OpenStreetMap
→ ( PostGIS | MySQL | Spatialite )
→ OpenLayers

From Map To Web

MySQL Conference & Expo 2011

Hartmut Holzgraefe <hartmut@php.net>
Me ...

- Hartmut Holzgraefe <hartmut@php.net>
- MySQL Support Engineer 2004-2010
- Currently working for FoeBuD e.V., a German NGO (http://foebud.org/)
- OSM-User since 2007, active since late 2009
Talk Outline

• What is OpenStreetMap
• What are PostGIS and OpenLayers
• OpenStreetMap Data Format
• OSM → Database
• Database → Web
• Ad Hoc Query Results
• Example Application
OpenStreetMap

- Open Geodata
- Open license (currently: CC-by-SA License)
- Data sources contain:
  - GPS Tracks
  - Arial Images (YaHoo!, Bing, AeroWest, …)
  - Other Imports (OpenGeoDB, USA: Tiger, …)
- See also http://openstreetmap.org/
What OpenStreetMap Looks like:

http://osm.org/go/TZMIDJLP
... compared to ...
But there is more than one map style

OpenWhateverMap provides a random mix of available styles

http://www.openwhatevermap.org/?zoom=12&lat=37.35624&lon=-121.92799&layers=B
OpenStreetMap Workflow
Of interest to us today:
PostGIS

• GIS – Extension for PostgreSQL
• Supports standard GIS data types like POINT, LINE, POLYGON ...
• 2D GIS Index Support
• Rich set of functions supporting geometry creation, combination and processing
• See also http://postgis.refractions.net/
OpenLayers

- Open Source JavaScript/Ajax API for Web Maps
- Similar to the Google Maps API in some ways
- But supporting way more formats:
  - Bitmap/Image: OpenStreetMap(*), Google, Bing, ...
  - Vektor: WKT, KML, GPX, OSM, GeoJSON, ...
  - WMS
- See also http://openlayers.org/
OSM XML File Format
Don't let it scare you, you won't have to deal with it directly

```xml
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' generator='JOSM'>
  <bounds minlat='52.02' minlon='8.51' maxlat='52.03' maxlon='8.52'/>
  <node id='27541371' timestamp='2010-04-11T12:02:42Z'
        uid='124466' user='Heros' visible='true' version='13'
        changeset='4392479' lat='52.0308503' lon='8.5233556' />
...
  <way id='4805339' timestamp='2010-02-18T19:18:18Z' uid='162220'
       user='SKit' visible='true' version='8' changeset='3911454'>
    <nd ref='30840398' />
    <nd ref='493865093' />
    <tag k='highway' v='residential' />
    <tag k='name' v='Weststraße' />
  </way>
...
  <relation id='29035' timestamp='2010-01-18T22:17:32Z' uid='162220'
            user='SKit' visible='true' version='10' changeset='3654770'>
    <member type='way' ref='5818594' role='inner' />
    <member type='way' ref='5818593' role='outer' />
    <tag k='natural' v='water' />
    <tag k='type' v='multipolygon' />
  </relation>
</osm>
```
OSM XML Building Blocks

- There are only three basic object types:
  - Nodes
  - Ways
  - Relations

- Each of these objects has:
  - ID, user, timestamp, version, changeset
  - … plus any number of additional key/value pairs

- Refer to the OSM wiki for key/value tag definitions and suggestions for various real world object types
OSM XML Format: Nodes

- A Node defines a single point on the map
- 2D latitude / longitude
- Nodes can stand for themselves (POI)
- … or can be part of a way
OSM XML Format: Ways

- Ways consist of an ordered list of Nodes

- Areas are just closed ways
OSM XML Format: Relations

- Relations combine other objects
- Relations can contain Nodes and Ways ...
- ... and other Relations, too
- Typical use cases
  - Multipolygons
  - Very long Ways
  - Local groups of Objects
  - Routs spanning multiple streets
Getting hold of OSM Data

- For small areas → OSM API
- For medium size areas → OSM XAPI
- For large areas → Downloads
  - Planet.osm → direktly from openstreetmap.org
  - Daily, hourly, minutely diffs → openstreetmap.org
  - $country.osm, $state.osm, $district.osm → e.g. from GeoFabrik.de

OSM XML -> Database

- Import tools: `osm2pgsql` and `ImpOSM`
- Both can import complete OSM files
- ... and daily/hourly/minutely diffs (`osm2pgsql`)
- Support OSM XML and the newer PBF Format
- Raw data is converted into a more usable form
  - By combining objects from relations
  - By creating separate tables for lines and polygons

Database support

- Currently PostgreSQL / PostGIS only
- Both tools are modular though so having backends for different GIS capable databases possible
- … and being worked on, but moving slowly
- Other open source options would be:
  - MySQL
  - SpatiaLite extension for SQLite
osm2pgsql

`osm2pgsql [options] planet.osm.{gz,bz2,pbf}`

  `--database=name`

  `--prefix=name`

  `--latlong`

  `--slim`

  `--cache=size`
Osm2pgsql Schema

- **Main tables**
  - Planet_osm_point
  - Planet_osm_line
  - Planet_osm_polygon
  - Planet_osm_roads
- **Helper tables (used during import)**
  - Planet_osm_nodes
  - Planet_osm_ways
  - Planet_osm_relations
## Schema Details

