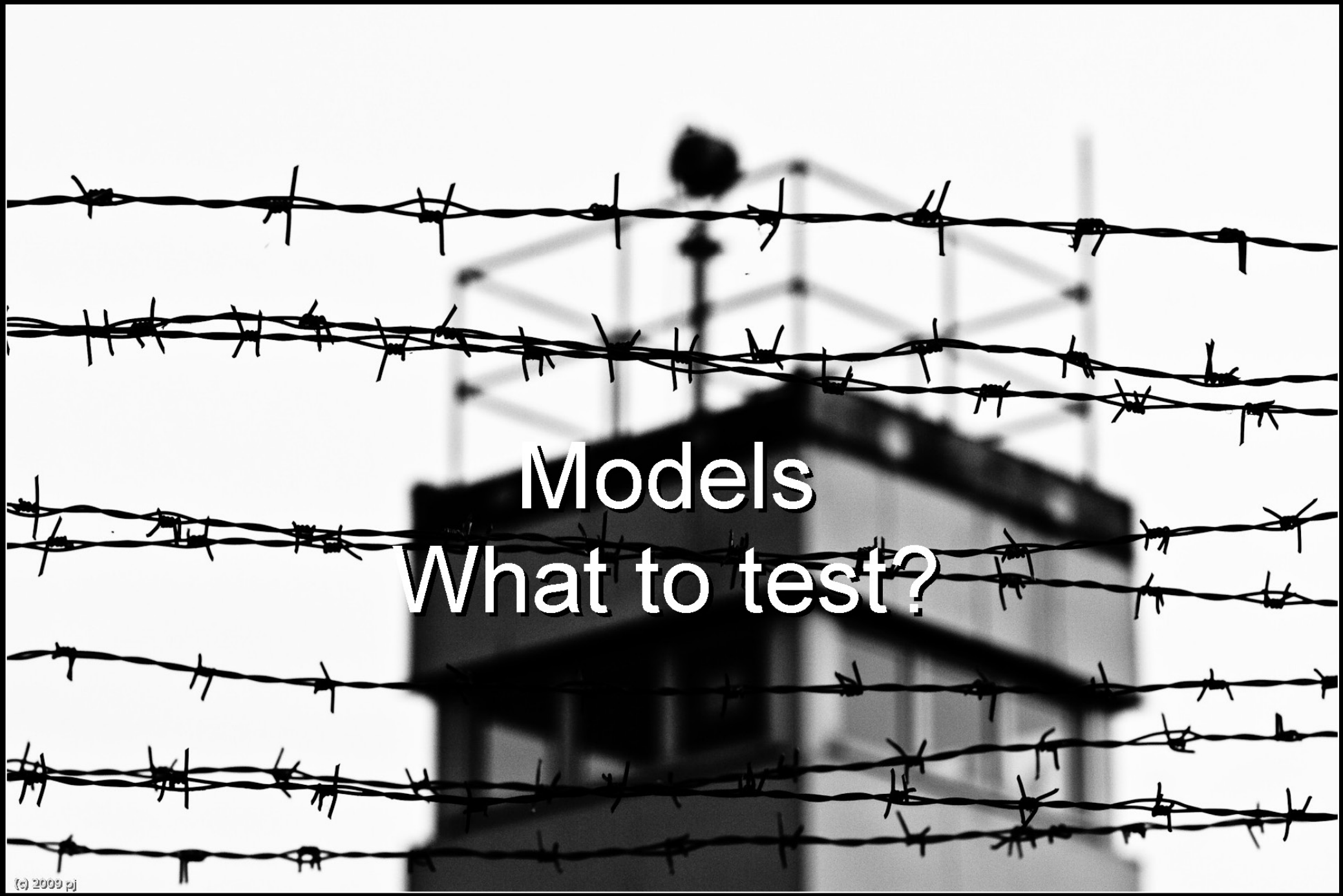
RubyFocus

Efficient Rails Test-Driven Development — Week 2

Wolfram Arnold
www.rubyfocus.biz

In collaboration with:
Sarah Allen, BlazingCloud.net
marakana.com


Models
What to test?
(c) 2009 pj

Test Models for...

