MAY 13, 2015

POSTGRES FULL TEXT SEARCH IN ECTO

IN ENGINEERING BY MITCHELL HENKE

Text search is often a core feature of web APIs, and Postgres and Ecto can fill the need without having to add additional technology (such as ElasticSearch or Solr) to the stack. We can lean on Postgres' pg_trgm to provide "good enough" search in early stages of API development.

My example will be on a User model, where we will be searching by username, so we'll set up a simple Ecto Model first:

```
1  defmodule User do
2    use Ecto.Model
3
4    schema "users" do
5      field :username, :string
6    end
7  end
```

user.ex hosted with ❤ by GitHub view raw

To make sure the search remains fast as the table grows, we're going to index the field using Postgres' pg_trgm:

```
1  defmodule Repo.Migrations.UserSearch do
2    use Ecto.Migration
3
4    def up do
5      execute "CREATE extension if not exists pg_trgm;"
6      execute "CREATE INDEX users_username_trgm_index ON users USING gin (username gin_t
7    end
8
9    def down do
```

```
10      execute "DROP INDEX users_username_trgm_index;"
11    end
12  end
```

migration.exs hosted with ❤ by GitHub view raw

Inspired by Drew Olson's post, we'll give the Ecto Model a function so we can easily compose search queries. Ecto doesn't have any shorthand for a trigram query over a text column, but using Ecto fragments is straightforward enough:

```
1   defmodule User do
2     use Ecto.Model
3
4     schema "users" do
5       field :username, :string
6     end
7
8     def search(query, search_term) do
9       from(u in query,
10      where: fragment("? % ?", u.username, ^search_term),
11      order_by: fragment("similarity(?, ?) DESC", u.username, ^search_term))
12    end
13  end
```

user.ex hosted with ❤ by GitHub view raw

This query will compare the search term, and order by similarity. Let's see it in action:

```
1   User |> User.search("mitch") |> Repo.all
2   # [debug] SELECT u0."id", u0."username" FROM "users" AS u0 WHERE (u0."name" % $1) ORDER
3   [%User{__meta__: %Ecto.Schema.Metadata{source: "users", state: :loaded},
4     username: "mitch"},
5    %User{__meta__: %Ecto.Schema.Metadata{source: "users", state: :loaded},
6     username: "mitch3"}]
```

iex.exs hosted with ❤ by GitHub view raw

We've been building more APIs in Elixir and Phoenix, and there have been questions on the Ecto mailing list about doing full text search in Ecto, so I thought I'd write up how we've done it. Hopefully it was helpful :)

Feel free to contact me on Twitter at @mitchellhenke or IRC in # elixir-lang with the same name.

ADDENDUM:

Someone on Reddit asked how to change the limit of the `%` operator in Postgres, which I've usually done via the `set_limit()` function on each connection, since it resets to the default on new connections.

Ecto doesn't have an after connect hook (created an issue here), and the user needed to be able to set the limit per query. A solution to this is to use the `similarity` function for both the filter and the order:

```elixir
defmodule User do
  use Ecto.Model

  schema "users" do
    field :username, :string
  end

  def search(query, search_term, limit = 0.3) do
    from(u in query,
    where: fragment("similarity(?, ?) > ?", u.username, ^search_term, ^limit),
    order_by: fragment("similarity(?, ?) DESC", u.username, ^search_term))
  end
end
```

user.ex hosted with ❤ by GitHub view raw