progress engine
we will develop it

#tceh

2016

Врубиться в Ruby

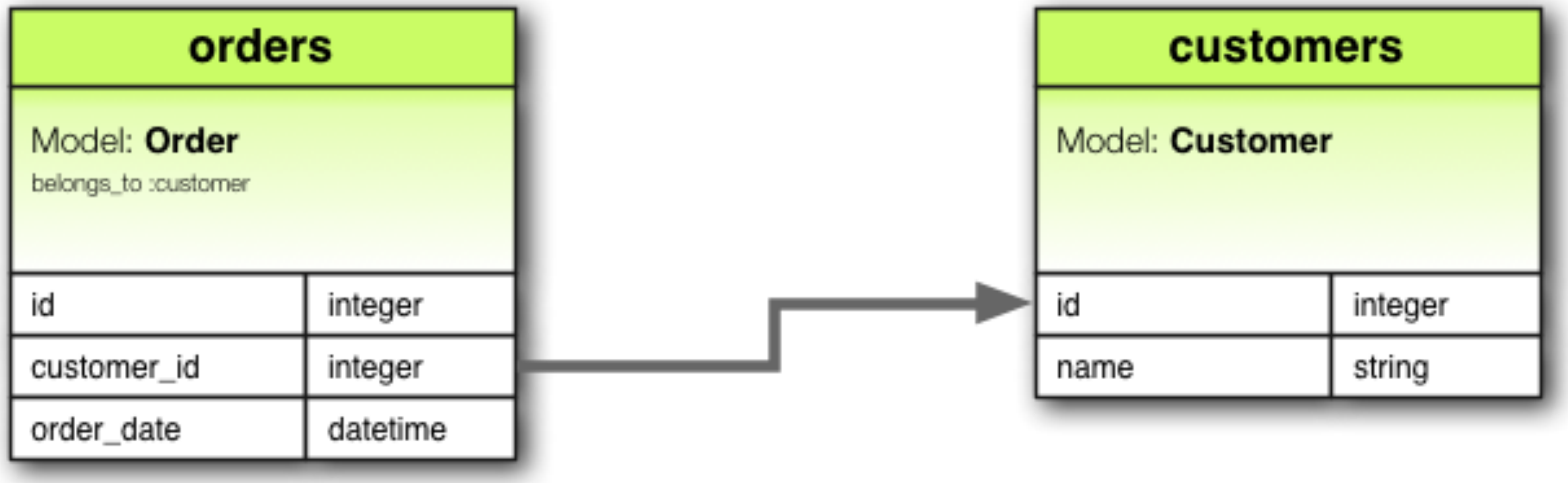
Лекция 7

Лекция 7

- Ассоциации ActiveRecord
- Язык запросов ActiveRecord

Ассоциации AR

belongs_to

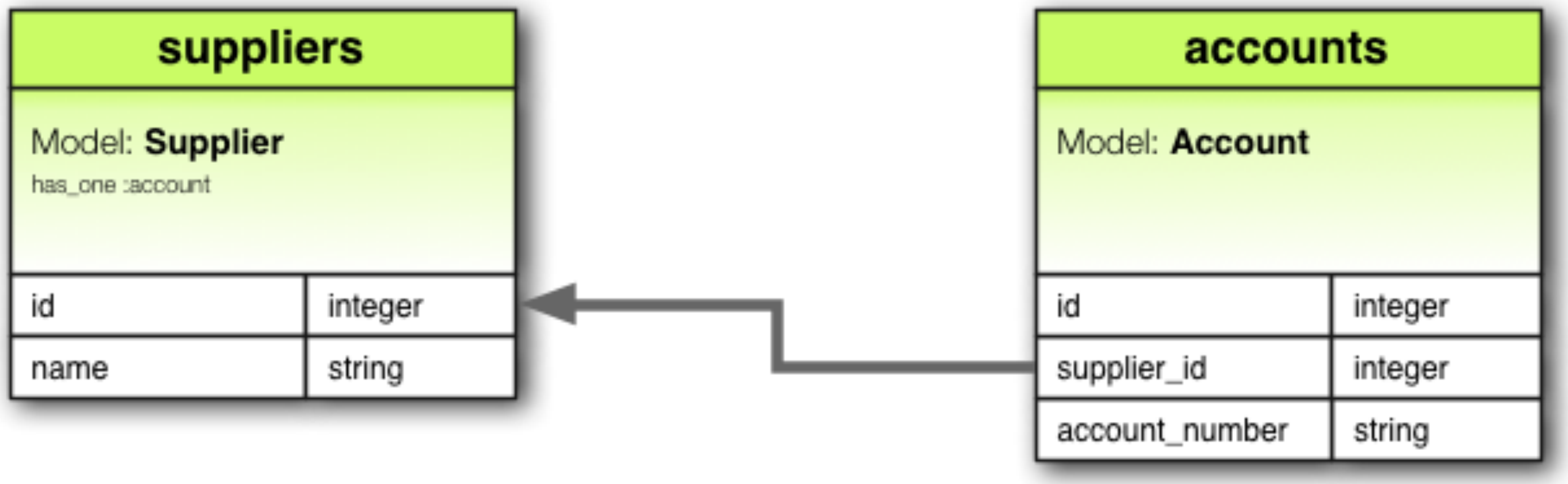


```
class Order < ActiveRecord::Base
  belongs_to :customer
end
```

