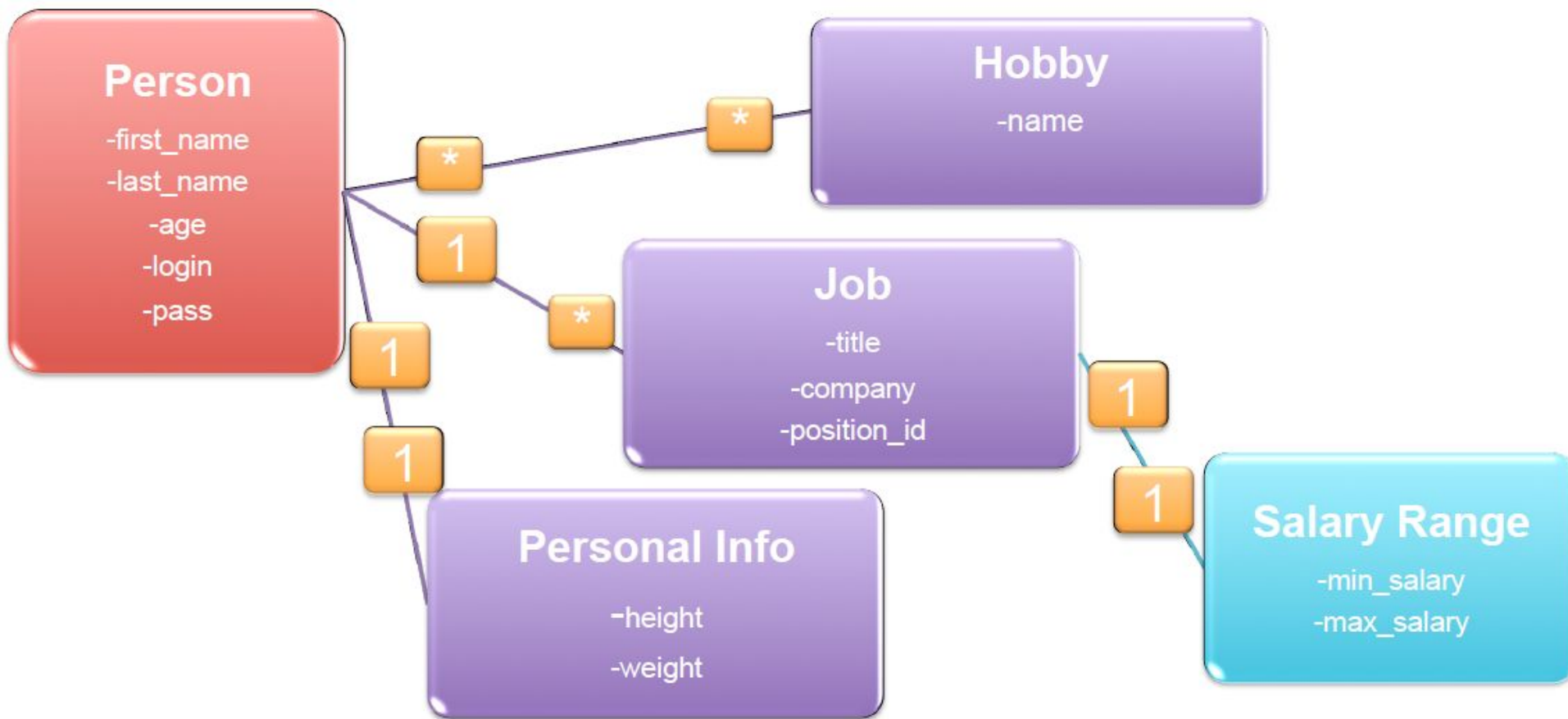

Ruby on Rails

— Active Record Relations —

Overview

- Relaciones entre entidades
 - one-to-one
 - one-to-many
 - many-to-many

Relaciones



Asociación one-to-one

One-to-One

- Un **person** tiene exactamente **una entrada** en **personal_info**
- Un entrada **personal_info** **pertenece** exactamente a un **person**.
- El lado “**belongs to**” o “pertenece” es el que tiene la **clave foránea**.

