Ruby on Rails

Introducción a Active Record

Overview

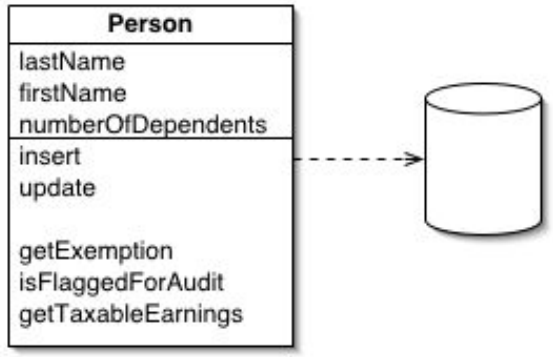
- El ORM Active Record
- Las convenciones de ActiveRecord.
- ActiveRecord CRUD
  - Create
  - Retrieve
  - Update
  - Delete

ORM

- ActiveRecord es el ORM por defecto de Rails.
- Un ORM es un mapeo entre el modelo relacional de la base de datos y los lenguajes orientados a objetos que tratan con objetos y sus comportamientos.

Patrón de Diseño

- Active Record fue definido como patrón de diseño por Martin Fowler.
- Encapsula la estructura del modelo y su comportamiento.



ActiveRecord

```
1  class Car < ActiveRecord::Base
2  end
3
4
5
```

¿Donde está todo el código?

Metaprogramación

+

Convenciones de código

ActiveRecord

- Configuración del **acceso a la base de datos**: config/database.yml
- Convención: existe una tabla con el nombre en **plural** que se corresponde con la subclase de ActiveRecord. (Car<ActiveRecord::Base --- cars)
- Convención: ActiveRecord espera que toda tabla tenga una **PK** con nombre **id**.

Rails Console : rails c

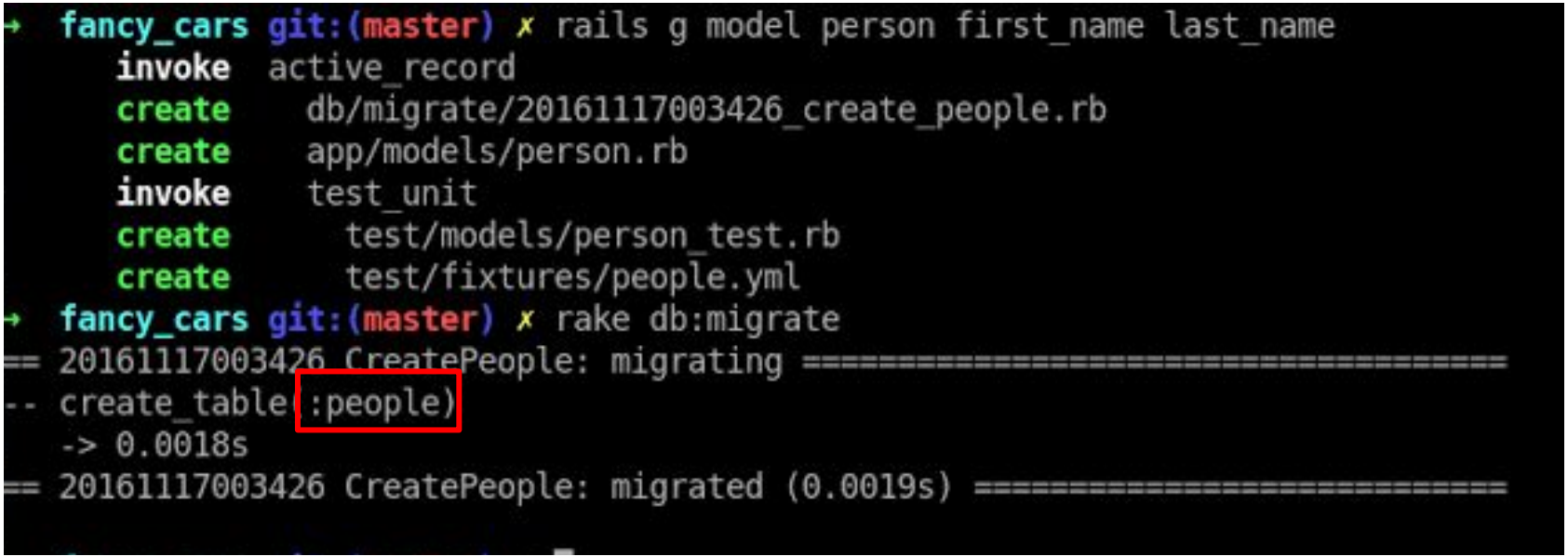
IRB con esteroides cuando nuestra aplicación rails está cargada!

```
→  ~ cd git/capacitaciones/ruby-on-rails-intro/modulo-4/sources/fancy_cars
→  fancy_cars git:(master) ✗ rails c
Loading development environment (Rails 4.0.4)
irb(main):001:0> Car.column_names
=> ["id", "company", "color", "year", "created_at", "updated_at", "price", "birth_date", "model"]
irb(main):002:0>
```

