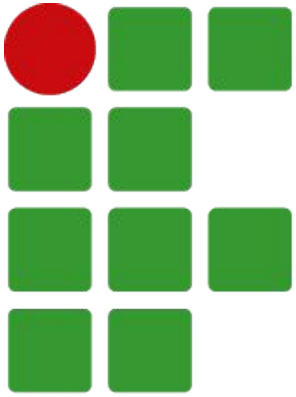


INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ruby on Rails

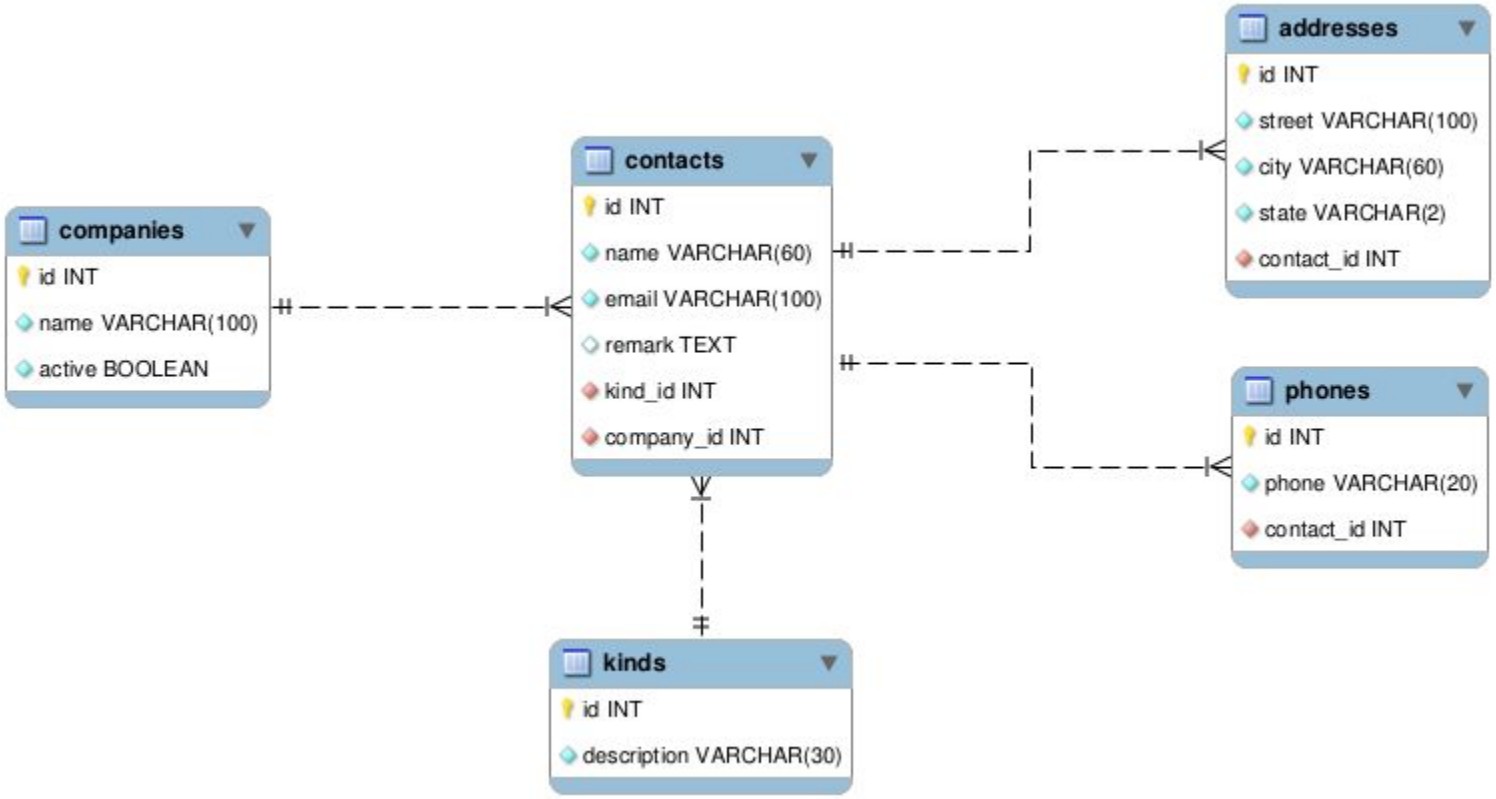
Associações

Desenvolvimento Web III

Prof. Diego Stiehl

INSTITUTO FEDERAL

Relembrando do Modelo



INSTITUTO FEDERAL

Arquivo de Seeds

- Quando criamos um novo BD para testes de desenvolvimento ainda não temos dados
- Se eu precisar “dropar” o BD, perco os dados
- Não precisamos criar todos dados na mão
- Rodar comando
  - rails db:seed
- Rails vai procurar e executar script do arquivo
  - db/seeds.rb

INSTITUTO FEDERAL

seeds.rb

- O script pode conter qualquer código Ruby
- Focar na criação de novas instâncias dos modelos da nossa aplicação
  - Persistir instâncias no BD
- Comando muito utilizado
  - MeuModel.create!
- Exemplo
  - Kind.create! description: 'Trabalho'

INSTITUTO FEDERAL

Prática

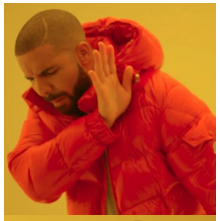
- Usando a gem Faker, crie (seeds.rb):
  - 5 Tipos
  - 10 Empresas