Опции belongs_to

```
class Order < ActiveRecord::Base
  belongs_to :customer, dependent: :destroy,
    counter_cache: true, class_name: “User”
end
```

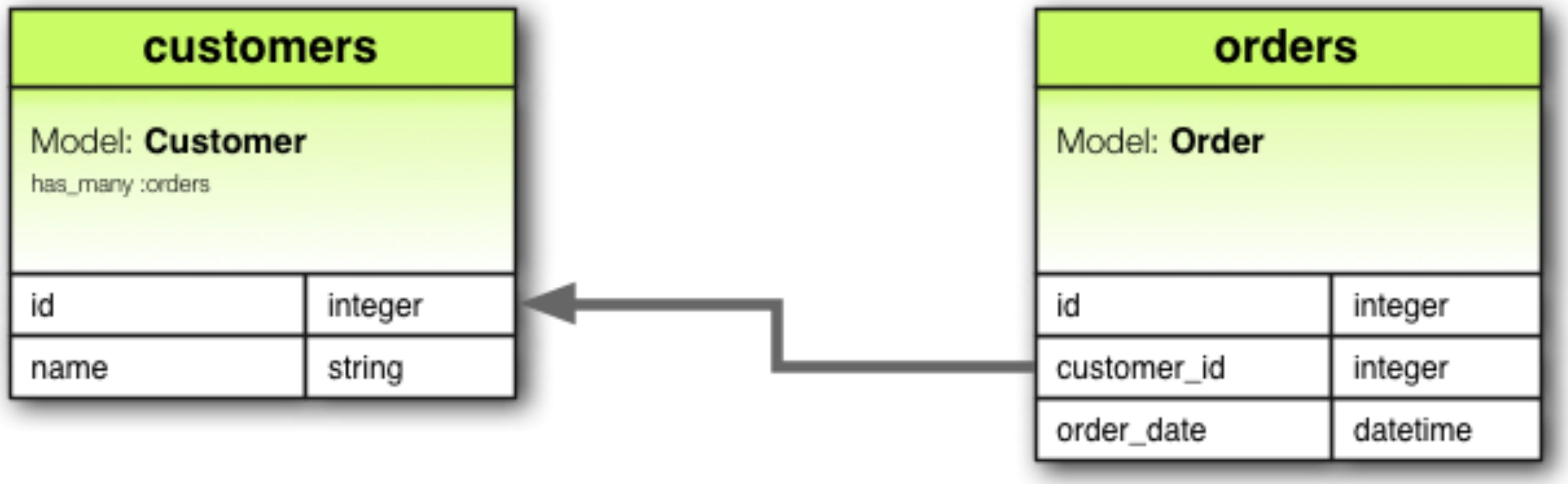
has_one



```
class Supplier < ActiveRecord::Base
  has_one :account
end
```

