PostGIS ♥ Protobuf

Introduction

- Björn Harrtell, Sweden
- Septima, Denmark

Note:
At Septima I work as a geospatial developer.

Subject

- Encoding Protobuf formats in PostGIS
- Mapbox Vector Tiles (lossy)
- Geobuf (lossless)

Note:
This talk is about a upcoming feature in PostGIS 2.4 to support outputting data in two Protobuf based formats, Mapbox Vector Tiles and Geobuf. It's important to note that vector tiles is a lossy format where as Geobuf is lossless.

Protobuf is a binary encoding specification from Google for structured data and I think that it can perhaps be seen as a binary equivalent of XML.

Why encode vector tiles in PostGIS?

- Custom projections
- Simplify toolchain
- Reduce I/O for potential performance wins

Note:
A few years back I had an assigment to build a mobile client with offline support and the ability to download and handle many layers of vector data with user controllable visiblity. Vector tiles seemed the best available choice. But I also had the requirement to use a swedish reference system which was not supported by the toolchains to create vector tiles at the time.

I did hack my own toolchain in Java and while it worked it wasn't code I wanted to see again. So I was interested in a more simple toolchain.

It bothered me that toolchains was essentially middleware requiring I/O and serialization of the source data, so I thought why not do this directly in PostGIS which had the potential to both simplify and improve efficiency.

Two new functions for vector tiles

- ST_AsMVTGeom (https://postgis.net/docs/manual-dev/ST_AsMVTGeom.html)
- ST_AsMVT (https://postgis.net/docs/manual-dev/ST_AsMVT.html)

Note:
The two new functions for vector tiles are ST_AsMVT and ST_ASMVTGeom.

- Show the docs, show the vector tile specification

ST_AsMVTGeom is used to transform a geometry into tile coordinate space.

ST_AsMVT is an aggregate function to encode a set of rows with transformed geometry + attributes into the binary vector tiles format.

ST_AsMVT was in initial proof of concept not an aggregate function, it took a plain SQL string as input and queried it internally. Having plain SQL strings as parameters is bad for many reasons of course and Paul suggested that I try to make it into aggregate function. That did indeed require some deep diving into Postgres internals.

The initial implementation was only a single function, ST_AsMVT. Due to good input from Blake Thompson at Mapbox about the importance of a strict geometry transformation step I reworked the implementation and split it into these two functions to make the transformation

step explicit. This has the additional benefit that you should be able to take advantage of the new Postgres parallel capabilities, though I haven't tested this myself.

Function for Geobuf

- ST_AsGeobuf (https://postgis.net/docs/manual-dev/ST_AsGeobuf.html)
- Similar to GeoJSON but more compact

Note:
I've also implemented ST_AsGeobuf, which I actually did before vector tiles because while also based on protobuf it's a more simple encoding so was a good start for me.

Can be a useful alternative to GeoJSON when you need larger volumes of data on the client side. Can complement vector tiles for when you need lossless access to the original source data.

Basic use

- Create the four tiles of zoom level 1 in spherical mercator

mvtile.sql

```
SELECT ST_AsMVT('land', 512, 'geom', q)
FROM (SELECT
  id,
  ST_AsMVTGeom(
    geometry,

ST_MakeEnvelope(:xmin,:ymin,:xmax,:ymax,3857),
    512, 0, true
  ) AS geom
FROM foss4g.ne_50m_land_3857
WHERE
  geometry &&
ST_MakeEnvelope(:xmin,:ymin,:xmax,:ymax,3857)
) as q WHERE geom IS NOT NULL
```

psql

```
psql -At \
  -v xmin=-20037508.342789244 \
  -v ymin=0 \
  -v xmax=0 \
  -v ymax=20037508.342789244 \
  -f mvtile.sql | xxd -r -p >./1/0/0.pbf
```

Note:
So the source is a Natural Earth dataset with the world continents as polygons.

I query that source for geometry intersecting tile bounds in geometrical coordinates.

Then ST_AsMVTGeom is used to transform the geometry into tile coordinate space with an extent of 512, buffer 0 and clipping set to true.

Finally I use ST_AsMVTGeom to aggregate the queried and transformed rows into a binary vector tile.

So that's the minimal bit of SQL needed. I can now use psql to make a tile for the first quadrant tile of the world.

Note that psql does not output raw binary, it's in hex encoded format so I use the xxd command to decode it to raw binary.

Results

- Single tile (https://bjornharrtell.github.io/presentations/vectortiles
- Four tiles with OL 4.0 (https://bjornharrtell.github.io/presentations/vectortiles
- Four tiles with OL 4.2 (https://bjornharrtell.github.io/presentations/vectortiles
- Higher resolution (https://bjornharrtell.github.io/presentations/vectortiles
- Random fill color (https://bjornharrtell.github.io/presentations/vectortiles

Postgres details

- Why not COPY?
- Does not output raw binary
- Optional use is through PG driver

Note:
You might ask why I'm not using the COPY command and it's because COPY does not output raw binary, it add header data.

I'd like to see it possible to opt out of these headers. Perhaps an opportunity for a contribution to Postgres?

So, the current optimal way to use ST_AsMVT is through a Postgres driver where you can read the binary directly.

How?

- protobuf-c (https://github.com/protobuf-c/protobuf-c)
- vector-tile-spec (https://github.com/mapbox/vector-tile-spec/tree/master/2.1)
- Pitch the idea and a POC on the postgis-devel list to get interest
- Sponsoring from CARTO (thank you!)
- Get away with no proper unit tests (!)
- At least a somewhat complete regression test suite
- Create millions of tiles

Future development

- Investigate integration of https://github.com/mapbox/wagyu
- Use of ST_AsMVT in tile servers like t-rex and others

End!

 @bjornharrtell (https://twitter.com/bjornharrtell)