Advanced Querying

Exact Searches

- We already know about basic retrieves:
  - `find(id)` or `find(id1, id2)`
  - `find_by(hash)`
  - `where(hash)`

These are nice if you know EXACTLY what you are looking for…

```
hazink1-ml1:advanced_ar hazink1$ rails c
Loading development environment (Rails 4.1.1)
irb(main):001:0> bron = Person.find_by last_name: "James"
  Person Load (0.1ms)  SELECT  "people".* FROM "people"  WHERE "people"."last_name" = 'James' LIMIT 1
=> #<Person id: 7, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-17 03:14:44", upda
ted_at: "2014-06-17 03:14:44">
```

Including SQL fragments

- Can specify a SQL fragment (as opposed to `hash`) inside the `where` and `find_by`

```
irb(main):001:0> Person.where("age BETWEEN 30 and 33").to_a
  Person Load (2.6ms)  SELECT "people".* FROM "people"  WHERE (age BETWEEN 30 and 33)
=> [#<Person id: 8, first_name: "Kalman", last_name: "Smith", age: 33, created_at: "20
06-17 03:51:50", updated_at: "2014-06-17 03:51:50", login: "kalman", pass: "abc123">,
erson id: 14, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-
03:51:51", updated_at: "2014-06-17 03:51:51", login: "lebron", pass: "lejames">]
```

```
irb(main):002:0> Person.find_by("first_name LIKE '%man'")
  Person Load (0.3ms)  SELECT  "people".* FROM "people"  WHERE (first_name LIKE '%man') LIMIT 1
=> #<Person id: 1, first_name: "Kalman", last_name: "Smith", age: 33, created_at: "2014-06-17 03:14:44", updated_at:
 "2014-06-17 03:14:44">
```

- Very powerful, but beware of SQL injection!

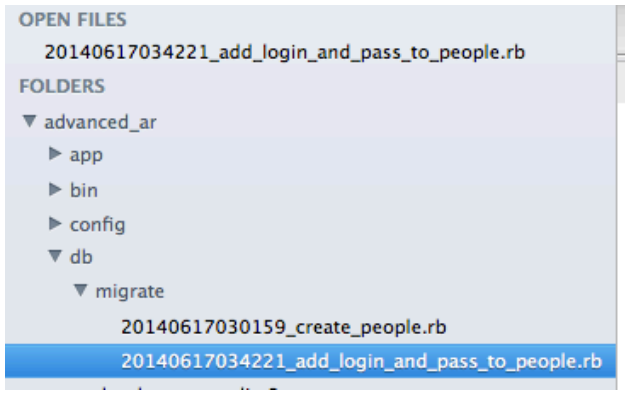
What is a SQL injection?

- Manipulating raw SQL to hack into a database.
- This includes maliciously dropping/deleting tables or gaining access to confidential information
- Wow! That sounds like fun – how easy is it to hack a database with one of these?

Modify people table

- Add login and pass fields to people table

```
hazink1-ml1:advanced_ar hazink1$ rails g migration add_login_and_pass_to_people login pass && rake db:migrate
      invoke  active_record
      create    db/migrate/20140617034221_add_login_and_pass_to_people.rb
== 20140617034221 AddLoginAndPassToPeople: migrating ==========================
-- add_column(:people, :login, :string)
   -> 0.0427s
-- add_column(:people, :pass, :string)
   -> 0.0004s
== 20140617034221 AddLoginAndPassToPeople: migrated (0.0432s) =================
```



20140617034221_add_login_and_pass_to_people.rb

```
1  class AddLoginAndPassToPeople < ActiveRecord::Migration
2    def change
3      add_column :people, :login, :string
4      add_column :people, :pass, :string
5    end
6  end
7
```

Reload data

```
1  Person.destroy_all
2
3  Person.create [
4    { first_name: "Kalman", last_name: "Smith", age: 33, login: "kalman", pass: "abc123" },
5    { first_name: "John", last_name: "Whatever", age: 27, login: "johnw", pass: "123abc" },
6    { first_name: "Michael", last_name: "Smith", age: 15, login: "michael", pass: "password" },
7    { first_name: "Josh", last_name: "Oreck", age: 57, login: "josh", pass: "not_telling" },
8    { first_name: "John", last_name: "Smith", age: 27, login: "john", pass: "secret" },
9    { first_name: "Bill", last_name: "Gates", age: 75, login: "bill", pass: "no_idea" },
10   { first_name: "LeBron", last_name: "James", age: 30, login: "lebron", pass: "lejames" }
11 ]
```

```
hazink1-ml1:advanced_ar hazink1$ rake db:seed
hazink1-ml1:advanced_ar hazink1$ _
```

Verifying the new data

```
sqlite> select * from people;
id          first_name  last_name   age         created_at                  updated_at                  login       pass
----------  ----------  ----------  ----------  --------------------------  --------------------------  ----------  ----------
8           Kalman      Smith       33          2014-06-17 03:51:50.771038  2014-06-17 03:51:50.771038  kalman      abc123
9           John        Whatever    27          2014-06-17 03:51:50.822071  2014-06-17 03:51:50.822071  johnw       123abc
10          Michael     Smith       15          2014-06-17 03:51:50.825872  2014-06-17 03:51:50.825872  michael     password
11          Josh        Oreck       57          2014-06-17 03:51:50.829914  2014-06-17 03:51:50.829914  josh        not_tellin
12          John        Smith       27          2014-06-17 03:51:51.040107  2014-06-17 03:51:51.040107  john        secret
13          Bill        Gates       75          2014-06-17 03:51:51.147159  2014-06-17 03:51:51.147159  bill        no_idea
14          LeBron      James       30          2014-06-17 03:51:51.187347  2014-06-17 03:51:51.187347  lebron      lejames
```

