daftcode

RUBY ON RAILS
Level UP

2. Bazy Danych

Praca domowa

RUBY ON RAILS
Level UP

Migracje

RUBY ON RAILS
Level UP

Schema

```ruby
# db/schema.rb

ActiveRecord::Schema.define(version: 2019_03_14_150921) do
  create_table "clans", force: :cascade do |t|
    t.string "name", null: false
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

  create_table "samurais", force: :cascade do |t|
    t.string "name", null: false
    t.integer "armor_quality", default: 0
    t.integer "number_of_battles", default: 0
    t.date "join_date"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end
end
```

Tworzenie migracji

```ruby
# db/migrate/YYYYMMDDHHMMSS_add_death_date_to_samurai.rb

class AddDeathDateToSamurai < ActiveRecord::Migration[5.2]
  def change
  end
end
```

Tworzenie migracji

```ruby
# db/migrate/YYYYMMDDHHMMSS_add_death_date_to_samurai.rb

class AddDeathDateToSamurai < ActiveRecord::Migration[5.2]
  def change
    add_column :samurais, :death_date, :date
  end
end
```

Schema

```ruby
# db/schema.rb

ActiveRecord::Schema.define(version: 2019_03_20_120546) do
  create_table "clans", force: :cascade do |t|
    t.string "name", null: false
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

  create_table "samurais", force: :cascade do |t|
    t.string "name", null: false
    t.integer "armor_quality", default: 0
    t.integer "number_of_battles", default: 0
    t.date "join_date"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
    t.date "death_date"
  end
end
```

Indeksy

RUBY ON RAILS
Level UP

Indeksy na relacjach

```ruby
# db/migrate/YYYYMMDDHHMMSS_reference_clan_from_samurais.rb

class ReferenceClanFromSamurais < ActiveRecord::Migration[5.2]
  def change
    add_reference :samurais, :clan
  end
end
```

```ruby
# db/schema.rb

create_table "samurais", force: :cascade do |t|
  ...
  t.integer "clan_id"
  t.index ["clan_id"], name: "index_samurais_on_clan_id"
end
```

Indeksy na relacjach

```ruby
# app/models/samurai.rb

class Samurai < ApplicationRecord
  belongs_to :clan
  ...
end


# app/models/clan.rb

class Clan < ApplicationRecord
  has_many :samurais, dependent: :destroy
  ...
end
```

Dodatkowe opcje

```ruby
# db/migrate/YYYYMMDDHHMMSS_index_name_on_clans.rb

class IndexNameOnClans < ActiveRecord::Migration[5.2]
  def change
    add_index :clans, :name
  end
end


# db/schema.rb

create_table "clans", force: :cascade do |t|
  ...
  t.index ["name"],
    name: "index_clans_on_name"
end
```

Dodatkowe opcje

```ruby
# db/migrate/YYYYMMDDHHMMSS_index_name_on_samurais.rb

class IndexNameOnClans < ActiveRecord::Migration[5.2]
  def change
    add_index :clans, :name, unique: true
  end
end


# db/schema.rb

create_table "clans", force: :cascade do |t|
  ...
  t.index ["name"],
    name: "index_clans_on_name",
    unique: true
end
```

Dodatkowe opcje

```
# rails console

> clan1 = Clan.create(name: 'Najlepszy klan')
=> #<Clan id: 1, name: "Najlepszy klan" ...>

> clan2 = Clan.create(name: 'Najlepszy klan')
# ActiveRecord::RecordNotUnique:
    SQLite3::ConstraintException:
      UNIQUE constraint failed:
      clans.name:
      INSERT INTO "clans" ("name", "created_at", "updated_at") VALUES (?, ?, ?)
```

Dodatkowe opcje

```ruby
# app/models/clan.rb

class Clan < ApplicationRecord
  ...
  validates :name, presence: true, uniqueness: true
end


# rails console

> clan1 = Clan.create(name: 'Najlepszy klan')
=> #<Clan id: 1, name: "Najlepszy klan" ...>

> clan2 = Clan.new(name: 'Najlepszy klan')
=> #<Clan id: nil, name: "Najlepszy klan", created_at: nil, updated_at: nil>
> clan2.valid?
=> false
> clan2.errors
=> {:name=>[{:error=>:taken, :value=>"Najlepszy klan"}]}
```

Dodatkowe opcje

```ruby
# db/migrate/YYYYMMDDHHMMSS_index_name_on_samurais.rb

class IndexNameOnClans < ActiveRecord::Migration[5.2]
  def change
    add_index :clans, :name, unique: true, where: "name != 'Bez nazwy'"
  end
end


# db/schema.rb

create_table "clans", force: :cascade do |t|
  ...
  t.index ["name"],
    name: "index_clans_on_name",
    unique: true,
    where: "name != 'Bez nazwy'"
end
```

Dodatkowe opcje

```ruby
# app/models/clan.rb

class Clan < ApplicationRecord
  ...
  validates :name,
    presence: true,
    uniqueness: { conditions: -> { where("name != 'Bez nazwy'") } }
end
```

