Client-side vector tiles rendering

Кравцун Андрей.

Основные способы хранения географических данных

- OSM:
  - nodes: объявление + список тэгов
  - ways: список нод
  - relations: ссылки на ноды или пути.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="12.25
 <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="46882"
  <tag k="traffic_sign" v="city_limit"/>
 </node>
 ...
 <way id="26659127" user="Masch" uid="55988" visible="true" version="5" changeset=
  <nd ref="292403538"/>
  ...
  <tag k="highway" v="unclassified"/>
 </way>
 <relation id="56688" user="kmvar" uid="56190" visible="true" version="28" changes
  <member type="node" ref="294942404" role=""/>
  ...
  <tag k="name" v="Küstenbus Linie 123"/>
...
```

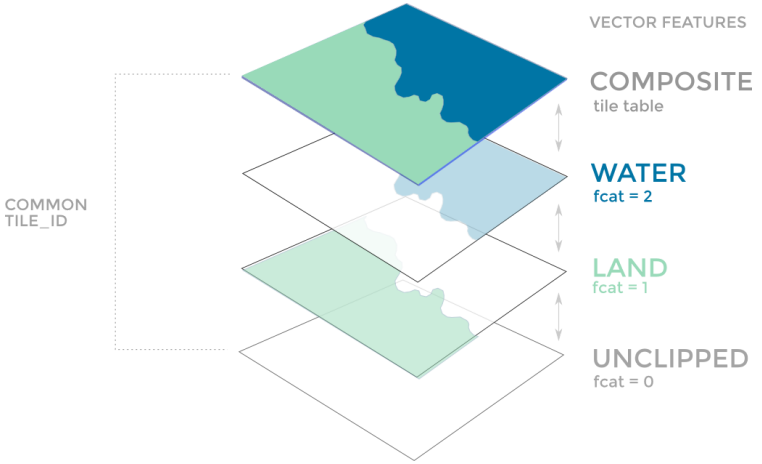
Основные способы хранения географических данных

- PostGIS: Геометрические примитивы для пространственных объектов, операции над ними и другие обслуживающие типы данных (напр. индексы). По факту: расширение SQL.

```
POINT(0 0)

LINESTRING(0 0,1 1,1 2)

POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))

MULTIPOINT((0 0),(1 2))

MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))

MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2

GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))
```

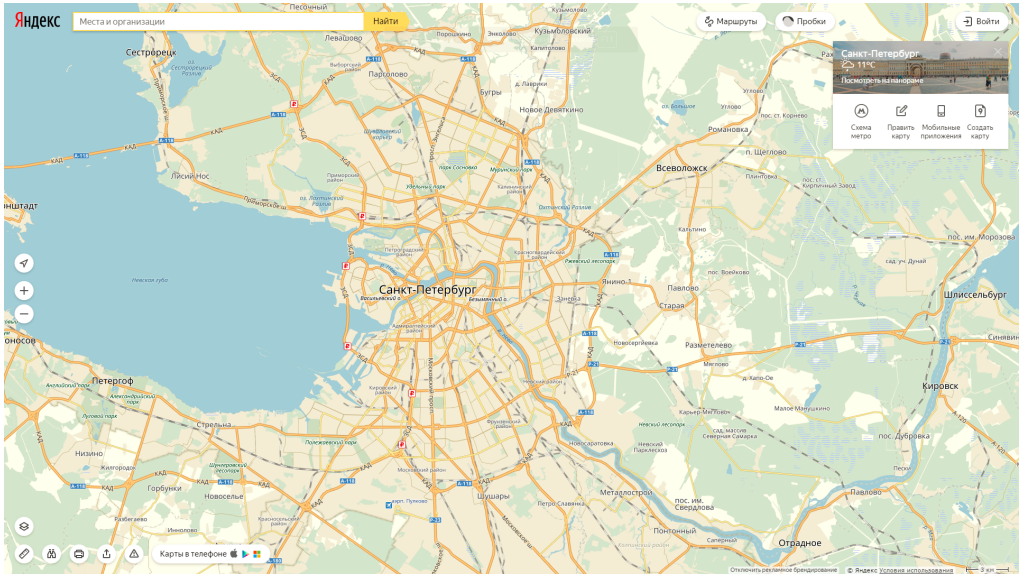
Основные способы хранения географических данных

- Vector tiles Более высокоуровневое, чем в OSM (но не всегда - более, чем в PostGIS) описание данных об объектах мира.



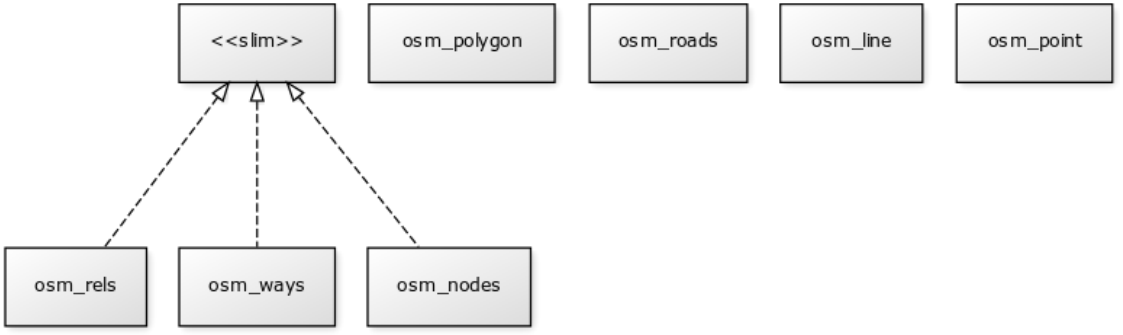
Основные способы хранения географических данных

- Растровое изображение



OSM -> POSTGIS

- osm2pgsql - Простая схема: line, point, polygon and roads, большие таблицы:



Подробнее

- imposm3 - более гибкая схема, с тематическим разделением: пример .

Применительно к openmaptiles:

