos/chorwedel/4398697300/
racing cars - http://www.flickr.com/photos/oizumi/2582189751/