Лекция 11
Планирование
Active Record 2

План

- Как спланировать проект?
- Блокировки
- Ассоциации
- Колбеки
- Миграции

С чего начать?

- Чего вам не хватает?
- Чего не хватает окружающим?
- Кто будет этим пользоваться?
- Как этим удобно пользоваться

Нарисуйте от руки интерфейс

- Окна, формы, кнопки

Какие потребуются данные?

- Спроектируйте таблицы
- Столбцы в таблицах
- Отношения между таблицами

API

- Выпишите API
- Желательно на Swagger

Обновление данных

- Web-приложение запускает несколько процессов, которые обращаются к общим данным
- Нужен механизм защиты от нарушения целостности
- Оптимистичная блокировка
- Пессимистичная блокировка

Оптимистичная блокировка

- Заводим поле целочисленное поле **lock_version**
- При каждом обновлении увеличиваем **lock_version** на 1
- Если объект с lock_version <= текущей пытается выполнить обновление, кидаем ошибку.

```
ActiveRecord::Base.lock_optimistically = false
```

```ruby
c1 = Client.find(1)
c2 = Client.find(1)

c1.first_name = "Michael"
c1.save

c2.name = "should fail"
c2.save # ActiveRecord::StaleObjectError
```

Настройки блокировок

```ruby
class Client < ApplicationRecord
  self.locking_column = :lock_client_column
end
```

```ruby
ActiveRecord::Base.lock_optimistically = false
```

Пессимистичные блокировки

```
Item.transaction do
  i = Item.lock.first
  i.name = 'Jones'
  i.save!
end
```

Поведение зависит от СУБД

```
Item.transaction do
  i = Item.lock("LOCK IN SHARE MODE").find(1)
  i.increment!(:views)
end
```

Блокировка инстансов

```ruby
item = Item.first
item.with_lock do
  item.increment!(:views)
end
```

Join

```
Author.joins("INNER JOIN posts
              ON posts.author_id = authors.id
              AND posts.published = 't'")
```

Ассоциации

```ruby
class Author < ApplicationRecord
end

class Book < ApplicationRecord
end
```

Без ассоциаций

```ruby
@book = Book.create(published_at: Time.now,
                    author_id: @author.id)


@books = Book.where(author_id: @author.id)
@books.each do |book|
  book.destroy
end
@author.destroy
```

Подсказываем

```ruby
class Author < ApplicationRecord
  has_many :books, dependent: :destroy
end

class Book < ApplicationRecord
  belongs_to :author
end
```

С ассоциациями

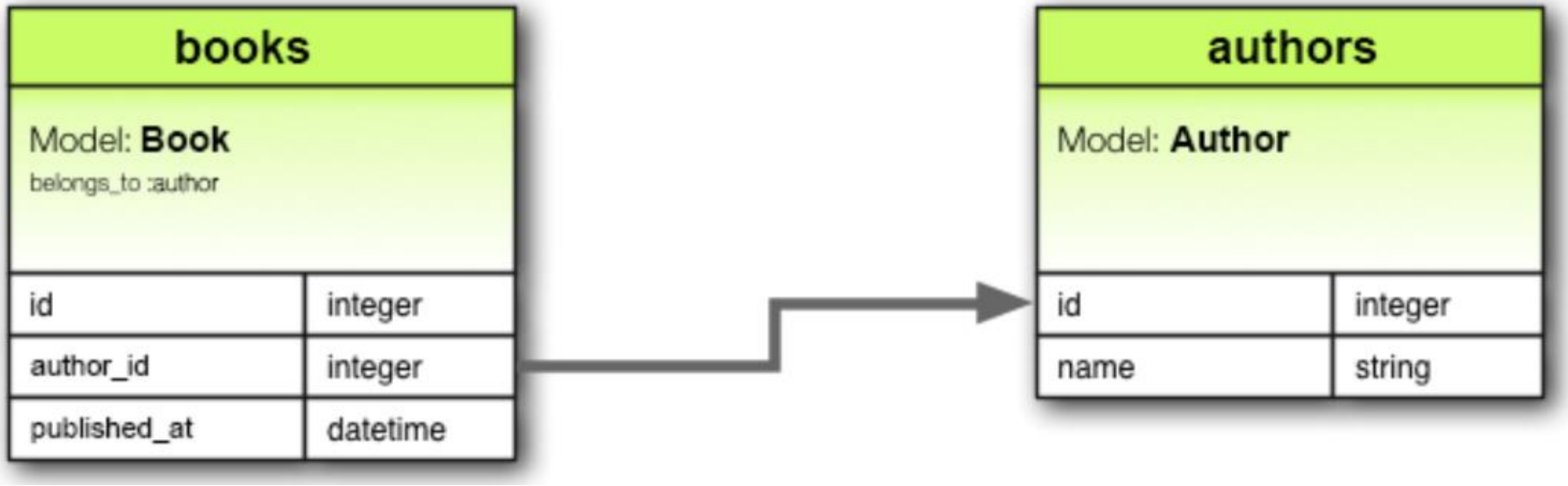
```
@book = @author.books
          .create(published_at: Time.now)

@author.destroy
```

