FOSSGIS 2020
Freiburg, 11.–14. März

WhereGroup

Verbindungen schaffen mit PostgreSQL Foreign Data Wrappern

12.3.2020 | Astrid Emde | FOSSGIS 2020 Freiburg

Astrid Emde

WhereGroup Bonn seit 2002, GIS-Consultant
FOSS-Academy-Schulungen
PostgreSQL/PostGIS, MapServer, GeoServer, QGIS,
QGIS Server, Mapbender, PostNAS-Suite
OSGeo Board, FOSSGIS e.V., Mapbender PSC,
OSGeoLive PSC, QGIS-DE

astrid.emde@wheregroup.com

Gegründet 2007 als Fusion der Firmen CCGIS, KARTA.GO GmbH und Geo-Consortium

40+ Mitarbeiter an 3 Standorten in Bonn, Berlin u. Freiburg (Projektleiter, Consultants, Trainer, Softwareentwickler)

Dienstleister in den Bereichen WebGIS, GDI, Kataster, Datenbanken mit freier Software

Schulungen, Infoveranstaltungen, Konferenzen

Unterstützer von OSGeo und FOSSGIS

https://wheregroup.com

https://fossacademy.com

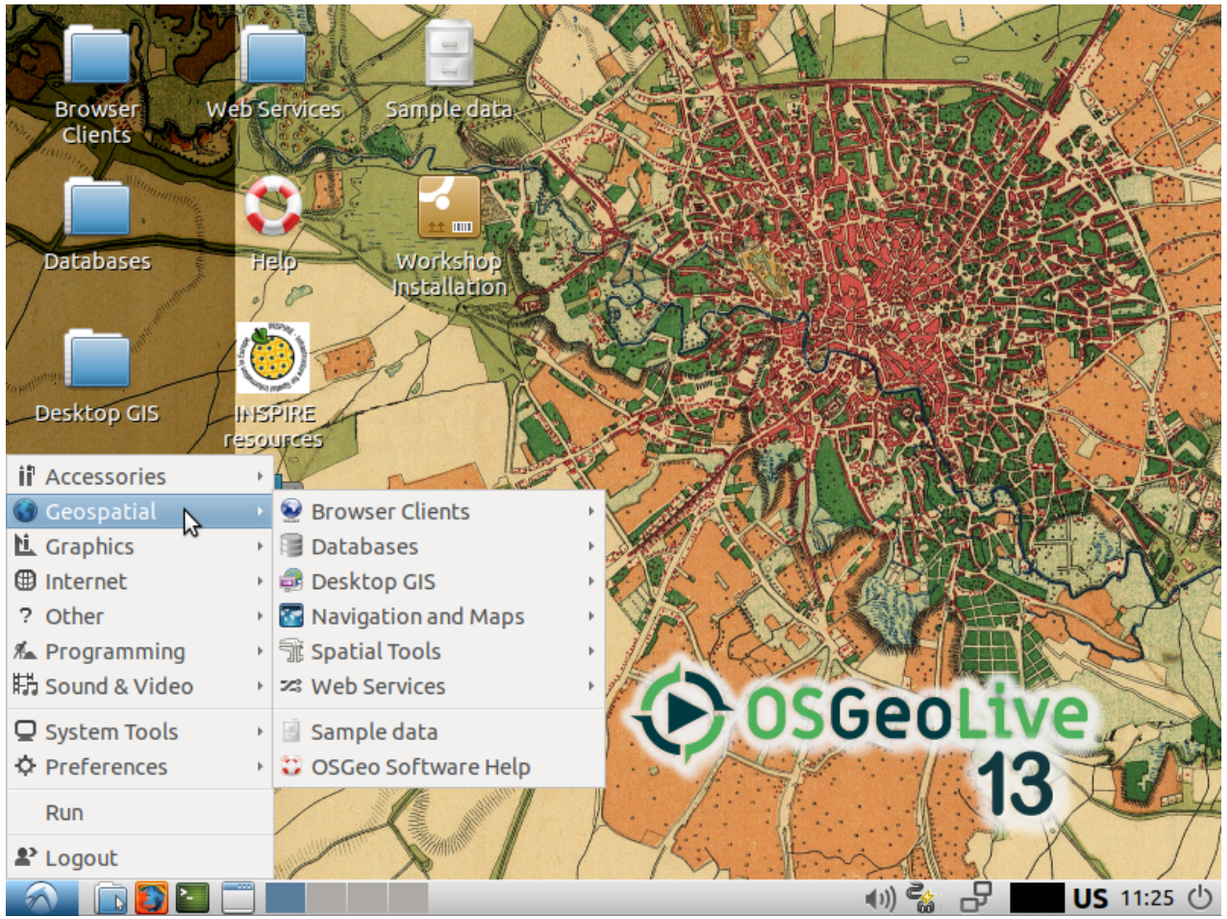
WhereGroup

FOSS Academy

Mapbender Metador Mops

OSGeoLive

Einfach mit OSGeoLive ausprobieren.