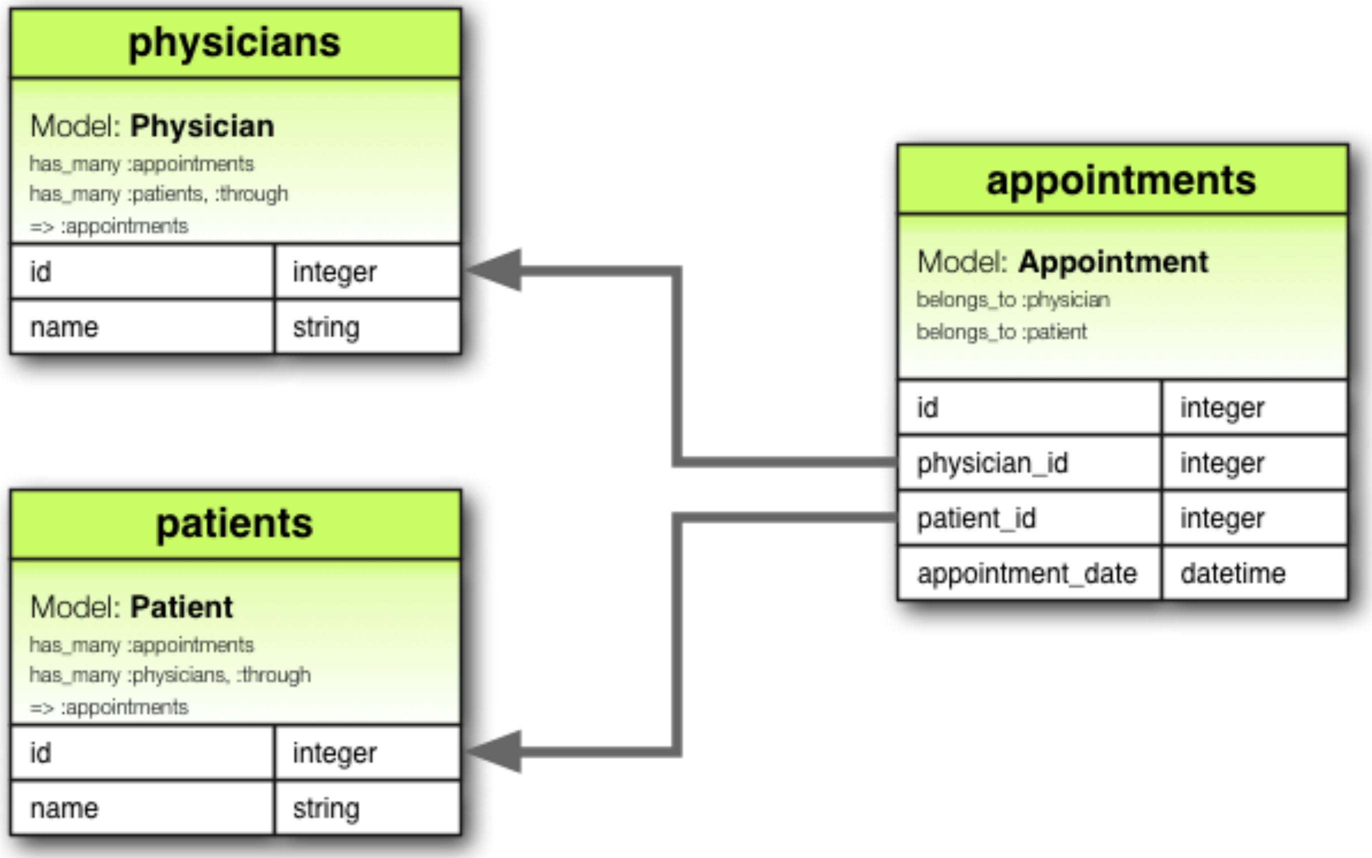
В чем различия между has_one и belongs_to?