Los métodos de clase lidian con las tablas enteras, mientras que los métodos de instancia con una fila en particular.

Models and Migrations

Ya estudiamos el scaffold generator y el migration generator, pero resulta de **model** tiene su propio generator, que también genera un **migration**.



Pluralización del nombre de la tabla

inflections.rb

```
1   # Be sure to restart your server when you modify this file.
2
3   # Add new inflection rules using the following format. Inflections
4   # are locale specific, and you may define rules for as many different
5   # locales as you wish. All of these examples are active by default:
6   # ActiveSupport::Inflector.inflections(:en) do |inflect|
7   #   inflect.plural /^(ox)$/i, '\1en'
8   #   inflect.singular /^(ox)en/i, '\1'
9   #   inflect.irregular 'person', 'people'
10  #   inflect.uncountable %w( fish sheep )
11  # end
12
13  # These inflection rules are supported but not enabled by default:
14  # ActiveSupport::Inflector.inflections(:en) do |inflect|
15  #   inflect.acronym 'RESTful'
16  # end
17
```

Recargando la consola rails

```
irb(main):008:0> begin
irb(main):009:1* Person.column_names
irb(main):010:1> rescue Exception => e
irb(main):011:1> puts "Exception: #{e.message}"
irb(main):012:1> end
Exception: Could not find table 'people'
=> nil
irb(main):013:0> reload!
Reloading...
=> true
irb(main):014:0> Person.column_names
=> ["id", "first_name", "last_name", "created_at", "updated_at"]
irb(main):015:0>
```

Después de un rake db:migrate

Entonces...

- Convenciones de ActiveRecord:
  - El nombre de la clase es **singular**
  - El nombre de la tabla es **plural**
  - Se necesita tener un **id** que será primary key.