RubyFocus

- validation
- side-effects before/after saving
- associations
- association proxy methods
- named scopes, custom finders
- nested attributes
- observers
- custom methods

Homework Last Week

RubyFocus

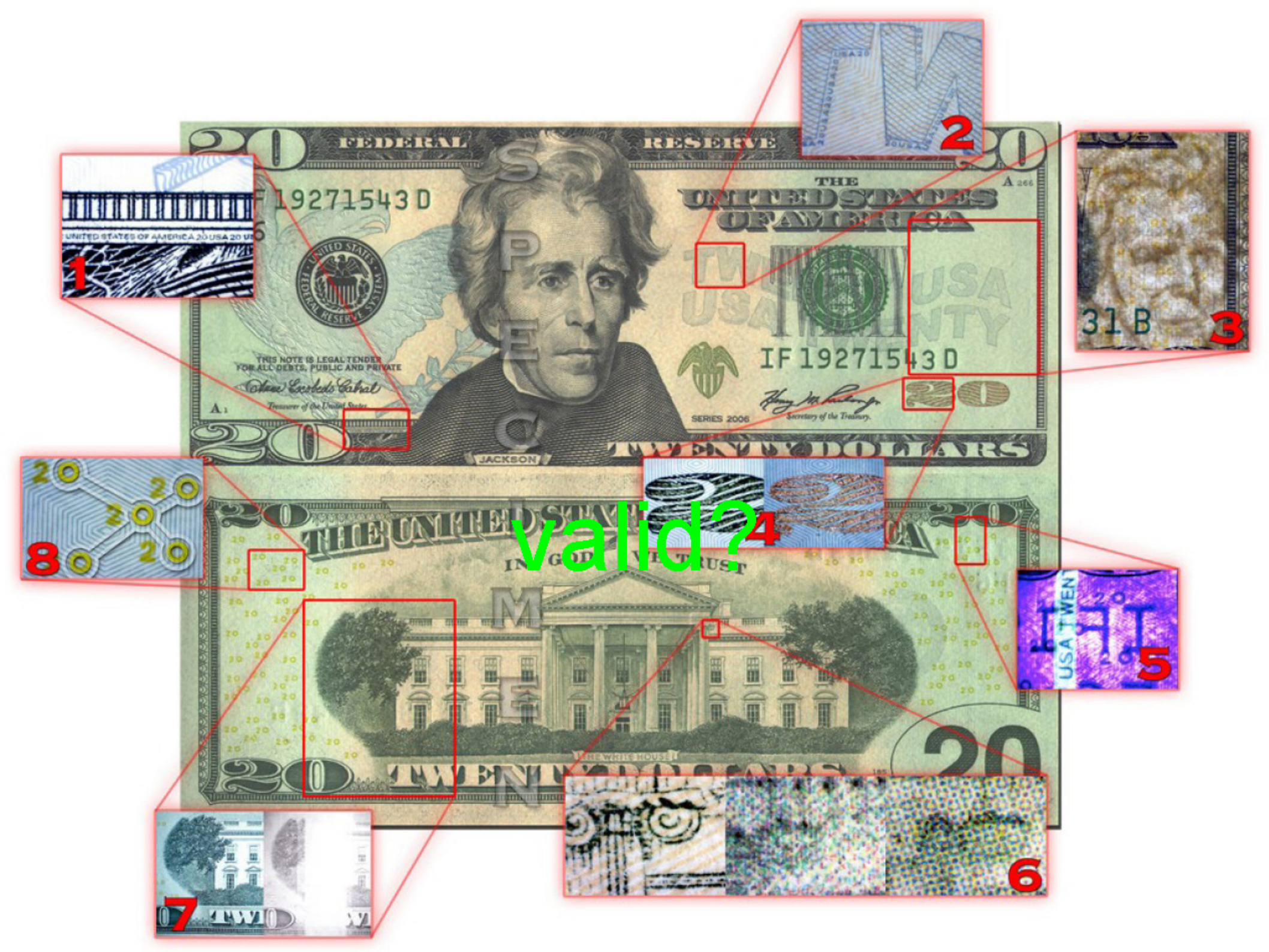
A Person should save correctly.

A Person can have many Addresses.

A Person's Address must have a street, city and zip.

A Person's Address can have an optional country.

If the country is left blank, it should default to “USA” prior to saving.


valid?
1
2
3
4
5
6
7
8

Model Validation

RubyFocus

Requirement:

presence, format, numerical?, length, uniqueness, ...

