–Mom, can we have G**gle Maps?
–We have G**gle Maps at home

Jimmy Angelakos

About me

- Systems & Database Architect
- Based in Edinburgh, Scotland
- Open Source user & contributor (25+ years)
- PostgreSQL exclusively (16+ years)
- Author, PostgreSQL Mistakes and How to Avoid Them
- Co-author, PostgreSQL 16 Administration Cookbook
- `pg_statviz` PostgreSQL extension

What I am not and what this talk is not

- Not a GIS expert
- Not an in-depth analysis
- Not a detailed HOWTO
- GIS subject is too extensive

So what is this about?

- Awareness of the combined potential of:
  - PostgreSQL
  - GIS
  - PostGIS
  - OpenStreetMap
- I've tried these things, and so can you!

Geographic Information Systems (GIS)

- Context: Software for geographic data (geodata)
  - Stores
  - Manages
  - Analyzes
  - Edits
  - Outputs
  - Visualizes

What can you use GIS for?

- Besides the obvious: Storing maps
- Associating data with locations → geodata
- Applications:
  - Governance
  - Environmental science
  - Health
  - History and archaeology
  - Cultural and social study

What can you use GIS in the database for?

- Develop location-aware services
  - Search for a POI (e.g. ATM) in Chicago
  - Search for the nearest ATMs
  - Time, weather, events where I am
- Associate things with GPS coordinates
  - Perform spatial queries (esp. with joins)
- Routing (how to get from A to C via B)

How do I get geodata in my PostgreSQL?

- PostGIS: extension for geographical objects
  - Supports probably any kind of spatial type and query you can think of
  - Based on “light-weight geometries” for optimal indexing, memory footprint
  - Makes Postgres the de facto industry standard in spatial databases
  - Open Geospatial Consortium hasn’t certified it 🤡

PostGIS

- Postgres can instantly return spatial containment result
  - Is this point (set of coordinates) inside the area of this geographical feature (lake, city, etc.)?
- Distance calculations
  - How far away are these two points?
- Advanced spatial queries such as k-nearest neighbor search
  - What are the N nearest <candidate features> to <query feature>?

Where do I get this geodata?

- Proprietary data
  - MapQuest, HERE, Google Maps, TomTom, Bing Maps, ESRI, etc.
  - Service providers: Mapbox, Amazon Location Service, etc.
- Open data
  - OpenStreetMap (OSM)
  - Wikimapia (?)

INTERMISSION

- Let’s talk about OpenStreetMap

What's OpenStreetMap?

- Free & open geographic database
- Created by Steve Coast in 2004
  - Ordnance Survey refusing to release data
- Accelerated adoption in 2012
  - Google started charging for Maps
- Collaboratively updated & maintained by community
- Database hosted by OpenStreetMap Foundation

Why is OpenStreetMap important?

- The Wikipedia of geographical knowledge
- Governance
  - UK-based non-profit with local chapters (e.g. US)
- Licensing
  - Open Database License (OdbL)
  - Attribution, Share-Alike, Keep Open (copyleft)
- Used by tons of websites, apps, tools

What's in OSM data? (Data Primitives)

- **Nodes**: WGS84 coordinates
  Features without size like POI
- **Ways**: Ordered lists of Nodes → lines or polygons
  Features like streets (linear) or lakes (areas)
- **Relations**: Ordered lists of Nodes, Ways, Relations
  Represent relationships of above
- **Tags**: Key-Value pairs for metadata of above objects

Where does PostgreSQL come in?