has_many



```
class Customer < ActiveRecord::Base
  has_many :orders
end
```

has_many :through

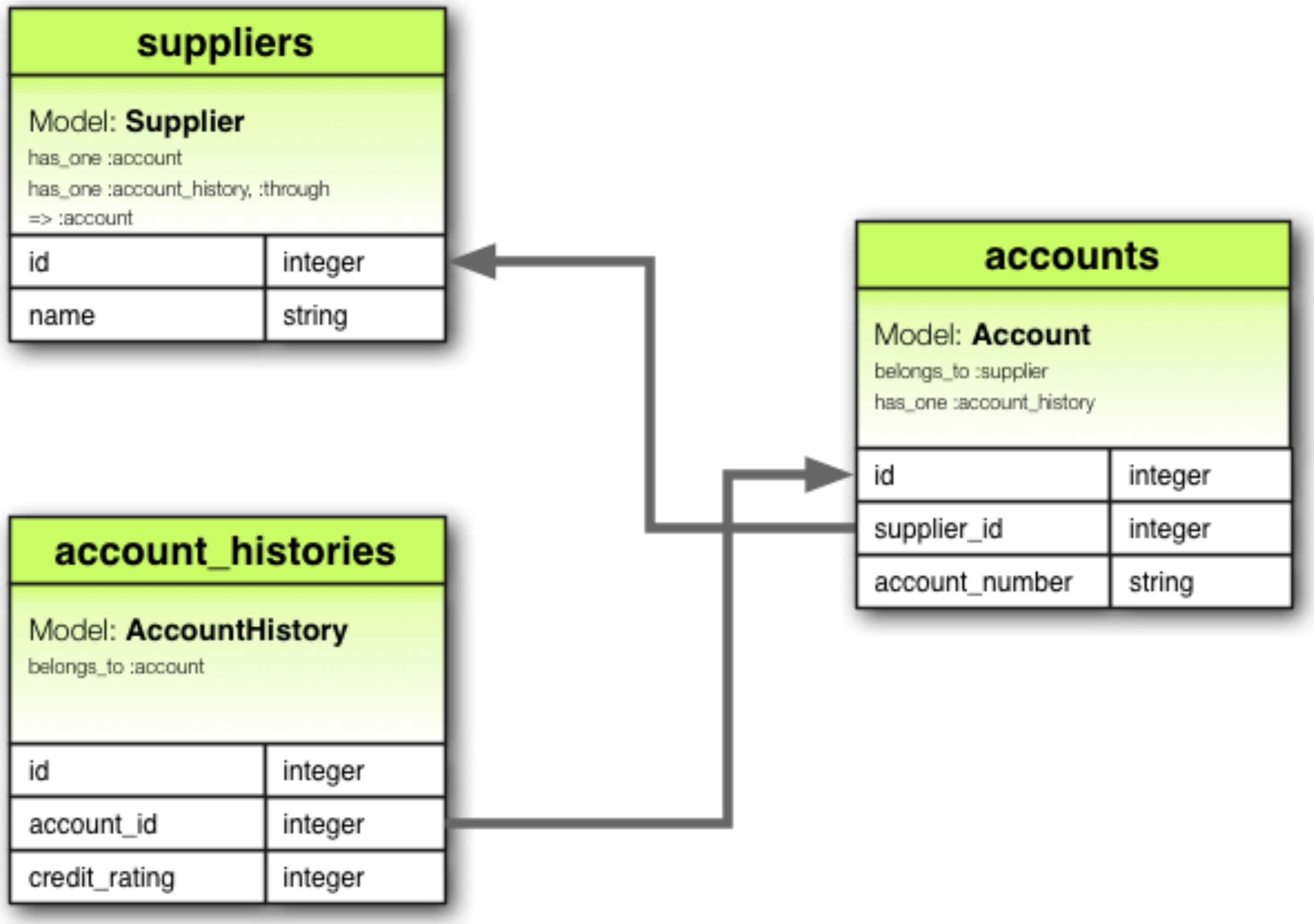


```
class Physician < ActiveRecord::Base
  has_many :appointments
  has_many :patients, :through => :appointments
end

class Appointment < ActiveRecord::Base
  belongs_to :physician
  belongs_to :patient
end

class Patient < ActiveRecord::Base
  has_many :appointments
  has_many :physicians, :through => :appointments
end
```