  - 10 Contatos para cada Empresa
    - Cada Contato tem um Tipo aleatório
  - 2 Endereços para cada Contato
  - 3 Telefones para cada Contato

INSTITUTO FEDERAL

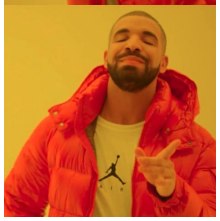
Associações

- Você usou os IDs para referenciar outras entidades?
  - Não é a melhor solução!
  - Referências com IDs remetem ao paradigma relacional (banco de dados)
- Devemos utilizar objetos Ruby

```
@contact.kind_id
```



```
@contact.kind.id
```



INSTITUTO FEDERAL

Mapeamento

- Associações mapeiam chaves (IDs) existentes
  - Fazem uma conversão automática para objetos Ruby (CoC)
- As associações são “coisas” de Model
  - Devem estar localizadas em suas respectivas classes de Model
    - app/models/*.rb

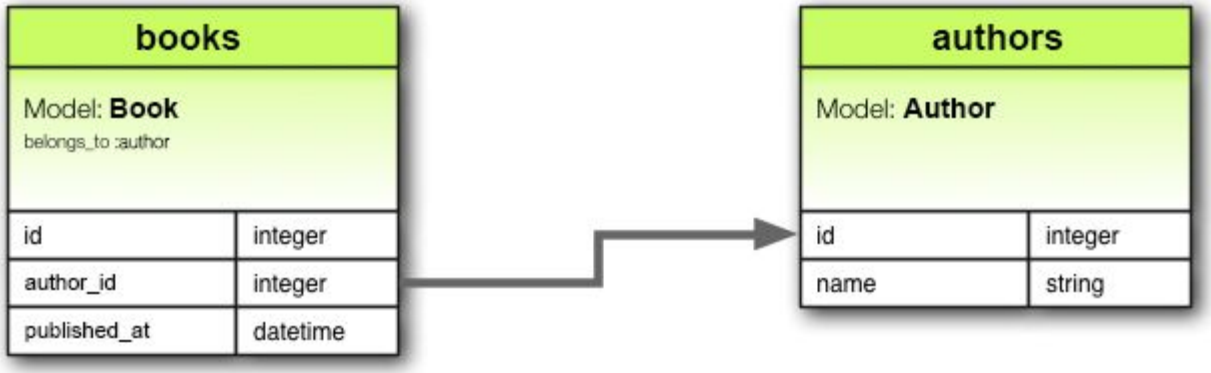
INSTITUTO FEDERAL

Tipos de Associações

- Em Rails, as associações podem ser dos tipos:
  - belongs_to
  - has_one
  - has_many
  - has_many :through
  - has_one :through
  - has_and_belongs_to_many

