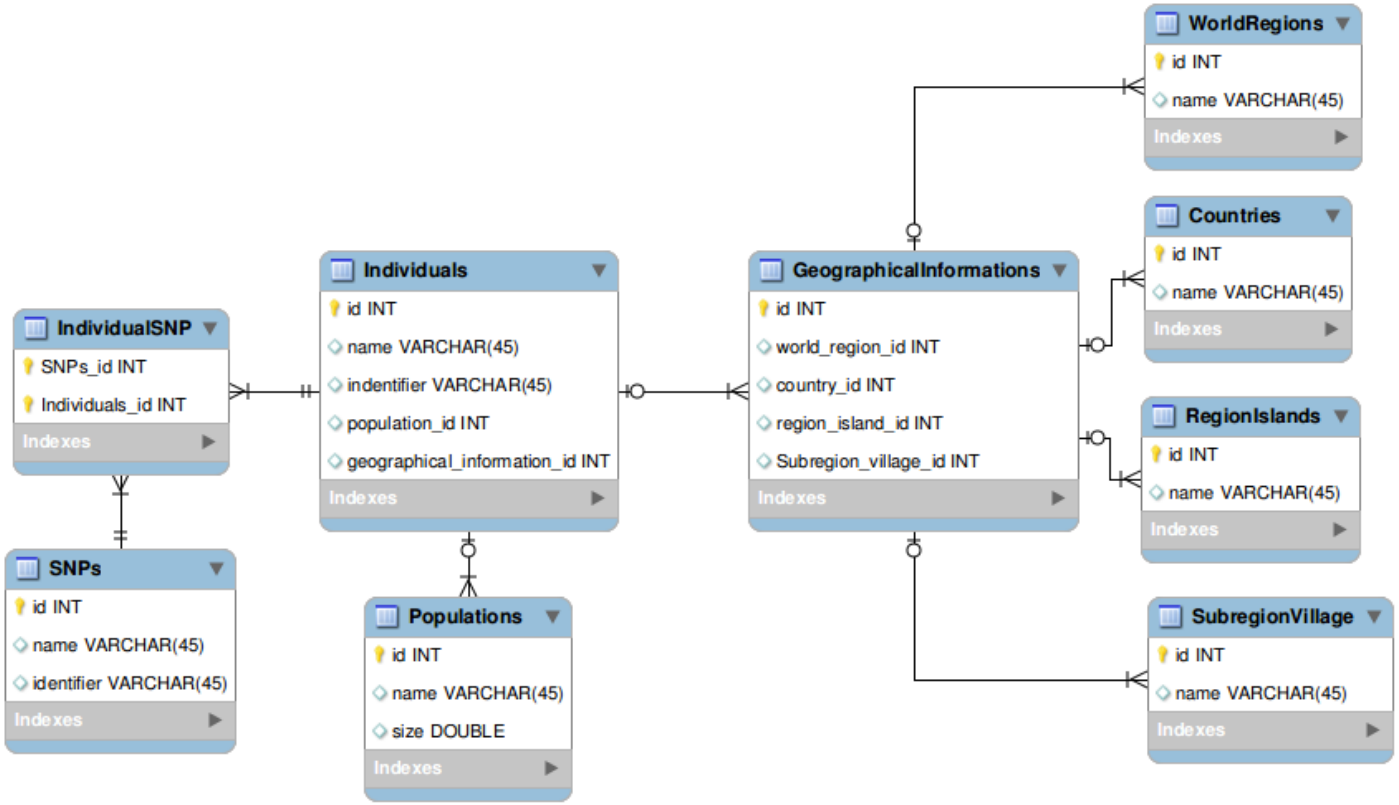
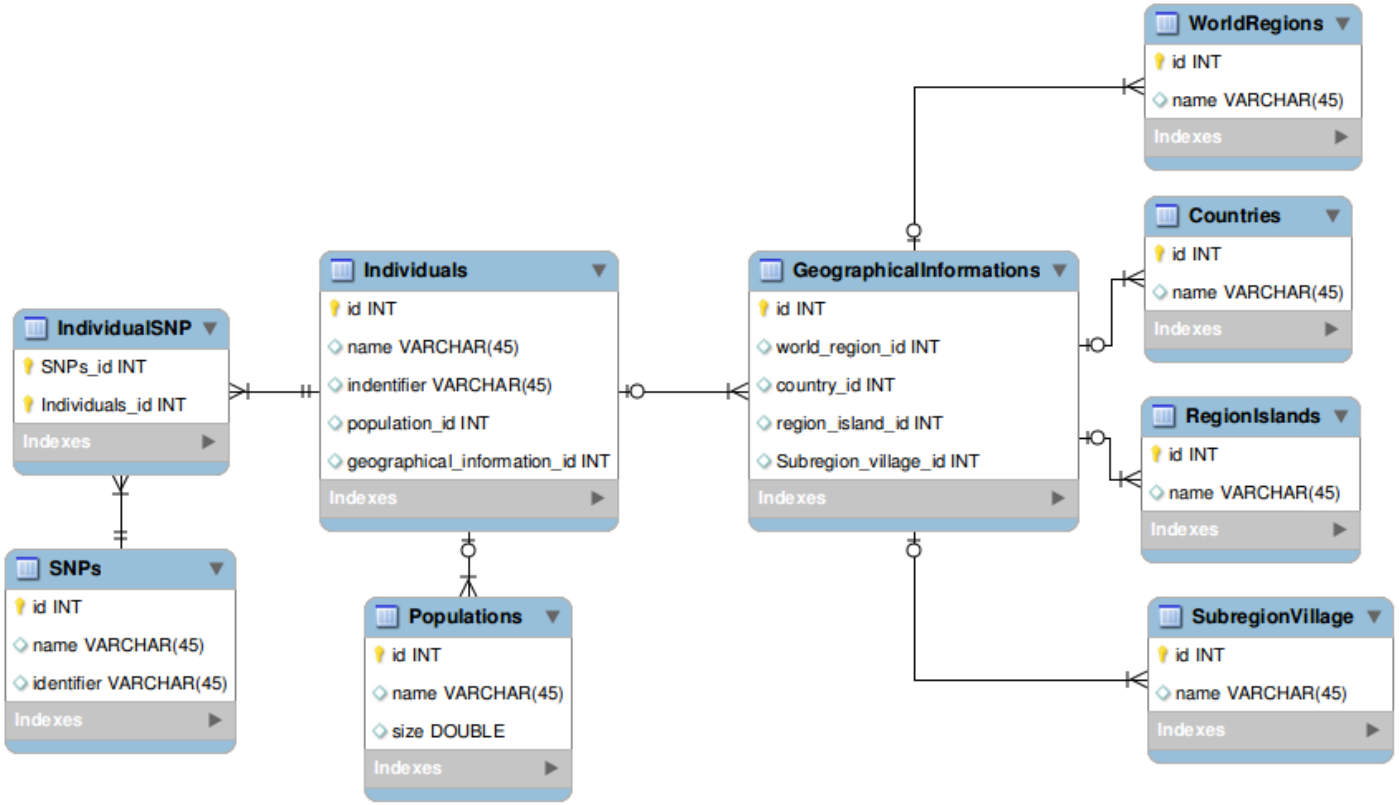
Introduction to Ruby on Rails as an ORM