has_one :through

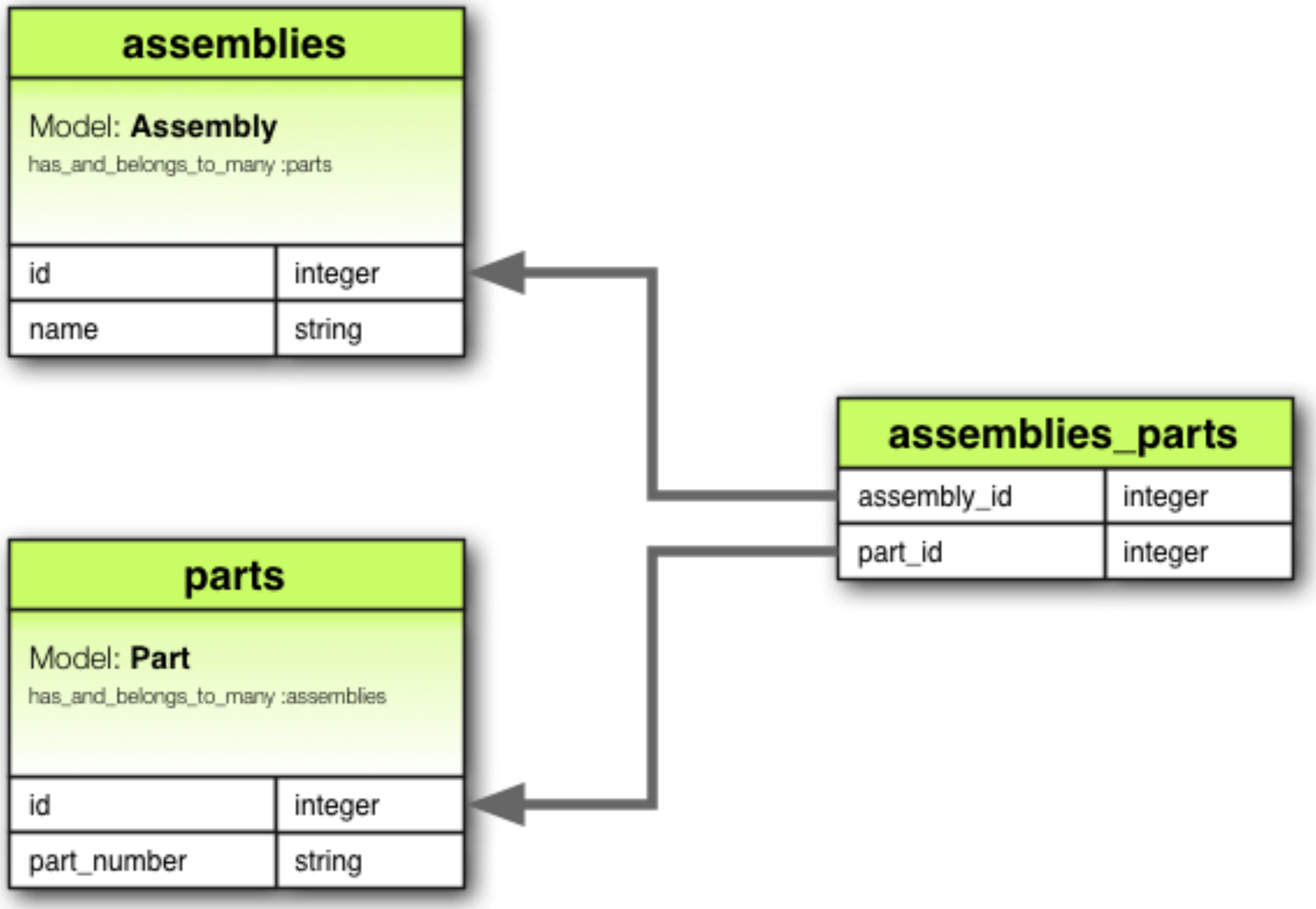


```
class Supplier < ActiveRecord::Base
  has_one :account
  has_one :account_history, :through => :account
end

class Account < ActiveRecord::Base
  belongs_to :supplier
  has_one :account_history
end

class AccountHistory < ActiveRecord::Base
  belongs_to :account
end
```