```
docker-compose run import-osm
```

Форматы vector tiles

- MVT: Mapnik Vector Tiles - оптимизирован под Mapnik-бэкэнды, поддерживается большинством известных vector-tiles-серверов.
- Mapbox vector tile format (mbtiles) - Сериализованный протобуферами MVT. Поддерживается Mapbox GL JS, Open Layers 3, Leaflet, Mapzen Tangram, Esri...
- Geopackage - аналог mbtiles от OGC (Open Geospatial Consortium). Поддерживается QGIS, ESRI and GDAL.
- OSciM-PBF: аналог mbtiles, по формату более низкоуровневый) от OpenScienceMap (разрабатывает картографические сервисы для научных исследований, Бремен), поддерживается TileStache OSciMap.

Форматы vector tiles

- GeoJSON, промежуточный формат, требуется перегонка между географической системой координат и проекцией меркатора. Поддерживается TileStache. Возможна конвертация в mbtiles с помощью https://github.com/mapbox/tippecanoe .
- Kothic JSON: Кастомный GeoJSON, поддерживается Kothic json_getter.py and TileStache JsonOSciMap (выглядел морально устаревшим).
- Mapsforge Ориентирован под устройства с ограниченными ресурсами (мобильные платформы). Главный файл разбивается на подфайлы с разными масштабами и уровнями детализации, со встроенной индексацией тайлов.
- Google Maps. Первый, но закрытый.

PostGIS -> vtiles

Сервера - много их. Перечислю некоторые:

- Google Maps (WebGL)
- Tessera, Kartotherian - основаны на node-mapnik (поэтому чреваты проблемами с версиями API Mapnik и node-mapnik)

vtiles -> изображение на экране