Pierre-Yves Dupont

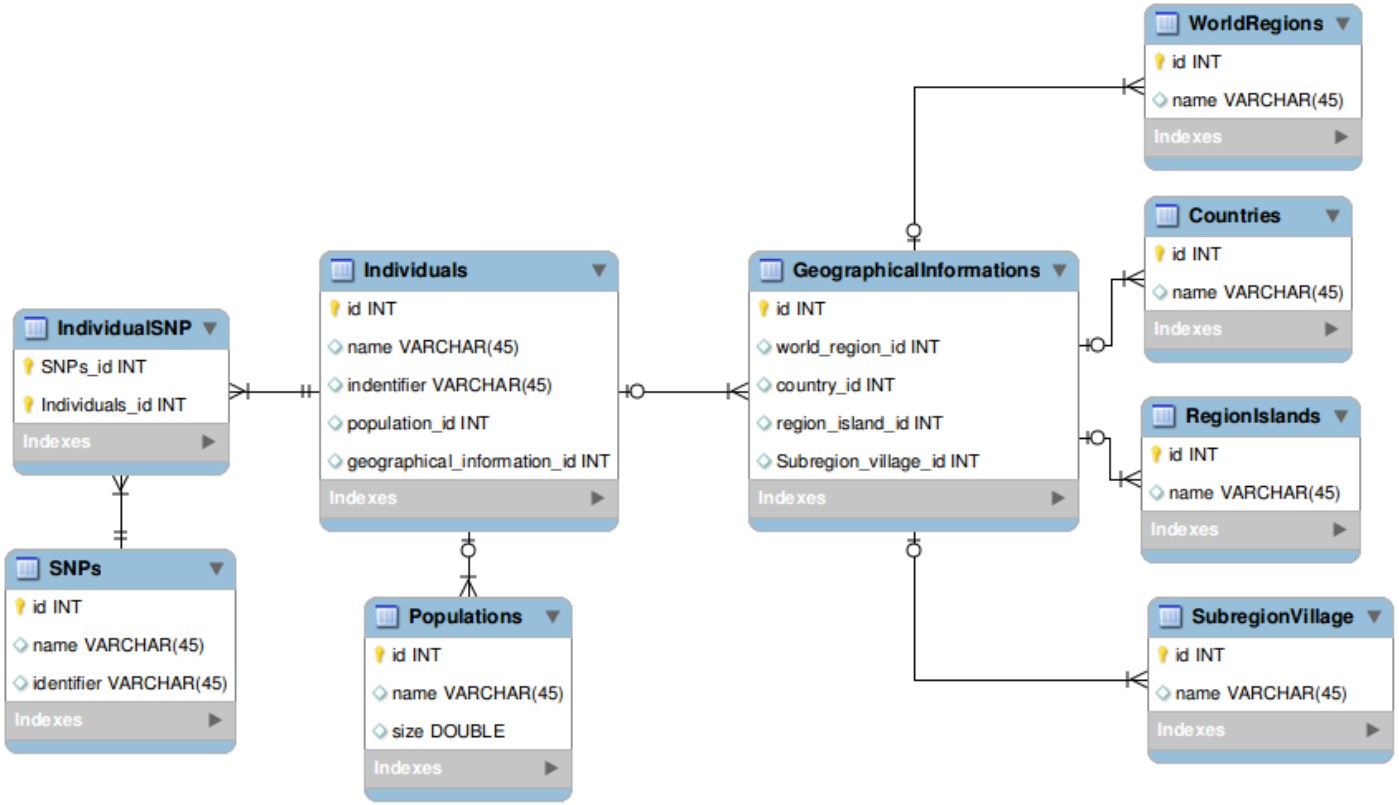
Simple example