**Table** "public.planet_osm_polygon"

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>osm_id</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z_order</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>way_area</td>
<td>real</td>
<td></td>
</tr>
<tr>
<td>way</td>
<td>geometry</td>
<td></td>
</tr>
</tbody>
</table>

**Indexes:**

- "planet_osm_polygon_index" gist (way)
- "planet_osm_polygon_pkey" btree (osm_id)
Simple Queries

Administrative town border of Stuttgart, Germany:

```sql
SELECT ASTEXT(way)
    FROM planet_osm_polygon
WHERE name='Stuttgart';
```

```
POLYGON((1006189.62 6231109.22,
    1006225.9  6231281.72,
    ...)
```
OpenLayers Basics

```html
<script src="OpenLayers.js"/>
<script src="OpenStreetMap.js"/>

<script type="text/javascript">
var map, vectorLayer;
function init() {
    map = new OpenLayers.Map('map');
    osmLayer = new OpenLayers.Layer.OSM.Mapnik();
    vectorLayer = new OpenLayers.Layer.Vector();
    map.addLayers([osmLayer, vectorLayer]);
    map.zoomToMaxExtent();
}
</script>

<body onload="init();">
    <div id="map"/>
</body>
```
WKT-Results as VectorLayer

OpenLayers directly supports the WKT format as returned by `ASTEXT()` for displaying vector features like points, lines and polygons:

```javascript
var wkt_format = new OpenLayers.Format.WKT();
var vectorFeature = wkt_format.read(WKT_text);
vectorLayer.addFeatures(vectorFeature);
map.zoomToExtent(vectorLayer.getDataExtent());
```
Markers as TextLayer

- TextLayer reads a TAB separated list
- ... and creates one map marker per input line
- Supported input fields:
  - Latitude and Longitude (mandatory)
  - Popup title and content (optional)
  - Icon image and relative position (optional)
Filling a TextLayer with Markers

- Example: Playgrounds in my home town

```sql
SELECT X(playground.way) AS lon,
       Y(playground.way) AS lat,
       playground.name AS title
FROM planet_osm_polygon AS city
JOIN planet_osm_point AS playground
ON CONTAINS(city.way, playground.way)
WHERE city.name = 'Bielefeld'
    AND playground.leisure = 'playground';
```
PostGIS Terminal

- Originally created by Marc Jansen and Till Adams for their (German) book on OpenLayers
- Simple query entry in a web form
- Results shown directly in the browser
- Slightly extended by me to support markers in addition to WKT vectors, and to support more than one WKT result row

- See also http://openlayers-buch.de/
Example: City borders

SELECT ASTEXT(way)
FROM planet_osm_polygon
WHERE name='Stuttgart';
SELECT ST_ASTEXT(way)
FROM planet_osm_polygon
WHERE name='Stuttgart';
Example: the main roads in ...

SELECT ASTEXT(r.way)
FROM planet_osm_polygon city
JOIN planet_osm_roads road
ON CONTAINS(city.way, road.way)
WHERE city.name = 'Stuttgart'
AND road.highway
IN ('primary','secondary','tertiary');
SELECT ST_ASTEXT(ST_Union(r.way))
    FROM planet_osm_polygon p
    JOIN planet_osm_roads  r
    ON ST_CONTAINS(p.way, r.way)
WHERE p.name = 'Stuttgart'
AND r.highway
    IN ('primary','secondary','tertiary');
SELECT X(playground.way) AS lon,
       X(playground.way) AS lat,
       playground.name AS title
FROM planet_osm_polygon AS city
JOIN planet_osm_point AS playground
  ON CONTAINS(city.way, playground.way)
WHERE city.name = 'Stuttgart'
  AND playground.leisure = 'playground';
SELECT st x(pt.way) AS lon, st y(pt.way) AS lat, pt.name AS title
FROM planet_osm_polygon AS p
JOIN planet_osm_point AS pt
ON ST_CONTAINS(p.way, pt.way)
WHERE p.name = 'Stuttgart'
AND pt.leisure = 'playground';
Example: combining with non-OSM data

```
SELECT ASTEXT( /* draw circle around center of gravity */
    BUFFER(CENTROID(city.way),
        sqrt(pop.population)*10)))
FROM planet_osm_polygon city
JOIN population pop
    ON city.name = pop.name
AND ( city.admin_level = '8'
    OR ( city.name = 'Bielefeld'
        AND city.admin_level = '6'))
/* German cities usually have admin_level 8, district-free cities are on the higher level 6 */
SELECT ST_ASTEXT(ST_UNION(ST_BUFFER(ST_CENTROID(c.way), sqrt(p.population)*10)))
FROM planet_osm_polygon c
JOIN population p
ON c.name = p.name
AND ( c.admin_level = '8' OR ( c.name = 'Bielefeld' AND c.admin_level = '6'))
Example Application

- “Cat inspectors”
Cat Inspectors cnt.