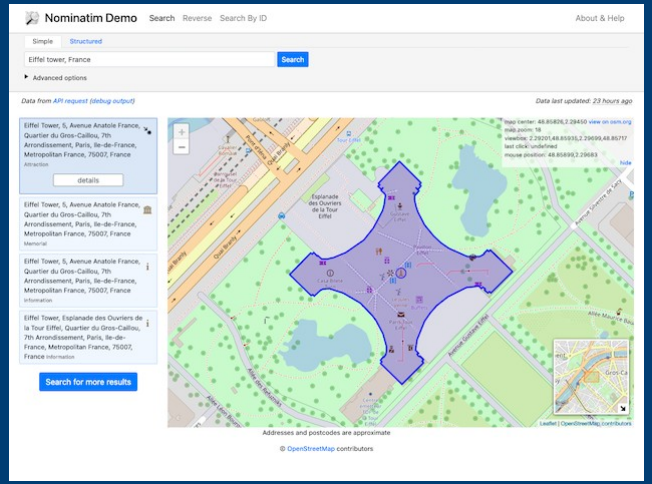
- OSM server uses PostgreSQL
- Tables of primitives
- Individual objects stored as rows
- Exports of data
  - Dumps of any size (incl. **`planet.osm`**)
  - Formats: PBF, XML

How can I use OpenStreetMap data?

- Direct access to objects (**`osm_id`**)
  - Spatial queries
- Geocoding
- Reverse geocoding
- Integrate map displays w/ a map server

INTERMISSION (again)

- Let’s talk about geocoding



What is geocoding?

- Search that returns the coordinates of a place/feature
  - By giving address or name
- Reverse: Search returns data on place/feature
  - By giving the coordinates

What are some geocoding tools?

- Nominatim
  - nominatim.openstreetmap.org
- Non-Postgres: photon
  - photon.komoot.io
- Others
  - wiki.openstreetmap.org/wiki/Geocoding

So what's the basic idea?

- Instead of relying on external/expensive Geodata APIs…
- Take this in-house by using OSM data inside PostgreSQL
- In conjunction with open source GIS tools

Getting the OSM data

```
transmission-cli -w . \
    -d 150000 \
https://planet.openstreetmap.org/pbf/planet-
latest.osm.pbf.torrent
```