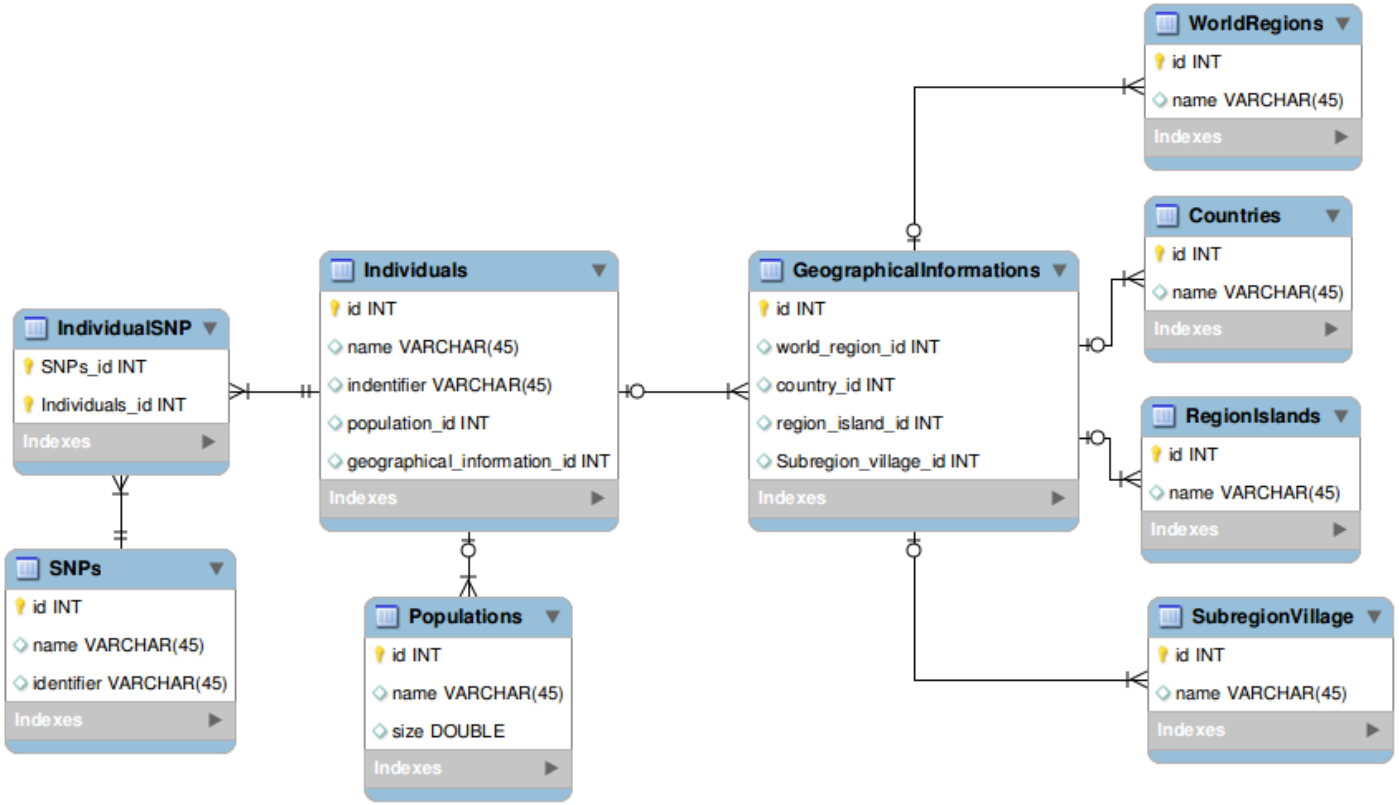
Individuals is a **Table**, having many **attributes** like *name*



