Rails: Models

Lecture 25
Recall: Rails Architecture
Recall: Rails Architecture
Mapping Tables to Objects

- General strategy for OO languages
  - Table in database -- a class
  - Table columns -- attributes of the class
  - Table rows -- instances of class (objects)

- Application works with database using ordinary language syntax
  - Class methods for finding row(s) in table

- Example: Java POJOs, Rails models
Directory Structure of Rails

depot/
   ...../app
   ........../controllers
   ........../helpers
   ........../models
   ........../views
   ................../layouts
   ....../components
   ....../config
   ....../db
   ....../doc
   ....../lib
   ....../log
   ....../public
   ....../script
   ....../test
   ....../tmp
   ....../vendor
   ....../README
   ....../Rakefile
A Bit of Configuration

- Which database to use?
  - SQLite is the easiest (no setup!)
  - MySQL has better performance
  - PostgreSQL favored for Heroku deployment

- Different environments: development, test, production
  - Default (for rake command) is development

- See config/database.yml

```yaml
default: &default
  adapter: sqlite3
  pool: 5
  timeout: 5000
development:
  <<: *default
  database: db/development.sqlite3
```
Database Tables

- A database is a collection of *tables*
  - Naming convention: Table names plural
- Each table has a list of *columns*
- Each column has a *name* and a *type*
- A table has a list of *rows*

<table>
<thead>
<tr>
<th>students</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fname</strong> (string)</td>
</tr>
<tr>
<td>Marco</td>
</tr>
<tr>
<td>Primo</td>
</tr>
<tr>
<td>Cher</td>
</tr>
</tbody>
</table>
# Column Types Mappings

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Postgres</th>
<th>MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>boolean</td>
<td>tinyint(1)</td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>date</td>
<td>date</td>
<td></td>
</tr>
<tr>
<td>decimal</td>
<td>decimal</td>
<td>decimal</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>float</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>integer</td>
<td>int(11)</td>
<td></td>
</tr>
<tr>
<td>varchar(255)</td>
<td>character</td>
<td>varchar(255)</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>clob(32768)</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>time</td>
<td>time</td>
<td></td>
</tr>
<tr>
<td>datetime</td>
<td>timestamp</td>
<td>datetime</td>
<td></td>
</tr>
</tbody>
</table>
Table Constraints

- Invariants on table entries beyond type information
  - "Lname is not null"
  - "Buckid is unique"

- Often want a unique identifier for each row (a "primary key")
  - Easy: Include another (integer) column
  - Database responsible for assigning this value every time a row is added
  - No way to change this value after creation
### Primary Key With Autoincrement

#### students

<table>
<thead>
<tr>
<th>id (key)</th>
<th>fname (string)</th>
<th>lname (string)</th>
<th>buckid (integer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Marco</td>
<td>Pantani</td>
<td>22352022</td>
</tr>
<tr>
<td>3</td>
<td>Primo</td>
<td>Carnera</td>
<td>334432</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Cher</td>
<td>34822039</td>
</tr>
</tbody>
</table>
Linking Tables

- Different tables can be related to each other
  - "Each student has exactly 1 major"
  - "Each student can own 1 (or more) vehicles"

- Keys are used to make this connection
  - Include a column in table X containing keys from table Y ("foreign keys")
  - For examples:
    - Student table includes a column identifying a student's major
    - Vehicle table includes a column identifying a (student) owner

- Association is an invariant between tables
### Association: Students & Vehicles

#### Students

<table>
<thead>
<tr>
<th>id</th>
<th>fname</th>
<th>lname</th>
<th>buckid</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Marco</td>
<td>Pantani</td>
<td>22352022</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Primo</td>
<td>Carnera</td>
<td>334432</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Cher</td>
<td></td>
<td>34822039</td>
<td>3</td>
</tr>
</tbody>
</table>

#### Vehicles

<table>
<thead>
<tr>
<th>id</th>
<th>owner</th>
<th>license</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>K3F 443L</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>F8L 220J</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>GOHBUX</td>
</tr>
</tbody>
</table>
## Associations

### students

<table>
<thead>
<tr>
<th>id (key)</th>
<th>major (for. key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

### vehicles

<table>
<thead>
<tr>
<th>id (key)</th>
<th>owner (for. key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

### programs

<table>
<thead>
<tr>
<th>id (key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
Schema

- Definition of table structure
  - Table name
  - Column names and types
  - Constraints
- Usually database manager-specific
- See db/schema.rb for Ruby-based schema description
  - Allows independence from particular DB manager
  - Schema is versioned by timestamp (really by "migration"...)

Computer Science and Engineering  ▪  The Ohio State University
Example schema.rb

ActiveRecord::Schema.define(version: 20170308051206) do

  create_table "students", force: :cascade do |t|
    t.string   "name"
    t.integer  "buckid"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

end
Migrations

Q. Who writes schema.rb?
   A. It is generated! (tool called rake)
   Golden rule: Never edit schema.rb directly
   Instead, write a migration

A migration is Ruby code (a class) that represents a change in schema
   Create new tables (including column names and column types)
   Modify existing tables (adding/removing columns, or changing associations)
   Delete ("drop") existing tables
Migration Classes