SQL Injection example

- **We want:** Pull out the information for a particular user based on his/her credentials
- **Hacker wants:** ALL users/passwords!

```
hazink1-ml1:advanced_ar hazink1$ rails c
Loading development environment (Rails 4.1.1)
irb(main):001:0> login = "lebron"; pass = "lejames"
=> "lejames"
irb(main):002:0> Person.where("login = '#{login}' AND pass = '#{pass}'")
  Person Load (0.8ms)  SELECT "people".* FROM "people"  WHERE (login = 'lebron' AND pass = 'lejames')
=> #<ActiveRecord::Relation [#<Person id: 14, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-17 03:51
:51", updated_at: "2014-06-17 03:51:51", login: "lebron", pass: "lejames">]>
irb(main):003:0> pass = "gotyou' OR 'x' = 'x"
=> "gotyou' OR 'x' = 'x"
irb(main):004:0> Person.where("login = '#{login}' AND pass = '#{pass}'")
  Person Load (0.2ms)  SELECT "people".* FROM "people"  WHERE (login = 'lebron' AND pass = 'gotyou' OR 'x' = 'x')
=> #<ActiveRecord::Relation [#<Person id: 8, first_name: "Kalman", last_name: "Smith", age: 33, created_at: "2014-06-17 03:51:
50", updated_at: "2014-06-17 03:51:50", login: "kalman", pass: "abc123">, #<Person id: 9, first_name: "John", last_name: "What
ever", age: 27, created_at: "2014-06-17 03:51:50", updated_at: "2014-06-17 03:51:50", login: "johnw", pass: "123abc">, #<Perso
n id: 10, first_name: "Michael", last_name: "Smith", age: 15, created_at: "2014-06-17 03:51:50", updated_at: "2014-06-17 03:51
:50", login: "michael", pass: "password">, #<Person id: 11, first_name: "Josh", last_name: "Oreck", age: 57, created_at: "2014
-06-17 03:51:50", updated_at: "2014-06-17 03:51:50", login: "josh", pass: "not_telling">, #<Person id: 12, first_name: "John",
 last_name: "Smith", age: 27, created_at: "2014-06-17 03:51:51", updated_at: "2014-06-17 03:51:51", login: "john", pass: "secr
et">, #<Person id: 13, first_name: "Bill", last_name: "Gates", age: 75, created_at: "2014-06-17 03:51:51", updated_at: "2014-0
6-17 03:51:51", login: "bill", pass: "no_idea">, #<Person id: 14, first_name: "LeBron", last_name: "James", age: 30, created_a
t: "2014-06-17 03:51:51", updated_at: "2014-06-17 03:51:51", login: "lebron", pass: "lejames">]>
```

Array Condition Syntax

- Lets you specify SQL fragment with ? followed by values
- Automagically performs conversions on the input values and escapes strings in the SQL
- Immune to SQL injection
- Similar to a PreparedStatement in Java

Array Condition Syntax

```
hazink1-ml1:advanced_ar hazink1$ rails c
Loading development environment (Rails 4.1.1)
irb(main):001:0> Person.find_by "age BETWEEN ? AND ?", 28, 32
  Person Load (1.0ms)  SELECT  "people".* FROM "people"  WHERE (age BETWEEN 28 AND 32) LIMIT 1
=> #<Person id: 14, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-17 03:51:51", updated_at: "201
-17 03:51:51", login: "lebron", pass: "lejames">
irb(main):002:0> Person.where "age BETWEEN ? AND ?", 28, 32
  Person Load (0.3ms)  SELECT "people".* FROM "people"  WHERE (age BETWEEN 28 AND 32)
=> #<ActiveRecord::Relation [#<Person id: 14, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-17 0
:51", updated_at: "2014-06-17 03:51:51", login: "lebron", pass: "lejames">]>
irb(main):003:0> Person.where("age BETWEEN ? AND ?", 28, 32).first
  Person Load (0.2ms)  SELECT  "people".* FROM "people"  WHERE (age BETWEEN 28 AND 32)  ORDER BY "people"."id" ASC LIMIT 1
=> #<Person id: 14, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-17 03:51:51", updated_at: "201
-17 03:51:51", login: "lebron", pass: "lejames">
```

Array Condition Syntax issues

- Array Condition Syntax is “SQL Injection safe” and easy to use, but there are now 2 (small) problems:

1. You have to keep track of the order of parameters “hiding” behind the “?”.
2. If you have *n* “?” – you need to pass in *n* values, even if they are a reference to the same value

Hash Condition Syntax

- Instead of "?", you specify symbols which map to the values in the hash passed in as a second parameter

```
irb(main):001:0> Person.find_by "last_name = :name OR first_name = :name", name: "James"
  Person Load (0.9ms)  SELECT  "people".* FROM "people"  WHERE (last_name = 'James' OR first_name = 'James') LIMIT 1
=> #<Person id: 14, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-17 03:51:51", updated_at:
-17 03:51:51", login: "lebron", pass: "lejames">
irb(main):002:0> Person.find_by "last_name = ? OR first_name = ?", "James", "James"
  Person Load (0.2ms)  SELECT  "people".* FROM "people"  WHERE (last_name = 'James' OR first_name = 'James') LIMIT 1
=> #<Person id: 14, first_name: "LeBron", last_name: "James", age: 30, created_at: "2014-06-17 03:51:51", updated_at:
-17 03:51:51", login: "lebron", pass: "lejames">
```