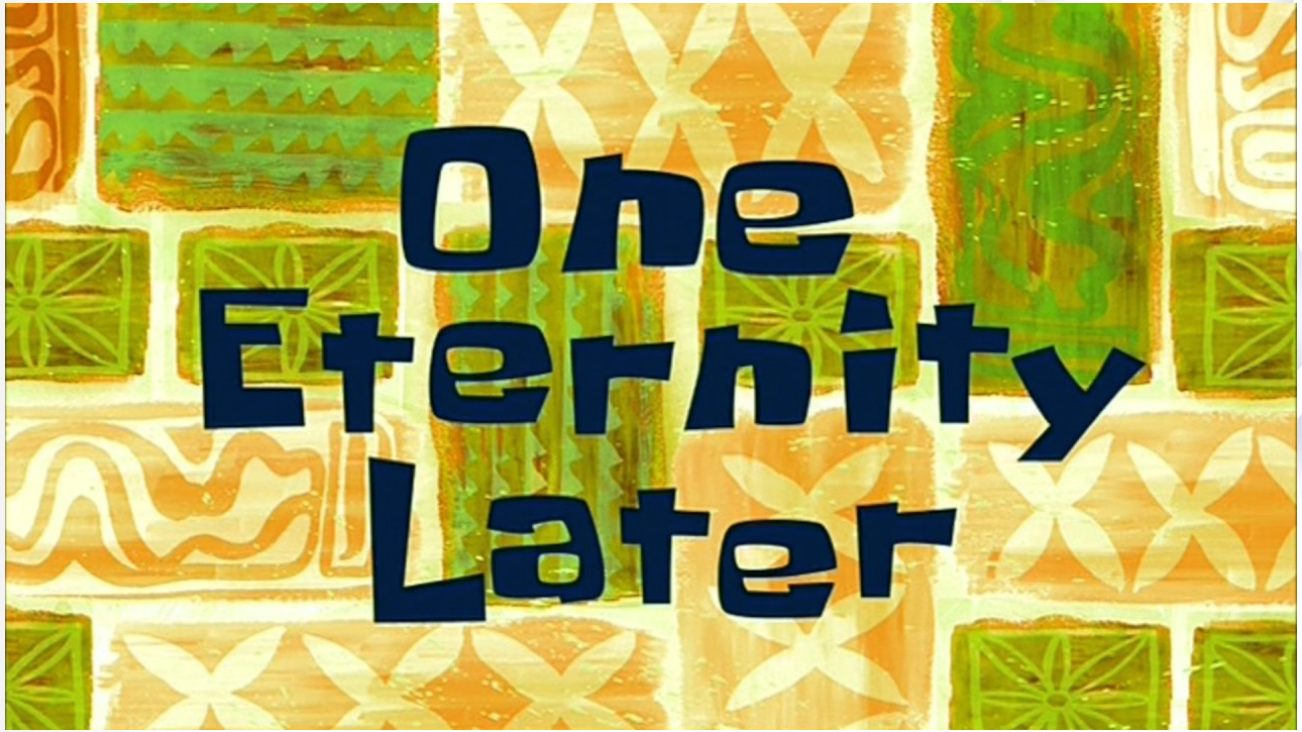
Getting the OSM data into Postgres

- There are standard ways of ingesting OSM data into PostgreSQL such as PgOSM Flex
- Ingestion takes 1.5 days for whole-planet data (~ 1TB)
- Subsequent updates to the data are much faster

PgOSM Flex

```
docker run --name pgosm -d -rm \
-v ~/devel/pgosm-data:/app/output \
-v /etc/localtime:/etc/localtime:ro \
-v ~/devel/pgosm-data/custom-layerset:/custom-layerset \
-e POSTGRES_PASSWORD=$POSTGRES_PASSWORD -p 5433:5432 \
-d rustprooflabs/pgosm-flex:latest \
-c shared_buffers=2GB -c work_mem=64MB -c maintenance_work_mem=10GB \
-c autovacuum_work_mem=2GB -c checkpoint_timeout=60min \
-c max_wal_senders=0 -c wal_level=minimal -c max_wal_size=10GB \
-c checkpoint_completion_target=0.9 -c random_page_cost=1.0 \
-c full_page_writes=off -c fsync=off
```

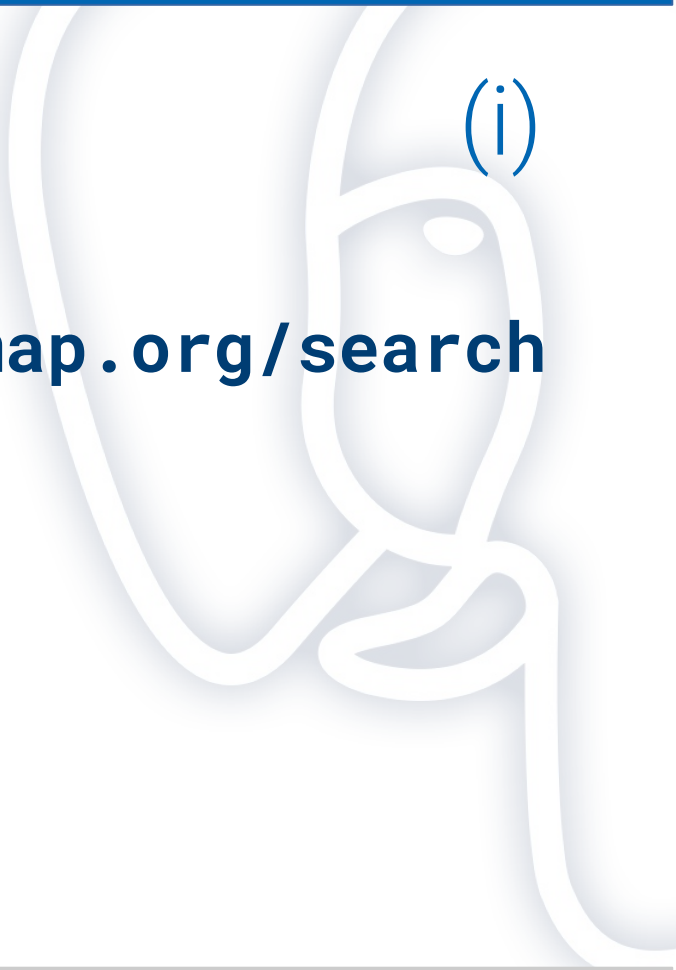
PgOSM Flex



Nominatim

- Comes in flavors:
  - (legacy) PHP
  - (whew) Python
- Install service as a Python ASGI application
- Serve via nginx