Типы ассоциаций

```
belongs_to
has_one
has_many
has_many :through
has_one :through
has_and_belongs_to_many
```

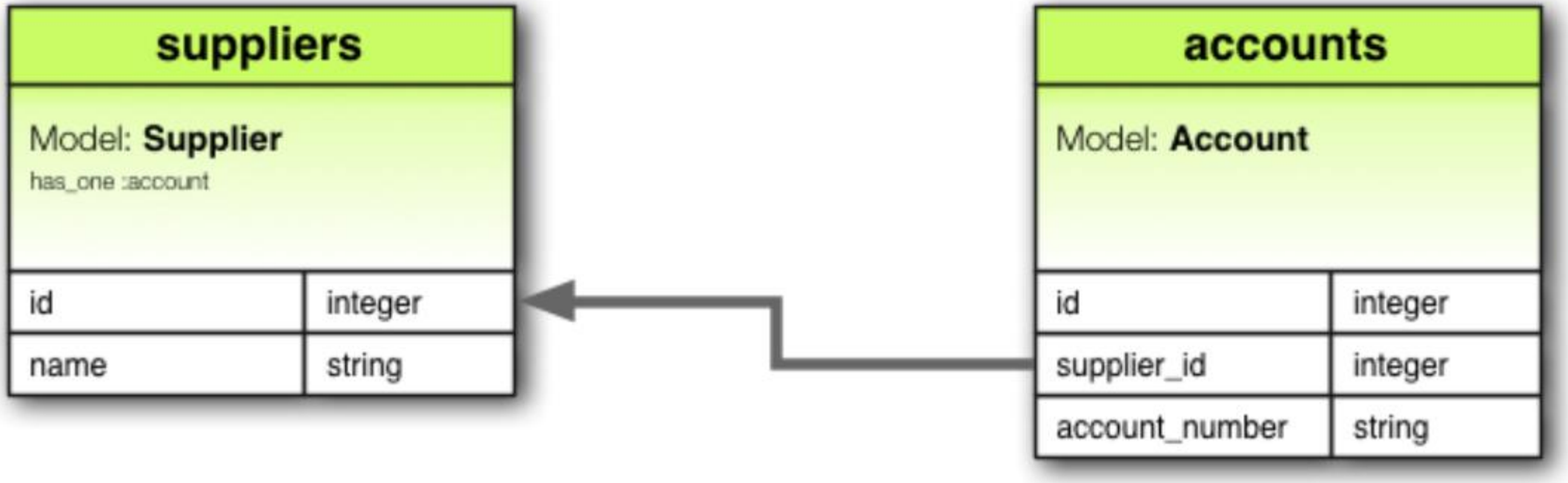
belongs_to

```ruby
class Book < ApplicationRecord
  belongs_to :author
end
```