Convención: El nombre por defecto de la clave foránea es {master_table_singular}_id, ejemplo: person_id

One-to-One

```
~/advanced_ar$ rails g model personal_info height:float weight:float person:references
  invoke  active_record
  create  db/migrate/20150908232650_create_personal_infos.rb
  create  app/models/personal_info.rb
```

FOLDERS

- ▼ advanced_ar
 - ▶ app
 - ▶ bin
 - ▶ config
 - ▼ db
 - ▼ migrate
 - 20150908214851_create_people.rb
 - 20150908221446_add_login_pass_to_people.rb
 - 20150908232650_create_personal_infos.rb

```
20150908232650_create_personal_infos.rb ✕

class CreatePersonalInfos < ActiveRecord::Migration
  def change
    create_table :personal_infos do |t|
      t.float :height
      t.float :weight
      t.references :person, index: true, foreign_key: true

      t.timestamps null: false
    end
  end
end
```

Clave foránea

```
~/advanced_ar$ rake db:migrate
== 20150908232650 CreatePersonalInfos: migrating =====
-- create_table(:personal_infos)
   -> 0.0014s
== 20150908232650 CreatePersonalInfos: migrated (0.0014s) =====
```

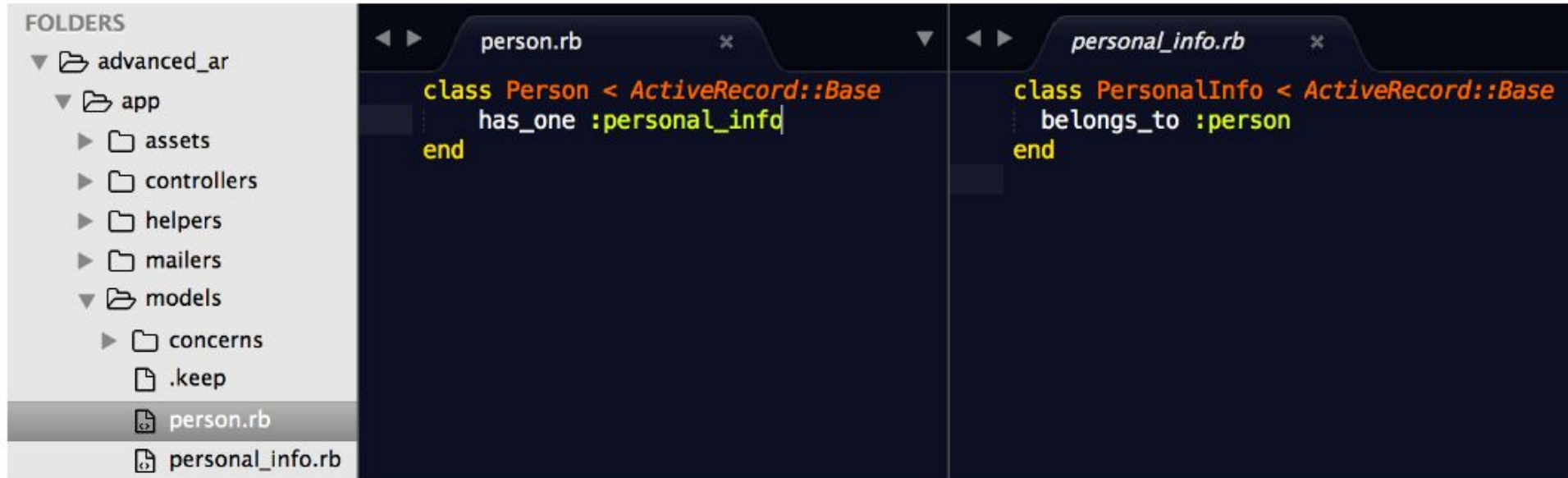
```
~/advanced_ar$ rails db
SQLite version 3.8.5 2014-08-15 22:37:57
```

```
Enter ".help" for usage hints.
```

```
sqlite> .schema personal_infos
```

```
CREATE TABLE "personal_infos" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "height" float, "weight" float, "person_id" integer
, "created_at" datetime NOT NULL, "updated_at" datetime NOT NULL);
CREATE INDEX "index_personal_infos_on_person_id" ON "personal_infos" ("person_id");
```