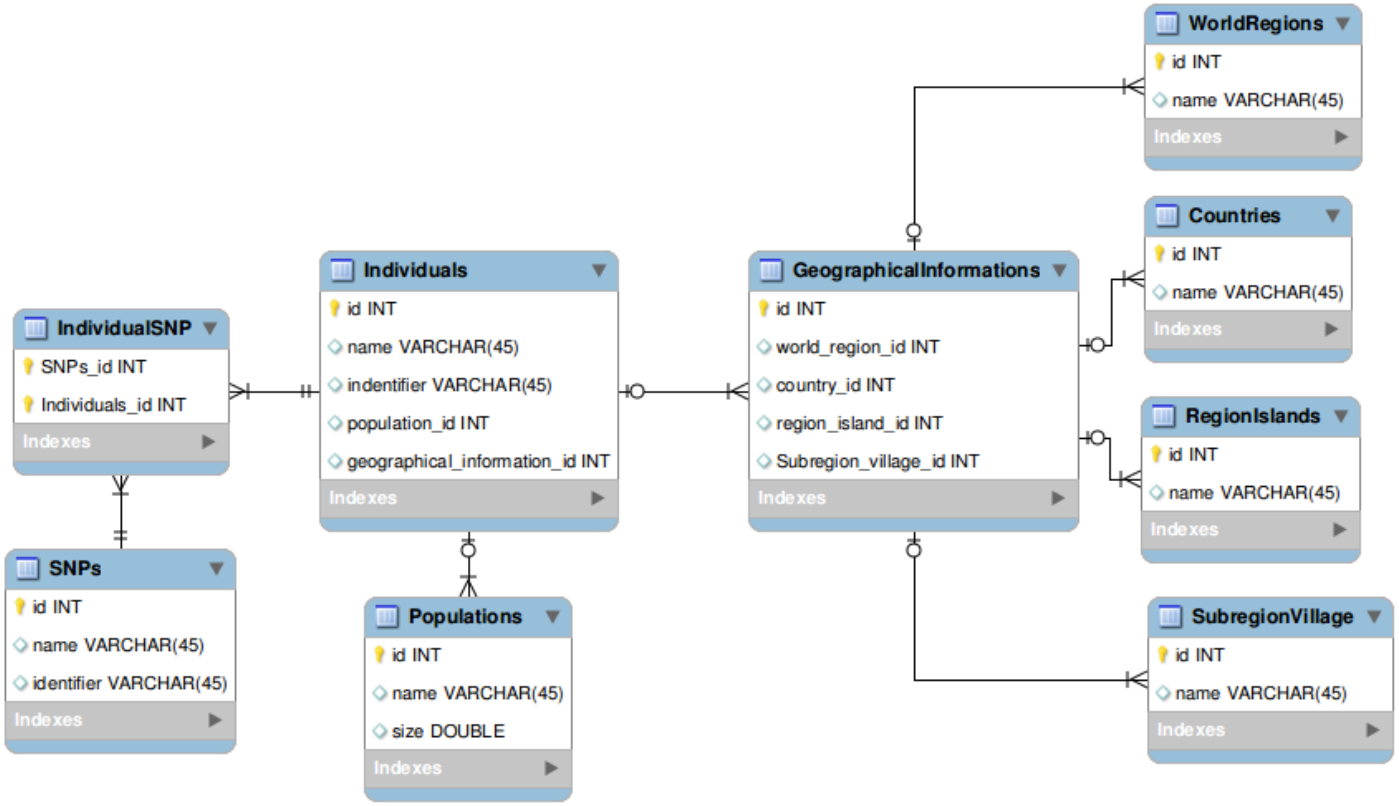
Individuals **belongs to** a Population



Population **has many** Individuals

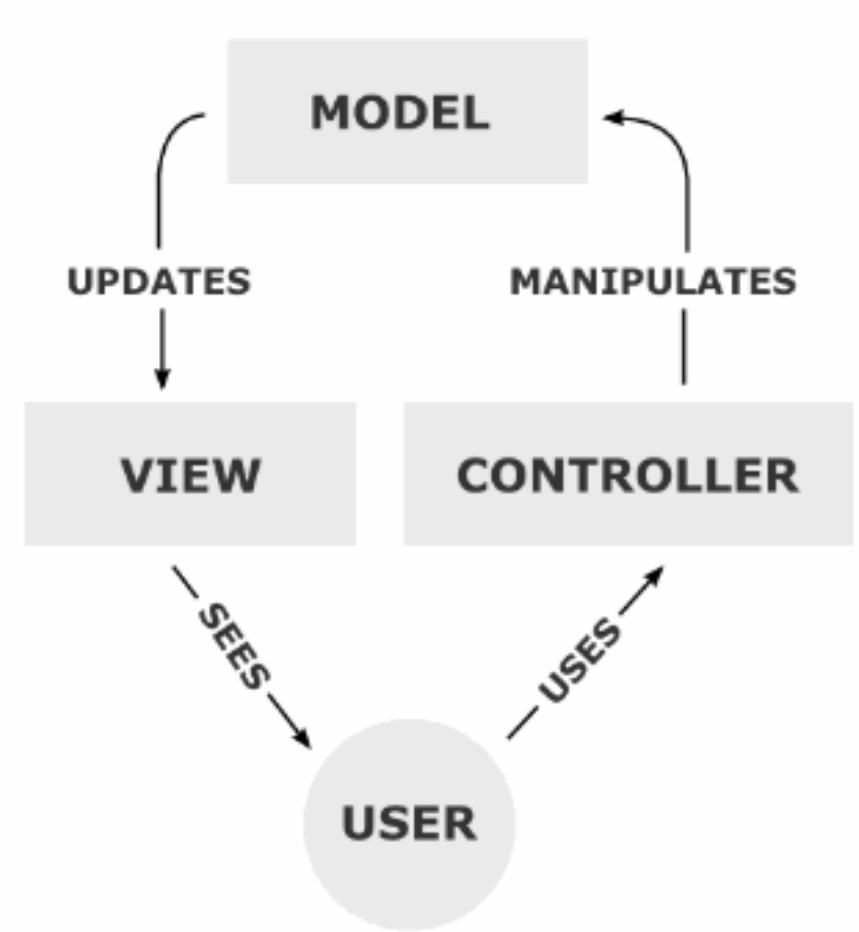


Individuals **has many** SNPs
SNPs **has many** Individuals



Individuals **has and belongs to many** SNPs
nn relation

Design Pattern MVC



MVC: Model View Controller

MVC: in RoR

app	6 items	folder
assets	3 items	folder
controllers	2 items	folder
application_controller.rb	80 bytes	Ruby script
individuals_controller.rb	2.1 kB	Ruby script
helpers	2 items	folder
mailers	0 items	folder
models	1 item	folder
individual.rb	175 bytes	Ruby script
views	2 items	folder
individuals	5 items	folder
edit.html.erb	128 bytes	HTML document
_form.html.erb	874 bytes	plain text document
index.html.erb	737 bytes	HTML document
new.html.erb	87 bytes	HTML document
show.html.erb	408 bytes	plain text document

Ruby on Rails ORM

ActiveRecod Models

```
#In app/models/individual.rb file
class Individual < ActiveRecord::Base
  attr_accessible :identifier, :name
  belongs_to :population
  belongs_to :geographical_information
  has_and_belongs_to_many :snps
end

#In app/models/population.rb file
class Population < ActiveRecord::Base
  attr_accessible :name, :size
  has_many :individuals
end
```

Database link

Convention over configuration (software design paradigm)

```
#In app/models/individual.rb file
#Model Individual -> table individuals
class Individual < ActiveRecord::Base
  #Attributes -> fields in the table
  attr_accessible :identifier, :name
  #population_id field in individuals table
  belongs_to :population
  belongs_to :geographical_information
  #join table individuals_populations
  has_and_belongs_to_many :snps
end
```

Database link

Some necessary configuration . . .

```
#In config/database.yml
development:
  adapter: mysql2
  encoding: utf8
  reconnect: false
  database: tutorails_development
  pool: 5
  username: root
  password: atadxam
  socket: /var/run/mysqld/mysqld.sock
```

Python ORM (SQLAlchemy)

```
class Individuals(Base):
    __tablename__ = 'individuals'

    id = Column(Integer, primary_key=True)
    name = Column(String(255), nullable=False)
    identifier = Column(Integer,nullable=False)