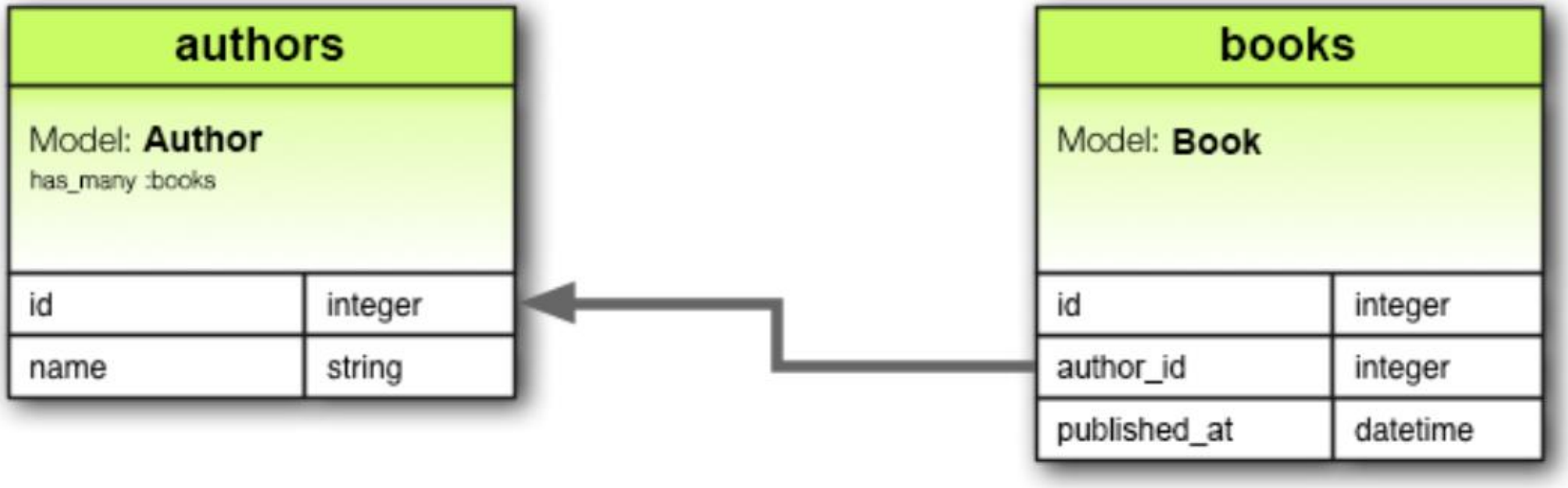
has_one

```
class Supplier < ApplicationRecord
  has_one :account
end
```