Verification:

valid?

errors.on(*attr)*

Enforcement:

validates_presence_of

validates_format_of

validates_numericality_of

validates_length_of

validates_uniqueness_of

...

How to Test for Validations?


```
it 'requires X' do
   n = Model.new
   n.should_not be_valid
   n.errors.on(:x).should_not be_nil
end
```

- Instantiate object with invalid property
- Check for not valid?
- Check for error on right attribute


Check for Side Effects

Model Callbacks

RubyFocus

Requirement:

- Default a value before saving
- Send an email after saving
- Post to a URL on delete
- ...

Callbacks:

- before_save
- after_save
- after_destroy
- ...

How to test Callbacks?

Through their Side Effects:

- Set up object in state before callback
- Trigger callback
- Check for side effect

```
it 'encrypts password on save' do
  n = User.new
  n.should_not be_valid
  n.errors.on(:x).should_not be_nil
end
```

How are Callbacks triggered?

Callback	Trigger event
before_validation	valid?
after_validation	valid?
before_save	save, create
after_save	save, create
before_create	create
after_create	create
before_destroy	destroy
after_destroy	destroy
after_find (see docs)	find
after_initialize (see docs)	new

Custom Finders

Custom Finders

RubyFocus

Requirement: Auto-suggest search

- Model requirement:
- A method to find records based on partial match on first or last name
- Typing “Jon” → Peter Jones, Jona Smith

Need:

- Fake data

How to test Custom Finders?

- Create a the smallest *non-trivial* dataset
- Check dataset is non-trivial
- Run finder
- Check results include a subset of dataset

```
it 'finds names with “Jo”' do
   @objs = [Factory(..), Factory(..), ...]
   Person.all.should == @objs
   res = Person.find_custom(“Jo”)
   res.should == [...]
end
```

Factories

Gem: github.com/thoughtbot/factory_girl

```
Factory.define (:person) do |p|
  p.first_name “Joe”
  p.last_name “Smith”
end
```

Important: Define factories for *valid* objects

named_scope

Better than: `def find_my_custom...`

Why?

Composable:

person.find_custom.addresses.count

Testable:

proxy_options

Associations

Model Associations

Requirement:

- Entities have relationships
- Given an object, I want to find all related objects

