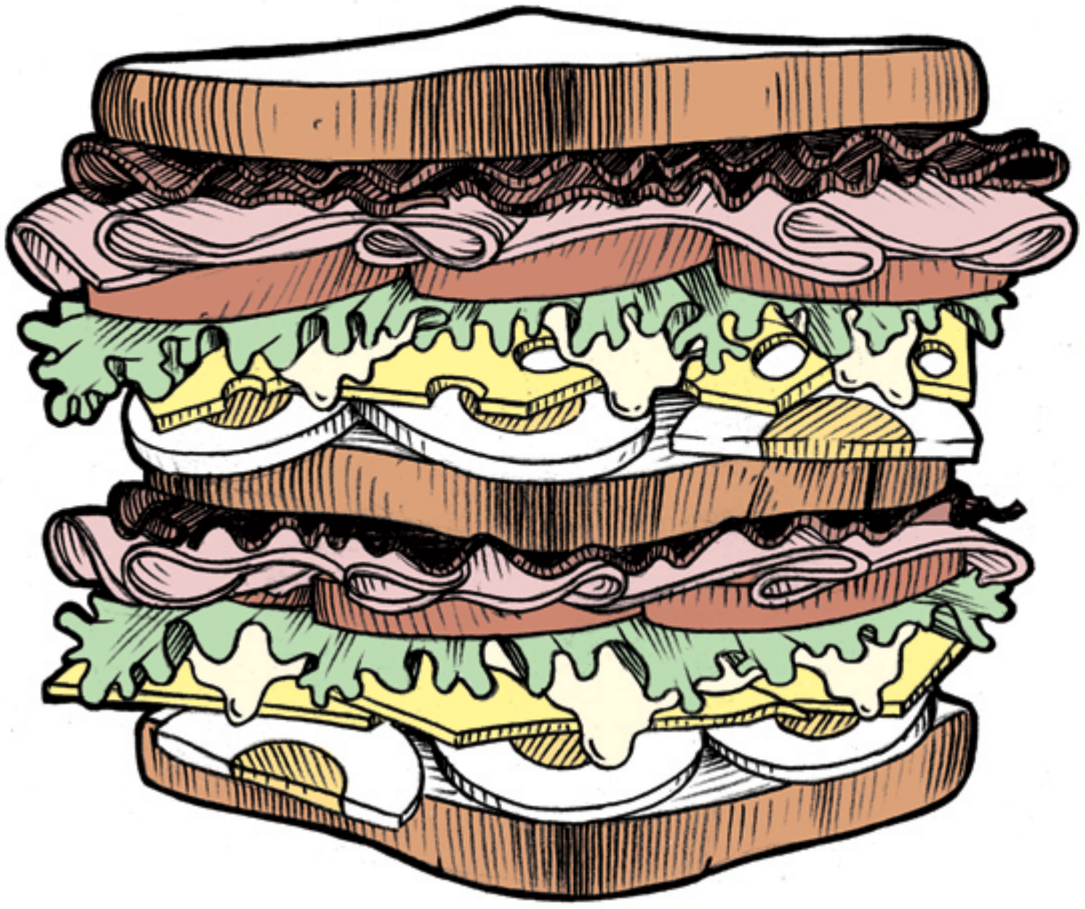
my search



for the perfect sandwich

combine

to unite, yoke together

…then requirements change

ORM

Caching
Structure
Migrations
Queries
Validations
Data manipulation
Callbacks
Caching
Relationships
DB Abstraction
4OZ.