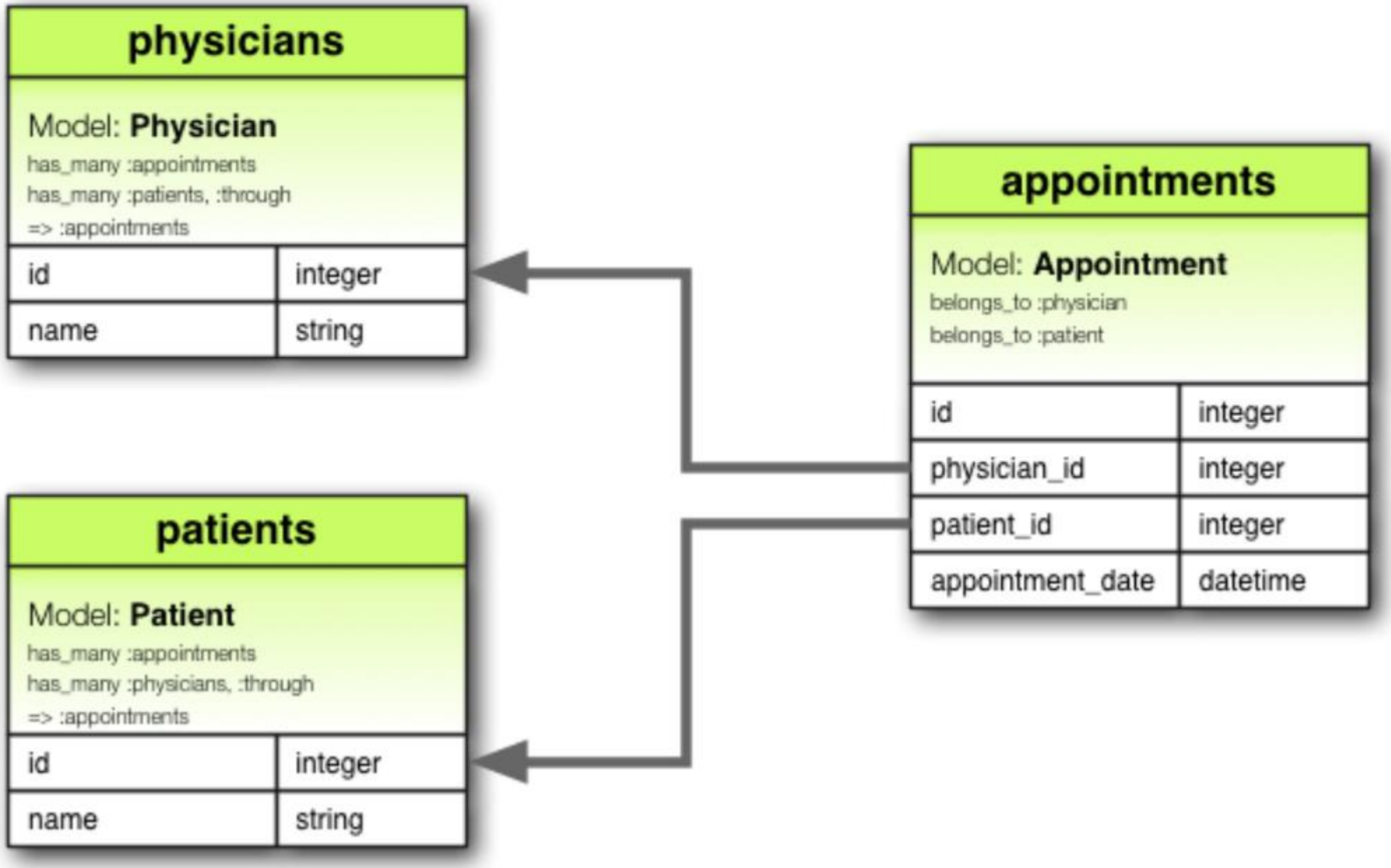
has_many

```ruby
class Author < ApplicationRecord
  has_many :books
end
```



