Introducción a Ruby on Rails

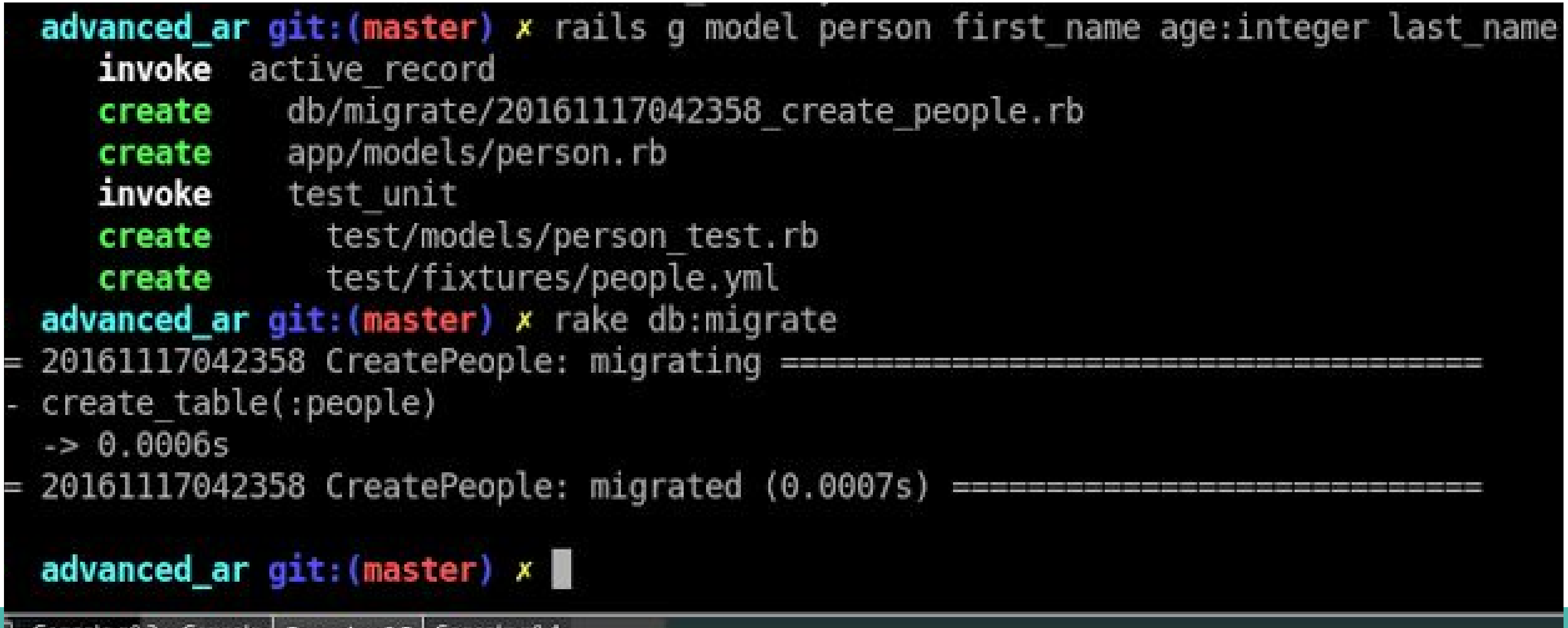
Seed / SQL / SQL Injection / SQL Literals

Overview

- Seed de la base de datos.
- Realizar consultas específicas incluyendo fragmentos SQL
- Los peligros del SQL Injection
- Alternativas para especificar Literales SQL:
    - Array Condition Syntax
    - Hash Condition Syntax

Aplicación de pruebas

- `rails new advanced_ar`
- `rails g model person first_name age:integer last_name`