has_one / belongs_to



One-to-One

```
irb(main):001:0> bill = Person.find_by first_name: "Bill"
  Person Load (0.2ms) SELECT "people".* FROM "people" WHERE "people"."first_name" = ? LIMIT 1 [["first_name", "Bill"]]
=> #<Person id: 13, first_name: "Bill", age: 75, last_name: "Gates", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08 22:22:51", login: "bill", pass: "windows3.1">
irb(main):002:0> bill.personal_info
  PersonalInfo Load (0.1ms) SELECT "personal_infos".* FROM "personal_infos" WHERE "personal_infos"."person_id" = ? LIMIT 1 [["person_id", 13]]
=> nil
irb(main):003:0> pi1 = PersonalInfo.create height: 6.5, weight: 220
  (0.1ms) begin transaction
  SQL (0.3ms) INSERT INTO "personal_infos" ("height", "weight", "created_at", "updated_at") VALUES (?, ?, ?, ?) [["height", 6.5], ["weight", 220.0], ["created_at", "2015-09-08 23:39:09.207265"], ["updated_at", "2015-09-08 23:39:09.207265"]]
  (1.4ms) commit transaction
=> #<PersonalInfo id: 1, height: 6.5, weight: 220.0, person_id: nil, created_at: "2015-09-08 23:39:09", updated_at: "2015-09-08 23:39:09">
irb(main):004:0> bill.personal_info = pi1
  (0.1ms) begin transaction
  SQL (0.3ms) UPDATE "personal_infos" SET "person_id" = ?, "updated_at" = ? WHERE "personal_infos"."id" = ? [["person_id", 13], ["updated_at", "2015-09-08 23:39:32.492655"], ["id", 1]]
  (0.7ms) commit transaction
=> #<PersonalInfo id: 1, height: 6.5, weight: 220.0, person_id: 13, created_at: "2015-09-08 23:39:09", updated_at: "2015-09-08 23:39:32">
```

Person and PersonalInfo

- Al generar la relación se tienen disponibles en una instancia de Person los métodos:
 - `build_personal_info(hash)`: no crea un registro en la base de datos, solo construye la instancia.
 - `create_personal_info(hash)`: crea un registro en la base de datos
- Ambos eliminan la referencia previa existente en la base de datos.

Loading development environment (Rails 4.2.3)

irb(main):001:0> bill = Person.find_by last_name: "Gates"

Person Load (0.2ms) SELECT "people".* FROM "people" WHERE "people"."last_name" = ? LIMIT 1 [["last_name", "Gates"]]

=> #<Person id: 13, first_name: "Bill", age: 75, last_name: "Gates", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08 22:22:51", login: "bill">

>
irb(main):002:0> bill.personal_info

PersonalInfo Load (0.5ms) SELECT "personal_infos".* FROM "personal_infos" WHERE "personal_infos"."person_id" = ? LIMIT 1 [["person_id", 13]]

=> #<PersonalInfo id: 1, height: 6.5, weight: 220.0, person_id: 13, created_at: "2015-09-08 23:39:09", updated_at: "2015-09-08 23:39:32">

irb(main):003:0> bill.build_personal_info height: 6.0, weight: 180

(0.2ms) begin transaction

SQL (0.5ms) UPDATE "personal_infos" SET "person_id" = ?, "updated_at" = ? WHERE "personal_infos"."id" = ? [["person_id", nil], ["updated_at", "2015-09-10 23:08:29.192565"], ["id", 1]]

(0.7ms) commit transaction

=> #<PersonalInfo id: nil, height: 6.0, weight: 180.0, person_id: 13, created_at: nil, updated_at: nil>

irb(main):004:0> bill.save

(0.1ms) begin transaction

SQL (1.1ms) INSERT INTO "personal_infos" ("height", "weight", "person_id", "created_at", "updated_at") VALUES (?, ?, ?, ?, ?) [["height", 6.0], ["weight", 180.0], ["person_id", 13], ["created_at", "2015-09-10 23:08:29.192565"], ["updated_at", "2015-09-10 23:08:29.192565"]]

(1.4ms) commit transaction

=> true

irb(main):005:0> josh = Person.find_by first_name: "Josh"; josh.create_personal_info height: 5.5, weight: 135

Person Load (0.3ms) SELECT "people".* FROM "people" WHERE "people"."first_name" = ? LIMIT 1 [["first_name", "Josh"]]

(0.0ms) begin transaction

SQL (0.9ms) INSERT INTO "personal_infos" ("height", "weight", "person_id", "created_at", "updated_at") VALUES (?, ?, ?, ?, ?) [["height", 5.5], ["weight", 135.0], ["person_id", 11], ["created_at", "2015-09-10 23:09:36.391913"], ["updated_at", "2015-09-10 23:09:36.391913"]]

(1.4ms) commit transaction

PersonalInfo Load (0.1ms) SELECT "personal_infos".* FROM "personal_infos" WHERE "personal_infos"."person_id" = ? LIMIT 1 [["person_id", 11]]