compose

put together, arrange

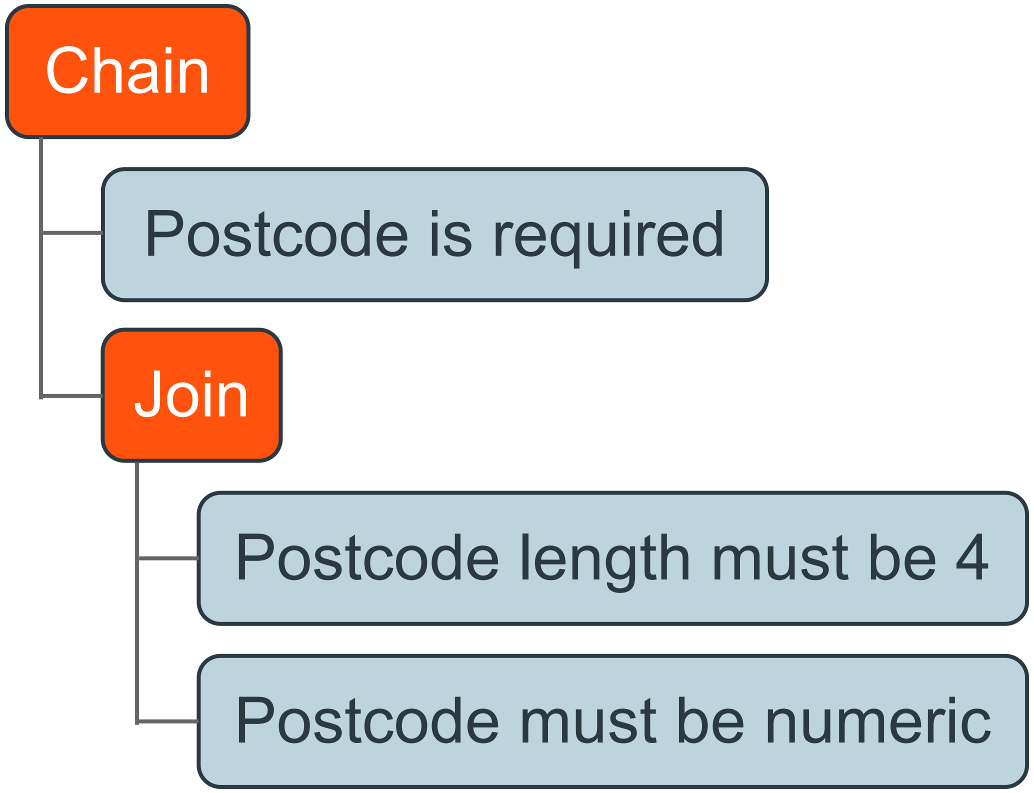
change becomes trivial

ORM

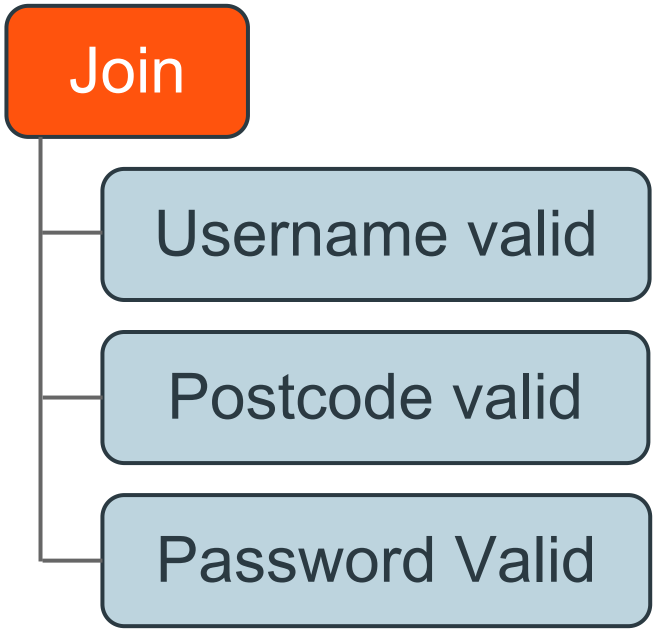
Caching
Structure
Migrations
Queries
Validations
Data manipulation
Callbacks
Caching
Relationships
DB Abstraction

```ruby
attr_accessor :being_edited

validates_presence_of :password,
  :unless => lambda{|u| u.being_edited }
```

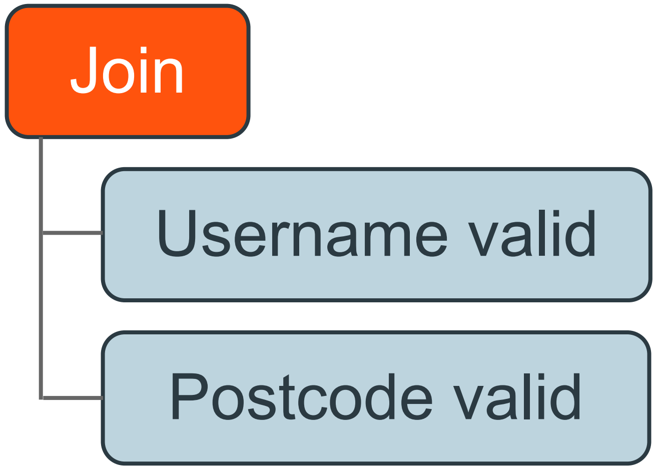
```clojure
(fact "When no name is provided an error should be given."
  (validate name-required {:age 2}) => ["Name may not be nil"])

(fact "When a name is provided there should be no errors."
  (validate name-required {:name "Fred" :age 2}) => [])
```


Chain
Postcode is required
Join
Postcode length must be 4
Postcode must be numeric

Signup



Edit profile



```
attr_accessor :being_edited

validates_presence_of :password,
  :unless => lambda{|u| u.being_edited }
```

```clojure
(def name-required
  {:validation :predicate
   :predicate  nil?
   :selector   [:name]
   :message    "Name may not be nil"})
```

```
(defmulti validate (fn [v d] (v :validation)))

(defmethod validate :predicate
  [{:keys [predicate selector message]} data]
  (let [value    (get-in data selector)
        invalid? (predicate value)]
    (if invalid? [message] [])))
```

```clojure
(defn arbitrary-validation [data]
  (if (and (first-thursday-of-the-month?)
           (the-clock-tower-has-struck-three?)
           (the-lonely-wolf-howls-in-the-night?)
           (has-crystals-aligned? data))
    [] ["Data is invalid"]))
```

```
(defrecord Predicate [selector predicate message]
  Validation
   (validate [{:keys [selector predicate message]} data]
     (if (predicate (get-in data (flatten selector)))
       [message] [])))
```

```clojure
(extend-type clojure.lang.IFn
  Validation
   (validate [self data]
     (self data)))
```

```clojure
(def name-required
  {:validation :predicate
   :predicate  nil?
   :selector   [:name]
   :message    "Name may not be nil"})
```

```
(defn present [name selector]
  (predicate selector str/blank?
             {:type ::present :name name :selector selector}))
```

```
(defmethod translate :vlad.validations/present
  [{:keys [name]}]
  (format "%s is required." name))
```