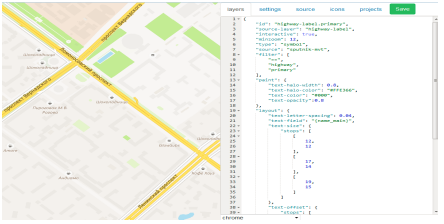
- Mapbox GL Native - библиотека для нативной отрисовки на множество платформ: Linux (GLFW), Android (OpenGL ES), iOS, macOS, Qt, React - для остального Web есть Mapbox GL JS. Большой минус: нет инструкции по билду на Windows.
- OpenLayers Огромная библиотека для отрисовки интерактивных карт в Web (JS).
- Tangram-ES Аналогично Mapbox GL Native, та же вторичность по отношению к своему web-собрату, несколько меньше целевых платформ.
- TileStache - монструозен, но морально устарел.

Обходные пути: OSM -> vtiles

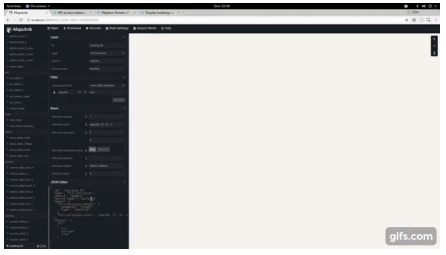
- TileMaker Мало возможностей для тюнинга, требуется время на освоение конфигурирования для корректного взаимодействия с инструментами рендеринга (специфицировать схему, настроить маппинги и т.д.)
- VectorTileCreator Часть огромной библиотеки KDE Marble, с невпечатляющими примерами применения.

Редакторы стилевых файлов

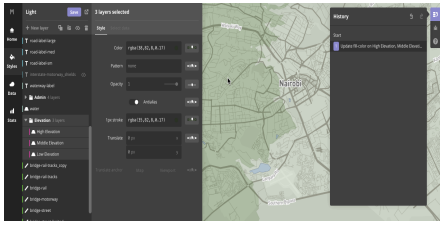
[mvt-styler (Sputnik)]



[maputnik]

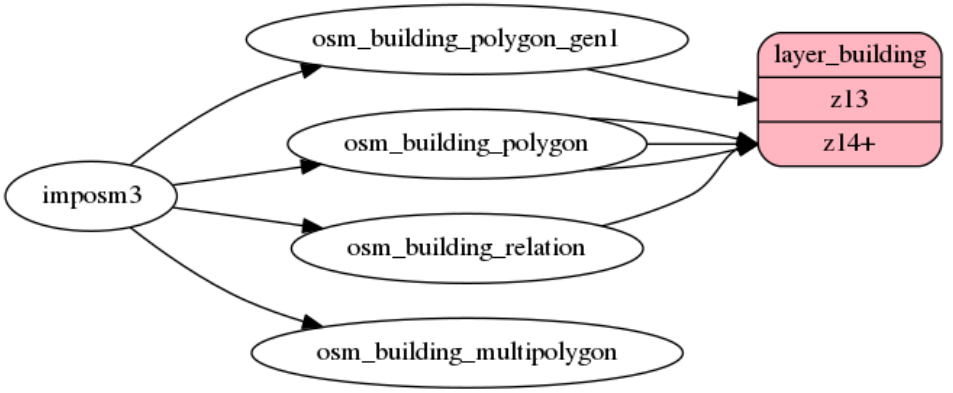


[Mapbox Studio]



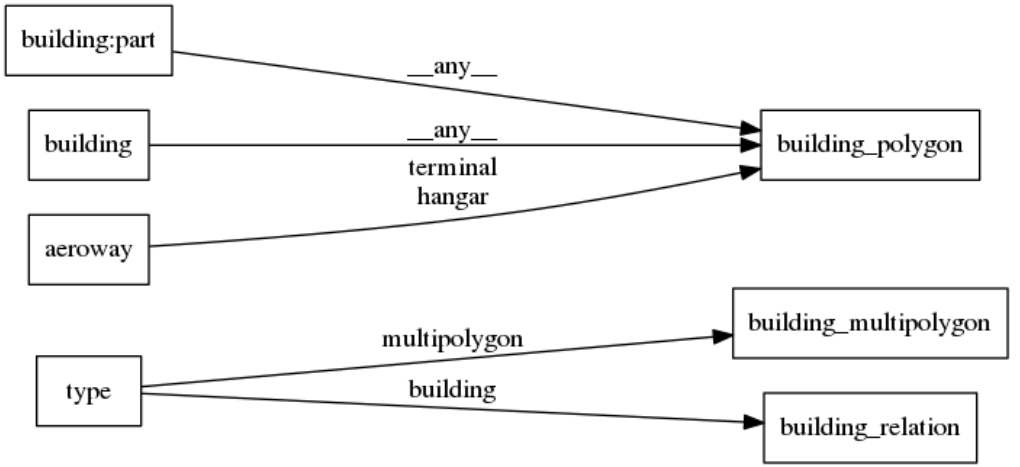
Data flow diagram одного конкретного слоя (OSM -> vtiles, *building*)

Конвертация нод OSM в таблицы PostGIS.



Data flow diagram одного конкретного слоя (OSM -> vtiles, *building*)

Маппинг



Работа с векторными тайлами с помощью openmaptiles-tools

Создание своего слоя из OSM-данных

С помощью imposm3 mapping file определить, как будет происходить конвертация сущностей OSM в данные PostGIS. Пример:

