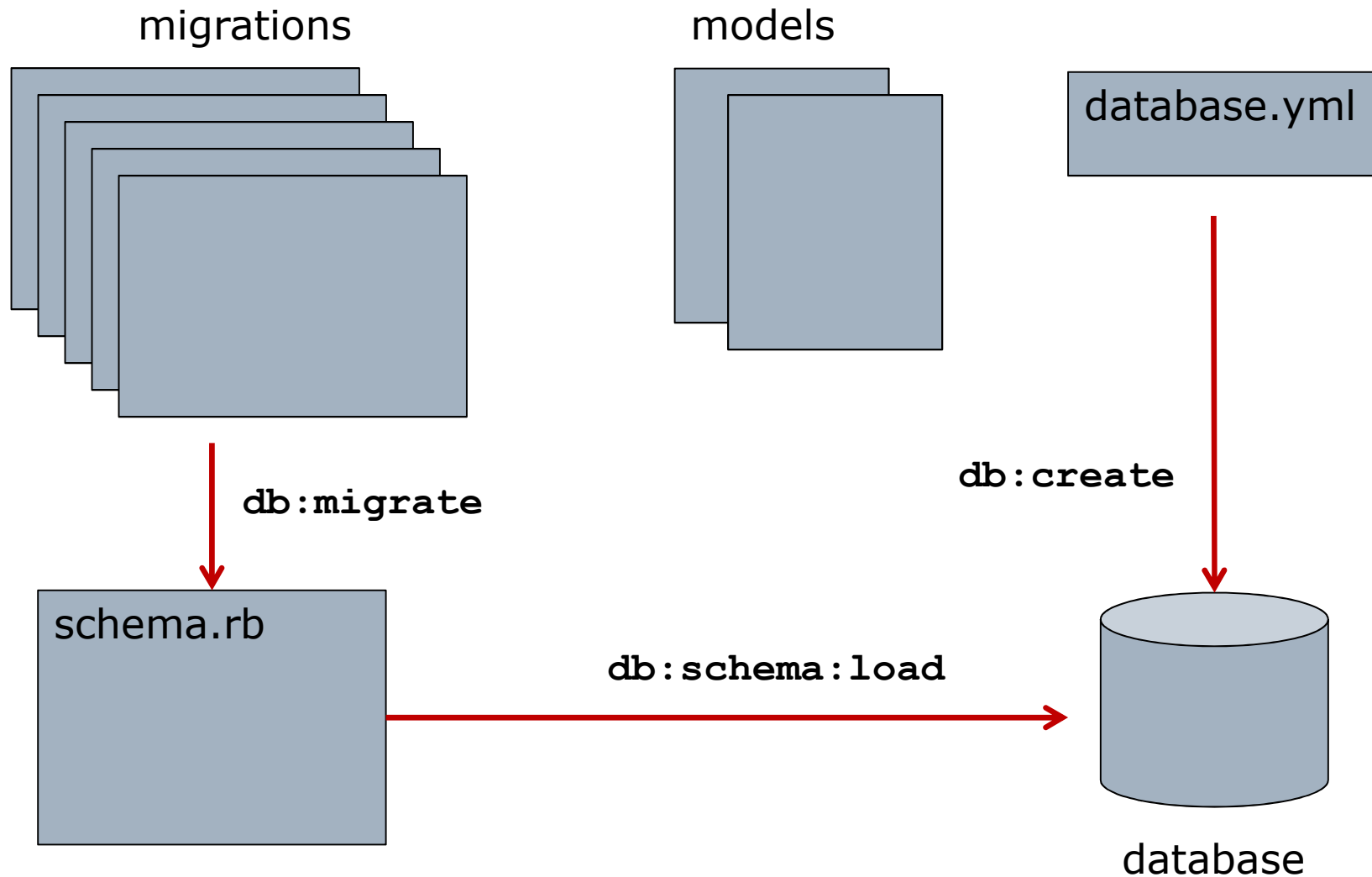


Rails: Associations and Validation

Computer Science and Engineering ■ College of Engineering ■ The Ohio State University

