Rails: Associations and Validation

Lecture 29
Schemas, Migrations, Models

- `schema.rb`
- `migrations`
- `models`
- `database.yml`

- `db:create`
- `db:migrate`
- `db:schema:load`
- `database`
Recall: Migrations

class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :name
      t.string :title
      t.text :content
      t.timestamps
    end
  end
end
Recall: Models

class Post < ApplicationRecord
  # attr_accessible :name, :title, :content
end
Generating Code: rails generate

- Notice: Two blobs of Ruby code need to be in sync
  - Migration (creates table and columns)
    `db/migrate/xxx_create_students.rb`
  - Model (with matching name)
    `app/models/student.rb`

- Easier: Generate *both* simultaneously
  
  ```
  $ rails generate model Student
  fname:string lname:string buckid:integer
  ```
  - Use model name (singular) and attributes
  - Note: this does *not* generate the schema.rb (use rails)

- Migrations for table edits can also be generated
  
  ```
  $ rails generate migration AddNickNameToStudent
  nick:string
  ```
  - Name is meaningful! (starts with add or remove)
  - Creates a migration that changes students table
Result of generate model

class CreateStudents < ActiveRecord::Migration
  def change
    create_table :students do |t|
      t.string :fname
      t.string :lname
      t.integer :buckid
      t.timestamps
    end
  end
end

class Student < ApplicationRecord
end
Demo with rails console

$ rails new demo  # creates directory
  # no schema, migrations, or models
$ cd demo
$ rails generate model Student \
  fname:string lname:string buckid:integer
  # see db/migrate, app/models
$ rails console
> Student.methods  # lots available!
> Student.all  # will this work?
> s = Student.new  # will this work?
Demo with rails console

$ rails new demo # creates directory
  # no schema, migrations, or models
$ cd demo
$ rails generate model Student
  fname:string lname:string buckid:integer
$ rails console
> Student.methods # lots available!
> Student.find :all # empty, no table
> s = Student.new # error, no table
$ rails db:migrate # creates schema.rb
$ rails console
> Student.all #=> []
Working With Models

> s = Student.new
> s2 = Student.new fname: 'Jo'
> s3 = Student.new fname: 'Xi',
    buckid: 23
> Student.all #=> ?
Working With Models

```ruby
> s = Student.new
> s2 = Student.new fname: 'Jo'
> s3 = Student.new fname: 'Xi', buckid: 23
> Student.all #=> [] still
> s.save
> Student.all #=> [<id: 1, ...>]
> s.fname = 'Mary'
> s.save
```
Seeding the Database

- Quickly populate using config/seeds.rb
  - $ rails db:seed  # run seeds.rb
  - $ rails db:reset # drop then reseed

- In db/seeds.rb:
  30.times do
    Student.create!(
      buckid: Faker::Number.unique
        .number(digits: 9),
      fname: Faker::Name.first_name,
      lname: Faker::Name.last_name)
  end

- Useful gem: Faker
  - Add to Gemfile: gem 'faker'
  - $ bundle install
## Associations (1:N Relationship)

<table>
<thead>
<tr>
<th>id (key)</th>
<th>buckid (integer)</th>
<th>team_id (foreign key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22352022</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>334432</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>34822039</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id (key)</th>
<th>name (string)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wicked Wicky</td>
</tr>
<tr>
<td>2</td>
<td>The Happy Crew</td>
</tr>
<tr>
<td>6</td>
<td>No Names</td>
</tr>
</tbody>
</table>
Invariants

- A student belongs to exactly 1 team
  - Weaker: A student belongs to at most 1 team

- Same representation for either invariant
  - A column (of foreign keys) in students table

- Maintaining stronger invariant
  - Students can only be added with team_id set to something valid
  - Deleting a team deletes member students!

- Maintaining weaker invariant
  - Students can be added with null team_id
  - Deleting a team null-ifies members' team_id
Rails Migration and Models

class AddTeamForeignKeys < ActiveRecord::Migration
def change
  add_reference :students, :team,
  index: true  # for quick load
end
end

class Student < ApplicationRecord
  belongs_to :team  # note singular form
      # adds Student#team method
end

class Team < ApplicationRecord
  has_many :students  # note plural form
      # adds Team#students method
end
Association Methods

- **Belongs_to** creates method for accessing owner
  
  ```ruby
  s = Student.find 1 #=> 22352022
  s.team #=> "The Happy Crew"
  s.team.name = 'The(tm) Happy Crew'
  ```

- **Has_many** creates method for accessing members
  
  ```ruby
  t = Team.find 1
  t.students #=> array of students
  t.students.first
  t.students.size
  t.students.destroy_all
  t.students.any? { |s| ... }
  ```
Asymmetry in Writes to Assoc.

