PostGIS on Rails

Matt Nemenman
@quarterdome

WHAT IS POSTGIS

- Geo-spatial extension to PostgreSQL
- Fast and easy location aware applications
- Scalable for app with complex location-aware business logic
- Scalable for large query volumes

In a Nutshell

- Geometry column type
  - Point, Linestring, Polygon, etc
- Spatial database index type (GiST)
- Hundreds of functions to operate on geometries in SQL
  - ... using spatial indices

How Fast and How Easy?

How Fast and How Easy?

- Should I bother for MySmallRailsApp or shall I do it old-fashioned way?

How Fast and How Easy?

- Should I bother for MySmallRailsApp or shall I do it old-fashioned way?
- Yes, you should (it is that easy!)

How Fast and How Easy?

- Should I bother for MySmallRailsApp or shall I do it old-fashioned way?
- Yes, you should (it is that easy!)
- Lets build a small app to prove it
  - No live coding (okay, it is not THAT easy!)

MySmallRailsApp

- Find all bars within the map viewport
  - NW and SE corners
- One million bars!
- Lets build it two ways
  - Using PostGIS
  - Old-fashioned way: using numeric to store location

Gemfile

- `gem 'pg'`
- `gem 'postgis_adapter', git: 'git://github.com/nofxx/postgis_adapter.git'`
- There are multiple alternative for `posgis_adapter`. Some are more maintained than other.

DATABASE.YML

```
development:
  adapter: postgresql
  template: template_postgis
```

Create Table

```
create_table :bars do |t|
  t.column :lat, :float
  t.column :lon, :float
  t.column :coordinates,
           :geometry, :srid => 4326
end
```

Create Indices

```
add_index :bars, :lat
add_index :bars, :lon
add_index :bars, [:lat, :lon]
add_index :bars, :coordinates,
                 :spatial => true
```

Model

- Nothing special here

```
class Bar < ActiveRecord::Base
  attr_accessible :lat,
                  :lon, :coordinates
  # ...
end
```

Seed the Test Data

```
1000000.times do
  lat = rand(-90.0 .. 90.0)
  lon = rand(-180.0 .. 180.0)
  coord = Point.from_lon_lat(lon, lat)
  Bar.create!(:lat => lat,
    :lon => lon,
    :coordinates => coord)
end
```

- `Point` is `GeoRuby::SimpleFeatures::Point`

SEARCH

- Old-fashioned way

```
def self.by_bbox(sw_lat, sw_lon,
                 ne_lat, ne_lon)
  self
    .where( :lat => sw_lat..ne_lat )
    .where( :lon => sw_lon..ne_lon )
end
```

SEARCH

- PostGIS way
- ```
  bbox_coords = [[[sw_lon, ne_lat],
                  [ne_lon, ne_lat],
                  [ne_lon, sw_lat],
                  [sw_lon, sw_lat],
                  [sw_lon, ne_lat]]]
  bbox = Polygon.from_coordinates(bbox_coords)
  self.where('ST_Intersects(coordinates, :bbox)',
                        :bbox => bbox)
  ```
- `Polygon is GeoRuby::SimpleFeatures::Polygon`
  ```

DB QUERIES

- ```
  SELECT "bars".* FROM "barss" WHERE
  ("bars"."lat" BETWEEN 35.487511 AND 40.33817)
  AND ("bars"."lon" BETWEEN -124.244385 AND
  -117.784424)
  ```