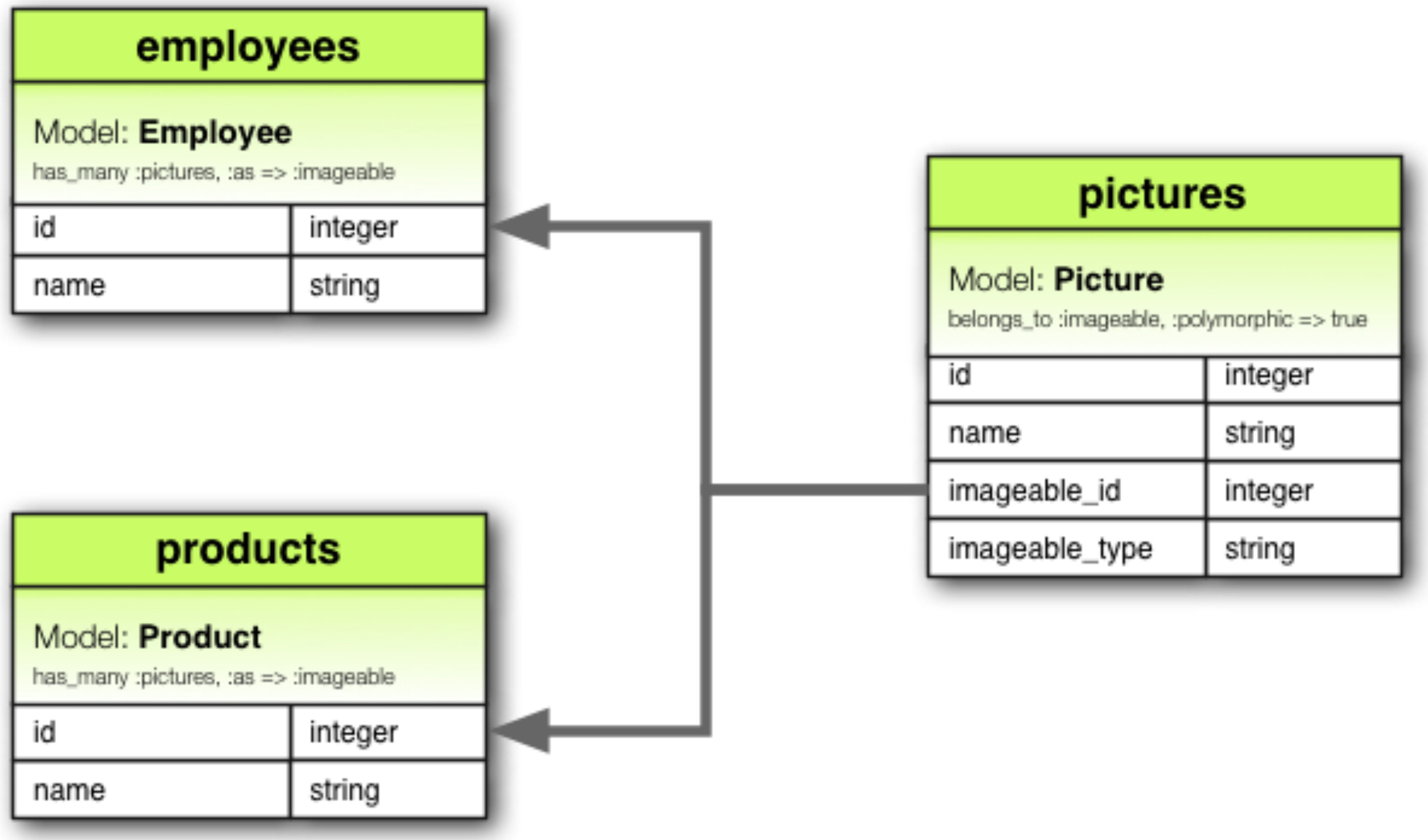
has_and_belongs_to_many



```
class Assembly < ActiveRecord::Base
  has_and_belongs_to_many :parts
end

class Part < ActiveRecord::Base
  has_and_belongs_to_many :assemblies
end
```