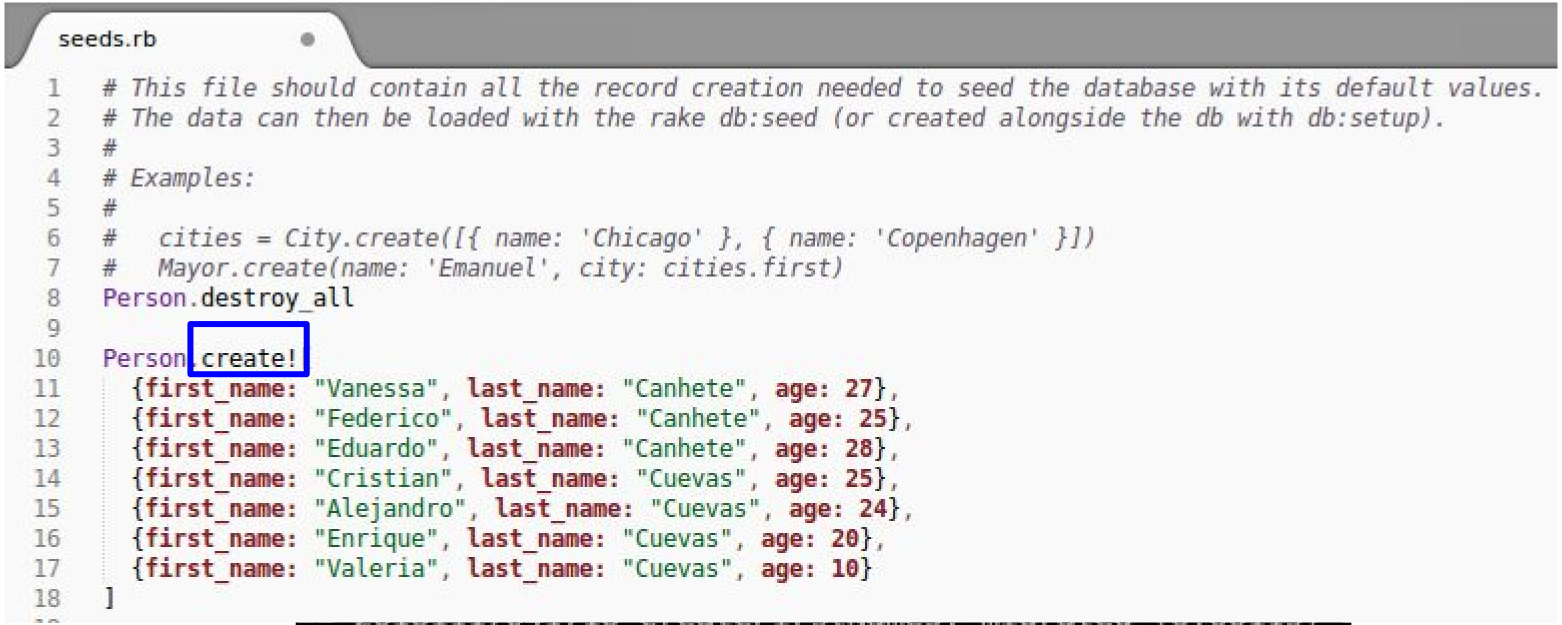


```
  advanced_ar git:(master) ✗ rails g model person first_name age:integer last_name
      invoke  active_record
      create    db/migrate/20161117042358_create_people.rb
      create    app/models/person.rb
      invoke    test_unit
      create      test/models/person_test.rb
      create      test/fixtures/people.yml
  advanced_ar git:(master) ✗ rake db:migrate
= 20161117042358 CreatePeople: migrating =====================================
- create_table(:people)
   -> 0.0006s
= 20161117042358 CreatePeople: migrated (0.0007s) ============================

  advanced_ar git:(master) ✗ █
```

db/seeds.rb

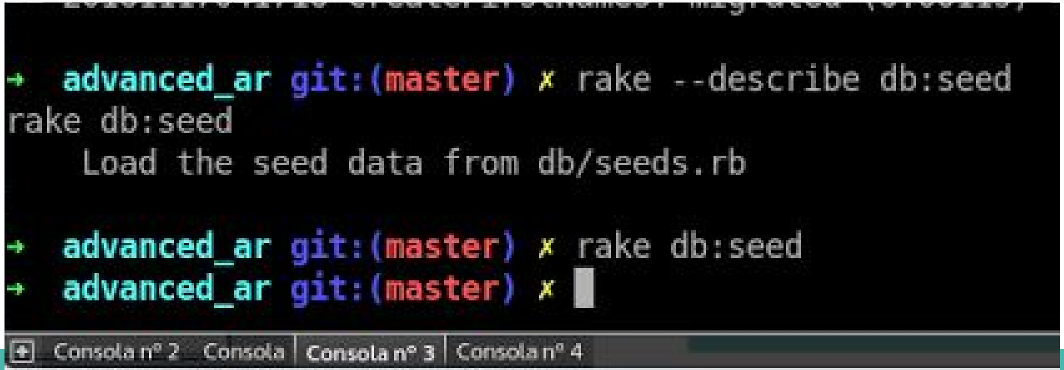
- Ya hemos visto cómo crear un modelo y su estructura en el migration, pero sería bueno saber cómo poblar la base de datos con datos de prueba.
- Rails provee db/seeds.rb para este propósito
- Para poblar la base de datos con algunos valores iniciales, solo se necesita correr: **rake db:seed**



seeds.rb

```ruby
# This file should contain all the record creation needed to seed the database with its default values.
# The data can then be loaded with the rake db:seed (or created alongside the db with db:setup).
#
# Examples:
#
#   cities = City.create([{ name: 'Chicago' }, { name: 'Copenhagen' }])
#   Mayor.create(name: 'Emanuel', city: cities.first)
Person.destroy_all

Person.create!
  {first_name: "Vanessa", last_name: "Canhete", age: 27},
  {first_name: "Federico", last_name: "Canhete", age: 25},
  {first_name: "Eduardo", last_name: "Canhete", age: 28},
  {first_name: "Cristian", last_name: "Cuevas", age: 25},
  {first_name: "Alejandro", last_name: "Cuevas", age: 24},
  {first_name: "Enrique", last_name: "Cuevas", age: 20},
  {first_name: "Valeria", last_name: "Cuevas", age: 10}
]
```