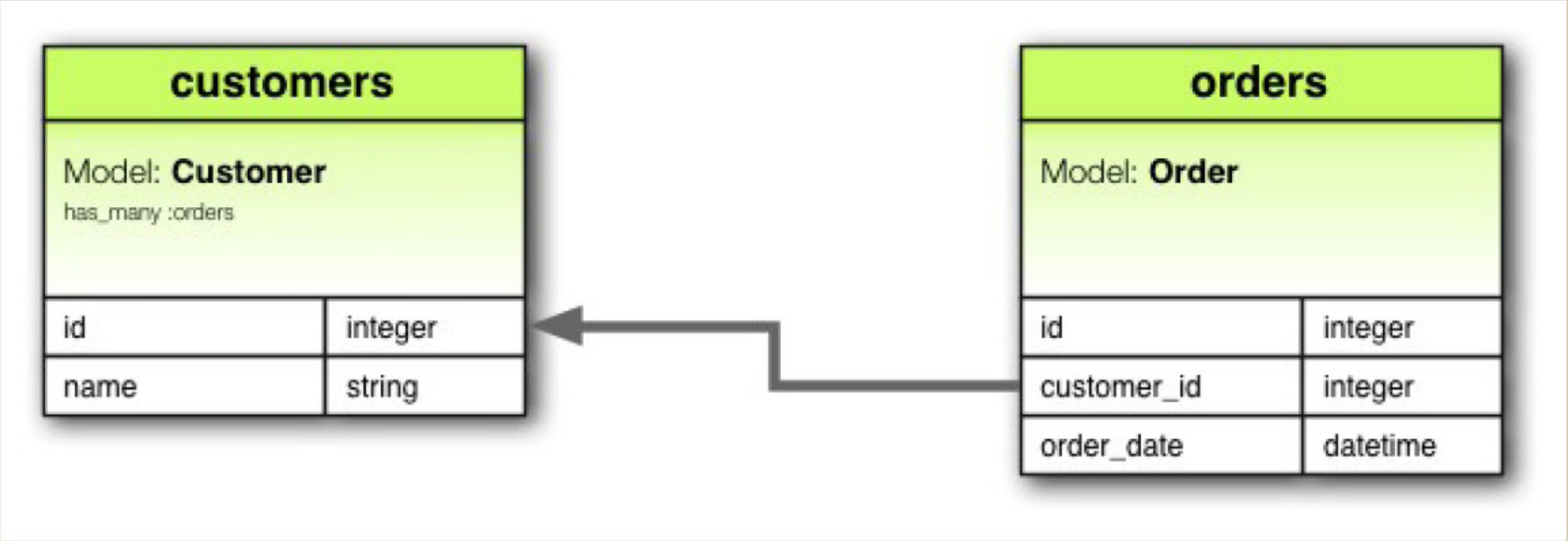
has_many

has_one

belongs_to

has_many :through

Tables and Associations




```ruby
class Customer < AR::Base
    has_many :orders
    ...
end
```

```ruby
class Order < AR::Base
    belongs_to :customer
    ...
end
```

Source: Rails Guides, http://guides.rubyonrails.org/association_basics.html

Migrations and Associations

RubyFocus

```
create_table :addresses do |t|
    t.belongs_to :person
    # same as:
    # t.integer :person_id
    ...
end


create_table :people do |t|
    ...
end
```

```
class Address < AR::Base
    belongs_to :person
    ...
end


class Person < AR::Base
    has_many :addresses
    ...
end
```

Association Methods

RubyFocus

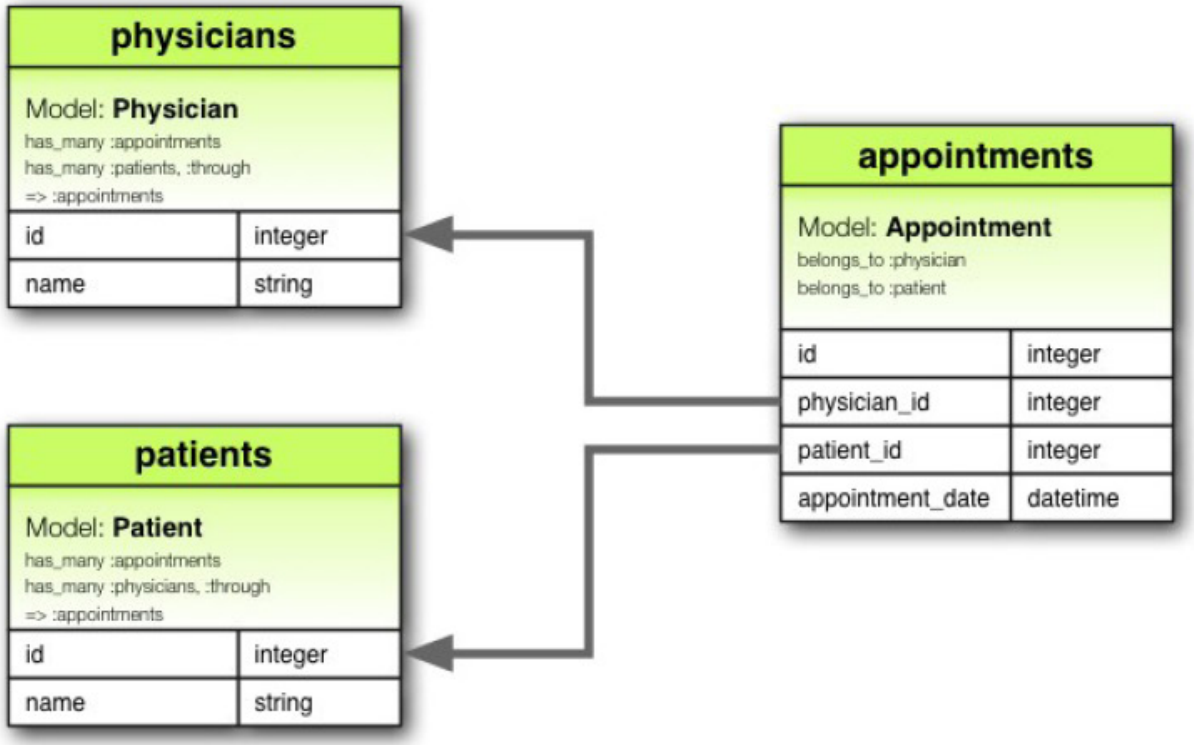
```
belongs_to :person
    .person
    .person =
    .build_person()
    .create_person()
```

```
has_many :assets
    .assets
    .assets <<
    .assets = [...]
    .assets.delete(obj,..)
    .assets.clear
    .assets.empty?
    .assets.create(...)
    .assets.build(...)
    .assets.find(...)
```