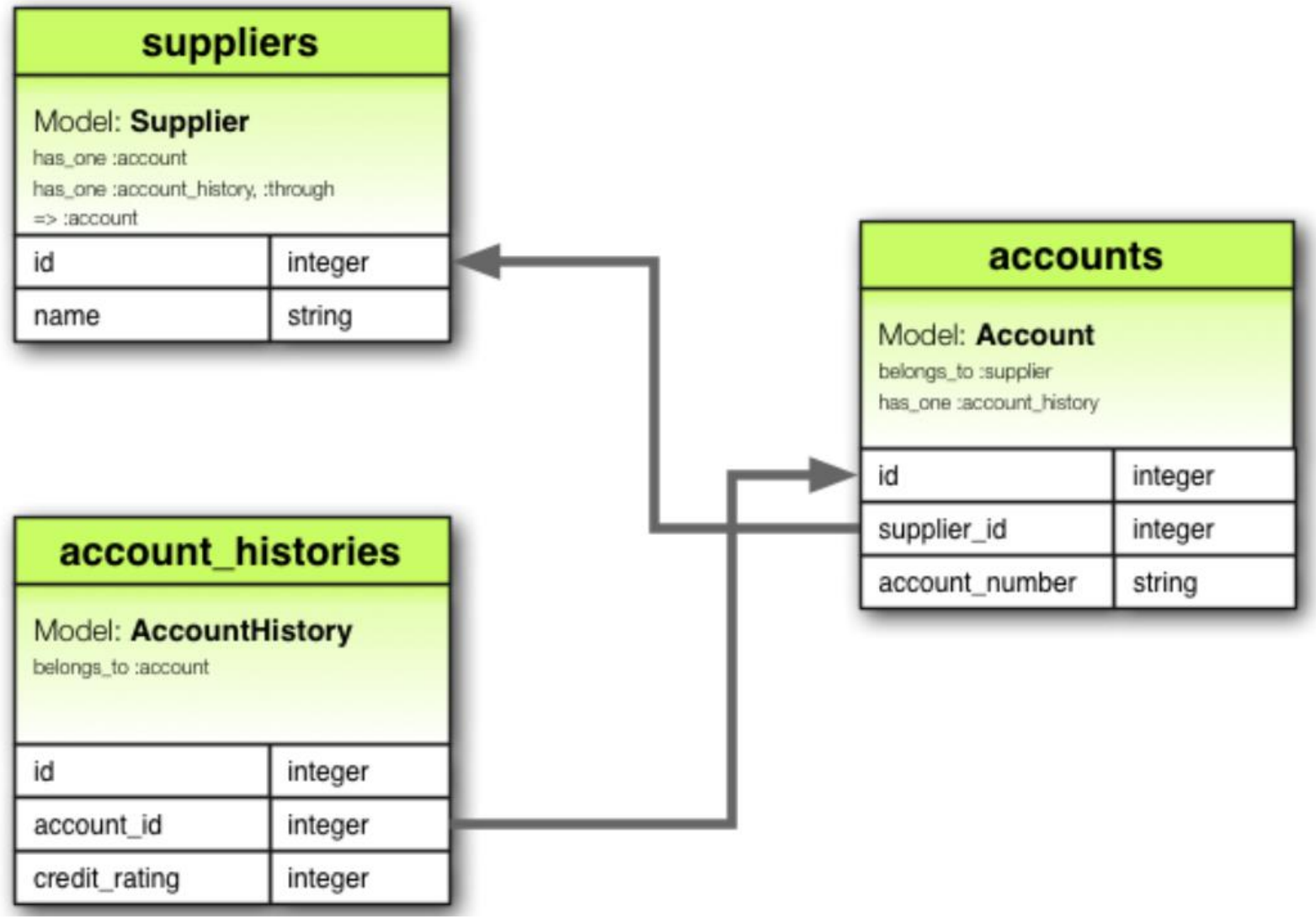
has_many :through

```ruby
class Physician < ApplicationRecord
  has_many :appointments
  has_many :patients, through: :appointments
end

class Appointment < ApplicationRecord
  belongs_to :physician
  belongs_to :patient
end

class Patient < ApplicationRecord
  has_many :appointments
  has_many :physicians, through: :appointments
end
```


physicians
Model: Physician
has_many :appointments
has_many :patients, :through
=> :appointments
id
integer
name
string
appointments
Model: Appointment
belongs_to :physician
belongs_to :patient
id
integer
physician_id
integer
patient_id
integer
appointment_date
datetime
patients
Model: Patient
has_many :appointments
has_many :physicians, :through
=> :appointments
id
integer
name
string