    population_id = Column(Integer,ForeignKey('population.id'))
    population = relation("Population", backref='individuals')

    def __init__(self, title=None, year=None):
        self.title = title
        self.year = year
```

Querying the database

Simple queries

```
Population.all

Population.first

Population.last

Population.order("size desc").limit(10)

Population.where("size <= ?", 1000).includes(:individuals)

Population.order("size").last

Population.first.name

Population.first.individuals
```

Defining a scope (subset)

```
class Population < ActiveRecord::Base
  attr_accessible :name, :size
  has_many :individuals

  #static method Population.smaller_than(1)
  def self.smaller_than(x)
    where("populations.size < ?", x)
  end

  #definition of the scope small
  scope :small, smaller_than(1000)
  #definitions of individuals belonging to ``Foo'' population
  scope :foos, joins(:population).\
   where('populations.name LIKE "Foo"')
end
```

```
%> Population.small
```

```
SELECT `populations`.*
  FROM `populations`
 WHERE (populations.size < 1000);
```

Defining a scope (subset)

```ruby
class Individual < ActiveRecord::Base
  attr_accessible :identifier, :name
  belongs_to :population
  belongs_to :geographical_information
  has_and_belongs_to_many :snps

  scope :in_small_population, \
   joins(:population).merge(Population.small)
end
```

```
%> Individual.in_small_population
```

```
    SELECT `individuals`.*
      FROM `individuals`
INNER JOIN `populations`
        ON `populations`.`id` = `individuals`.`population_id`
     WHERE (populations.size < 1000);
```

Populate the database

Simple queries

```
1 #create a new individual
2 i = Individual.create :name => 'my_name', :identifier => 'my_id'
3 #find a population having name 'foo', create it if not
4 p = Population.find_or_create_by_name :name => 'foo'
5 #add the created individuals in the population
6 p.individuals << i
```

line 2

```
1    INSERT INTO `individuals`
2    (`created_at`, `geographical_information_id`,
3    `identifier`, `name`, `population_id`, `updated_at`)
4    VALUES ('2012-09-12 01:51:32', NULL, 'my_id',
5    'my_name', NULL, '2012-09-12 01:51:32');
```