Source: http://api.rubyonrails.org/classes/ActiveRecord/Associations/ClassMethods.html

has_many :through

many-to-many relationships




```
class Physician < ActiveRecord::Base
  has_many :appointments
  has_many :patients, :through => :appointments
end

class Appointment < ActiveRecord::Base
  belongs_to :physician
  belongs_to :patient
end

class Patient < ActiveRecord::Base
  has_many :appointments
  has_many :physicians, :through => :appointments
end
```

Indices for Associations

RubyFocus

Rule: Any database column that can occur in a WHERE clause should have an index

```
create_table :addresses do |t|
  t.belongs_to :person
  # same as:
  # t.integer :person_id
  ...
end
add_index :addresses, :person_id
```

How to test for Associations?

- Are the association methods present?
- Checking for one is enough.
- No need to “test Rails” unless using associations with options
- Check that method runs, if options used

```
it “has many addresses” do
  p = Person.new
  p.should respond_to(:addresses)
end
```

Association Options

RubyFocus

Ordering

```
has_many :people, :order => "last_name ASC"
```

Class Name

```
belongs_to :customer, :class_name => "Person"
```

Foreign Key

```
has_many :messages, :foreign_key => "recipient_id"
```

Conditions

```
has_many :unread_messages,
  :class_name => "Message",
  :conditions => {:read_at => nil}
```

How to test Assn's with Options?

- Set up a non-trivial data set.
- Verify that it's non-trival.
- Run association method having options
- Verify result

```
it “sorts addresses by zip” do

   p = Factory(:person)
   # Factory for addrs with zip 23456, 12345
   Address.all.should == [addr1, addr2]
   p.addresses.should == [addr2, addr1]
   p.should respond_to(:addresses)
end
```

More Association Options

RubyFocus

Joins

```
has_many :popular_items,
   :class_name => “Item”,
   :include => :orders,
   :group => “orders.customer_id”,
   :order => “count(orders.customer_id) DESC”
```

Joins

RubyFocus

Items Table

id	name
1	Samsung TV
2	GE Dishwasher
3	Dell Laptop

Orders Table

id	customer_id	item_id
1	1	1
2	1	2
5	1	1
3	2	2
4	2	1

Inner Join


```
SELECT * FROM items INNER JOIN orders
                    ON items.id=orders.item_id


orders              items
id   name           id          item_id     customer_id
---  ----------     ----------  ----------  -----------
1    Samsung TV     1           1           1
2    GE Dishwas     2           2           1
2    GE Dishwas     3           2           2
1    Samsung TV     4           1           2
1    Samsung TV     5           1           1
```

Outer Join

RubyFocus

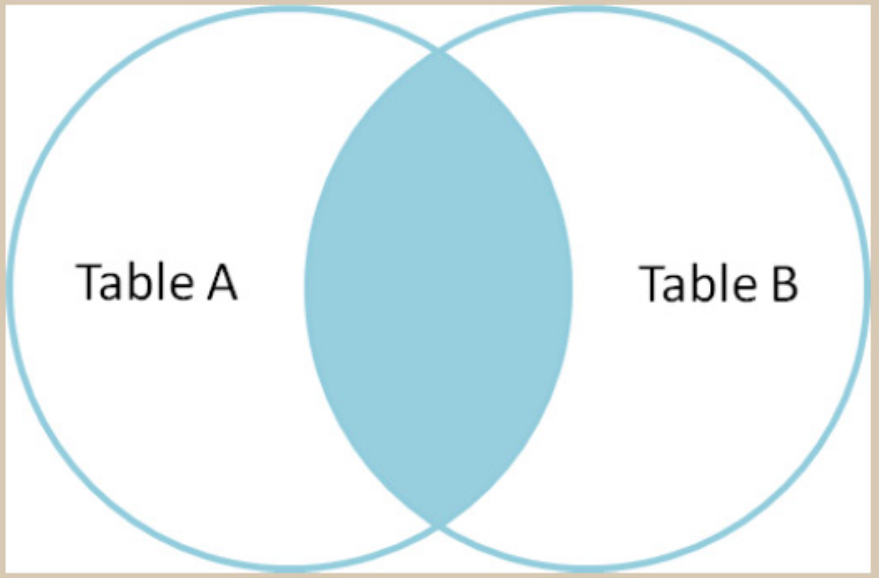
```
SELECT * FROM items LEFT OUTER JOIN orders
                    ON items.id=orders.item_id


items                orders
id   name            id          item_id     customer_id
---  ----------      ----------  ----------  ----------
3    Dell Laptop     NULL        NULL        NULL
1    Samsung TV      1           1           1
2    GE Dishwash     2           2           1
2    GE Dishwash     3           2           2
1    Samsung TV      4           1           2
1    Samsung TV      5           1           1
```