Querying Nominatim



```
curl \
"https://nominatim.openstreetmap.org/search
?q=Chicago"
```

```
curl \
"https://localhost/search?
q=Chicago&format=geojson"
```

Querying Nominatim (ii)


```
{
  "type": "FeatureCollection",
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0. http://osm.org/copyright",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "place_id": 26249932,
        "osm_type": "relation",
        "osm_id": 122604,
        "place_rank": 16,
        "category": "boundary",
        "type": "administrative",
        "importance": 0.7515295727100249,
        "addresstype": "city",
        "name": "Chicago",
        "display_name": "Chicago, Cook County, Illinois, United States"
      },
      "bbox": [
        -87.9400876,
        41.644531,
        -87.5240812,
        42.0230396
      ],
      "geometry": {
        "type": "Point",
        "coordinates": [
          -87.6244212,
          41.8755616
        ]
      }
    }
  ]
}
```

Querying Nominatim (iii)


```
"features": [
  {
    "type": "Feature",
    "properties": {
      "place_id": 26249932,
      "osm_type": "relation",
      "osm_id": 122604,
      "place_rank": 16,
      "category": "boundary",
      "type": "administrative",
      "importance": 0.7515295727100249,
      "addresstype": "city",
      "name": "Chicago",
      "display_name": "Chicago, Cook County, Illinois, United States"
    },
```

Querying Nominatim (iv)


```
    "bbox": [
      -87.9400876,
      41.644531,
      -87.5240812,
      42.0230396
    ],
    "geometry": {
      "type": "Point",
      "coordinates": [
        -87.6244212,
        41.8755616
      ]
    }
  }
]
```