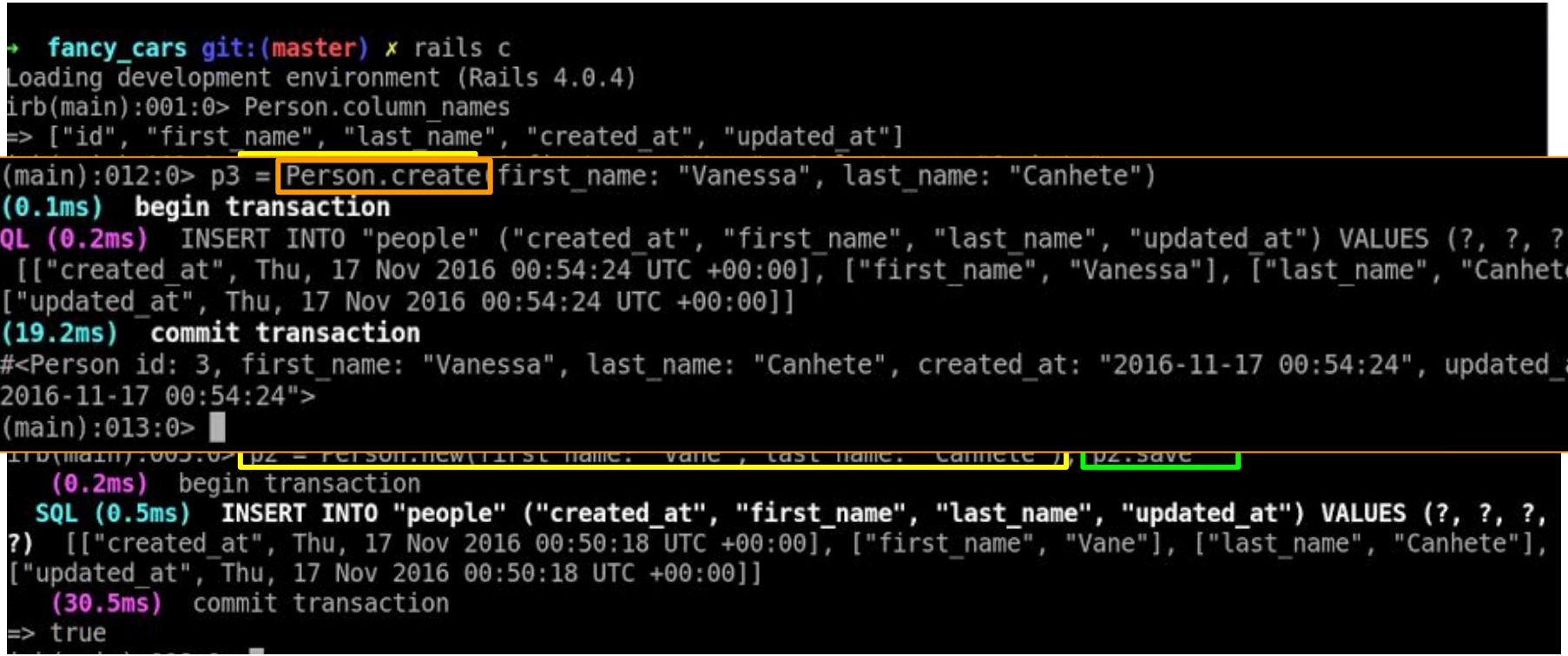
Operaciones con ActiveRecord

Create (CRUD)

Existen tres formas de **crear un registro** en la base de datos:

1. Utilizando un **constructor vacío** y **atributos fantasmas** para setear valores y luego llamar a `save`.
2. Pasar como argumento un **hash de atributos** al constructor y luego llamar a `save`.
3. Utilizar el método `create` con un **hash de atributos** para crear un objeto y guardarlo en la base de datos en **un solo paso**.

Ejemplo


fancy_cars git:(master) ✗ rails c
Loading development environment (Rails 4.0.4)
irb(main):001:0> Person.column_names
=> ["id", "first_name", "last_name", "created_at", "updated_at"]
irb(main):012:0> p3 = Person.create(first_name: "Vanessa", last_name: "Canhete")
(0.1ms) begin transaction
SQL (0.2ms) INSERT INTO "people" ("created_at", "first_name", "last_name", "updated_at") VALUES (?, ?, ?, ?) [["created_at", Thu, 17 Nov 2016 00:54:24 UTC +00:00], ["first_name", "Vanessa"], ["last_name", "Canhete"], ["updated_at", Thu, 17 Nov 2016 00:54:24 UTC +00:00]]
(19.2ms) commit transaction
=> #<Person id: 3, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 00:54:24", updated_at: "2016-11-17 00:54:24">
irb(main):013:0>
(0.2ms) begin transaction
SQL (0.5ms) INSERT INTO "people" ("created_at", "first_name", "last_name", "updated_at") VALUES (?, ?, ?, ?) [["created_at", Thu, 17 Nov 2016 00:50:18 UTC +00:00], ["first_name", "Vane"], ["last_name", "Canhete"], ["updated_at", Thu, 17 Nov 2016 00:50:18 UTC +00:00]]
(30.5ms) commit transaction
=> true

Retrieve / Read (CRUD)

- find(id) o find(id1, id2)
  - Lanza una excepción **RecordNotFound** si no encuentra datos.
- first, last, take, all
  - Retornan los resultados esperados o nil si no encuentra datos.
- order(:column) o order(column: :desc)
  - Permite el ordenamiento de los resultados de forma Ascendente o Descendente.
- pluck
  - Permite disminuir la cantidad de campos que serán retornados por la base de datos.
  - Se necesita llamarlo siempre al final.

```
irb(main):006:0> Person.all.order(first_name: :desc)
  Person Load (0.2ms)  SELECT "people".* FROM "people" ORDER BY "people"."first_name" DESC
=> #<ActiveRecord::Relation [#<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">, #<Person id: 5, first_name: "Iván", last_name: "Perez", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">, #<Person id: 6, first_name: "Gabriela", last_name: "Gaona", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">]>
irb(main):007:0> Person.all.order(first_name: :desc).to_a
  Person Load (0.2ms)  SELECT "people".* FROM "people" ORDER BY "people"."first_name" DESC
=> [#<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">, #<Person id: 5, first_name: "Iván", last_name: "Perez", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">, #<Person id: 6, first_name: "Gabriela", last_name: "Gaona", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">]
irb(main):008:0> Person.first
  Person Load (0.2ms)  SELECT "people".* FROM "people" ORDER BY "people"."id" ASC LIMIT 1
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">
irb(main):009:0> Person.all.first
  Person Load (0.3ms)  SELECT "people".* FROM "people" ORDER BY "people"."id" ASC LIMIT 1
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">
irb(main):010:0> Person.all[0]
  Person Load (0.2ms)  SELECT "people".* FROM "people"
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at: "2016-11-17 01:03:46">
irb(main):011:0>
```