=> #<PersonalInfo id: 3, height: 5.5, weight: 135.0, person_id: 11, created_at: "2015-09-10 23:09:36", updated_at: "2015-09-10 23:09:36">

Cómo quedaron los datos en la BD?

```
sqlite> select * from personal_infos;
```

id	height	weight	person_id	created_at	updated_at
1	6.5	220.0		2015-09-08 23:39:09.207265	2015-09-10 23:08:11.687362
2	6.0	180.0	13	2015-09-10 23:08:29.192565	2015-09-10 23:08:29.192565
3	5.5	135.0	11	2015-09-10 23:09:36.391913	2015-09-10 23:09:36.391913

```
sqlite> select * from people;
```

id	first_name	age	last_name	created_at	updated_at	login	pass
8	Kalman	33	Smith	2015-09-08 22:22:51.990586	2015-09-08 22:22:51.990586	kman	abc123
9	John	27	Whatever	2015-09-08 22:22:51.992746	2015-09-08 22:22:51.992746	john1	123abc
10	Michael	15	Smith	2015-09-08 22:22:51.994324	2015-09-08 22:22:51.994324	mike	not_tellin
11	Josh	57	Oreck	2015-09-08 22:22:51.995846	2015-09-08 22:22:51.995846	josh	password1
12	John	27	Smith	2015-09-08 22:22:51.997415	2015-09-08 22:22:51.997415	john2	no_idea
13	Bill	75	Gates	2015-09-08 22:22:51.999069	2015-09-08 22:22:51.999069	bill	windows3.1
14	LeBron	30	James	2015-09-08 22:22:52.000502	2015-09-08 22:22:52.000502	bron	need more

Asociación one-to-many

One-to-many

- Una **persona** tiene uno o varios trabajos (**job**)
- Un registro de job **pertenece** exactamente a una persona.
- El lado “**belongs to**” (pertenece) es el que tiene la **clave foránea**.


Convención: El nombre por defecto de la clave foránea es {master_table_singular}_id, ejemplo: person_id

Crear el model Job con su migration

```
~/advanced_ar$ rails g model job title company position_id person:references
  invoke  active_record
  create  db/migrate/20150922141356_create_jobs.rb
  create  app/models/job.rb
  invoke  test_unit
  create  test/models/job_test.rb
  create  test/fixtures/jobs.yml
~/advanced_ar$ rake db:migrate
== 20150922141356 CreateJobs: migrating =====
-- create_table(:jobs)
   -> 0.0020s
== 20150922141356 CreateJobs: migrated (0.0020s) =====
```

Crear el model Job con su migration

FOLDERS

▼  advanced_ar


▶  app

▶  bin

▶  config

▼  db

▼  migrate

 20150908214851_create_people.r

 20150908221446_add_login_pass

 20150908232650_create_persona

 20150922141356_create_jobs.rb



20150922141356_create_jobs.rb *

```
1 class CreateJobs < ActiveRecord::Migration
2   def change
3     create_table :jobs do |t|
4       t.string :title
5       t.string :company
6       t.string :position_id
7       t.references :person, index: true, foreign_key: true
8     end
9     t.timestamps null: false
10  end
11 end
12 end
```


Modificando los modelos de Person y Job

FOLDERS

- ▼ advanced_ar
 - ▼ app
 - ▶ assets
 - ▶ controllers
 - ▶ helpers
 - ▶ mailers
 - ▼ models
 - ▶ concerns
 - .keep
 - job.rb
 - person.rb
 - personal_info.rb

person.rb

```
1 class Person < ActiveRecord::Base
2   has_one :personal_info
3   has_many :jobs
4 end
5
```

job.rb

```
1 class Job < ActiveRecord::Base
2   belongs_to :person
3 end
4
5
6
```