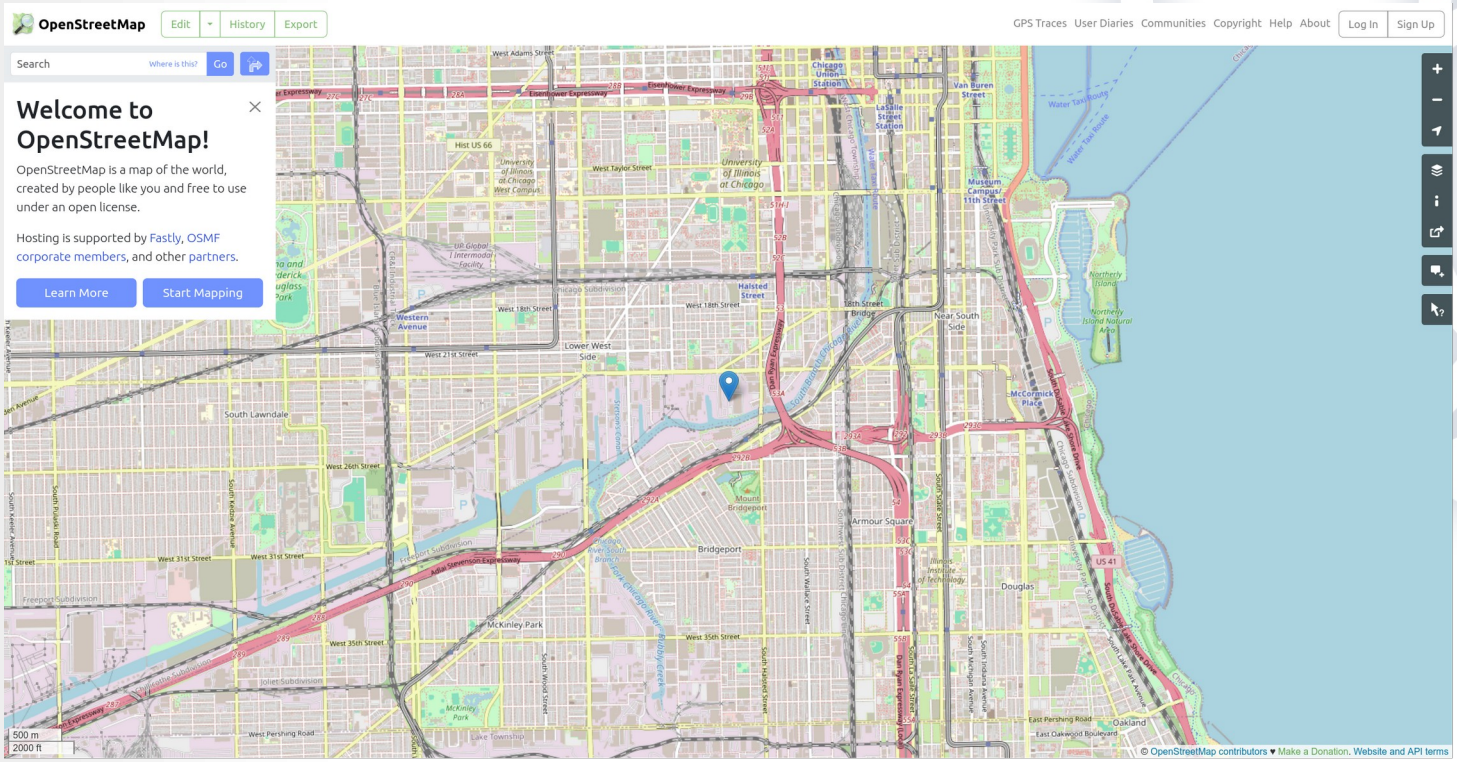
Photon

- Java/ElasticSearch
  - Search as you type
  - Typo tolerant (fuzzy search)
  - Multilingual
  - Ready made indexes, regularly updated
  - BUT: Updatable via Postgres/Nominatim
- Python library: github.com/astagi/pyphoton