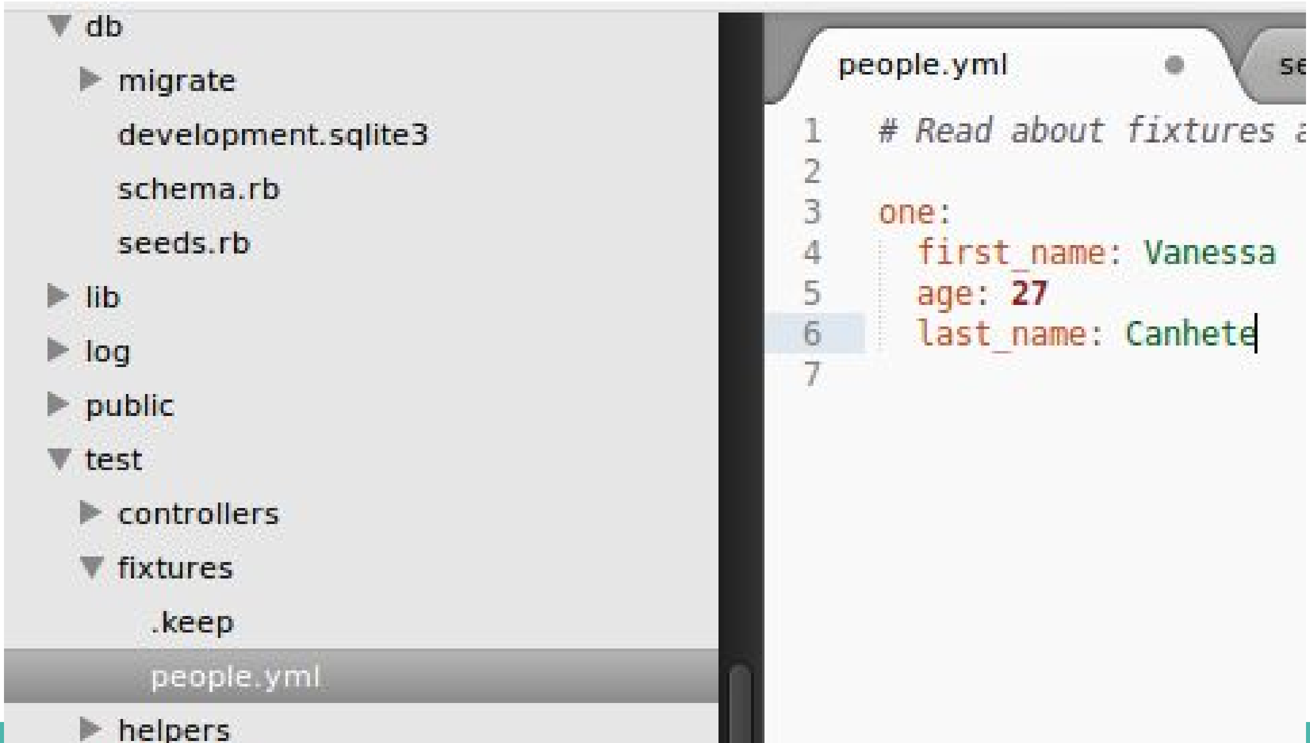


```
→  advanced_ar git:(master) ✗ rake --describe db:seed
rake db:seed
    Load the seed data from db/seeds.rb

→  advanced_ar git:(master) ✗ rake db:seed
→  advanced_ar git:(master) ✗ 
```

Seed utilizando Fixtures

```
ActiveRecord::Fixtures.create_fixtures("#{Rails.root}/test/fixtures/people")
```



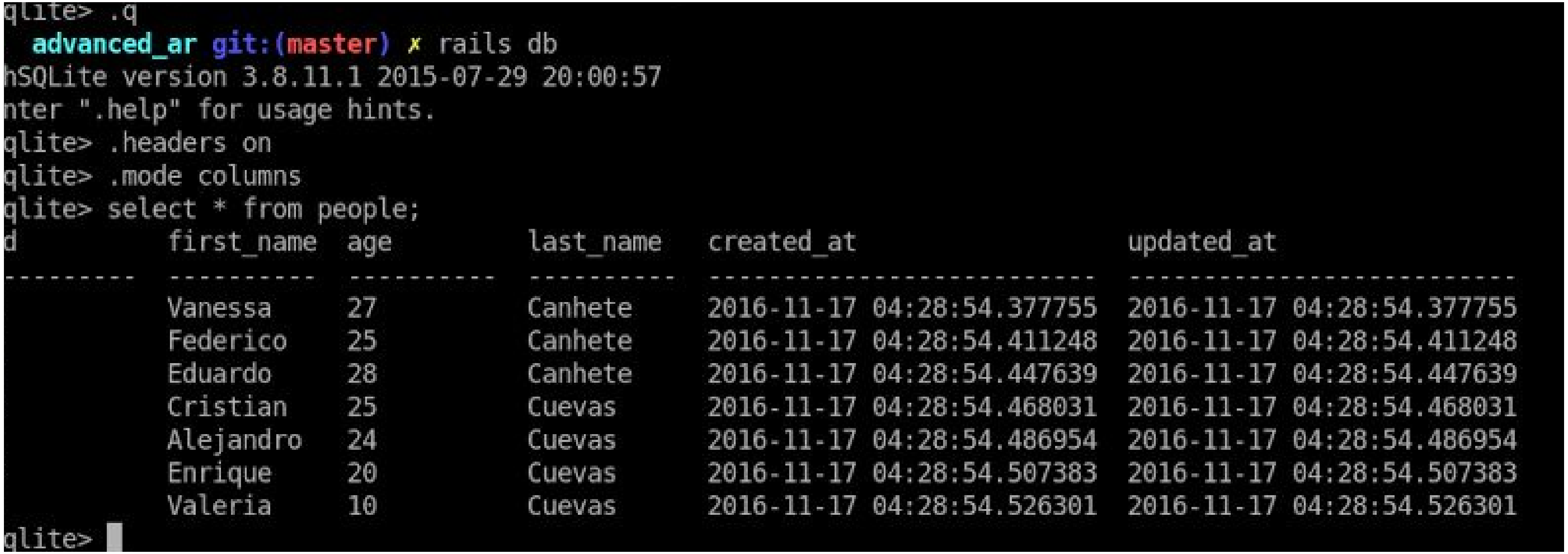
```
▼ db
    ▶ migrate
      development.sqlite3
      schema.rb
      seeds.rb
▶ lib
▶ log
▶ public
▼ test
    ▶ controllers
    ▼ fixtures
        .keep
        people.yml
    ▶ helpers
```