Person y Job

```
~/advanced_ar$ rails c
Loading development environment (Rails 4.2.3)
irb(main):001:0> ActiveRecord::Base.logger = nil
=> nil
irb(main):002:0> Job.create company: "MS", title: "Developer", position_id: "#1234"
=> #<Job id: 1, title: "Developer", company: "MS", position_id: "#1234", person_id: nil, created_at: "2015-09-22 14:30:49", updated_at: "2015-09-22 14:30:49">
>
irb(main):003:0> p1 = Person.first
=> #<Person id: 8, first_name: "Kalman", age: 33, last_name: "Smith", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08 22:22:51", ss: "abc123">
irb(main):004:0> p1.jobs
=> #<ActiveRecord::Associations::CollectionProxy []>
irb(main):005:0> p1.jobs << Job.first
=> #<ActiveRecord::Associations::CollectionProxy [#<Job id: 1, title: "Developer", company: "MS", position_id: "#1234", person_id: nil, created_at: "2015-09-22 14:30:49", updated_at: "2015-09-22 14:31:45">]>
irb(main):006:0> Job.first.person
=> #<Person id: 8, first_name: "Kalman", age: 33, last_name: "Smith", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08 22:22:51", ss: "abc123">
irb(main):007:0> █
```

Mas métodos

- **person.jobs = jobs**
 - Reemplaza los jobs existentes con un nuevo array
 - A diferencia de `person.jobs << job(s)` donde los jobs son agregados
- **person.jobs.clear**
 - Desasocia los jobs del registro de person seteando el foreign key a null
- Los métodos **create** y **where** para jobs son un **scope** de person.

Scoped Jobs (rake db:seed)



The image shows a code editor interface with a file explorer on the left and a code editor on the right. The file explorer, titled "FOLDERS", shows a directory structure with folders like "advanced_ar", "app", "bin", "config", "db", "lib", "log", and "public". The "db" folder is expanded, showing files "migrate", "development.sqlite3", "schema.rb", and "seeds.rb". The "seeds.rb" file is selected and its contents are displayed in the code editor. The code defines three scopes for the Person model: "Person.destroy_all", "Person.create!" (with two records), "Person.first.jobs.create!" (with two records), and "Person.last.jobs.create!" (with two records).

```
seeds.rb
1 Person.destroy_all
2
3 Person.create! [
11 ]
12
13 Person.first.jobs.create! [
14   { title: "Developer", company: "MS", position_id: "#1234" },
15   { title: "Developer", company: "MS", position_id: "#1235" }
16 ]
17
18 Person.last.jobs.create! [
19   { title: "Sr. Developer", company: "MS", position_id: "#5234" },
20   { title: "Sr. Developer", company: "MS", position_id: "#5235" }
21 ]
```

Scoped Jobs: Where

```
~/advanced_ar$ rails c
Loading development environment (Rails 4.2.3)
irb(main):001:0> ActiveRecord::Base.logger = nil
=> nil
irb(main):002:0> Person.first.jobs.where(company: "MS").count
=> 2
irb(main):003:0> Person.last.jobs.where(company: "MS").count
=> 2
irb(main):004:0> Person.last.jobs.where(company: "MS").to_a
=> [#<Job id: 4, title: "Sr. Developer", company: "MS", position_id: "#5234", person_id: 21, created_at: "2013-03-19T19:39:19">, #<Job id: 5, title: "Sr. Developer", company: "MS", position_id: "#5235", person_id: 21, created_at: "2013-03-19T19:39:19">]
irb(main):005:0> █
```

Cambio de nombre del campo

FOLDERS

- ▼ advanced_ar
 - ▼ app
 - ▶ assets
 - ▶ controllers
 - ▶ helpers
 - ▶ mailers
 - ▼ models
 - ▶ concerns
 - .keep

person.rb

```
1 class Person < ActiveRecord::Base
2   has_one :personal_info
3   has_many :jobs
4   has_many :my_jobs, class_name: "Job"
5 end
6
```












```
~/advanced_ar$ rails c
Loading development environment (Rails 4.2.3)
irb(main):001:0> Person.first.my_jobs
Person Load (0.1ms) SELECT "people".* FROM "people" ORDER BY "people"."id" ASC LIMIT 1
Job Load (0.1ms) SELECT "jobs".* FROM "jobs" WHERE "jobs"."person_id" = ? [["person_id", 15]]
=> #<ActiveRecord::Associations::CollectionProxy [#<Job id: 2, title: "Developer", company: "MS", position_id: "#1234", person_id: 15,
2 14:39:19", updated_at: "2015-09-22 14:39:19">, #<Job id: 3, title: "Developer", company: "MS", position_id: "#1235", person_id: 15,
14:39:19", updated_at: "2015-09-22 14:39:19">]>
```

:dependent / Cascade

- **has_many**, **has_one** y **belongs_to** soportan la opción :dependent que permite especificar el destino de la asociación cuando se destruye el padre.
- **:delete** – eliminar los objetos asociados
- **:destroy** – lo mismo que el anterior, pero elimina la asociación llamando al método destroy.
- **:nullify** – setea el FK a NULL