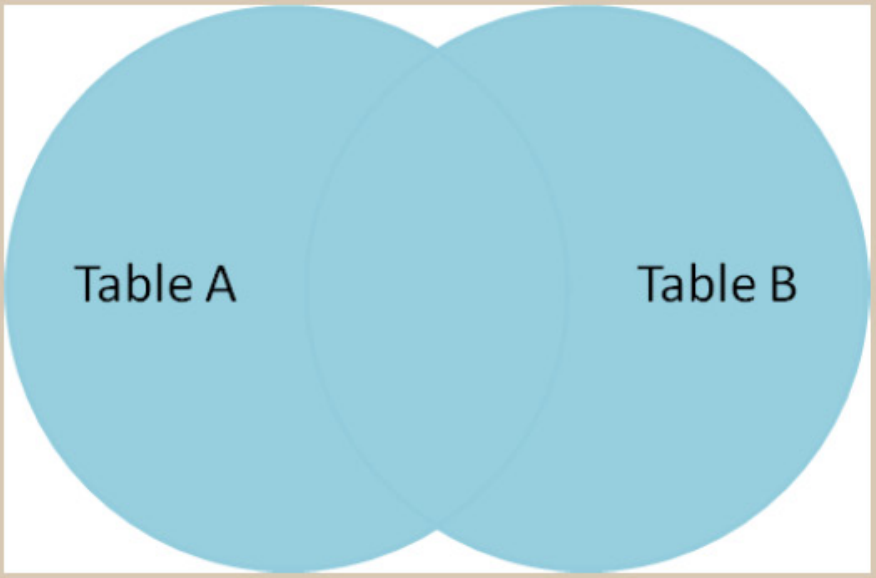
Inner Join vs. Outer Join

RubyFocus

Inner Join: Intersection



Outer Join: Superset



Source: http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

Grouping


```
SELECT *, COUNT(orders.item_id) AS cnt
                    FROM items LEFT OUTER JOIN orders
                    ON items.id=orders.item_id
                    GROUP BY orders.item_id
```