people.yml

```yaml
# Read about fixtures a

one:
  first_name: Vanessa
  age: 27
  last_name: Canhete

```



```
qlite> .q
  advanced_ar git:(master) ✗ rails db
hSQLite version 3.8.11.1 2015-07-29 20:00:57
nter ".help" for usage hints.
qlite> .headers on
qlite> .mode columns
qlite> select * from people;
d          first_name  age         last_name   created_at                  updated_at
---------  ----------  ----------  ----------  --------------------------  --------------------------
           Vanessa     27          Canhete     2016-11-17 04:28:54.377755  2016-11-17 04:28:54.377755
           Federico    25          Canhete     2016-11-17 04:28:54.411248  2016-11-17 04:28:54.411248
           Eduardo     28          Canhete     2016-11-17 04:28:54.447639  2016-11-17 04:28:54.447639
           Cristian    25          Cuevas      2016-11-17 04:28:54.468031  2016-11-17 04:28:54.468031
           Alejandro   24          Cuevas      2016-11-17 04:28:54.486954  2016-11-17 04:28:54.486954
           Enrique     20          Cuevas      2016-11-17 04:28:54.507383  2016-11-17 04:28:54.507383
           Valeria     10          Cuevas      2016-11-17 04:28:54.526301  2016-11-17 04:28:54.526301
qlite> 
```

Entonces..

- db/seeds.rb permite que creemos datos de prueba.
- Utilizar create! - de otra forma fallará silenciosamente.

Fragmentos SQL / Peligros del SQL Injection

Búsquedas exactas

Ya conocemos algunos métodos que nos permiten buscar registros en la base de datos:

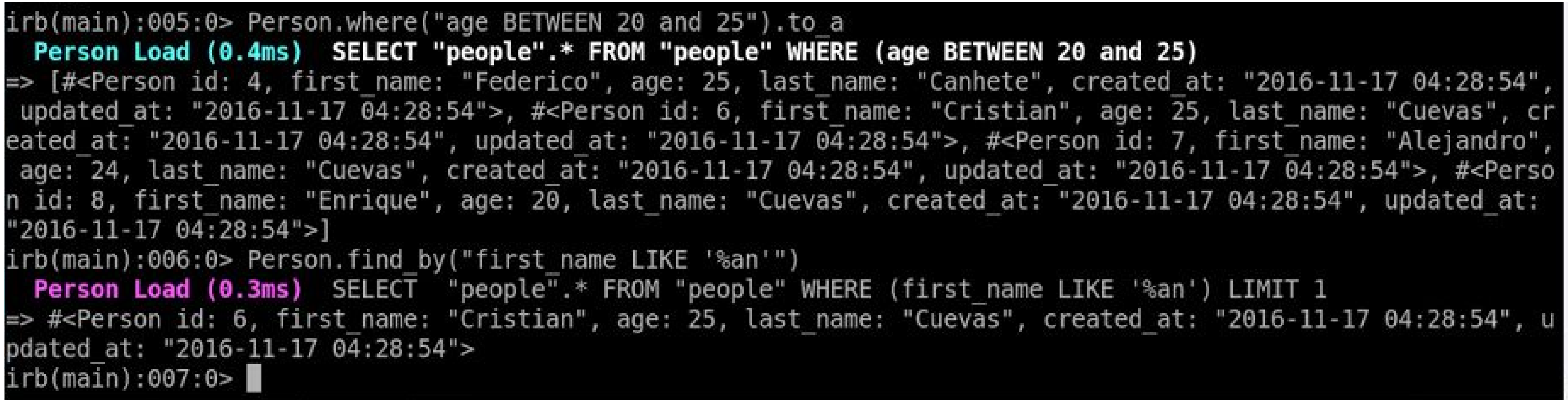
- `find(id)` o `find(id1, id2)`
- `find_by(hash)`
- `where(hash)`

Pero solamente son buenos si sabemos exactamente qué estamos buscando.