Simple queries

```
1 #create a new individual
2 i = Individual.create :name => 'my_name', :identifier => 'my_id'
3 #find a population having name 'foo', create it if not
4 p = Population.find_or_create_by_name :name => 'Bar'
5 #add the created individuals in the population
6 p.individuals << i
```

Search in MySQL is not case sensitive

line 4

```
1 SELECT `populations`.*
2   FROM `populations`
3  WHERE `populations`.`name` = 'Bar'
4  LIMIT 1;
5
6  INSERT INTO `populations`
7   (`created_at`, `name`, `size`, `updated_at`)
8  VALUES ('2012-09-12 02:35:00', 'Bar',
9  NULL, '2012-09-12 02:35:00');
```

Simple queries

```
1 #create a new individual
2 i = Individual.create :name => 'my_name', :identifier => 'my_id'
3 #find a population having name 'foo', create it if not
4 p = Population.find_or_create_by_name :name => 'Bar'
5 #add the created individuals in the population
6 p.individuals << i
```

line 6

```
1 UPDATE `individuals`
2    SET `population_id` = 5, `updated_at` = '2012-09-12 02:37:11'
3  WHERE `individuals`.`id` = 13;
```

Rails generators

Generate the scaffold of the application

To generate the “tutorails” application:

```
%>rails new tutorails
```

Will make all the folders, configuration and template files of the application

Create a new model

```
%>rails generate scaffold SNPs name:string identifier:string
      invoke  active_record
      create    db/migrate/20120912025110_create_snps.rb
      create    app/models/snp.rb
      invoke    test_unit
      create      test/unit/snp_test.rb
      create      test/fixtures/snps.yml
      invoke  resource_route
       route    resources :snps
      invoke  scaffold_controller
      create    app/controllers/snps_controller.rb
      invoke    erb
      create      app/views/snps
      create      app/views/snps/index.html.erb
      create      app/views/snps/edit.html.erb
      create      app/views/snps/show.html.erb
      create      app/views/snps/new.html.erb
      create      app/views/snps/_form.html.erb
      ...
```

Migration files

```
#db/migrate/20120912025110_create_snps.rb
class CreateSnps < ActiveRecord::Migration
  def change
    #Generated code
    create_table :snps do |t|
      t.string :name
      t.string :identifier
      t.timestamps
    end
    #The code for the join table has to be written by hand
    create_table :individuals_snps, :id => false do |t|
   t.references :individual, :null => false
   t.references :snp, :null => false
end
add_index(:individuals_snps, [:individual_id, :snp_id])
  end
end
```

Migrate the database

```
%>rake db:migrate
==  CreateSnps: migrating ================================
-- create_table(:snps)
   -> 0.0789s
-- create_table(:individuals_snps, {:id=>false})
   -> 0.0586s
-- add_index(:individuals_snps, [:individual_id, :snp_id])
   -> 0.1343s
==  CreateSnps: migrated (0.2721s) =======================
```

Gem Rails

Gem: plugin system

To add a gem in your application, edit the Gemfile file:

```
gem "nifty-generators"
```

And run the command:

```
%> bundle install
```

Your gem is installed and you can use it

```
%> rails g nifty:scaffold georgraphical_information \
world_region_id:integer country_id:integer \
region_island_id:integer subregion_village_id:integer
```

Views

GeographicalInformation Index

Geographical Informations

World Region	Country	Region Island	Subregion Village	
Asia	Indonesia	Sulawesi	Central Sulawesi	Show Edit Destroy
Asia	Indonesia	Sumba	Dieng	Show Edit Destroy
Asia	Indonesia	Sumba		Show Edit Destroy
Asia	Indonesia	Sumatra	Anakalang	Show Edit Destroy
Asia	Indonesia	Flores		Show Edit Destroy

New Geographical Information

GeographicalInformation Show

Geographical Information

World Region: Asia

Country: Indonesia

Region Island: Sulawesi

Subregion Village: Central Sulawesi

Edit | Destroy | View All

GeographicalInformation Edit

New Geographical Information

World region: Asia

Country: Indonesia

Region island: Sumatra

Subregion village: Anakalang

Create Geographical information

Back to List

Documentation

Documentation

- http://rubyonrails.org/
- http://railscasts.com/
- http://railsapi.com/doc/rails-v3.2.6/