



# OpenStreetMap-Daten in PostGIS

## FROSCON 2012

Samstag, 25. August 2012

*Sven Geggus* <[sven@gegg.us](mailto:sven@gegg.us)>





# OpenStreetMap-Daten in PostGIS

## Gliederung

- Definition von PostGIS und Openstreetmap
- Datenmodell von PostGIS
- Datenmodell von OSM
- Mapping der Datenmodelle
- Importwerkzeuge für PostGIS
- Kartenprojektionssysteme
- PostGIS Anwendungsbeispiele



# OpenStreetMap-Daten in PostGIS

## Was ist PostGIS?

PostGIS ist eine Erweiterung von PostgreSQL, die es ermöglicht geografische Objekte in der Datenbank zu speichern.

Bei diesen Objekten handelt es sich um Geometrien nach dem ***Simple features for SQL*** Standard des Open Geospatial Consortium (OGC). Seit PostGIS 2.0 ist auch die Speicherung von Rasterdaten (z.B. Luftbildern) möglich.



# OpenStreetMap-Daten in PostGIS

## Was ist Openstreetmap?



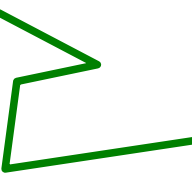
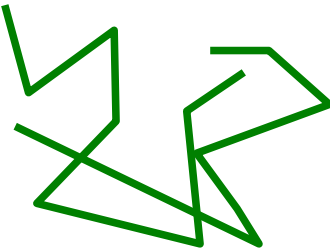
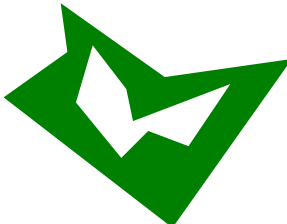

Openstreetmap (OSM) ist ein Projekt, das für jeden frei nutzbare Geodaten sammelt. Anfangs waren das hauptsächlich Straßendaten.

Weil OSM Daten mit Topologie (routingfähig) erfasst aber auch aus historischen Gründen sind diese nicht kompatibel zu OGC Simple features.



# OpenStreetMap-Daten in PostGIS


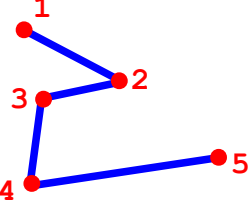
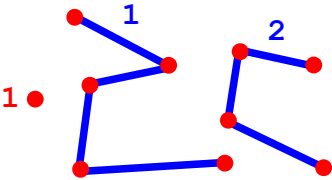
## OGC „simple features“ Geometrietypen

<b>Point</b> 	<b>Multi-Point</b> 
<b>LineString</b> 	<b>Multi-LineString</b> 
<b>Polygon</b> 	<b>Multi-Polygon</b> 



# OpenStreetMap-Daten in PostGIS



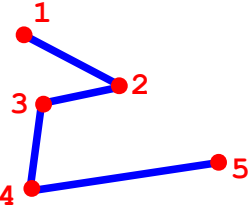
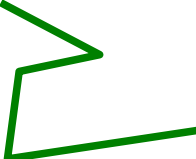
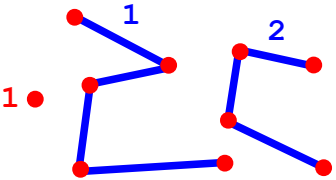
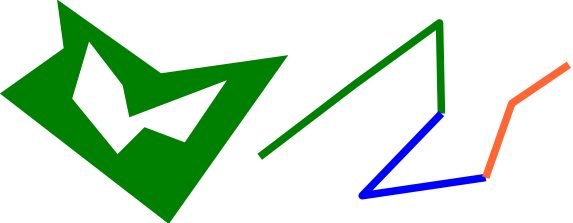
## OSM Datentypen

<p><b>Node</b></p> 	<pre>&lt;node id='1' Lat='0.0' Lon='1.0'&gt;   &lt;tag k='name' v='node #1' /&gt; &lt;/node&gt;</pre>
<p><b>Way</b></p> 	<pre>&lt;way id='1'&gt;   &lt;nd ref='1' /&gt;   &lt;nd ref='2' /&gt;   &lt;nd ref='3' /&gt;   &lt;nd ref='4' /&gt;   &lt;nd ref='5' /&gt;   &lt;tag k='name' v='way #1' /&gt; &lt;/way&gt;</pre>
<p><b>Relation</b></p> 	<pre>&lt;relation id='1'&gt;   &lt;member type='node' ref='1' role='' /&gt;   &lt;member type='way' ref='1' role='' /&gt;   &lt;member type='way' ref='2' role='' /&gt;   &lt;tag k='name' v='relation #1' /&gt; &lt;/relation&gt;</pre>