belongs_to

- Define uma associação um-para-um com outro modelo
  - A instância que declara pertence à instância de outro modelo




```
class Book < ActiveRecord: :Base
  belongs_to :author
end
```

INSTITUTO FEDERAL

belongs_to

- Migration correspondente

```
class CreateBooks < ActiveRecord::Migration[5.0]
  def change
    create_table :authors do |t|
      t.string :name
      t.timestamps
    end

    create_table :books do |t|
      t.belongs_to :author, index: true
      t.datetime :published_at
      t.timestamps
    end
  end
end
```

t.references também funciona

INSTITUTO FEDERAL

Importante

- Associações belongs_to DEVEM usar o nome do atributo em SINGULAR
- Exemplos:
  - belongs_to :author
  - belongs_to :company
  - belongs_to :contact

INSTITUTO FEDERAL

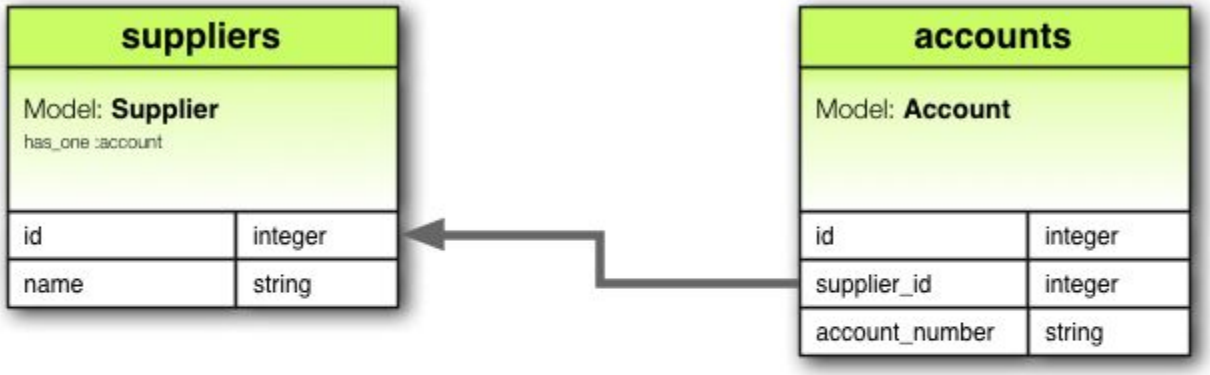
Prática

- Detecte e aplique as associações belongs_to do modelo de dados da aplicação de contatos
- Utilize o Rails Console para testar seus objetos

INSTITUTO FEDERAL

has_one

- Também define uma associação um-para-um com outro modelo
  - Diferentes semântica e consequências
  - A instância tem uma instância de outro modelo




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

INSTITUTO FEDERAL

has_one

- Migration correspondente

```
class CreateSuppliers < ActiveRecord::Migration[5.0]
  def change
    create_table :suppliers do |t|
      t.string :name
      t.timestamps
    end

    create_table :accounts do |t|
      t.belongs_to :supplier, index: true
      t.string :account_number
      t.timestamps
    end
  end
end
```

INSTITUTO FEDERAL

has_one

- Em resumo
  - O **has_one** é o “outro lado” do **belongs_to** em uma relação um-para-um bidirecional
- Comum ser índice único
  - Exemplo de migration:

```
create_table :accounts do |t|
  t.belongs_to :supplier, index: { unique: true }, foreign_key: true
  # ...
end
```