```
layer:
  id: "building"
  description: Buildings from OpenStreetMap
  buffer_size: 4
  datasource:
    query: (SELECT geometry FROM layer_building(!bbox!, z(!scale_denominator!))) AS t
  fields:
    render_height: An approximated height from levels and height of building.
schema:
  - ./building.sql
datasources:
  - type: imposm3
    mapping_file: ./mapping.yaml
```

Конфигурирование тайлсета (какие слои будут нужны в финальной выборке)

Пример:

```
tileset:
  layers:
    - layers/building/building.yaml
    - layers/housenumber/housenumber.yaml
    - layers/poi/poi.yaml
  name: Street Level
  description: A tileset showing street level info like building, housenumbers and
  attribution: "OpenStreetMap contributors"
  maxzoom: 14
  minzoom: 13
  center: [-12.2168, 28.6135, 4]
```

Для генерирования файла маппинга для imposm3 необходимо запустить

```
generate-imposm3 <tileset>
```

Пример файла объявления тайлсета в `openmaptiles.yml` - в нём как минимум наверняка понадобится изменить поля `attribution` и `center`.

Создание тайлсета на основе своих OSM-данных

Пошаговое руководство. Для данного примера использовался файл `data/taganrog.osm.pbf`

Поместить свой OSM-PBF файл в папку `data/` и добавить ее в в подключаемые тайма в контейнер postgis, например (файл `docker-compose.yml`):

```
version: "2"
volumes:
  pgdata:
  cache:
services:
  postgres:
    image: "openmaptiles/postgis:2.5"
    volumes:
    - pgdata:/var/lib/postgresql/data
    - ./data:/data
    networks:
    ...
```

Добавить свои слои на основе предыдущих двух инструкций.

Запустить контейнер postgres:

```
docker-compose up -d postgres
```

Но вышеописанное проще сделать так:

```
./quickstart.sh <name-without-osm.pbf>
```

По умолчанию создается файл `data/tiles.tiles`.

Обновление тайлсета

поместить файл с изменениями (в формате OsmChange) в папку `import/` и запустить

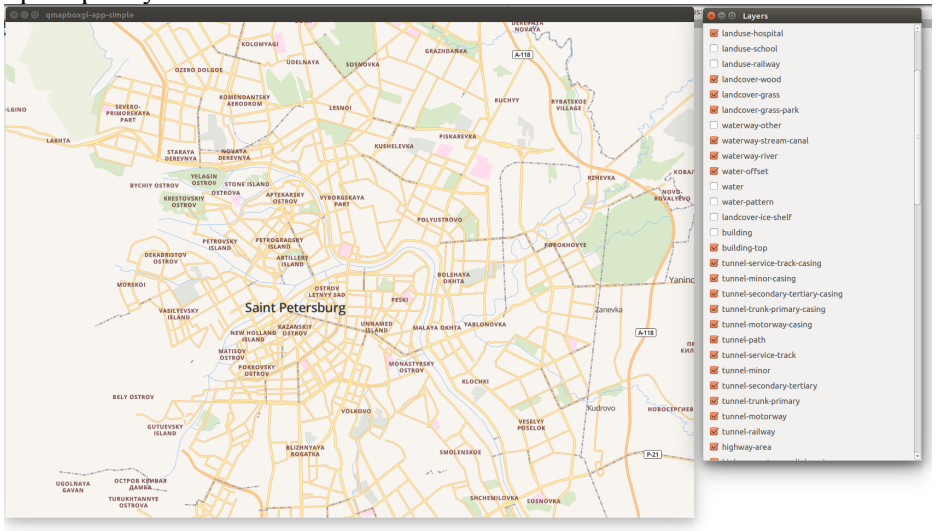
```
docker-compose run import-osm-diff
```

После этого сгенерированный в папке `diffdir/` файл поместить под именем `tiles.txt` в папку `import/` и запустить

```
docker-compose run generate-changed-vectortiles
```

Прототип приложения рендера:

- репозиторий
- конфигурационный файл стилей (можно взять из примеров отсюда).
- пример запуска:



Спасибо за внимание!