Take and Pluck

```
irb(main):016:0> Person.take
  Person Load (0.4ms)  SELECT "people".* FROM "people" LIMIT 1
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at
: "2016-11-17 01:03:46">
irb(main):017:0> Person.take 2
  Person Load (0.1ms)  SELECT "people".* FROM "people" LIMIT 2
=> [#<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_a
t: "2016-11-17 01:03:46">, #<Person id: 5, first_name: "Iván", last_name: "Perez", created_at: "2016-11-17 01
:03:46", updated_at: "2016-11-17 01:03:46">]
irb(main):018:0> Person.all.map{|person| person.first_name}
  Person Load (0.3ms)  SELECT "people".* FROM "people"
=> ["Vanessa", "Iván", "Gabriela"]
irb(main):019:0> Person.pluck(:first_name)
   (0.2ms)  SELECT "people"."first_name" FROM "people"
=> ["Vanessa", "Iván", "Gabriela"]
irb(main):020:0>
```

Retrieve / Read (CRUD)

- `where(hash)`
  - Permite proveer condiciones para la búsqueda
  - Retorna un `ActiveRecord::Relation` (lo mismo que `all`), pero siempre se puede **segregar** utilizando `first` o tratarlo como un `Array`.

Where

```
irb(main):020:0> Person.where(last_name: "Canhete")
  Person Load (0.1ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'Canhete'
=> #<ActiveRecord::Relation [#<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-1
1-17 01:03:46", updated_at: "2016-11-17 01:03:46">]>
irb(main):021:0> Person.where(last_name: "Canhete").first
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'Canhete' ORDER BY "peopl
e"."id" ASC LIMIT 1
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at
: "2016-11-17 01:03:46">
irb(main):022:0> Person.where(last_name: "Canhete")[0]
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'Canhete'
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at
: "2016-11-17 01:03:46">
irb(main):023:0> Person.where(last_name: "Canhete").pluck(:first_name)
   (0.2ms)  SELECT "people"."first_name" FROM "people" WHERE "people"."last_name" = 'Canhete'
=> ["Vanessa"]
```

Find_by

- `find_by(conditions_hash)`
  - Lo mismo que `where`, pero retorna un solo resultado o nil si no se encontró un registro con esas condiciones.
- `find_by!(conditions_hash)`
  - Lo mismo que el `find_by`, pero lanza una excepción en caso de no encontrar el registro.