INSTITUTO FEDERAL

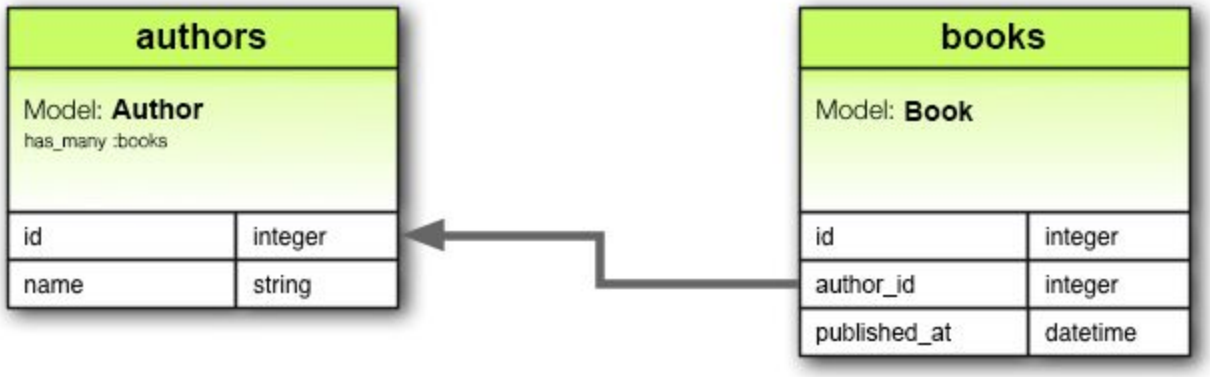
Prática

- Detecte e aplique as associações has_one do modelo de dados da aplicação de contatos
  - Dica:
    - Digamos que, no nosso modelo, um usuário poderá ter apenas um endereço
- Utilize o Rails Console para testar seus objetos

INSTITUTO FEDERAL

has_many

- Define uma associação um-para-muitos com outro modelo
  - Quase sempre é o “outro lado” do belongs_to
  - A instância tem zero ou mais instâncias de outro modelo




```
class Author < ActiveRecord: :Base
  has_many :books
end
```

INSTITUTO FEDERAL

has_many

- Migration correspondente

```
class CreateAuthors < ActiveRecord::Migration[5.0]
  def change
    create_table :authors do |t|
      t.string :name
      t.timestamps
    end

    create_table :books do |t|
      t.belongs_to :author, index: true
      t.datetime :published_at
      t.timestamps
    end
  end
end
```

INSTITUTO FEDERAL

Prática

- Detecte e aplique as associações has_many do modelo de dados da aplicação de contatos
- Utilize o Rails Console para testar seus objetos

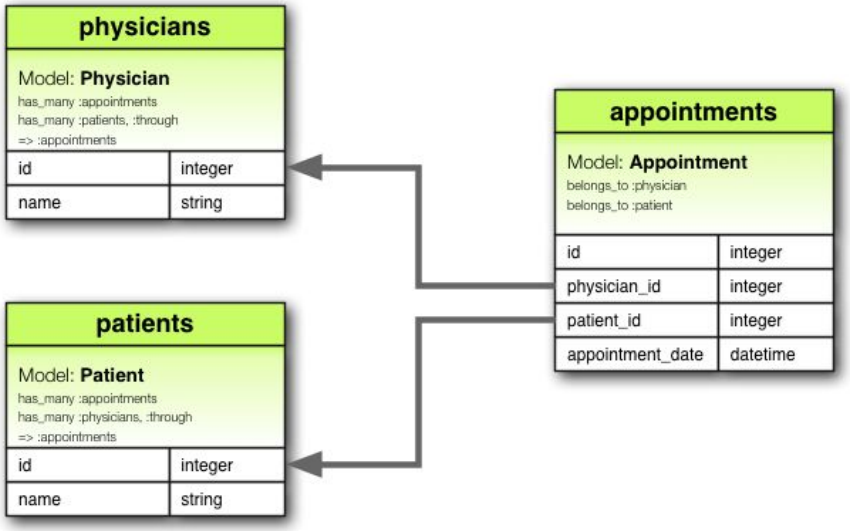
INSTITUTO FEDERAL

has_many :through

- Frequentemente usado para definir uma associação muitos-para-muitos com outro modelo
- Indica que existe uma relação com outro modelo através de um terceiro modelo