Incluyendo fragmentos SQL

Se puede especificar un fragmento SQL como parte de la sentencia **`where`** y **`find_by`**.

Es muy poderoso, pero es susceptible a la inyección SQL.

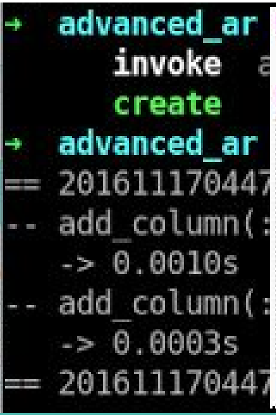


```
irb(main):005:0> Person.where("age BETWEEN 20 and 25").to_a
  Person Load (0.4ms)  SELECT "people".* FROM "people" WHERE (age BETWEEN 20 and 25)
=> [#<Person id: 4, first_name: "Federico", age: 25, last_name: "Canhete", created_at: "2016-11-17 04:28:54",
 updated_at: "2016-11-17 04:28:54">, #<Person id: 6, first_name: "Cristian", age: 25, last_name: "Cuevas", cr
eated_at: "2016-11-17 04:28:54", updated_at: "2016-11-17 04:28:54">, #<Person id: 7, first_name: "Alejandro",
 age: 24, last_name: "Cuevas", created_at: "2016-11-17 04:28:54", updated_at: "2016-11-17 04:28:54">, #<Perso
n id: 8, first_name: "Enrique", age: 20, last_name: "Cuevas", created_at: "2016-11-17 04:28:54", updated_at:
"2016-11-17 04:28:54">]
irb(main):006:0> Person.find_by("first_name LIKE '%an'")
  Person Load (0.3ms)  SELECT  "people".* FROM "people" WHERE (first_name LIKE '%an') LIMIT 1
=> #<Person id: 6, first_name: "Cristian", age: 25, last_name: "Cuevas", created_at: "2016-11-17 04:28:54", u
pdated_at: "2016-11-17 04:28:54">
irb(main):007:0> 
```

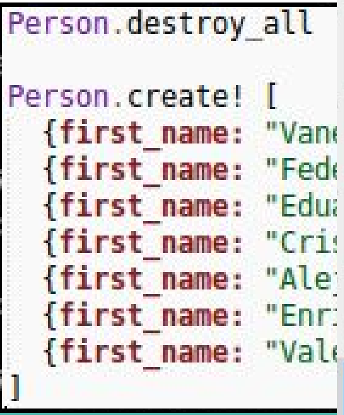
Qué es una Inyección SQL?

- Manipular las consultas SQL para hackear a la base de datos.
- Esto incluye borrar maliciosamente tablas o ganar acceso a información confidencial.

Para mostrar un ejemplo vamos a agregar login y pass a nuestra tabla persona, modificamos el seed, y volvemos a correr rake db:seed

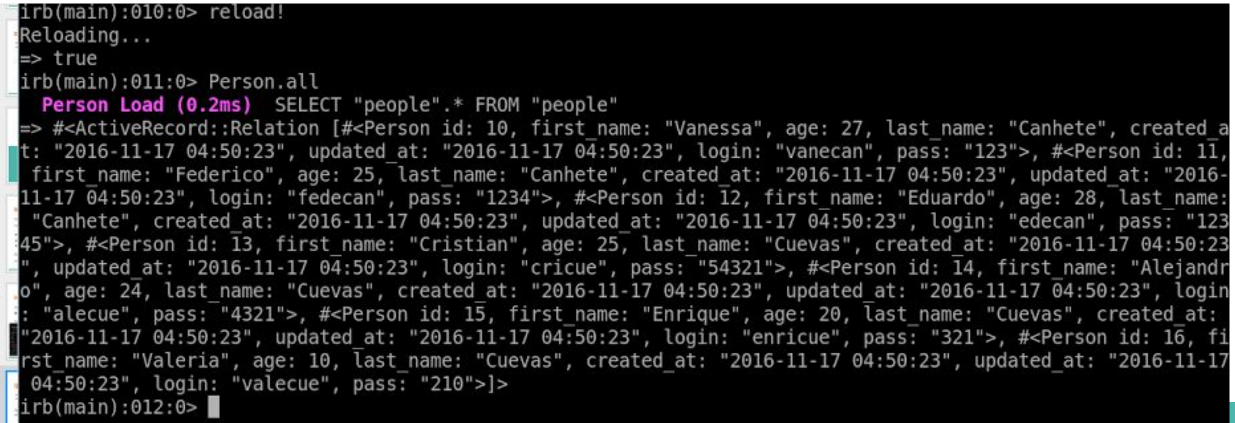


```
→  advanced_ar
     invoke   a
     create
→  advanced_ar
== 201611170447
-- add_column(:
   -> 0.0010s
-- add_column(:
   -> 0.0003s
== 201611170447
```



