

# Dealing with a Monster Ecto Query

Mackenzie Morgan

Axios

Elixir Wizards Conference 2021

# About Me

A year ago, I learned Elixir because we launched the Axios mobile app and immediately crashed Apollo.

A year ago, I learned Elixir because we launched the Axios mobile app and immediately crashed Apollo.  
Every morning.

A year ago, I learned Elixir because we launched the Axios mobile app and immediately crashed Apollo.

Every morning.

At 6AM.

# The problem

A single complex query with a lot of ORs is responsible for the majority of our database load.

# The problem

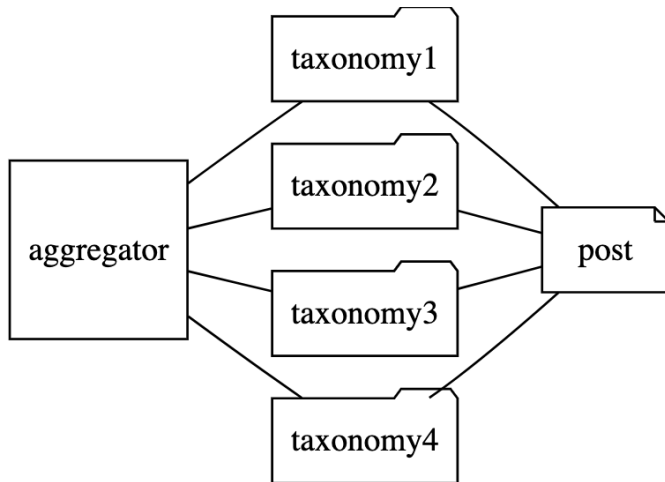
A single complex query with a lot of ORs is responsible for the majority of our database load.

And the biggest day in US political news is next week.

# The problem

A single complex query with a lot of ORs is responsible for the majority of our database load.

And the biggest day in US political news is next week.  
(the presidential election)





# Starting code

```
Repo.all(  
  from p in Post,  
    left_join: t1 in assoc(p, :taxonomy1 ),  
    left_join: t2 in assoc(p, :taxonomy2 ),  
    left_join: t3 in assoc(p :taxonomy3 ),  
    left_join: t4 in assoc(p :taxonomy4 ),  
    left_join: a1 in assoc(t1, :aggregators ),  
    left_join: a2 in assoc(t2, :aggregators ),  
    left_join: a3 in assoc(t3, :aggregators ),  
    left_join: a4 in assoc(t4, :aggregators ),  
  where:  
    a1.id == ^id or  
    a2.id == ^id or  
    a3.id == ^id or  
    a4.id == ^id  
)
```

# Break it down

What if we do 4 smaller queries?

# Break it down

What if we do 4 smaller queries?

```
Repo.all(  
  from p in Post,  
    left_join: t1 in assoc(p, :taxonomy1 ),  
    left_join: a1 in assoc(t1, :aggregators ),  
    where:  
      a1.id == ^id  
)
```

# Break it down

What if we do 4 smaller queries?

```
Repo.all(  
  from p in Post,  
    left_join: t1 in assoc(p, :taxonomy1 ),  
    left_join: a1 in assoc(t1, :aggregators ),  
    where:  
      a1.id == ^id  
)
```

Written out 4 times, once for each taxonomy

# Not DRY enough

What if we take advantage of atoms and the pin operator?

# Not DRY enough

What if we take advantage of atoms and the pin operator?

```
defp query_articles(id, taxonomy) do
  Repo.all(
    from p in Post,
      left_join: t in assoc(p, ^taxonomy),
      left_join: a in assoc(t, :aggregators ),
      where: a.id == ^id
  )
end
```

# Not DRY enough

What if we take advantage of atoms and the pin operator?

```
defp query_articles(id, taxonomy) do
  Repo.all(
    from p in Post,
      left_join: t in assoc(p, ^taxonomy),
      left_join: a in assoc(t, :aggregators ),
      where: a.id == ^id
  )
end
```

And call it 4 times, once for each taxonomy

# Just a little further

What if we use Elixir's famed concurrency?



# Just a little further

What if we use Elixir's famed concurrency?

```
taxonomies = [  
  :taxonomy1,  
  :taxonomy2,  
  :taxonomy3,  
  :taxonomy4  
]
```

```
Task.async_stream(  
  taxonomies,  
  fn taxonomy ->  
    query_stories(id, taxonomy)  
  end,  
  timeout: 30_000  
)  
> Enum.flat_map(fn  
  {:ok, posts} -> posts  
  _ -> []  
end)
```

# Results

- DB CPU utilization: 50%  $\rightarrow$  40% (-20%)
- Postgres analyze cost: 3614  $\rightarrow$  16
- Postgres analyze execution time: 8.424ms  $\rightarrow 0.125ms \times 4 = 0.5ms$
- Max requests per second: 700%
- Stress-free Election Night

# Find Me Online

- **Twitter:** @maco\_nix
- **GitHub:** @maco
- **Homepage:** [mackenzie.morgan.name](http://mackenzie.morgan.name)
- **Elixir Slack:** @maco