:dependent - Ejemplo

FOLDERS

- ▼  advanced_ar
- ▼  app
 - ▶  assets
 - ▶  controllers
 - ▶  helpers
 - ▶  mailers
 - ▼  models
 - ▶  concerns
 -  .keep
 -  job.rb
 -  person.rb



person.rb



```
1  class Person < ActiveRecord::Base
2      has_one :personal_info, dependent: :destroy
3      has_many :jobs
4      has_many :my_jobs, class_name: "Job"
5  end
6
```


:dependent - Ejemplo

```
irb(main):001:0> mike = Person.find_by first_name: "Michael"
  Person Load (0.2ms) SELECT "people".* FROM "people" WHERE "people"."first_name" = ? LIMIT 1 [["first_name", "Michael"]]
=> #<Person id: 31, first_name: "Michael", age: 15, last_name: "Smith", created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15",
pass: "not_telling">
irb(main):002:0> mike.personal_info
  PersonalInfo Load (0.1ms) SELECT "personal_infos".* FROM "personal_infos" WHERE "personal_infos"."person_id" = ? LIMIT 1 [["person_id", 31]]
=> #<PersonalInfo id: 13, height: 5.5, weight: 200.0, person_id: 31, created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15",
pass: "not_telling">
irb(main):003:0> mike.destroy
  (0.2ms) begin transaction
  SQL (0.6ms) DELETE FROM "personal_infos" WHERE "personal_infos"."id" = ? [["id", 13]]
  SQL (0.1ms) DELETE FROM "people" WHERE "people"."id" = ? [["id", 31]]
  (1.4ms) commit transaction
=> #<Person id: 31, first_name: "Michael", age: 15, last_name: "Smith", created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15",
pass: "not_telling">
irb(main):004:0> PersonalInfo.find 13
  PersonalInfo Load (0.2ms) SELECT "personal_infos".* FROM "personal_infos" WHERE "personal_infos"."id" = ? LIMIT 1 [["id", 13]]
ActiveRecord::RecordNotFound: Couldn't find PersonalInfo with 'id'=13
```

Asociaciones Many-to-Many

Many-to-Many

- Una persona puede tener **muchos** hobbies
- Un hobby puede ser compartido por **muchas** personas
- Para definir este tipo de asociaciones se utiliza: habtm (**has_and_belongs_to_many**).
- Se necesita una tabla extra para el join (**sin un modelo**, solo un migration)

Convención: los nombres de los modelos en plural separados por un guión bajo en orden alfabético

Hobbies y Hobbies_People

```
~/advanced_ar$ rails g model hobby name
  invoke  active_record
  create  db/migrate/20150922154738_create_hobbies.rb
  create  app/models/hobby.rb
  invoke  test_unit
  create  test/models/hobby_test.rb
  create  test/fixtures/hobbies.yml
~/advanced_ar$ rails g migration create_hobbies_people person:references hobby:references
  invoke  active_record
  create  db/migrate/20150922154813_create_hobbies_people.rb
```

Hobbies_People Migration

FOLDERS

▼ advanced_ar

▶ app

▶ bin

▶ config

▼ db

▼ migrate

20150908214851_create_people.rb

20150908221446_add_login_pass_to_people

20150908232650_create_personal_infos.rb

20150922141356_create_jobs.rb

20150922154738_create_hobbies.rb

20150922154813_create_hobbies_people.rb



20150922154813_create_hobbies_people.rb *

```
1 class CreateHobbiesPeople < ActiveRecord::Migration
2   def change
3     create_table :hobbies_people, id: false do |t|
4       t.references :person, index: true, foreign_key: true
5       t.references :hobby, index: true, foreign_key: true
6     end
7   end
8 end
9
```

```
~/advanced_ar$ rake db:migrate
```

```
== 20150922154738 CreateHobbies: migrating ==
```

```
-- create_table(:hobbies)
```

```
-> 0.0009s
```

```
== 20150922154738 CreateHobbies: migrated (0.0010s) ==
```

```
== 20150922154813 CreateHobbiesPeople: migrating ==
```

```
-- create_table(:hobbies_people, {:id=>false})
```

```
-> 0.0014s
```

```
== 20150922154813 CreateHobbiesPeople: migrated (0.0014s) ==
```