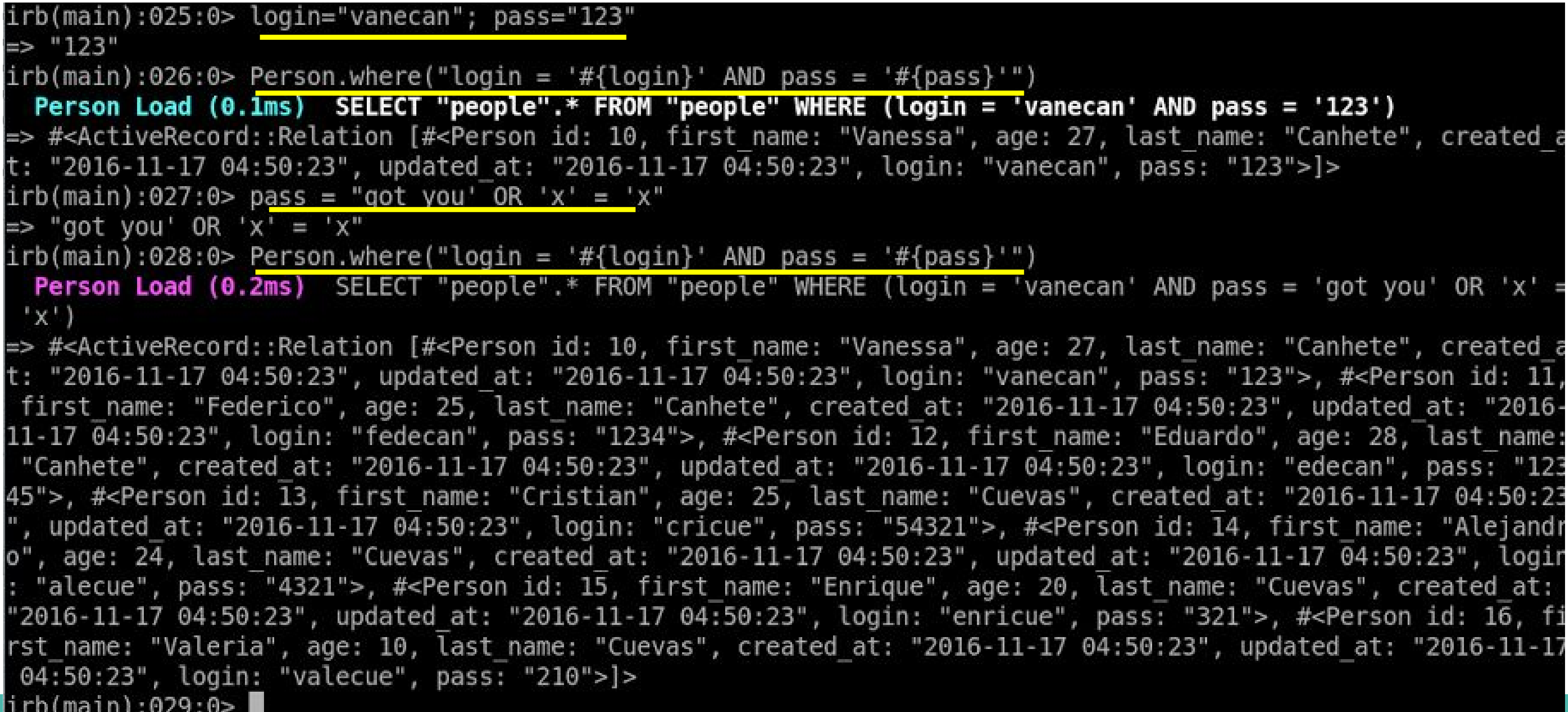
```
Person.destroy_all

Person.create! [
  {first_name: "Vane
  {first_name: "Fede
  {first_name: "Edua
  {first_name: "Cris
  {first_name: "Ale
  {first_name: "Enri
  {first_name: "Vale
]
```



```
irb(main):010:0> reload!
Reloading...
=> true
irb(main):011:0> Person.all
  Person Load (0.2ms)  SELECT "people".* FROM "people"
=> #<ActiveRecord::Relation [#<Person id: 10, first_name: "Vanessa", age: 27, last_name: "Canhete", created_a
t: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "vanecan", pass: "123">, #<Person id: 11,
 first_name: "Federico", age: 25, last_name: "Canhete", created_at: "2016-11-17 04:50:23", updated_at: "2016-
11-17 04:50:23", login: "fedecan", pass: "1234">, #<Person id: 12, first_name: "Eduardo", age: 28, last_name:
 "Canhete", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "edecan", pass: "123
45">, #<Person id: 13, first_name: "Cristian", age: 25, last_name: "Cuevas", created_at: "2016-11-17 04:50:23
", updated_at: "2016-11-17 04:50:23", login: "cricue", pass: "54321">, #<Person id: 14, first_name: "Alejandr
o", age: 24, last_name: "Cuevas", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login
: "alecue", pass: "4321">, #<Person id: 15, first_name: "Enrique", age: 20, last_name: "Cuevas", created_at:
"2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "enricue", pass: "321">, #<Person id: 16, fi
rst_name: "Valeria", age: 10, last_name: "Cuevas", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17
 04:50:23", login: "valecue", pass: "210">]>
irb(main):012:0> 
```