INSTITUTO FEDERAL

Exemplo




```
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
```

INSTITUTO FEDERAL

has_many :through

- Migration correspondente

```
class CreateAppointments < ActiveRecord::Migration[5.0]
  def change
    create_table :physicians do |t|
      t.string :name
      t.timestamps
    end

    create_table :patients do |t|
      t.string :name
      t.timestamps
    end

    create_table :appointments do |t|
      t.belongs_to :physician, index: true
      t.belongs_to :patient, index: true
      t.datetime :appointment_date
      t.timestamps
    end
  end
end
```

INSTITUTO FEDERAL

has_many :through - Atalhos

- O has_many :through também é útil para criarmos **atalhos** através de associações aninhadas
- Exemplo:
  - Se um Documento tem muitas Sessões
    - E se uma Sessão tem muitos Parágrafos
      - Podemos dizer que um Documento tem muitos Parágrafos
        - » Podemos mapear esse “atalho”

INSTITUTO FEDERAL

has_many :through - Atalhos

Exemplo:

```
class Document < ApplicationRecord
  has_many :sections
  has_many :paragraphs, through: :sections
end

class Section < ApplicationRecord
  belongs_to :document
  has_many :paragraphs
end

class Paragraph < ApplicationRecord
  belongs_to :section
end
```

INSTITUTO FEDERAL

Prática

- Encontre e mapeie as possibilidades de atalhos com **has_many :through** em nossa aplicação de contatos
- Utilize o Rails Console para testar seus objetos

INSTITUTO FEDERAL

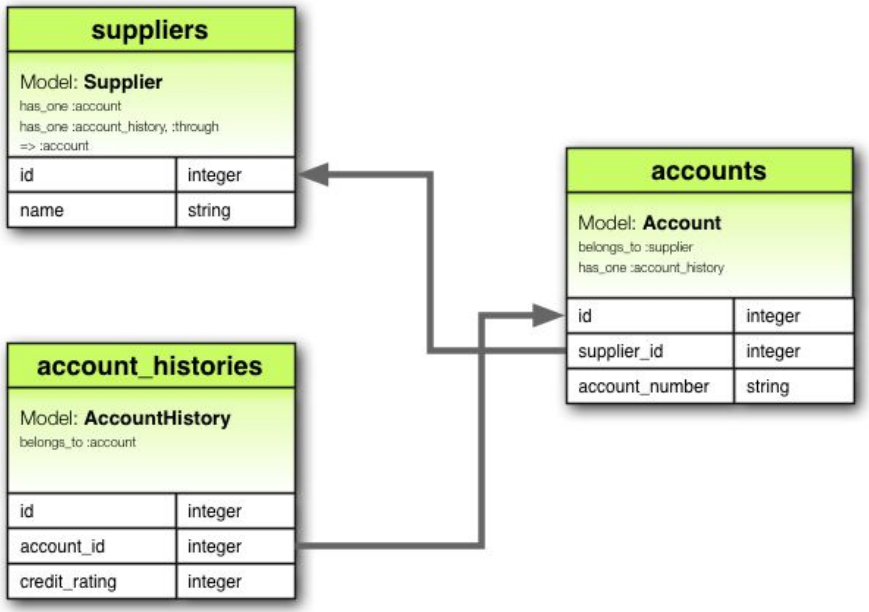
has_one :through

- Define uma relação um-para-um com outro modelo através de um terceiro modelo
  - Deve haver uma relação **has_one** prévia
- O has_one :through vai criar o “atalho” diretamente para a entidade informada