Полиморфные ассоциации



```
class Picture < ActiveRecord::Base
  belongs_to :imageable, :polymorphic => true
end

class Employee < ActiveRecord::Base
  has_many :pictures, :as => :imageable
end

class Product < ActiveRecord::Base
  has_many :pictures, :as => :imageable
end
```

Замыкание на себя

```
class Employee < ApplicationRecord
  has_many :subordinates, class_name: "Employee",
                          foreign_key: "manager_id"

  belongs_to :manager, class_name: "Employee"
end
```

Язык запросов AR

find

```
@post = Post.find(1)

SELECT * from posts WHERE (posts.id == 1) LIMIT 1
```

take

```
@post = Post.take(2)

SELECT * from posts LIMIT 2
```

first

```
@post = Post.first(5)

SELECT * from posts ORDER BY posts.id ASC LIMIT 5
```

last

```
@post = Post.last(5)

SELECT * from posts ORDER BY posts.id DESC LIMIT 5
```

find_by

```
@post = Post.find_by user_id: 1

SELECT * from posts where user_id == 1
```

where

```
@post = Post.where(user_id: 1)

SELECT * from posts where user_id == 1
```

where not

```
@post = Post.where.not(user_id: 1)

SELECT * from posts where user_id != 1
```

AR scopes

scope

```
class Article < ApplicationRecord
    scope :published, -> { where(published: true) }

    scope :published_and_commented, ->
{ published.where("comments_count > 0") }

    scope :created_before, ->
          (time) { where("created_at < ?", time) }
end
```

default_scope

```
class Article < ApplicationRecord
    default_scope { where("removed_at IS NULL") }
end
```

unscoped

```
Post.where(published: false).unscoped.all
```

```
SELECT * FROM posts
```

AR hacks

pluck

Post.all.**pluck**(:title)

синоним

Post.all.select(:title).map(&:title)

Не пересекающиеся коллекции

```
Article < ActiveRecord::Base
  scope :unchecked, where(:checked => false)

 scope :unchecked2, lambda { |checked| where(["id not in (?)",
checked.pluck(:id)]) }

end
```

```
@unchecked_articles = Article.unchecked
Article.unchecked2(@unchecked_articles)
```

first_or_create и first_or_initialize

```
Art.where(name: "Black square").first_or_create
```

```
Art.where(name: "Black square").first_or_create do |art|
  art.author = "Malevich"
end
```

```
@art = Art.where(name: "Black square").first_or_initialize
```

find_each

```
Article.where(published: true).each do |article|
  #do something
end
```

```
Article.where(published: true).find_each do |article|
  #do something
end
```