Ejemplo de Inyección SQL



```
irb(main):025:0> login="vanecan"; pass="123"
=> "123"
irb(main):026:0> Person.where("login = '#{login}' AND pass = '#{pass}'")
  Person Load (0.1ms)  SELECT "people".* FROM "people" WHERE (login = 'vanecan' AND pass = '123')
=> #<ActiveRecord::Relation [#<Person id: 10, first_name: "Vanessa", age: 27, last_name: "Canhete", created_a
t: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "vanecan", pass: "123">]>
irb(main):027:0> pass = "got you' OR 'x' = 'x"
=> "got you' OR 'x' = 'x"
irb(main):028:0> Person.where("login = '#{login}' AND pass = '#{pass}'")
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE (login = 'vanecan' AND pass = 'got you' OR 'x' =
 'x')
=> #<ActiveRecord::Relation [#<Person id: 10, first_name: "Vanessa", age: 27, last_name: "Canhete", created_a
t: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "vanecan", pass: "123">, #<Person id: 11,
 first_name: "Federico", age: 25, last_name: "Canhete", created_at: "2016-11-17 04:50:23", updated_at: "2016-
11-17 04:50:23", login: "fedecan", pass: "1234">, #<Person id: 12, first_name: "Eduardo", age: 28, last_name:
 "Canhete", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "edecan", pass: "123
45">, #<Person id: 13, first_name: "Cristian", age: 25, last_name: "Cuevas", created_at: "2016-11-17 04:50:23
", updated_at: "2016-11-17 04:50:23", login: "cricue", pass: "54321">, #<Person id: 14, first_name: "Alejandr
o", age: 24, last_name: "Cuevas", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login
: "alecue", pass: "4321">, #<Person id: 15, first_name: "Enrique", age: 20, last_name: "Cuevas", created_at:
"2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "enricue", pass: "321">, #<Person id: 16, fi
rst_name: "Valeria", age: 10, last_name: "Cuevas", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17
 04:50:23", login: "valecue", pass: "210">]>
irb(main):029:0>
```

Entonces...

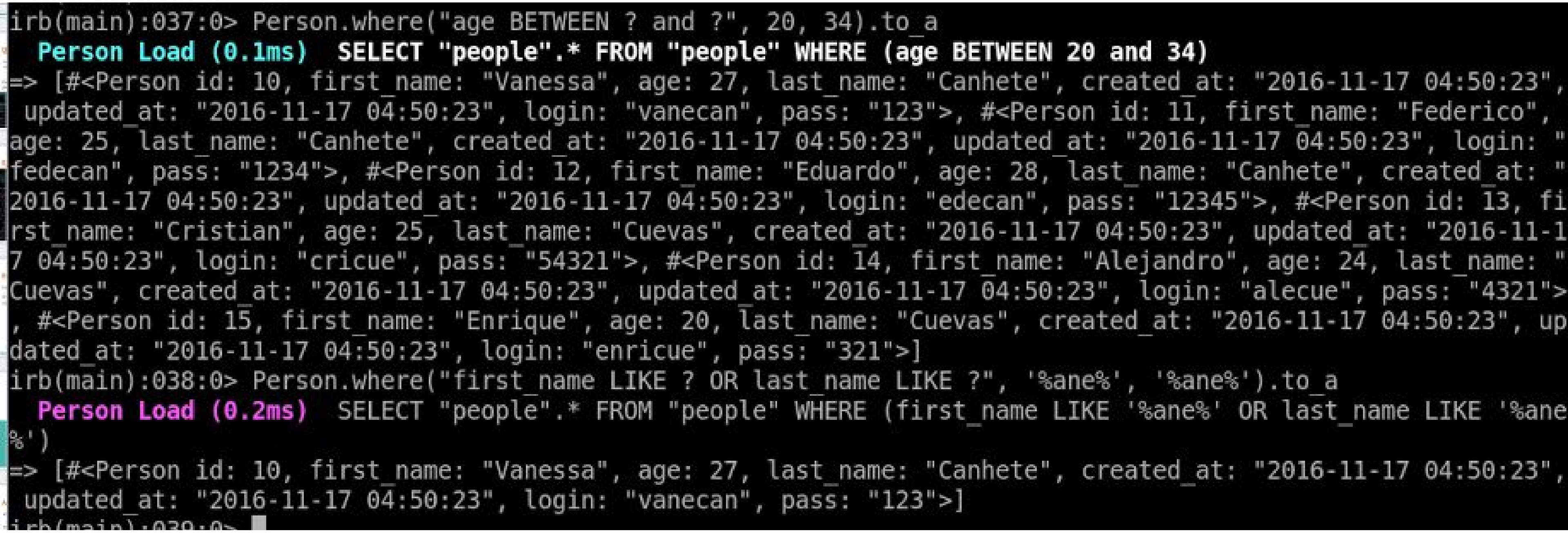
Se puede introducir fácilmente fragmentos sql en las consultas.

Desafortunadamente, esta aproximación puede dejarnos susceptibles a una inyección SQL.

