Time	3 hours
Learning Goals	• Learn how to set up database associations with Active Record. • Reinforce learning on Active Record validations.

Now that you're familiar with the basics of Active Record migrations and simple Active Record models, it's time to learn about Active Record Associations.

You probably remember how to associate tables in a database using foreign keys. This challenge will build on that knowledge and extend it into the realm of the ActiveRecord ORM.

Note: please work on top of the repo from the previous challenge.

Objectives

Add a Teacher Model

Create a `Teacher` model that extends `ActiveRecord::Base`. It should have attributes to store the teacher's name, email address, and phone number. Remember to keep things i18n - friendly!

This model should be created in its own Ruby file and in the right folder. You'll also need to create a migration to make sure the table gets created.

Create some test data for Teachers

Without resorting to using SQL or SQLite, write some Ruby code that uses ActiveRecord commands to create some test data. Create 9 teachers, each with unique names and email addresses.

While you're at it, please ensure that no 2 teachers can share the same email address using an ActiveRecord validation. Write a test to be sure that it works.

Create a One-to-Many association between Teachers and Students

Let's assume for the time being that each student has only 1 teacher, and each teacher can have many students. Does your Teacher model need to change? How about your student model? Do you need a migration?

Make the necessary changes to your code to support this new constraint on the data.

Once you've done that, write some code that will arbitrarily distribute the students fairly evenly across the teachers (note: don't worry about getting equal number of students for each teacher).

Write tests to ensure that your association is working correctly. For example, given a student, can you find a teacher? Can you find all of her students?

Uh-oh! The requirements have changed

The customer for whom you're building this system just changed her mind. It turns out that the system needs to support the notion that a student can have more than one teacher.

Make the necessary changes to your models (along with any necessary migrations) to support this. (Hint: Remember the many-to-many relationship challenge you did on week 2?)

Write tests to ensure that your association is working correctly. For example, given a student, can you find her teachers? Can you find all students for a given teacher?

DRY it up! (Optional)

Does it annoy you that both tables (`students` and `teachers`) are storing `name`, `email`, and `phone` fields? That's good!

Can you think of a way to fix it? Go ahead and do it! Here's a start:

```
class Person < ActiveRecord::Base
end


class Student < Person
  has_many :student_teachers, :foreign_key => :student_id
  has_many :teachers, :through => :student_teachers
end


class Teacher < Person
  # What relations go here?
end
```

```
class StudentTeacher < ActiveRecord::Base
  belongs_to :student
  belongs_to :teacher
end
```

Only read this link if you need a hint.