Many-to-Many en la DB

```
~/advanced_ar$ rails db
SQLite version 3.8.5 2014-08-15 22:37:57
Enter ".help" for usage hints.
sqlite> .schema %hobbies%
CREATE TABLE "hobbies" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "name" varchar, "created_at" datetime NOT NULL, "updated_at" datetime NOT NULL);
CREATE TABLE "hobbies_people" ("person_id" integer, "hobby_id" integer);
CREATE INDEX "index_hobbies_people_on_person_id" ON "hobbies_people" ("person_id");
CREATE INDEX "index_hobbies_people_on_hobby_id" ON "hobbies_people" ("hobby_id");
```

Person and Hobby Models

FOLDERS

- ▼ advanced_ar
- ▼ app
 - ▶ assets
 - ▶ controllers
 - ▶ helpers
 - ▶ mailers
 - ▼ models
 - ▶ concerns
 - .keep
 - hobby.rb
 - job.rb
 - person.rb

person.rb

```
1 class Person < ActiveRecord::Base
2   has_one :personal_info, dependent: :destroy
3   has_many :jobs
4   has_many :my_jobs, class_name: "Job"
5   has_and_belongs_to_many :hobbies
6 end
7
```

hobby.rb

```
1 class Hobby < ActiveRecord::Base
2   has_and_belongs_to_many :people
3 end
4
```


How-to Queries

```
irb(main):002:0> josh = Person.find_by first_name: "Josh"
=> #<Person id: 32, first_name: "Josh", age: 57, last_name: "Oreck", created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15", login: "josh", pass: "password1">

irb(main):003:0> lebron = Person.find_by first_name: "LeBron"
=> #<Person id: 35, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15", login: "bron", pass: "need more rings">

irb(main):004:0> programming = Hobby.create name: "Programming"
=> #<Hobby id: 3, name: "Programming", created_at: "2015-09-22 16:18:52", updated_at: "2015-09-22 16:18:52">

irb(main):005:0> josh.hobbies << programming; lebron.hobbies << programming
=> #<ActiveRecord::Associations::CollectionProxy [#<Hobby id: 3, name: "Programming", created_at: "2015-09-22 16:18:52", updated_at: "2015-09-22 16:18:52">]>

irb(main):006:0> programming.people
=> #<ActiveRecord::Associations::CollectionProxy [#<Person id: 32, first_name: "Josh", age: 57, last_name: "Oreck", created_at: "2015-09-22 15:06:15", login: "josh", pass: "password1">, #<Person id: 35, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-22 15:06:15", login: "bron", pass: "need more rings">]>
```


Entonces

- **ONE-TO-ONE:** `has_one` / `belongs_to` (y una `columna integer` en la BD) es todo lo que se necesita para establecer una asociación one-to-one.
- **ONE-TO-MANY:** utiliza `has_many` y `belongs_to`.
 - Gestiona las relaciones “huérfanas” especificando la opción `:dependent` en la asociación principal
- **MANY-TO-MANY:** cuenta con 2 modelos y 3 migrations
 - La tabla join debe existir en la base de datos, pero no en código ruby (modelo)

Rich many to many associations

Rich many-to-many association

- A veces, necesitamos mantener algunos datos en la tabla join de una relación many-to-many
- O necesitamos almacenar grandchild relationships (tablas nietas) en un modelo.
- Por ejemplo: user → articles → comments
- En nuestro caso, todos los salary-ranges (rangos de salario) para una persona particular.
 - Es decir: una persona tiene varios trabajos (jobs) y cada trabajo tiene un y solo un rango salarial. Por ende, lo que queremos saber es cuales son los rangos de salario de una persona.

Rich Many-to-Many Association

- ActiveRecord provee una opción **:through** para este propósito.
- La idea básica es:
 - Primero se **crea** una relación regular **padre-hijo**
 - Luego se **utiliza al modelo hijo para como “join”** entre el padre y el nieto.