# OpenStreetMap-Daten in PostGIS

## „Mapping“ OSM/OGC Datentypen

<b>Node</b> 	<b>Point</b> 
<b>Way</b> 	<b>LineString</b> 
<b>Relation</b> 	<b>Polygon, LineStrings, ...</b> 



# OpenStreetMap-Daten in PostGIS

## Flächen in OSM

- Derzeit existiert kein spezieller Datentyp für Flächen
- Einfache Polygone ergeben sich aus geschlossenen Wegen mit passendem tagging und ggf. area=yes

```
<way id='1'>
  <nd ref='1' />
  <nd ref='2' />
  <nd ref='3' />
  <nd ref='4' />
  <tag k='highway' v='pedestrian' />
  <tag k='area' v='yes' />
</way>
```

```
<way id='1'>
  <nd ref='1' />
  <nd ref='2' />
  <nd ref='3' />
  <nd ref='4' />
  <tag k='building' v='yes' />
</way>
```

- Aufwenigere Polygone (Flächen mit “Loch”) werden als Relation dargestellt

```
<relation id="1">
  <member type="way" ref="1" role="outer"/>
  <member type="way" ref="2" role="inner"/>
  <member type="way" ref="3" role="inner"/>
  <tag k="natural" v="water"/>
  <tag k="type" v="multipolygon"/>
</relation>
```





# OpenStreetMap-Daten in PostGIS

## Attributsdaten für Geometrieobjekte

### **OSM:**

Freies key/value Taggingschema vergleichbar mit assoziativen Datenfeldern (perl „hash“, python „dictionary“) in Skriptsprachen

### **OGC:**

Attribute in festen Spalten vergleichbar mit Tabellen in relationalen Datenbanken (z.B. Name, Höhe, ...)

**Elegante Lösung für PostgreSQL:  
„hstore“ key/value Extension**



# OpenStreetMap-Daten in PostGIS

## Übersicht Importwerkzeuge

Name	Inkrementelle updates	Datenbankschema	hstore	Zahlen	Simple features
osm2pgsql	möglich	fest	optional	ja, hardcoded	Point, Linestring, Polygon
osmosis	möglich	fest	ja	nein	Point
imposm	nicht möglich	beliebig (configfile/python)	optional	beliebig (python code)	Point, Linestring, Polygon
ogr2ogr (nur svn-Version)	nicht möglich	fest	ja	nein	Point, Linestring, Polygon
osmium	nicht möglich	beliebig (C++ code)	ja	beliebig (C++ code)	Point, Linestring, Polygon



# OpenStreetMap-Daten in PostGIS

## Kartenprojektionen im Schnelldurchlauf

- Die Erde ist ~~eine Kugel~~ näherungsweise ein Rotationselipsoid
- Traditionell wurden gut zum eigenen Land passende Elipsoiden und Koordinatensysteme verwendet
- Die European Petroleum Survey Group (**EPSG**) sammelt alle gebräuchlichen „Spatial Reference Systems“ (SRS) und vergibt Referenznummern
- Die FOSS Bibliothek proj4, PostGIS, etc. verwenden diese EPSG Nummern
- Derzeit kennt proj4 rund 4000 solcher Systeme



# OpenStreetMap-Daten in PostGIS

## Kartenprojektionen im Schnelldurchlauf

- Migration zu World Geodetic System 1984 (WGS 84) und Universaler Transversaler Merkatorprojektion (UTM) durch Nutzung des GPS Systems
- UTM Standard ist leider ungeeignet zur Darstellung der ganzen Welt auf einer Karte. Als diese Standards entwickelt wurden gab es noch keine Webkarten
- Google hat „World Mercator“ (auch „Google Mercator“ oder „Web Mercator“ genannt) für „Google Maps“ entwickelt
- „World Mercator“ wurde zum Defakto Standard für Webkarten



# OpenStreetMap-Daten in PostGIS

## Wichtige EPSG Codes

EPSG:4326	Geografische Längen- und Breitengrade bezogen auf WGS 84 Elipsoid (verwendet bei GPS und OSM)
EPSG:3857	World Merkator auch Google Mercator oder Web Mercator
EPSG:900913	Alter Pseudo Code für World Mercator (GOOGLE in 1337-speech)
EPSG:31466	Gauss-Krüger Zone2 bezogen auf Bessel Elipsoid "deutsches Hauptdreiecksnetz"
EPSG:31467	Gauss-Krüger Zone3 bezogen auf Bessel Elipsoid "deutsches Hauptdreiecksnetz"
EPSG:31468	Gauss-Krüger Zone4 bezogen auf Bessel Elipsoid "deutsches Hauptdreiecksnetz"
EPSG:31469	Gauss-Krüger Zone5 bezogen auf Bessel Elipsoid "deutsches Hauptdreiecksnetz"
EPSG:25832	UTM Zone 32N ETRS 89 "Europäisches Terrestrisches Referenzsystem"
EPSG:25833	UTM Zone 33N ETRS 89 "Europäisches Terrestrisches Referenzsystem"