- See db/migrate
- Filename consists of
  - Timestamp (UTC) of creation
  - Class name (descriptive of delta)
  - Example: class CreatePosts in 20151026180030_create_posts.rb
- Consequence: Migrations are run in a consistent order
  - Deltas do not commute, so order is important
- Class extends ActiveRecord::Migration
  - Contains method change
  - This method invoked by rake db:migrate
Example Migration Class

class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :name
      t.string :title
      t.text :content
    
      t.timestamps null: false
    end
  end
end
Result of raking this Migration

<table>
<thead>
<tr>
<th>:id (key)</th>
<th>:name (string)</th>
<th>:title (string)</th>
<th>:content (text)</th>
<th>:created_at (time)</th>
<th>:updated_at (time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:posts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Column Type Mappings

<table>
<thead>
<tr>
<th>Column Type</th>
<th>Ruby</th>
<th>SQLite</th>
<th>Postgres</th>
<th>MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>:boolean</td>
<td>Boolean</td>
<td>boolean</td>
<td>boolean</td>
<td>tinyint(1)</td>
</tr>
<tr>
<td>:date</td>
<td>Date</td>
<td>date</td>
<td>date</td>
<td>date</td>
</tr>
<tr>
<td>:decimal</td>
<td>BigDecimal</td>
<td>decimal</td>
<td>decimal</td>
<td>decimal</td>
</tr>
<tr>
<td>:float</td>
<td>Float</td>
<td>float</td>
<td>float</td>
<td>float</td>
</tr>
<tr>
<td>:integer</td>
<td>FixNum</td>
<td>integer</td>
<td>integer</td>
<td>int(11)</td>
</tr>
<tr>
<td>:string</td>
<td>String</td>
<td>varchar(255)</td>
<td>character</td>
<td>varchar(255)</td>
</tr>
<tr>
<td>:text</td>
<td>String</td>
<td>text</td>
<td>clob(32768)</td>
<td>text</td>
</tr>
<tr>
<td>:time</td>
<td>Time</td>
<td>time</td>
<td>time</td>
<td>time</td>
</tr>
<tr>
<td>:timestamp</td>
<td>Time</td>
<td>datetime</td>
<td>timestamp</td>
<td>datetime</td>
</tr>
</tbody>
</table>
Schema Deltas In Migrations

- In addition to creating tables, the change method can also
  - Modify columns of an existing table
    - add_column, remove_column, rename_column, change_column
  - Modify and delete tables
    - change_table, drop_table

- Example: xxx_add_author_to_posts.rb
  ```ruby
class AddAuthorToPosts < ActiveRecord::Migration

  def change
    add_column :posts, :author, :string
  end
  end
  ```
Migrations as History

- Change defined by migration can be undone
  - Migrations give a linear history of deltas
  - Schema is the result of applying them (in order)
- Can move forward/backward in history
  - Create database only (no schema) defined in config/database.yml
    - $ rake db:create
  - Update schema.rb (compare its version number to list of migrations) and apply to database
    - $ rake db:migrate
  - Rollback schema.rb to earlier point in history
    - $ rake db:rollback
  - Load schema defined in db/schema.rb
    - $ rake db:schema:load
Schemas, Migrations, Models

- `schema.rb`
- `migrations`
- `models`
- `database.yml`
- `db:create`
- `db:migrate`
- `db:schema:load`
- `db:schema:dump`
- `database`
Migrations vs Schema

- Golden rule: Never edit schema.rb
  - It is regenerated every time you do a migration
  - Every change in schema means writing a migration

- Commit schema.rb to version control
  - Deployment in fresh environment means loading schema, not reliving the full migration history

- Commit migrations to version control
  - Once a migration has been shared, to undo it you should create a new migration (preserve the linear history)
Models

- Programmatic way for application to interact with database
  - Collection of Ruby classes
  - Extend `ActiveRecord::Base`
  - Found in app/models

- Each class corresponds to a table
  - Note: Models are *singular* (tables are *plural*)
  - Includes attributes corresponding to columns *implicitly*

```ruby
class Post < ActiveRecord::Base
  #attr_accessible :name,:title,:content
end
```
Class Methods for Models

- Create a new instance with `new`
  
  ```ruby
  p1 = Post.new
  p2 = Post.new author: "Xi", title: "Hola"
  ```

  **Warning:** this only creates the model (object) it does *not* modify the database

- Create instance *and* add it to database
  
  ```ruby
  p3 = Post.create author: "Zippy"
  ```

- Retrieve a particular row from table
  
  ```ruby
  @post = Post.find 4 #search for id
  @post = Post.find_by author: "Xi"
  @blog = Post.all
  @student = Student.find_by_buckid 543333
  @post = Post.first
  @post = Post.last
  ```
Instance Methods for Models

- To save a model (object) as a row in the database
  ```ruby
  p = Post.new author: 'Xi'
  p.save # commits change to database
  ```

- Read/write attributes like an ordinary Ruby class
  ```ruby
  @post = Post.find_by author: 'Xi'
  t = @post.title # => nil
  @post.title = 'A Successful Project'
  @post.save # don't forget to save
  ```

- To delete a row from the table
  ```ruby
  @post.destroy # no save needed
  ```
Summary

- Databases: Tables, columns, rows
  - Structure defined in a schema
  - Rails uses Ruby code to generate schema

- Migrations
  - Ruby code describing change to schema
  - Syntax look declarative

- Models
  - Ruby classes that mirror database tables
  - Class names from table (singular vs plural)
  - Attributes from columns

- Code generation
  - Schema generated by schema.rb
  - Schema.rb generated by rake on migrations
  - Migrations and models generated by rails generate