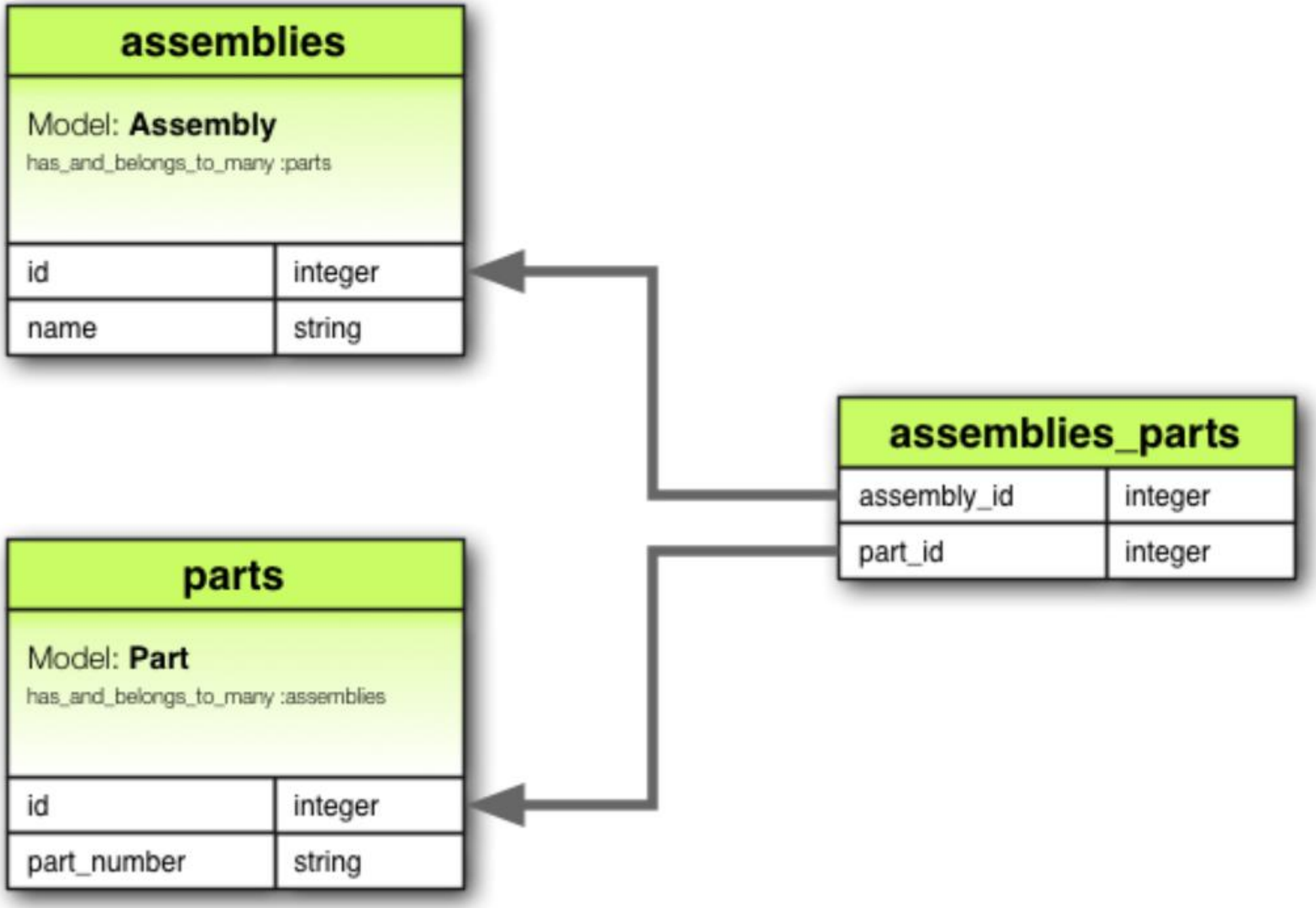
has_one :through

```ruby
class Supplier < ApplicationRecord
  has_one :account
  has_one :account_history, through: :account
end

class Account < ApplicationRecord
  belongs_to :supplier
  has_one :account_history
end

class AccountHistory < ApplicationRecord
  belongs_to :account
end
```


suppliers
Model: Supplier
has_one :account
has_one :account_history, :through
=> :account
id
integer
name
string
accounts
Model: Account
belongs_to :supplier
has_one :account_history
id
integer
supplier_id
integer
account_number
string
account_histories
Model: AccountHistory
belongs_to :account
id
integer
account_id
integer
credit_rating
integer

has_and_belongs_to_many

```ruby
class Assembly < ApplicationRecord
  has_and_belongs_to_many :parts
end

class Part < ApplicationRecord
  has_and_belongs_to_many :assemblies
end
```


assemblies
Model: Assembly
has_and_belongs_to_many :parts
id
integer
name
string
assemblies_parts
assembly_id
integer
part_id
integer
parts
Model: Part
has_and_belongs_to_many :assemblies
id
integer
part_number
string

Callbacks

Что мы хотим?