items

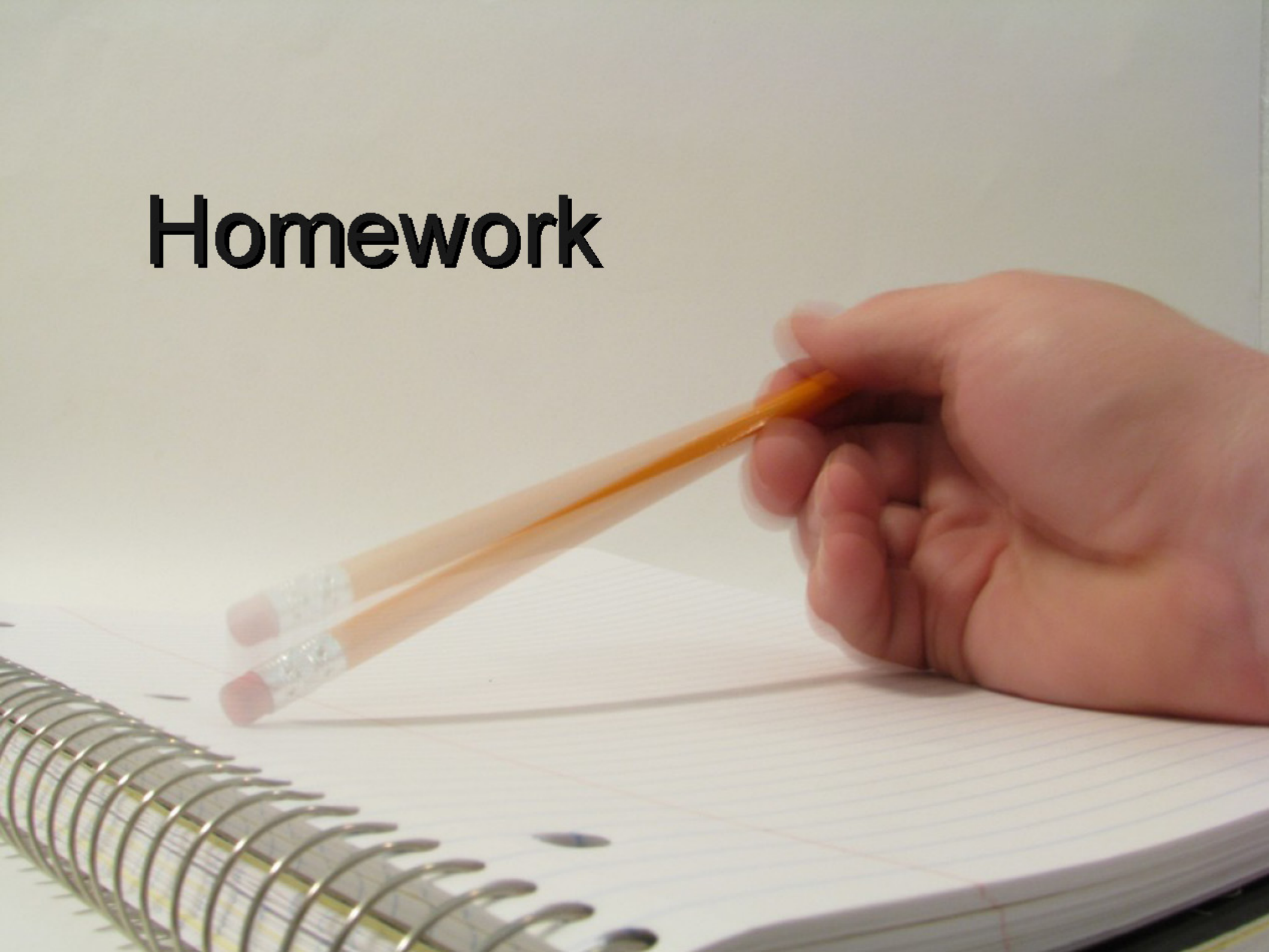
id	name	orders.id	orders.item_id	orders.customer_id	cnt
3	Dell Laptop	NULL	NULL	NULL	NULL
1	Samsung TV	1	1	1	3
2	GE Dishwash	2	2	1	2

Grouping generally requires an aggregate function, e.g. count.

How to test Assication Joins?

- Set up a non-trivial data set.
- Verify that it's non-trival.
- Run association method having options
- Verify result

Homework



Homework

RubyFocus

Addresses have a 2-letter State field.

The system can store orders by a customer.

A customer is a person in the database.

An order can consist of many items.

An order requires a customer and at least one item.

An item has a name, description and a price field. The price should accommodate 2 decimals. Name and price are required.

Given a person, I want to find all the items they bought.

Extra Credit I: The system can retrieve a ranked list of most popular items (i.e. most often ordered items).

Extra Credit II: For a loyalty program, the system must retrieve a list of customers who ordered 2 or more items in the last 90 days.

Recommended Reading

http://guides.rubyonrails.org/association_basics.html

http://guides.rubyonrails.org/active_record_querying.html

http://guides.rubyonrails.org/activerecord_validations_callbacks.html

RSpec Book: Chapter 13

RSpec Book: Chapter 12 (for background)

Flickr Photo Attribution

Watch Tower/East German Border
http://www.flickr.com/photos/peter_jakobs/3965422831/sizes/l/in/photostream/

$20 Bill Security features:
http://www.flickr.com/photos/jackofspades/4500411648/sizes/l/

Drugs
http://www.flickr.com/photos/monster/132597882/sizes/o/

Flickr Relationship map
http://www.flickr.com/photos/gustavog/9708628/

Signpost
http://www.flickr.com/photos/jenny-pics/4266714722/sizes/l/

Notebook with pencil
http://www.flickr.com/photos/tomsaint/2987926396/sizes/l/