Salary Range / Model and Migration

```
~/advanced_ar$ rails g model salary_range min_salary:float max_salary:float job:references
  invoke  active_record
  create  db/migrate/20150922165025_create_salary_ranges.rb
  create  app/models/salary_range.rb
  invoke  test_unit
  create  test/models/salary_range_test.rb
  create  test/fixtures/salary_ranges.yml
```

```
~/advanced_ar$ rake db:migrate
```

```
== 20150922165025 CreateSalaryRanges: migrating =====
```

```
-- create_table(:salary_ranges)
```

```
-> 0.0014s
```

```
== 20150922165025 CreateSalaryRanges: migrated (
```

```
class CreateSalaryRanges < ActiveRecord::Migration
  def change
    create_table :salary_ranges do |t|
      t.float :min_salary
      t.float :max_salary
      t.references :job, index: true, foreign_key: true

      t.timestamps null: false
    end
  end
end
```

Job and SalaryRange models

```
job.rb
1 class Job < ActiveRecord::Base
2   belongs_to :person
3   has_one :salary_range
4 end

salary_range.rb
1 class SalaryRange < ActiveRecord::Base
2   belongs_to :job
3 end
4
```

Person and SalaryRange Models

FOLDERS

▼ advanced_ar

▼ app

▶ assets

▶ controllers

▶ helpers

▶ mailers

▼ models

▶ concerns

📄 .keep

📄 hobby.rb

📄 job.rb

📄 person.rb

person.rb

```
1 class Person < ActiveRecord::Base
2   has_one :personal_info, dependent: :destroy
3   has_many :my_jobs, class_name: "Job"
4   has_and_belongs_to_many :hobbies
5   has_many :jobs
6   has_many :approx_salaries, through: :jobs, source: :salary_range
7 end
```

job.rb

```
1 class Job < ActiveRecord::Base
2   belongs_to :person
3   has_one :salary_range
4 end
```



Rangos de salario por persona

```
irb(main):002:0> lebron = Person.find_by(first_name: "LeBron")
=> #<Person id: 35, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15", ass: "need more rings">
irb(main):003:0> lebron.jobs.count
=> 2
irb(main):004:0> lebron.jobs.pluck(:id)
=> [12, 13]
irb(main):005:0> Job.find(12).create_salary_range(min_salary: 10000.00, max_salary: 20000.00)
=> #<SalaryRange id: 1, min_salary: 10000.0, max_salary: 20000.0, job_id: 12, created_at: "2015-09-22 17:25:01", updated_at: "2015-09-22 17:25:01">
irb(main):006:0> Job.find(13).create_salary_range(min_salary: 15000.00, max_salary: 35000.00)
=> #<SalaryRange id: 2, min_salary: 15000.0, max_salary: 35000.0, job_id: 13, created_at: "2015-09-22 17:25:33", updated_at: "2015-09-22 17:25:33">
irb(main):007:0> lebron.approx_salaries
=> #<ActiveRecord::Associations::CollectionProxy [#<SalaryRange id: 1, min_salary: 10000.0, max_salary: 20000.0, job_id: 12, created_at: "2015-09-22 17:25:01", updated_at: "2015-09-22 17:25:01">, #<SalaryRange id: 2, min_salary: 15000.0, max_salary: 35000.0, job_id: 13, created_at: "2015-09-22 17:25:33", updated_at: "2015-09-22 17:25:33">]>
```


ActiveRecord Calculations

FOLDERS

- ▼ advanced_ar
 - ▼ app
 - ▶ assets
 - ▶ controllers
 - ▶ helpers
 - ▶ mailers
 - ▼ models
 - ▶ concerns
 - .keep
 - hobby.rb
 - job.rb
 - person.rb

```
person.rb
1  class Person < ActiveRecord::Base
2    has_one :personal_info, dependent: :destroy
3    has_many :my_jobs, class_name: "Job"
4    has_and_belongs_to_many :hobbies
5    has_many :jobs
6    has_many :approx_salaries, through: :jobs, source: :salary_range
7
8    def max_salary
9      approx_salaries.maximum(:max_salary)
10   end
11 end
12
```

El máximo salario de una persona..

```
irb(main):001:0> lebron = Person.find_by last_name: "James"
  Person Load (0.2ms) SELECT "people".* FROM "people" WHERE "people"."last_name" = ? LIMIT 1
[["last_name", "James"]]
=> #<Person id: 35, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15", login: "bron", pass: "need more rings">
irb(main):002:0> lebron.max_salary
  (0.2ms) SELECT MAX("salary_ranges"."max_salary") FROM "salary_ranges" INNER JOIN "jobs" ON "salary_ranges"."job_id" = "jobs"."id" WHERE "jobs"."person_id" = ? [["person_id", 35]]
=> 35000.0
```

Entonces...

Se puede obtener buenos resultados utilizando `:through`

Los cálculos de ActiveRecord hacen que la base de datos haga el trabajo duro.