- Add a student to a team’s association: Student is automatically saved (assuming team is stored in database)
  \[
  t = \text{Team.find 1} \\
t.\text{students} \Rightarrow [] \\
t.\text{students} \leftarrow \text{Student.new} \ # \text{gets an id} \\
t.\text{students} \Rightarrow [\#<\text{Student id: 1, ...}>]
  \]

- Assign a team student’s association: Student is not automatically saved
  \[
  s = \text{Student.find 1} \\
s.\text{team} = \text{my\_team} \\
s.\text{reload} \Rightarrow s's \text{ team is unchanged!}
  \]
Modifiers for `belongs_to`

class Student < ApplicationRecord

  belongs_to :greek_house,
    optional: true
    # allows foreign key to be null
  belongs_to :project_group,
    class_name: 'Team'
    # default is ProjectGroup
  belongs_to :major,
    foreign_key: 'OSU_code'
    # default is major_id
  belongs_to :team,
    touch: :membership_updated

end
Modifiers for `has_many`

class Team < ApplicationRecord
  has_many :students,
    limit: 5,
    # max number of members
  dependent: :destroy,
    # what happens to students
    # when team is destroyed?
  class_name: 'OSUStudent'
    # default is Student
end
More Relationships

- **1:1 (one-to-one)**
  - Use `belongs_to` with `has_one`
  - `has_one` is just `has_many` with limit of 1
  - Same asymmetry in writing exists

- **N:M (many-to-many)**
  - A third, intermediary table is used with 2 columns (for foreign keys from two tables)
  - In rails, use `has_many :through` association
Validations

- An invariant on data in a single table
  - Every student has a (non-null) buckid
  - Buckids are unique
  - Team names are less than 30 characters
  - Usernames match a given regular expression

- To maintain invariant:
  - Must be true initially
  - Must be satisfied by each insertion

- These validations are in the model
  - A model instance can be checked
  - Invalid objects cannot be saved

```ruby
student = Student.new lname: 'Vee'
student.valid? #=> false (no buckid)
student.save #=> false
```
Example

class Post < ApplicationRecord

  validates :name, presence: true
  validates :title, presence: true,
  length: { minimum: 5,
            maximum: 50 }

end
Rails Implementation

- Model object has an **errors** attribute
  - This attribute is a hash (of problems)
- Failing a validity check adds an item to the errors hash
  - Empty hash corresponds to valid object
  - Each attribute is a key in the errors hash (plus there is a general key, :base)
  
```
  s.errors[:buckid] = "is not a number"
```

- The **valid?** method does the following:
  - Empties errors hash
  - Runs validations
  - Returns **errors.empty?**
Validates Method in Model

validates :column, condition

□ Uniqueness
  uniqueness: true
  uniqueness: {message: 'Username already taken'}

□ Non-nullness (not the same as truth, see next)
  presence: {message: 'Title needed'}

□ Truth of a boolean field
  acceptance: {message: 'Accept the terms'}

□ Matching a regular expression
  format: {with: /[A-Z].*/, message: ...}
  format: /[^A-Za-z0-9]+/

□ Being a number
  numericality: {only_integer: true}

□ Having a length
  length: {minimum: 5}
Alternative: Declarative Style

- Special methods for each flavor of validation

  validates_uniqueness_of :username
  validates_presence_of :password
  validates_acceptance_of :terms
  validates_format_of :name,
      with: /[A-Z].*/
  validates_numericality_of :buckid,
      only_integer: true
Summary

- **Code generation**
  - Database schema generated by `schema.rb`
  - `Schema.rb` generated by `rails` on migrations
  - Migrations and models can be generated by `rails`

- **Associations**
  - 1:N (or 1:1) relationships via foreign keys
  - Rails methods `belongs_to`, `has_many`
  - Create association attributes, which can be read and written
  - Asymmetry in writing owner vs member

- **Validations**
  - Invariants checked before saving
  - Errors hash contains list of problems
  - Declarative style for common case checks
  - Custom validity checkers possible too