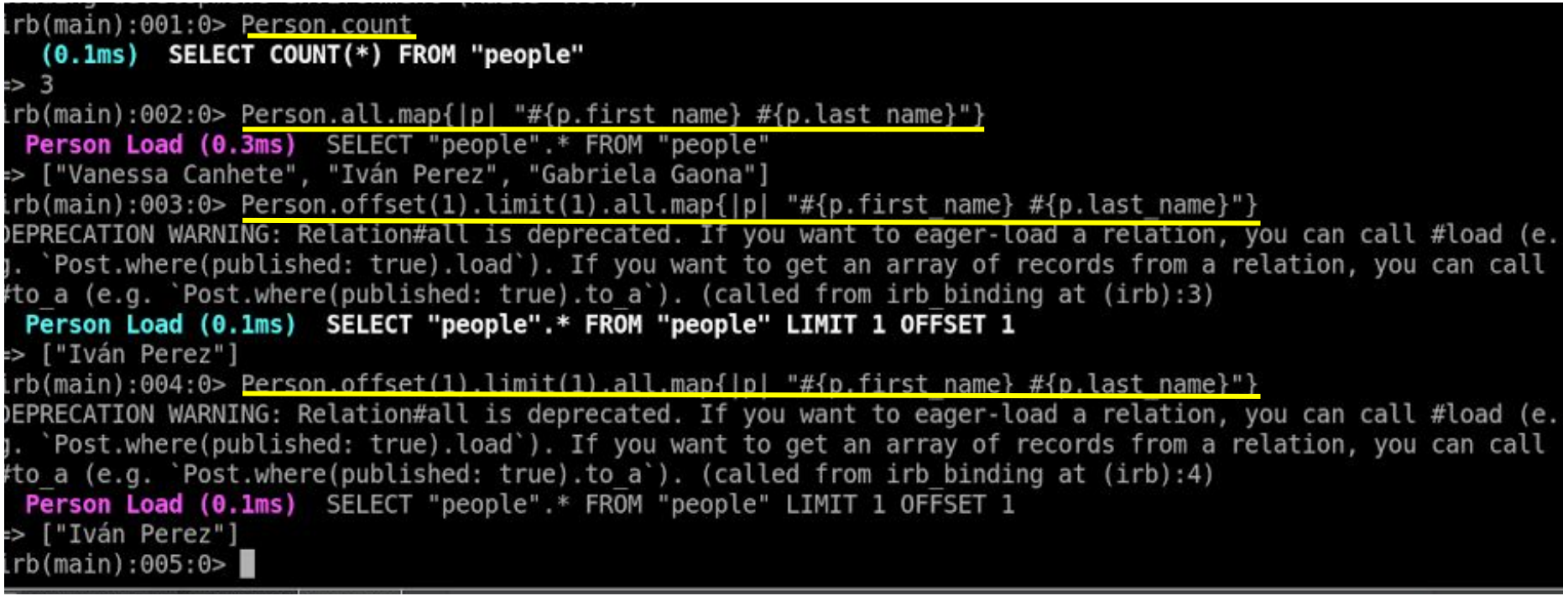
Find_by

```
irb(main):024:0> Person.find_by(last_name: "Canhete")
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'Canhete' LIMIT 1
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at
: "2016-11-17 01:03:46">
irb(main):025:0> Person.where(last_name: "Canhete")
  Person Load (0.1ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'Canhete'
=> #<ActiveRecord::Relation [#<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-1
1-17 01:03:46", updated_at: "2016-11-17 01:03:46">]>
irb(main):026:0> Person.find_by(last_name: "SomeLastName")
  Person Load (0.3ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'SomeLastName' LIMIT 1
=> nil
irb(main):027:0> Person.find_by!(last_name: "SomeLastName")
  Person Load (0.4ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'SomeLastName' LIMIT 1
ActiveRecord::RecordNotFound: ActiveRecord::RecordNotFound
        from /home/vanessa/.rbenv/versions/2.0.0-p481/lib/ruby/gems/2.0.0/gems/activerecord-4.0.4/lib/active_
```

Consola nº 2 | Consola | Consola nº 3 | Consola nº 4

limit / offset

- limit(n)
  - Permite limitar la cantidad de registros que devolverá la consulta.
- offset(n)
  - No comienza desde el inicio, salta algunos registros.
- Se puede **combinar** estos dos para paginar colecciones demasiado grandes de registros de la base de datos.


irb(main):001:0> Person.count
(0.1ms) SELECT COUNT(*) FROM "people"
=> 3
irb(main):002:0> Person.all.map{|p| "#{p.first_name} #{p.last_name}"}
Person Load (0.3ms) SELECT "people".* FROM "people"
=> ["Vanessa Canhete", "Iván Perez", "Gabriela Gaona"]
irb(main):003:0> Person.offset(1).limit(1).all.map{|p| "#{p.first_name} #{p.last_name}"}
DEPRECATION WARNING: Relation#all is deprecated. If you want to eager-load a relation, you can call #load (e.
g. `Post.where(published: true).load`). If you want to get an array of records from a relation, you can call
#to_a (e.g. `Post.where(published: true).to_a`). (called from irb_binding at (irb):3)
Person Load (0.1ms) SELECT "people".* FROM "people" LIMIT 1 OFFSET 1
=> ["Iván Perez"]
irb(main):004:0> Person.offset(1).limit(1).all.map{|p| "#{p.first_name} #{p.last_name}"}
DEPRECATION WARNING: Relation#all is deprecated. If you want to eager-load a relation, you can call #load (e.
g. `Post.where(published: true).load`). If you want to get an array of records from a relation, you can call
#to_a (e.g. `Post.where(published: true).to_a`). (called from irb_binding at (irb):4)
Person Load (0.1ms) SELECT "people".* FROM "people" LIMIT 1 OFFSET 1
=> ["Iván Perez"]
irb(main):005:0>