Lecture 28

Schemas, Migrations, Models



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`
- Single point of control: 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 # error, no table
> 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 # error, no table
> s = Student.new # error, no table
$ rails db:migrate # creates schema.rb
$ rails console
> Student.all #=> []
```

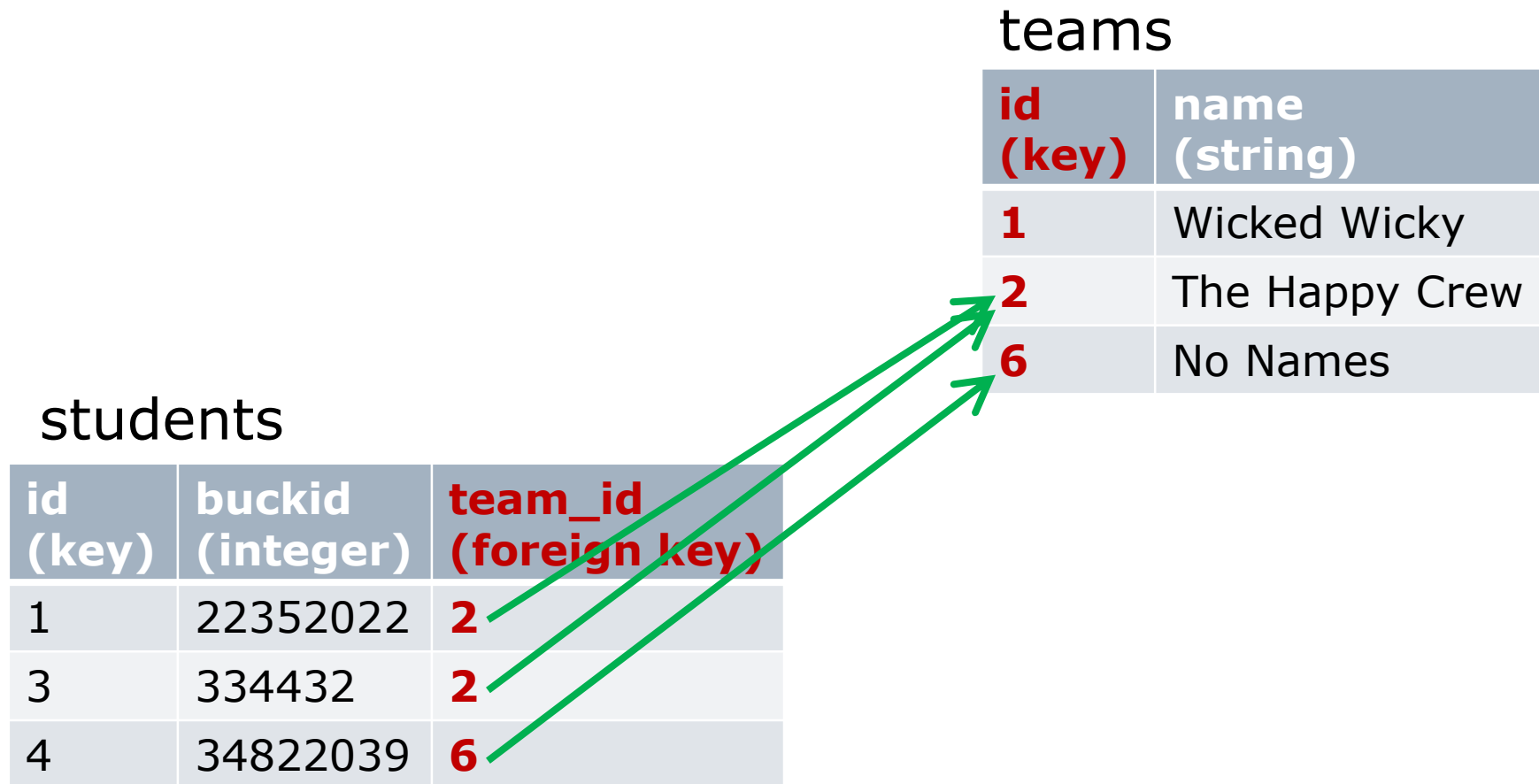

Working With Models

```
> s = Student.new
> s2 = Student.new fname: "Jo"
> s3 = Student.new fname: "Xi",
           buckid: 23
> Student.all #=> ?
```

Working With Models

```
> 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
```

Associations (1:N Relationship)



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
               # have Student#team method
end
```

```
class Team < ApplicationRecord
  has_many :students # note plural form
               # have Team#students method
end
```

Association Methods

- *Belongs_to* creates method for accessing owner

```
@student = Student.find 1 #=> 22352022
```

```
@student.team #=> 'The Happy Crew'
```

```
@student.team.name = 'The(tm) Happy Crew'
```

- *Has_many* creates method for accessing members

```
@team = Team.find 1
```

```
@team.students #=> array of students
```

```
@team.students.first
```

```
@team.students.size
```

```
@team.students.destroy_all
```

```
@team.students.any? { |s| ... }
```

Asymmetry in Writes to Assoc.

- Add a student to a team's association:
student automatically saved (assuming
team is stored in database)

```
t = Team.find 1
```

```
t.students #=> []
```

```
t.students << Student.new # gets an id
```

```
t.students #=> [#<Student id: 1, ...>]
```

- Assign a team student's association:
student is not automatically saved

```
s = Student.find 1
```

```
s.team = my_team
```

```
s.reload #=> s's 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 Project_Group
  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 dependents
    # when parent 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 the 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 can not be saved
- ```
student = Student.new lname: 'Vee'
student.valid? #=> false (no buckid)
student.save #=> false
```

# 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's 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?**

# Example

```
class Post < ApplicationRecord

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

end
```

# Validates Method in Model

`validates :column, condition`

- ❑ Uniqueness
  - `uniqueness: true`
  - `uniqueness: {message: 'Username already taken'}`
- ❑ Non-nullness (not the same as being true!)
  - `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