WEEK 5

RUBY + SQL

CLASSES & METHODS

"Hello, world!"

"Tacos are delish."

"This is the way."

5

2

[4, 8, 15, 16, 23, 42]

["Rachel", "Monica", "Phoebe", "Ross", "Chandler", "Joey"]

{ color: "purple", number: 17, computer: "Apple" }

{ name: "Ben", location: "Chicago, IL", status: "Staying warm!" }

"Hello, world!"
"Tacos are delish."
"This is the way."

} **`String`**

5
2

} **`Integer`**

[4, 8, 15, 16, 23, 42]
["Rachel", "Monica", "Phoebe", "Ross", "Chandler", "Joey"]

} **`Array`**

{ color: "purple", number: 17, computer: "Apple" }
{ name: "Ben", location: "Chicago, IL", status: "Staying warm!" }

} **`Hash`**

A CLASS...

- is like a factory
  - it builds similar types of things (aka *instances of the class*)
- begins with a capital letter
- is named in the singular
  - a "Car" factory builds cars (not a "Cars" factory)
- defines shared behavior

EXAMPLE

- *5* and *2* are both integers - instances of the *Integer* class.
- Although they have different values, they have shared methods as defined by the *Integer* class.
- For example, *to_f* converts an integer to a float:

```
5.to_f
# 5.0

2.to_f
# 2.0
```

HTTPS://GITHUB.COM/ENTR451-WINTER2026/RUBY-AND-SQL

REMINDER HOW TO:
SHARE YOUR GITPOD WORKSPACE

code-along/0-classes.rb

WRITING SQL WITH RUBY

WRITING SQL WITH RUBY

Why am I supposed to care?

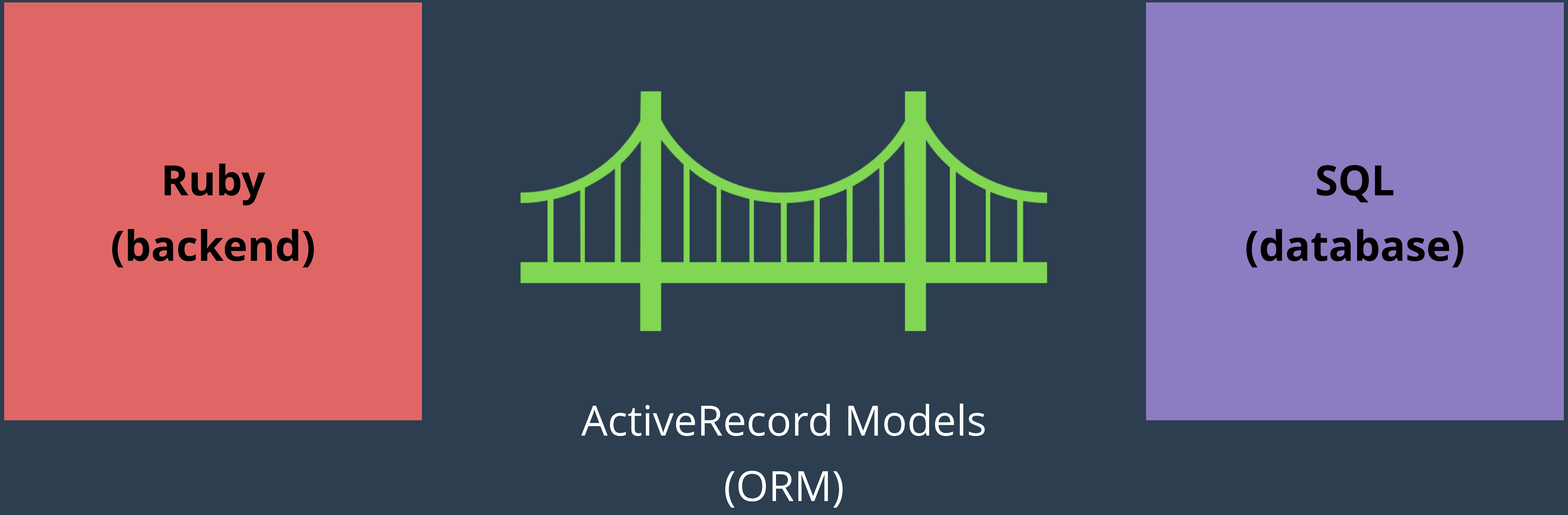
WRITING SQL WITH RUBY

- It's easier
- It's more consistent
- It's more flexible

WRITING SQL WITH RUBY

- It's easier
  - SQL is ugly
  - SQL is not very "programming-like"
- It's more consistent
  - Organizational perspective – labor/cost savings
  - Leverage existing expertise
- It's more flexible
  - Change your database, don't have to rewrite
  - Different database for development vs. production

MODELS


Ruby
(backend)
ActiveRecord Models
(ORM)
SQL
(database)

A MODEL…

- is a class dedicated to representing a table in the database
  - the model name is singular
  - the table name is plural

CREATING TABLES

create a table *things* with columns for *name* and *age*.

```
CREATE TABLE things (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT,
  age INTEGER
);
```

now in ruby code!

CREATING TABLES IN RUBY
(AKA MIGRATIONS)

1. generate the model and database migration files.

```
rails generate model Thing
```

2. open up the generated *db/migrate* file.