to_sql и explain

Post.all.**to_sql**

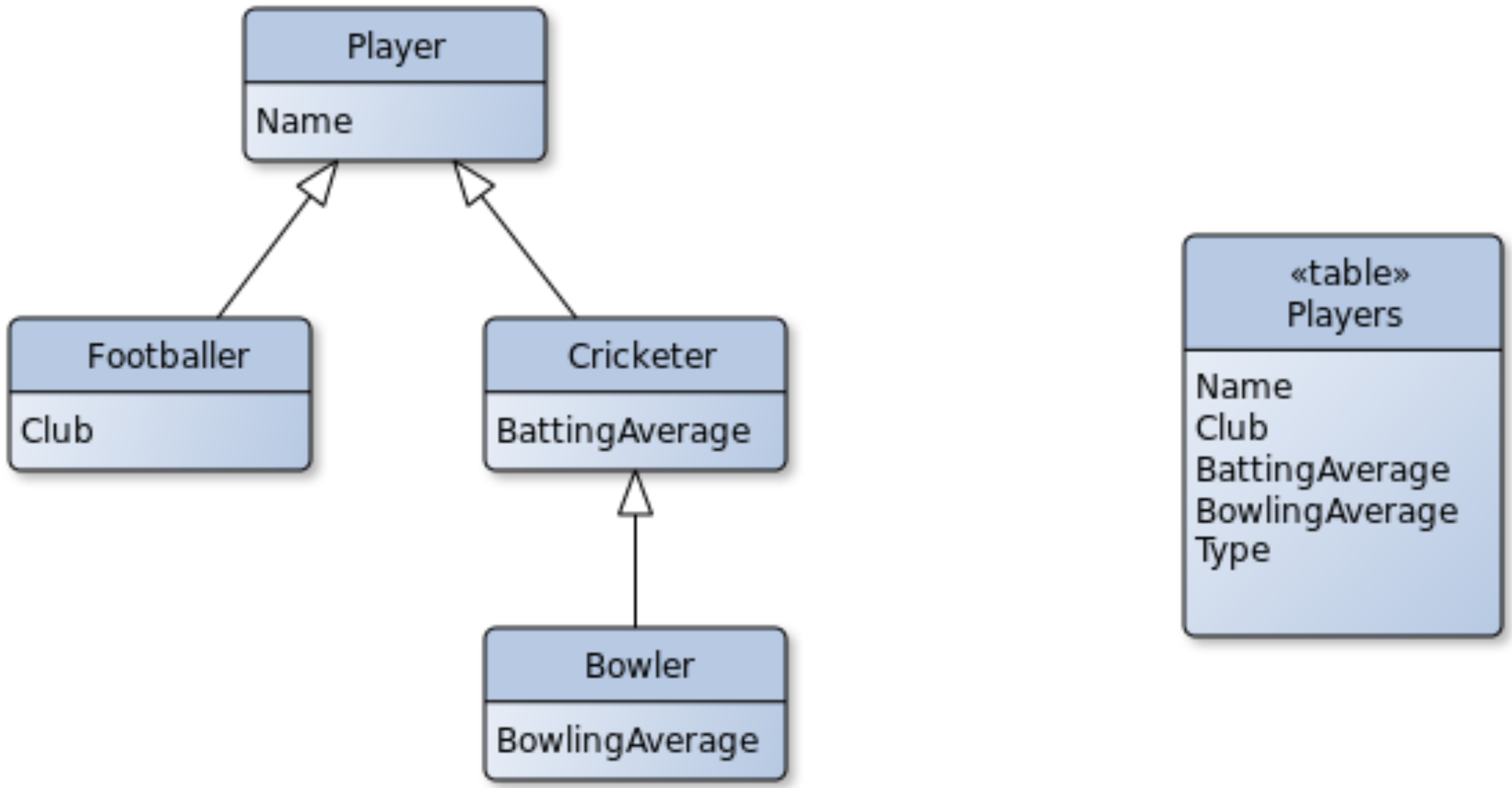
Post.all.**explain**

scoping

```
Article.where(published: true).scoping do
  Article.first
end
```

```
SELECT * FROM articles WHERE published = true LIMIT 1
```

STI


Player
Name
Footballer
Club
Cricketer
BattingAverage
Bowler
BowlingAverage
«table»
Players
Name
Club
BattingAverage
BowlingAverage
Type

```
class CreateSports < ActiveRecord::Migration
  def change
    create_table :sports do |t|
      t.string :type
      t.string :act_primary
      t.string :act_secondary
      # ... more column fields #
      t.timestamps
    end
  end
end
```

```
class Sport < ActiveRecord::Base
  # Methods, variables and constants
end
```

```
class ProBasketball < Sport
  # Methods, variables and constants
end


class ProFootball < Sport
  # Methods, variables and constants
end
```

Домашнее задание

1 - Прочитать
http://guides.rubyonrails.org/association_basics.html

2 - Прочитать
http://guides.rubyonrails.org/active_record_querying.html

3 - Прочитать
http://guides.rubyonrails.org/active_model_basics.html

4 - Прочитать
http://www.davidverhasselt.com/set-attributes-in-activerecord/

5 - Прочитать
https://samurails.com/tutorial/single-table-inheritance-with-rails-4-part-1/
https://samurails.com/tutorial/single-table-inheritance-with-rails-4-part-2/
https://samurails.com/tutorial/single-table-inheritance-with-rails-4-part-3/

Вопросы?

+7 (926) 889-16-32

alec@alec-c4.com

http://alec-c4.com/