Querying Photon

```
curl http://localhost:2322/api?q=chicago
```

Let’s choose a location 41.85003, -87.65005



–Mom, can we have G**gle Maps? –We have G**gle Maps at home

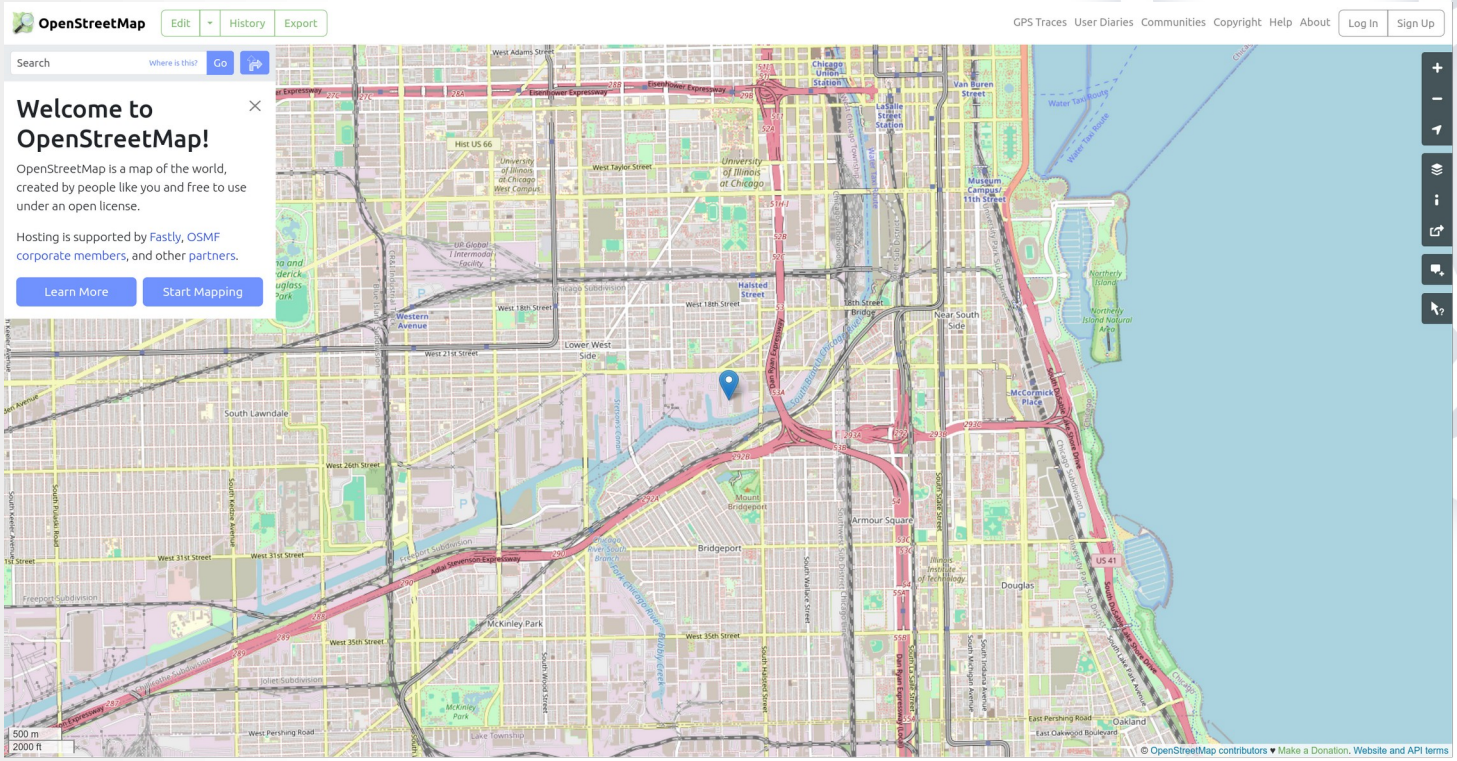
Spatial query

(i)

```
SELECT ST_Contains(geometry,
   ST_SetSRID(
      ST_MakePoint(-87.65005, 41.85003), 4326))
FROM place
WHERE osm_id=122604;

 st_contains
-------------
 t
(1 row)
```

Another location 42.00697, -87.72319



–Mom, can we have G**gle Maps? –We have G**gle Maps at home

Spatial query (ii)

```
SELECT ST_Contains(geometry,
   ST_SetSRID(
      ST_MakePoint(-87.72319, 42.00697), 4326))
FROM place
WHERE osm_id=122604;

 st_contains
-------------
 f
(1 row)
```

Spatial query with JOIN

```
SELECT customer_id
FROM customer_addresses ca
JOIN place
ON ST_Contains(geometry,
   ST_SetSRID(ST_MakePoint(ca.long, ca.lat), 4326))
AND osm_id=122604;
```

Various use cases

- Find objects in area/jurisdiction
- Passing the object type is super powerful
- Deduplicate addresses
  - e.g. if geocoded coordinates are within 300ft
- Normalize addresses
  - With geocoding you don’t have to worry about parsing addresses

Other tools

- Leaflet
- QGIS
- GeoServer
- MapServer
- Mapnik

Let's keep in touch!

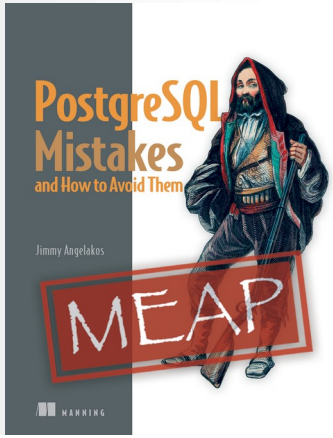
Mastodon: https://fosstodon.org/@vyruss

LinkedIn: https://linkedin.com/in/vyruss

YouTube: https://youtube.com/@JimmyAngelakos

Thank you ❤️

45% off for 3 months!
Code: **PGDC24**



25% off!
On: **Amazon.com**