3. add the relevant columns (commented out code is generated by step 1 above).

```
# class CreateThings < ActiveRecord::Migration
#   def change
#     create_table :things do |t|
        t.string "name"
        t.integer "age"

#       t.timestamps
#     end
#   end
# end
```

4. execute the migration file.

```
rails db:migrate
```

MODIFYING TABLES IN RUBY

(AKA MIGRATIONS)

1. generate the model and database migration files.

```
rails generate migration AddSomethingToThing
```

** Note, migration file name must be unique*

2. open up the generated *db/migrate* file.

3. add or remove columns (commented out code is generated by step 1 above).

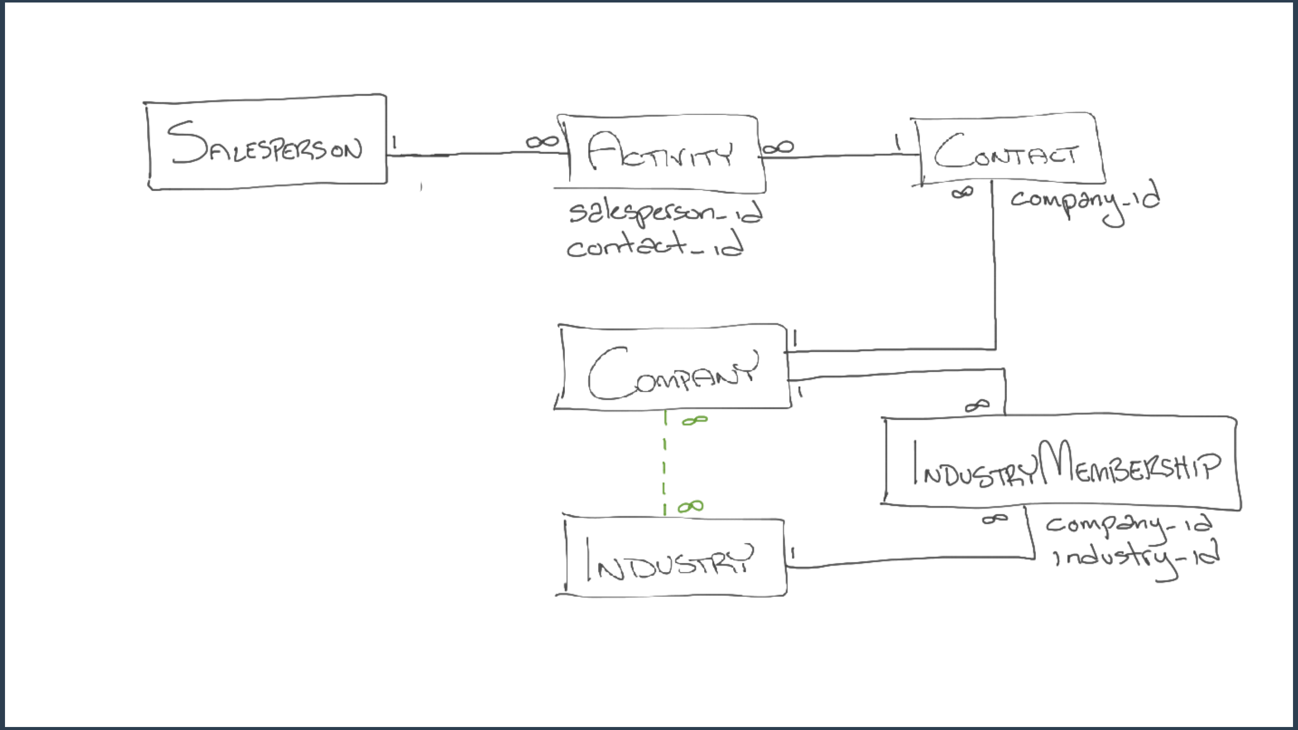
```
# class AddSomethingToThing < ActiveRecord::Migration
#   def change
      add_column "things", "something", "string"
      remove_column "things", "something", "string"
#   end
# end
```

4. execute the migration file.

```
rails db:migrate
```

CRM - USER STORIES & WIREFRAMES

CRM - DOMAIN MODEL & SQL



code-along/1-migrations.rb

A MODEL...

- is a class dedicated to representing a table in the database
  - the model name is singular; the table name is plural
- has built-in methods to:
  - query rows in the table
  - read a single row's column values
  - create a new row
  - update a row
  - delete a row
- writes SQL so we don't have to

SQL	ActiveRecord (Ruby)	Note
SELECT * FROM things	Thing.all	returns an array of all rows
SELECT COUNT(*) FROM things	Thing.all.count	returns an integer

SELECT * From things
WHERE name = "It"
	Thing.where({"name" => "It"})	returns an array of rows
		matching criteria

SELECT * FROM things
WHERE id = 1 LIMIT 1
	thing = Thing.find_by({"id" => 1})
	thing["name"]
		returns a single row
		and reads column from row

INSERT INTO things (name)
VALUES ("It")
	thing = Thing.new
	thing["name"] = "It"
	thing.save
		inserts column(s) as
		a new row into table

UPDATE things
SET name = "That"
	thing["name"] = "That"
	thing.save
		updates column(s) in an
		existing row

DELETE FROM things
WHERE id = 1
	thing.destroy	deletes a single row

code-along/2-models.rb

ASSOCIATIONS

(AKA RELATIONSHIPS BETWEEN MODELS)

code-along/3-associations.rb

ASSIGNMENT #2

- Posted in Canvas

NEXT WEEK

- Intro to Web Apps

REMINDER

- "Crash course" on HTML by next week