INSTITUTO FEDERAL

has_one :through

- Exemplo:




```
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
```

INSTITUTO FEDERAL

has_one :through

- Migration correspondente

```
class CreateAccountHistories < ActiveRecord::Migration[5.0]
  def change
    create_table :suppliers do |t|
      t.string :name
      t.timestamps
    end

    create_table :accounts do |t|
      t.belongs_to :supplier, index: true
      t.string :account_number
      t.timestamps
    end

    create_table :account_histories do |t|
      t.belongs_to :account, index: true
      t.integer :credit_rating
      t.timestamps
    end
  end
end
```

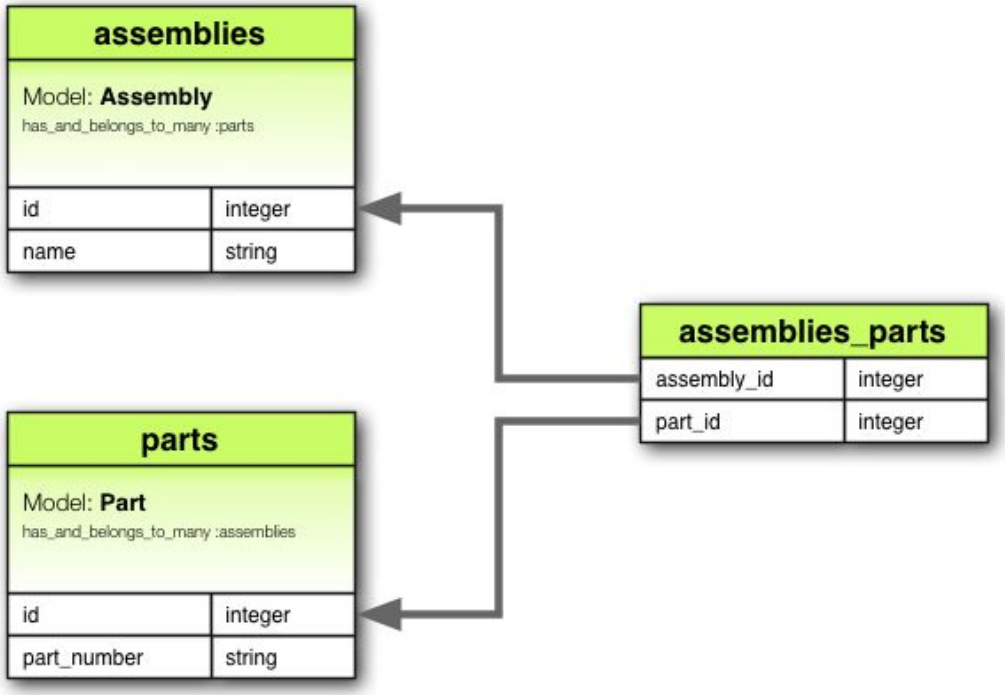
INSTITUTO FEDERAL

has_and_belongs_to_many

- Define uma relação muitos-para-muitos direta entre dois modelos
  - Não há a intervenção de outro modelo
  - Útil para quando não há atributos relacionados à associação entre os dois modelos

has_and_belongs_to_many

- Exemplo:




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

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

INSTITUTO FEDERAL

has_and_belongs_to_many

- Migration correspondente

```
class CreateAssembliesAndParts < ActiveRecord::Migration[5.0]
  def change
    create_table :assemblies do |t|
      t.string :name
      t.timestamps
    end

    create_table :parts do |t|
      t.string :part_number
      t.timestamps
    end

    create_table :assemblies_parts, id: false do |t|
      t.belongs_to :assembly, index: true
      t.belongs_to :part, index: true
    end
  end
end
```

INSTITUTO FEDERAL

Prática

- Adapte nosso modelo para o seguinte: