Rails Models
Advanced Associations

...

Devpoint Labs - Jake Sorce / Dave Jungst

Has Many Through Association

A has_many :through association is often used to set up a many-to-many connection with another model.

This association indicates that the declaring model can be matched with zero or more instances of another model by proceeding through a third model.

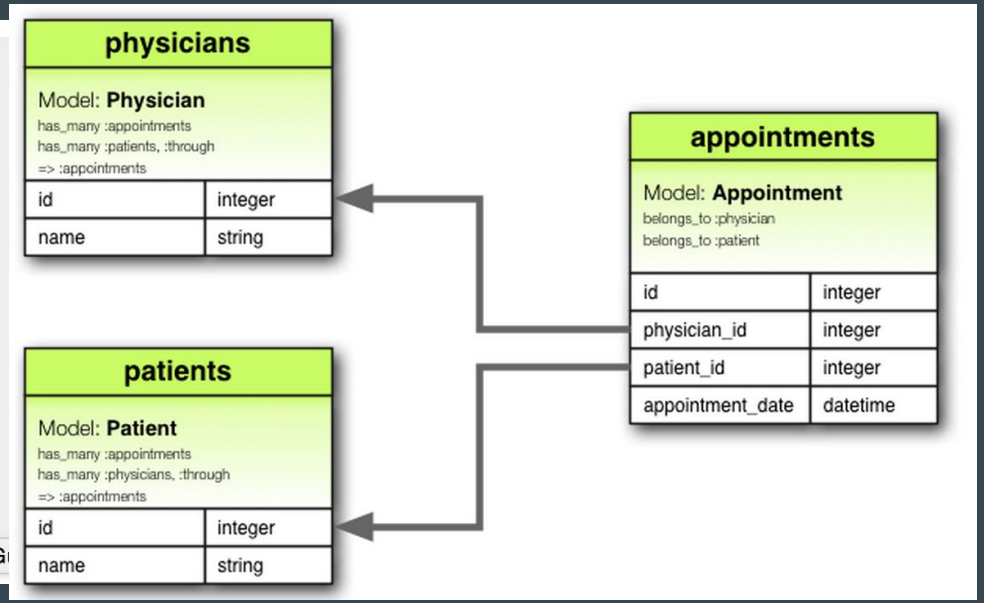
Has Many Through Association - Diagram

```ruby
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
```

Active Record Associations — Ruby on Rails G



Has Many Through Association - Migration

```
class CreateAppointments < ActiveRecord::Migration
  def change
    create_table :physicians do |t|
      t.string :name
      t.timestamps null: false
    end

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

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

Has One Through Association

A has_one :through association sets up a one-to-one connection with another model.

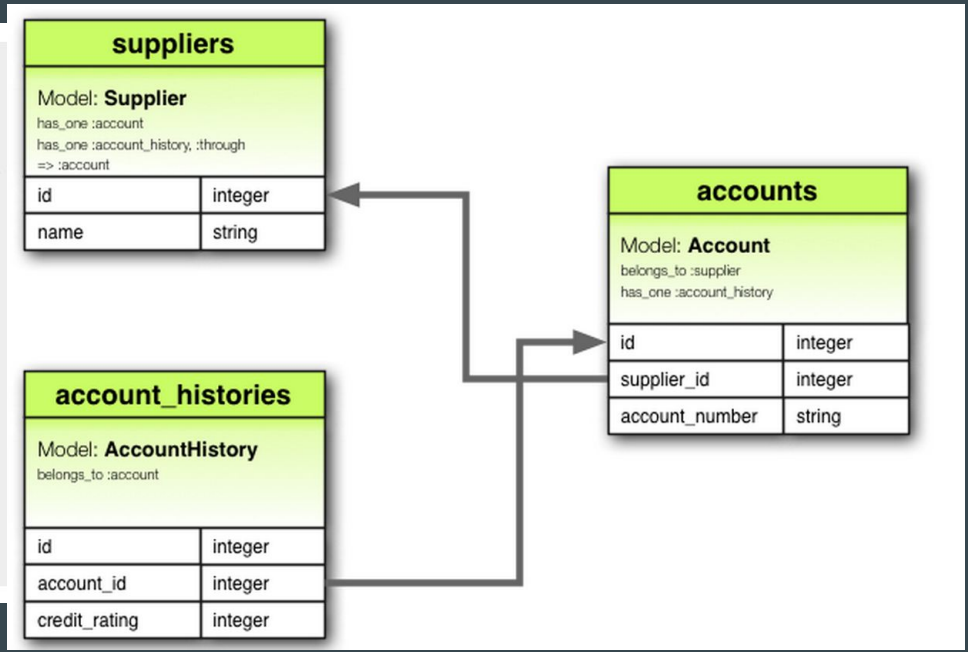
This association indicates that the declaring model can be matched with one instance of another model by proceeding through a third model.

Has One Through Association - Diagram