- ```
  SELECT "bars".* FROM "bars" WHERE
  (ST_Intersects(coordinates,
  '0103000020E6100000010000000500000082A8FB00A40
  F5FC0813E9127492B4440A48CB80034725DC0813E91274
  92B4440A48CB80034725DC05AB8ACC266BE414082A8FB0
  0A40F5FC05AB8ACC266BE414082A8FB00A40F5FC0813E9
  127492B4440'))
  ```

SO HOW FAST IS IT?

SO HOW FAST IS IT?

- `explain analyze` …

SO HOW FAST IS IT?

- `explain analyze` …
- old-fashioned:
  - cost=1008.80..2445.58; time=6.150..8.720

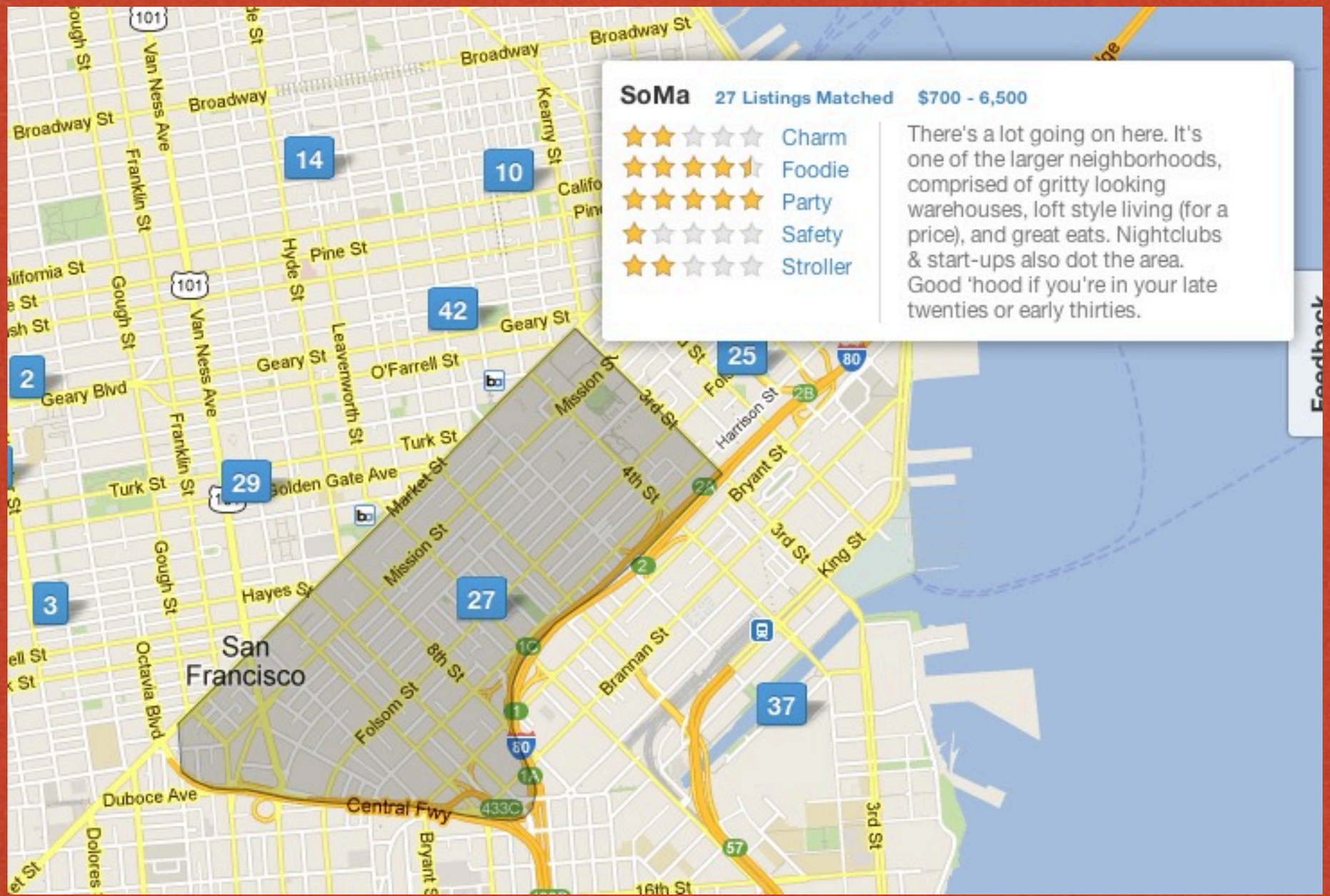
SO HOW FAST IS IT?

- `explain analyze` …
- old-fashioned:
  - cost=1008.80..2445.58; time=6.150..8.720
- PostGIS:
  - cost=16.15..1756.92; time=0.361..3.140

What else can PostGIS do

- Geometry overlap/intersection/touching detection and calculation
- Distance calculation and ordering
- 2D and 3D
- Geometry editing (stretch, expand, simplify)
- Encoding in industry formats (e.g. GeoJSON)

PostGIS in Action



How can I try PostGIS out?

- Postgres.app - http://postgresapp.com
- Heroku Postgres - https://devcenter.heroku.com/articles/is-postgis-available

QUESTIONS?

- Now is the time!