Array y Hash Syntax

Array Syntax

- Nos permite **especificar** el fragmento SQL con **?** seguido de valores (que serían los parámetros).
- Automáticamente **realiza una conversión** de los valores de entrada y **escapa** los strings en el SQL.
- Es **inmune** a una inyección SQL
- Es similar a un **PreparedStatement** de Java.



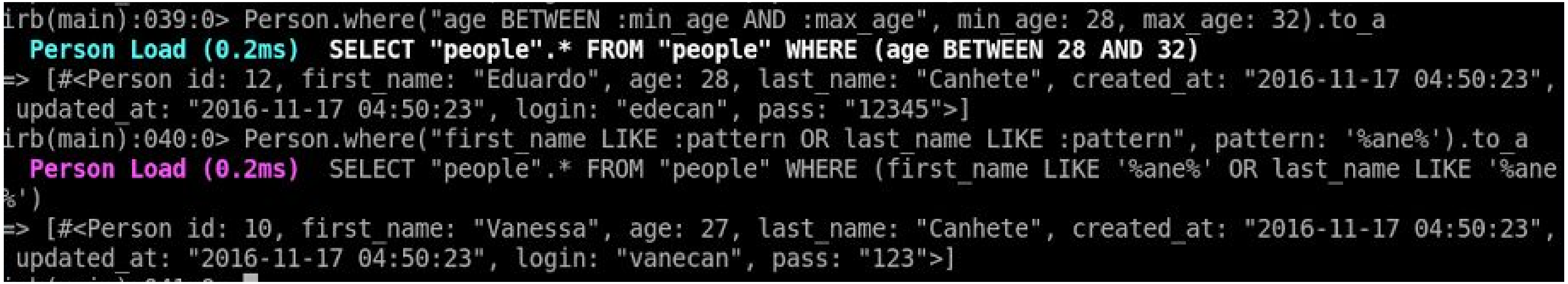
```
irb(main):037:0> Person.where("age BETWEEN ? and ?", 20, 34).to_a
  Person Load (0.1ms)  SELECT "people".* FROM "people" WHERE (age BETWEEN 20 and 34)
=> [#<Person id: 10, first_name: "Vanessa", age: 27, last_name: "Canhete", created_at: "2016-11-17 04:50:23",
 updated_at: "2016-11-17 04:50:23", login: "vanecan", pass: "123">, #<Person id: 11, first_name: "Federico",
age: 25, last_name: "Canhete", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "
fedecan", pass: "1234">, #<Person id: 12, first_name: "Eduardo", age: 28, last_name: "Canhete", created_at: "
2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "edecan", pass: "12345">, #<Person id: 13, fi
rst_name: "Cristian", age: 25, last_name: "Cuevas", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-1
7 04:50:23", login: "cricue", pass: "54321">, #<Person id: 14, first_name: "Alejandro", age: 24, last_name: "
Cuevas", created_at: "2016-11-17 04:50:23", updated_at: "2016-11-17 04:50:23", login: "alecue", pass: "4321">
, #<Person id: 15, first_name: "Enrique", age: 20, last_name: "Cuevas", created_at: "2016-11-17 04:50:23", up
dated_at: "2016-11-17 04:50:23", login: "enricue", pass: "321">]
irb(main):038:0> Person.where("first_name LIKE ? OR last_name LIKE ?", '%ane%', '%ane%').to_a
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE (first_name LIKE '%ane%' OR last_name LIKE '%ane
%')
=> [#<Person id: 10, first_name: "Vanessa", age: 27, last_name: "Canhete", created_at: "2016-11-17 04:50:23",
 updated_at: "2016-11-17 04:50:23", login: "vanecan", pass: "123">]
irb(main):039:0>
```

Array Condition Syntax

- La sintaxis del array contidion es “SQL Injection safe” y fácil de usar, pero existen dos pequeños problemas:
  - Se debe mantener el orden de los parámetros.
  - Si se tienen *n* “?” - se necesita pasar n valores, inclusive cuando haya alguna referencia al mismo valor.

Hash Condition Syntax

En lugar de “?” se especifican símbolos donde el mapeo de valores en el hash es pasado en un segundo parámetro.



```
irb(main):039:0> Person.where("age BETWEEN :min_age AND :max_age", min_age: 28, max_age: 32).to_a
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE (age BETWEEN 28 AND 32)
=> [#<Person id: 12, first_name: "Eduardo", age: 28, last_name: "Canhete", created_at: "2016-11-17 04:50:23",
 updated_at: "2016-11-17 04:50:23", login: "edecan", pass: "12345">]
irb(main):040:0> Person.where("first_name LIKE :pattern OR last_name LIKE :pattern", pattern: '%ane%').to_a
  Person Load (0.2ms)  SELECT "people".* FROM "people" WHERE (first_name LIKE '%ane%' OR last_name LIKE '%ane
%')
=> [#<Person id: 10, first_name: "Vanessa", age: 27, last_name: "Canhete", created_at: "2016-11-17 04:50:23",
 updated_at: "2016-11-17 04:50:23", login: "vanecan", pass: "123">]
```

Entonces...

Utilizar siempre Array o Hash condition syntax para evitar el SQL Injection.

Hash syntax parece ser más intuitivo para la mayoría de las personas.