# OpenStreetMap-Daten in PostGIS

## Einsatzbeispiele PostGIS

**Allgemein: Datenbankabfragen mit Geobezug**

**Beispiel: Alle Restaurants im Umkreis von 2km zum Standort (Längengrad: 7,18 Breitengrad: 50,78) ermitteln (Datenbankschema, osm2pgsql mit hstore, Spatial Reference System, Google Mercator)**

**Vorüberlegung: Restaurants können in der Datenbank sowohl Gebäude als auch Punkte sein**

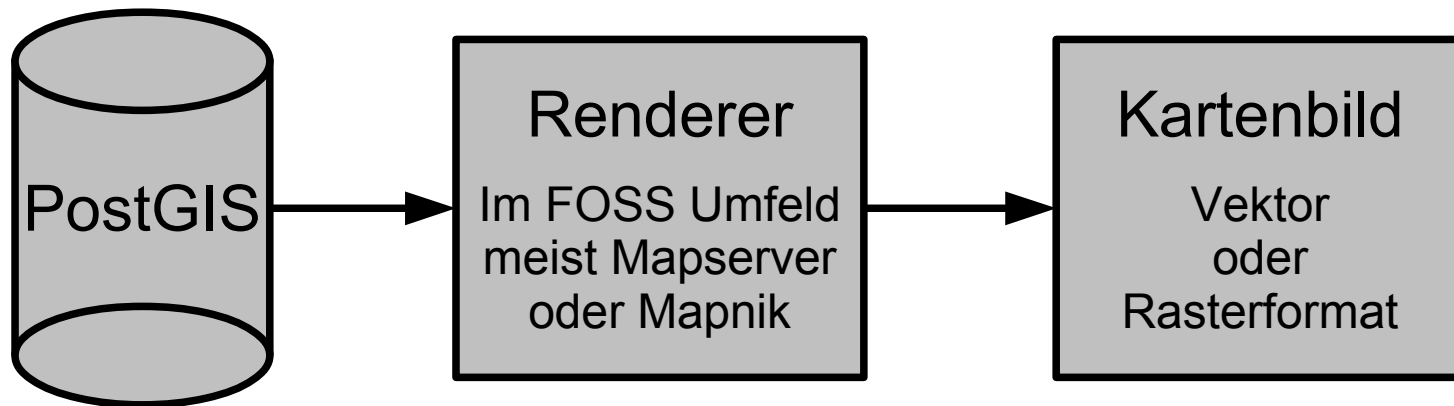
```
SELECT tags->'name' FROM planet_osm_point
WHERE amenity='restaurant' AND ST_DWithin(way,
ST_Transform(GeomFromText('POINT(7.18 50.78)',4326),3857), 2000)
UNION ALL
SELECT tags->'name' FROM planet_osm_polygon
WHERE amenity='restaurant' AND building='yes' AND ST_DWithin(way,
ST_Transform(GeomFromText('POINT(7.18 50.78)',4326),3857), 2000);
```



# OpenStreetMap-Daten in PostGIS

## Einsatzbeispiele PostGIS

### Kartenrendering







# OpenStreetMap-Daten in PostGIS

## Einsatzbeispiele PostGIS Kartenrendering mit Mapserver

MAP

OUTPUTFORMAT

NAME agg

DRIVER AGG/PNG

MIMETYPE "image/png"

END

IMAGECOLOR "#FFFFFF"

PROJECTION

"init=epsg:3857"

END

LAYER

TYPE POLYGON

STATUS ON

NAME "forest"

GROUP "default"

CONNECTIONTYPE POSTGIS

CONNECTION "dbname=gis"

DATA "way from (select way,osm\_id,landuse from planet\_osm\_polygon where landuse='forest')  
as foo using unique osm\_id using srid=3857"

CLASS

STYLE

COLOR "#c6e2b5"

END

END

END

END





# OpenStreetMap-Daten in PostGIS

## Einsatzbeispiele PostGIS Kartenrendering mit Mapnik

```
<?xml version="1.0" encoding="utf-8"?>
<Map background-color="#FFFFFF" srs="+init=epsg:3857" minimum-version="2.0.0">
  <Style name="forest">
    <Rule>
      <PolygonSymbolizer fill="#c6e2b5"/>
    </Rule>
  </Style>
  <Layer name="forest" status="on" srs="+init=epsg:3857">
    <StyleName>forest</StyleName>
    <Datasource>
      <Parameter name="type">postgis</Parameter>
      <Parameter name="dbname">gis</Parameter>
      <Parameter name="estimate_extent">>false</Parameter>
      <Parameter name="extent">-20037508,-19929239,20037508,19929239</Parameter>
      <Parameter name="table">
        (select way from planet_osm_polygon where landuse='forest') as forest
      </Parameter>
    </Datasource>
  </Layer>
</Map>
```