```ruby
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 One Through Association - Migration

```
class CreateAccountHistories < ActiveRecord::Migration
  def change
    create_table :suppliers do |t|
      t.string :name
      t.timestamps null: false
    end

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

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

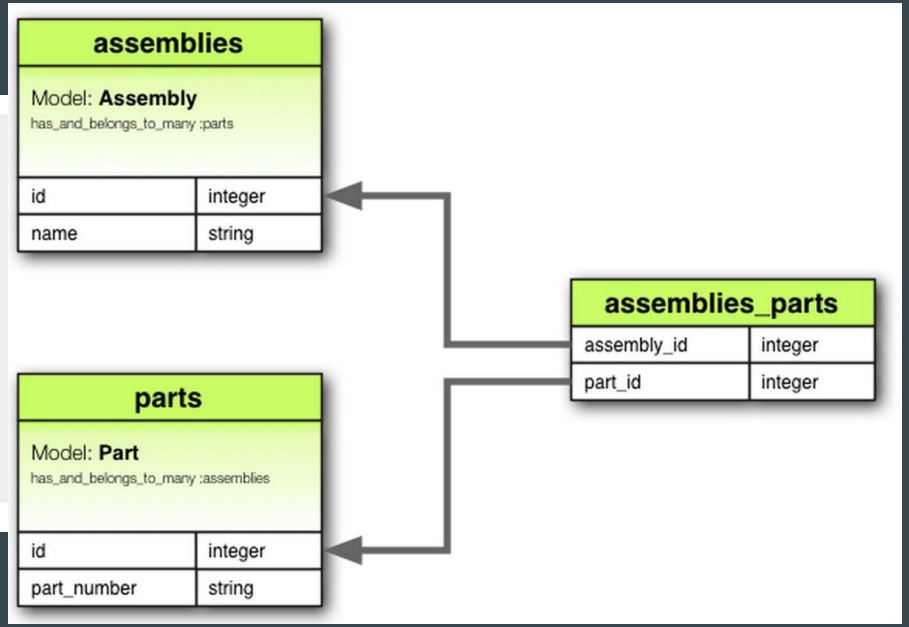
Has And Belongs To Many Association

A has_and_belongs_to_many association creates a direct many-to-many connection with another model, with no intervening model.

Has And Belongs To Many Association - Diagram

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

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



Has And Belongs To Many Association - Migration

```
class CreateAssembliesAndParts < ActiveRecord::Migration
  def change
    create_table :assemblies do |t|
      t.string :name
      t.timestamps null: false
    end

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

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

Choosing Between has_many :through and has_and_belongs_to_many

The simplest rule of thumb is that you should set up a has_many :through relationship if you need to work with the relationship model as an independent entity. If you don't need to do anything with the relationship model, it may be simpler to set up a has_and_belongs_to_many relationship (though you'll need to remember to create the joining table in the database).

You should use has_many :through if you need validations, callbacks, or extra attributes on the join model.

Has Many Through Real World Example

Suppose we have 2 models Users and Magazines

A user can subscribe to many magazines

A magazine can be subscribed to by many users

Here we have a perfect example of a has_many_through relationship

A user could have many magazines through subscriptions

A magazine could have many users through subscriptions

Example

rails new doctors_office -d postgresql --quiet

I am going to use the scaffold generator which will generate a model view and controller along with many other things. Some of which are not desired like scaffold. css

rails g scaffold Physician name:string

rails g scaffold Patient name:string

rails g scaffold Appointment physician_id:integer patient_id:integer appointment_date:datetime

bundle exec rake db:create db:migrate

Physician Model

app/models/physician.rb

```
class Physician < ActiveRecord::Base

  has_many :appointments

  has_many :patients, through: :appointments

end
```

Patient Model

app/models/patient.rb

```
class Patient < ActiveRecord::Base

  has_many :appointments

  has_many :physicians, through: :appointments

end
```

Appointment Model

app/models/patient.rb

```
class Patient < ActiveRecord::Base

  belongs_to :patient

  belongs_to :physician

end
```

Testing the relationship

bundle exec rails c

>> doctor1 = Physician.create(name: ‘Dre’)

>> patient1 = Patient.create(name: ‘Dave’)

>> doctor1.patients << patient1 (What are we doing here?)

>> Appointment.all (What will this return?)

>> patient1.physicians (What will this return?)

>> doctor1.patients (What will this return?)

A word of warning about scaffold

Scaffold is a great way to prototype an app or build a quick example app.

In practice it is better to only build out what you need and not create a bunch of files that you don't need / want.

There are many files in a scaffold that can produce undesired behavior.

Use the scaffold generation with extreme caution especially when first learning rails because you don't want to rely on it and not learn how to build an app naturally.