- Валидацию
- Бизнес-логику
- Понятный порядок вызовов
- Структурированную обработку ошибок

```ruby
class User < ApplicationRecord
  validates :login, :email, presence: true

  before_validation :ensure_login_has_a_value

  private
  def ensure_login_has_a_value
    if login.nil?
      self.login = email unless email.blank?
    end
  end
end
```

Фильтруем вызовы

```ruby
class User < ApplicationRecord
  before_validation :normalize_name, on: :create

  after_validation :set_location, on: [ :create, :update ]

  private
  def normalize_name
    self.name = name.downcase.titleize
  end

  def set_location
    self.location = LocationService.query(self)
  end
end
```

При создании объекта

before_validation
after_validation
before_save
around_save
before_create
around_create
after_create
after_save
after_commit
after_rollback

При обновлении объекта

before_validation
after_validation
before_save
around_save
before_update
around_update
after_update
after_save
after_commit
after_rollback

При удалении объекта

before_destroy
around_destroy
after_destroy
after_commit/after_rollback

Остальные

```ruby
class User < ApplicationRecord
  after_initialize do |user|
    puts "You have initialized an object!"
  end

  after_find do |user|
    puts "You have found an object!"
  end
end

>> User.new
#You have initialized an object!
>> User.first
#You have found an object!
#You have initialized an object
```

Методы, которые вызывают колбеки

create
create!
destroy
destroy!
destroy_all
save
save!

save(validate: false)
toggle!
touch
update_attribute
update
update!
valid?

Методы, которые НЕ вызывают колбеки

- decrement!
- decrement_counter
- delete
- delete_all
- increment!
- increment_counter
- update_column
- update_columns
- update_all
- update_counters

Прерывание цепочки колбеков

```
throw :abort
```

```
ActiveRecord::Rollback
ActiveRecord::RecordInvalid
```

Колбеки в ассоциациях

```ruby
class User < ApplicationRecord
  has_many :articles, dependent: :destroy
end

class Article < ApplicationRecord
  after_destroy :log_destroy_action

  def log_destroy_action
    puts 'Article destroyed'
  end
end
```

Условные колбеки

```ruby
class Order < ApplicationRecord
  before_save :normalize_card_number,
              if: :paid_with_card?
end
```

```ruby
class Order < ApplicationRecord
  before_save :normalize_card_number,
              if: Proc.new { |order| order.paid_with_card? }
end
```

Классы колбеков

```
class PictureFileCallbacks
  def after_destroy(picture_file)
    if File.exist?(picture_file.filepath)
      File.delete(picture_file.filepath)
    end
  end
end
```

Подключение классов-колбеков

```
class PictureFile < ApplicationRecord
  after_destroy PictureFileCallbacks.new
end
```

Со статическими методами

```
class PictureFileCallbacks
  def self.after_destroy(picture_file)
    if File.exist?(picture_file.filepath)
      File.delete(picture_file.filepath)
    end
  end
end

class PictureFile < ApplicationRecord
  after_destroy PictureFileCallbacks
end
```

Колбеки + Транзакции

```ruby
PictureFile.transaction do
  picture_file_1.destroy
  picture_file_2.save!
end
```

Цепочка колбеков будет общей
Обрабатывать исключения можно в колбеке
Колбеки after_commit и after_rollback не попадут в транзакцию

Миграции

```ruby
class CreateProducts < ActiveRecord::Migration[5.0]
  def change
    create_table :products do |t|
      t.string :name
      t.text :description

      t.timestamps
    end
  end
end
```

```ruby
class CreateProducts < ActiveRecord::Migration
  def up
    create_table :products do |t|
      t.string :name
      t.text :description
      t.timestamps
    end
  end

  def down
    drop_table :products
  end
end
```

```ruby
class AddReceiveNewsletterToUsers < ActiveRecord::Migration
  def up
    change_table :users do |t|
      t.boolean :receive_newsletter, :default => false
    end
    User.update_all ["receive_newsletter = ?", true]
  end

  def down
    remove_column :users, :receive_newsletter
  end
end
```