Update (CRUD)

- Existen dos formas de actualizar un registro en la base de datos:
  - Obtener el registro, modificar los valores y luego invocar a save.
  - Obtener el registro y luego invocar a update pasando un hash con los atributos y los nuevos valores.
- Existe también el método update_all para updates en batch:
  - Se puede encadenar esto al final de un where.

```
irb(main):005:0> vane = Person.find_by first_name: "Vanessa"
  Person Load (0.4ms)  SELECT "people".* FROM "people" WHERE "people"."first_name" = 'Vanessa' LIMIT 1
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at
: "2016-11-17 01:03:46">
irb(main):006:0> vane.last_name = "Cuevas"
=> "Cuevas"
irb(main):007:0> vane.save
   (0.2ms)  begin transaction
  SQL (5.4ms)  UPDATE "people" SET "last_name" = ?, "updated_at" = ? WHERE "people"."id" = 4  [["last_name",
"Cuevas"], ["updated_at", Thu, 17 Nov 2016 03:47:20 UTC +00:00]]
   (28.1ms)  commit transaction
=> true
irb(main):008:0> vane = Person.find(4)
  Person Load (0.3ms)  SELECT "people".* FROM "people" WHERE "people"."id" = ? LIMIT 1  [["id", 4]]
=> #<Person id: 4, first_name: "Vanessa", last_name: "Cuevas", created_at: "2016-11-17 01:03:46", updated_at:
 "2016-11-17 03:47:20">
irb(main):009:0> Person.find_by(last_name: "Cuevas").update(last_name: "Canhete")
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE "people"."last_name" = 'Cuevas' LIMIT 1
   (0.1ms)  begin transaction
  SQL (0.2ms)  UPDATE "people" SET "last_name" = ?, "updated_at" = ? WHERE "people"."id" = 4  [["last_name",
"Canhete"], ["updated_at", Thu, 17 Nov 2016 03:49:40 UTC +00:00]]
   (15.4ms)  commit transaction
=> true
irb(main):010:0>
```

Delete (CRUD)

- `destroy(id)` o `destroy`
  - Elimina una instancia particular de la base de datos.
  - Primer instancia un objeto luego realiza callbacks antes de la eliminación
  - Revisar: http://guides.rubyonrails.org/active_record_callbacks.html
- `delete(id)`
  - Elimina una fila de la base de datos.
- Existe también `delete_all`

```
irb(main):012:0> Person.count
   (0.1ms)  SELECT COUNT(*) FROM "people"
=> 3
irb(main):013:0> vane = Person.find_by_first_name: "Vanessa"
  Person Load (0.4ms)  SELECT "people".* FROM "people" WHERE "people"."first_name" = 'Vanessa' LIMIT 1
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at
: "2016-11-17 03:49:40">
irb(main):014:0> vane.destroy
   (0.1ms)  begin transaction
  SQL (0.2ms)  DELETE FROM "people" WHERE "people"."id" = ?  [["id", 4]]
   (28.9ms)  commit transaction
=> #<Person id: 4, first_name: "Vanessa", last_name: "Canhete", created_at: "2016-11-17 01:03:46", updated_at
: "2016-11-17 03:49:40">
irb(main):015:0> ivan = Person.find_by_first_name: "Iván"
  Person Load (0.4ms)  SELECT "people".* FROM "people" WHERE "people"."first_name" = 'Iván' LIMIT 1
=> #<Person id: 5, first_name: "Iván", last_name: "Perez", created_at: "2016-11-17 01:03:46", updated_at: "20
16-11-17 01:03:46">
irb(main):016:0> Person.delete(ivan.id)
  SQL (20.9ms)  DELETE FROM "people" WHERE "people"."id" = 5
=> 1
irb(main):017:0>
```

Entonces...

- ActiveRecord es muy intuitivo cuando se trata de interacción con la base de datos.
- Siempre hay que tener en mente si se está retornando un solo resultado (`find`) o un `ActiveRecord::Relation` (`where`)
- Update y Delete son simples de usar.
- Delete tiene:
  - “Ir directo a la base de datos” (`delete`)
  - Instanciar un objeto Ruby y dejarlo interactuar con la base de datos (`destroy`)