Klucze obce

RUBY ON RAILS
Level UP

Prosty klucz obcy

```ruby
# db/migrate/YYYYMMDDHHMMSS_add_clan_foreign_key_to_samurais.rb

class AddClanForeignKeyToSamurais < ActiveRecord::Migration[5.2]
  def change
    add_foreign_key :samurais, :clans
  end
end


# db/schema.rb

...
add_foreign_key "samurais", "clans"
```

Prosty klucz obcy

```ruby
# db/migrate/YYYYMMDDHHMMSS_create_samurais.rb

class CreateSamurais < ActiveRecord::Migration[5.2]
  def change
    create_table :samurais do |t|
      ...
      t.references :clan, foreign_key: true
    end
  end
end


# db/schema.rb

...
add_foreign_key "samurais", "clans"
```

Efekty

```
class Samurai < ApplicationRecord
  belongs_to :clan
end

> Samurai.create!(clan_id: -1)
# ActiveRecord::RecordInvalid:
    Validation failed:
      Clan must exist


> Samurai.create!(clan_id: -1)
# ActiveRecord::InvalidForeignKey:
    PG::ForeignKeyViolation: ERROR:
      insert or update on table "samurais" violates foreign key constraint
```

Efekty

```ruby
class Samurai < ApplicationRecord
  belongs_to :clan
end

class Clan < ApplicationRecord
  has_many :samurais, dependent: :destroy
end

> clan_with_samurais.destroy!
# 👍
```

```ruby
class Samurai < ApplicationRecord
  belongs_to :clan
end

class Clan < ApplicationRecord
  has_many :samurais
end

> clan_with_samurais.destroy!
# ActiveRecord::InvalidForeignKey:
    PG::ForeignKeyViolation: ERROR:
      update or delete on table “clans"
      violates foreign key constraint
```

Zaawansowane relacje

RUBY ON RAILS
Level UP

STI - dziedziczenie w bazie danych

```ruby
# app/models/samurai.rb

class Samurai < ApplicationRecord
  belongs_to :clan
  ...
end

# app/models/hussar.rb

class Hussar < ApplicationRecord
  belongs_to :clan
  ...
end
```

STI - dziedziczenie w bazie danych

```ruby
# app/models/samurai.rb

class Samurai < ApplicationRecord
  belongs_to :clan
  ...

  def attack
    "#{name} used katana"
  end
end

# app/models/hussar.rb

class Hussar < ApplicationRecord
  belongs_to :clan
  ...

  def attack
    "CHARGE!!"
  end
end
```

STI - dziedziczenie w bazie danych

```ruby
# app/models/warrior.rb

class Warrior < ApplicationRecord
  belongs_to :clan
  ...
end

# app/models/samurai.rb

class Samurai < Warrior
 def attack
    "#{name} used katana"
  end
end

# app/models/hussar.rb

class Hussar < Warrior
  def attack
    "CHARGE!!"
  end
end
```

RUBY ON RAILS
Level UP

STI - dziedziczenie w bazie danych

```ruby
# db/migrate/YYYYMMDDHHMMSS_change_samurais_to_warriors.rb

class ChangeSamuraisToWarriors < ActiveRecord::Migration[5.2]
  def change
    rename_table :samurais, :warriors
    add_column :warriors, :type, :string, default: 'Samurai'
  end
end


# app/models/clan.rb

class Clan < ApplicationRecord
  has_many :warriors, dependent: :destroy
  ...
end
```

STI - dziedziczenie w bazie danych

```
# rails console

> clan = Clan.create(name: 'Common clan')
> Samurai.create(name: 'Mister Samurai', clan: clan)
> Hussar.create(name: 'Pan Husarz', clan: clan)
> Samurai.create(name: 'Mister Samurai', clan: clan)

> clan.warriors.map(&:attack)
=> ["Mr. Samurai used katana", "CHARGE!!", "Mr. Samurai used katana"]
```

Polimorfizm

```ruby
# db/migrate/YYYYMMDDHHMMSS_add_defensible_to_warriors.rb

class AddDefensibleToWarriors < ActiveRecord::Migration[5.2]
  def change
    add_reference :warriors, :defensible, polymorphic: true, index: true
  end
end

# db/schema.rb

create_table "warriors", force: :cascade do |t|
  ...
  t.string "defensible_type"
  t.integer "defensible_id"
  t.index ["defensible_type", "defensible_id"],
    name: "index_warriors_on_defensible_type_and_defensible_id"
end
```

Polimorfizm

```ruby
# app/models/barricade.rb

class Barricade < ApplicationRecord
  has_many :warriors, as: :defensible
end

# app/models/stronghold.rb

class Stronghold < ApplicationRecord
  has_many :warriors, as: :defensible
end

# app/models/warrior.rb

class Warrior < ApplicationRecord
  ...
  belongs_to :defensible, polymorphic: true
end
```

Thanks!