Подвох!

```
class AddFlagToProduct < ActiveRecord::Migration
  def change
    add_column :products, :flag, :boolean
    Product.all.each do |product|
      product.update_attributes!(:flag => 'false')
    end
  end
end
```

Обновляем модель

```
class Product < ActiveRecord::Base
  validates :flag, :presence => true
end
```

Снова мигрируем

```
class AddFuzzToProduct < ActiveRecord::Migration
  def change
    add_column :products, :fuzz, :string
    Product.all.each do |product|
      product.update_attributes! :fuzz => 'fuzzy'
    end
  end
end
```

И снова обновляем модель

```
class Product < ActiveRecord::Base
  validates :flag, :fuzz, :presence => true
end
```

Решение

```ruby
class AddFlagToProduct < ActiveRecord::Migration
  class Product < ActiveRecord::Base
  end

  def change
    add_column :products, :flag, :integer
    Product.reset_column_information
    Product.all.each do |product|
      product.update_attributes!(:flag => false)
    end
  end
end
```

Что можно делать в миграции?

add_column
add_index
change_column
change_table
create_table
drop_table
remove_column
remove_index
rename_column

Порядок миграций

- 20190424163039_initial_migration.rb
- 20190811123621_create_active_storage_tables.active_storage.rb
- 20190811123733_add_cards.rb
- 20190811125830_devise_create_admin_users.rb
- 20190811125839_create_active_admin_comments.rb
- 20190811173104_add_template_file_name.rb
- 20190811174806_fix_card_template_key.rb

Доступные типы столбцов

`:binary`	`:float`
`:boolean`	`:integer`
`:date`	`:primary_key`
`:datetime`	`:string`
`:decimal`	`:text`
	`:time`

```
create_table :products do |t|
  t.column :name, 'polygon', :null => false
end
```

Генераторы

```
rails generate model
                    Product name:string description :text
```

```ruby
class CreateProducts < ActiveRecord::Migration
  def change
    create_table :products do |t|
      t.string :name
      t.text :description

      t.timestamps
    end
  end
end
```

Добавляем столбец

```
rails generate migration AddPartNumberToProducts
                                part_number:string
```

Удаляем столбец

```
rails generate migration RemovePartNumberFromProducts
                                part_number:string
```

Результат

```
class RemovePartNumberFromProducts < ActiveRecord::Migration
  def up
    remove_column :products, :part_number
  end

  def down
    add_column :products, :part_number, :string
  end
end
```

Создание таблиц

```ruby
create_table :products do |t|
  t.column :name, :string, :null => false
end

create_table :products do |t|
  t.string :name, :null => false
end
```

Изменение таблиц

```
change_table :products do |t|
  t.remove :description, :name
  t.string :part_number
  t.index :part_number
  t.rename :upccode, :upc_code
end
```

Helpers

```ruby
change_table :products do |t|
  t.timestamps
end

create_table :products do |t|
  t.references :category
end
```

Если ничего не помогло

```
execute <<-SQL
        ALTER TABLE products
          ADD CONSTRAINT fk_products_categories
          FOREIGN KEY (category_id)
          REFERENCES categories(id)
SQL
```

Выполнение миграций

```
rake db:migrate
rake db:rollback
rake db:rollback STEP=3
rake db:migrate:redo STEP=3

db:reset
```

Schema.rb

```ruby
ActiveRecord::Schema.define(:version => 20080906171750) do
  create_table "authors", :force => true do |t|
    t.string   "name"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "products", :force => true do |t|
    t.string   "name"
    t.text "description"
    t.datetime "created_at"
    t.datetime "updated_at"
    t.string "part_number"
  end
end
```

Ссылки

https://guides.rubyonrails.org/active_record_querying.html

https://guides.rubyonrails.org/association_basics.html

https://guides.rubyonrails.org/v3.2/migrations.html