Was sind Foreign Data Wrapper - FDW?

& wo können sie zum Einsatz kommen?

Was sind Foreign Data Wrapper - FDW?

Aufbau von Verbindungen zu externen Datenquellen

Warum?

...es gibt doch schon dblink.

dblink

```
CREATE EXTENSION dblink;

SELECT *
FROM dblink('dbname=osm_local',
            'SELECT osm_id, name, way as geom
            FROM public.planet_osm_point
            WHERE amenity = ''cafe''')
     AS foo
     ( osm_id int8, name text, geom geometry(point,4326));
```

FDW basieren auf dem SQL/MED Standard

- SQL Management of External Data
- Definiert, wie sich Datenbanken mit externen Datenquellen verbinden können
- SQL Standard von ISO/IEC 9075-9:2008
- https://wiki.postgresql.org/wiki/SQL/MED

FDW allgemein

- nicht viele Implementationen
- MariaDB CONNECT - folgt nicht dem Standard
- IBM/DB2

FDW in PostgreSQL

2011 mit lesender Unterstützung in PostgreSQL 9.1 eingefügt

2013 schreibende Unterstützung in PostgreSQL 9.3

FDW mit PostgreSQL

Generic SQL Database Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
ODBC	Native		github			CartoDB took over active development of the ODBC FDW for PG 9.5+
JDBC	Native		github			Not maintained ?
JDBC2	Native		github			
SQL_Alchemy	Multicorn	PostgreSQL	GitHub	PGXN	documentation	Can be used to access data stored in any database supported by the sqlalchemy python toolkit.
VirtDB	Native	GPL	GitHub			A generic FDW to access VirtDB data sources (SAP ERP, Oracle RDBMS)

Specific SQL Database Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
PostgreSQL	Native	PostgreSQL	git.postgresql.org		documentation	
Oracle	Native	PostgreSQL	github	PGXN	website	
MySQL	Native		github	PGXN	example	FDW for MySQL
Informix	Native	PostgreSQL	github			
Firebird	Native	PostgreSQL	github	PGXN	README	version 1.1 released (2019-05)
SQLite	Native		github			An FDW for SQLite3 (read-only)
SQLite	Native	PostgreSQL	github	PGXN	README	An FDW for SQLite3 (write support and several pushdown optimization)
Sybase / MS SQL Server	Native		github	PGXN		An FDW for Sybase and Microsoft SQL server
MonetDB	Native		github			

NoSQL Database Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
BigTable or HBase	Native Rust Binding (RPGFFI)	MIT	Github			
Cassandra	Multicorn	MIT	Github	Rankactive		
Cassandra2	Native	MIT	Github			
Cassandra	Multicorn	PostgreSQL	Github			
ClickHouse	Multicorn	BSD	Github		README	
ClickHouse	Native	Apache	Github		README	
CouchDB	Native	PostgreSQL	Github	PGXN		Original version
CouchDB	Native	PostgreSQL	Github			golgauth version (9.1 - 9.2+ compatible)
GridDB	Native	PostgreSQL	Github		README	
InfluxDB	Native	PostgreSQL	Github		README	
Kafka	Native	PostgreSQL	GitHub		README	
Kyoto Tycoon	Native	MIT	Github			
MongoDB	Native	GPL3+	Github	PGXN	README	EDB version
MongoDB	Multicorn	MIT	Github			
MongoDB	Multicorn		Github			Yet Another Postgres FDW for MongoDB
Neo4j	Multicorn	GPLv3	Github		README	FWD for Neo4j and also add a Cypher function to Pg

FDW für viele Datenquellen

- andere SQL-Datenbanken
- einfache Dateien
- unterschiedlichste Geodaten
- Twitter
- Verschiedene Installationen, einige über das PGEX PostgreSQL Extension Network
- https://wiki.postgresql.org/wiki/Foreign_data_wrappers

Wie sieht der Zugriff auf unterschiedliche Quellen aus?

Gugga mr amole!

PostgreSQL- mit PostgreSQL-Datenbank verbinden

Verbinden der Datenbanken natural_earth2 mit osm_local

- Laden der Erweiterung postgresql_fdw
- Foreign Server anlegen
- User Mapping anlegen
- Foreign Table anlegen
- Viel Spaß!

Laden der Erweiterung postgresql_fdw

```
CREATE EXTENSION postgres_fdw;
```

Foreign Server anlegen

```
CREATE SERVER fdw_pg_server_osm_local
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host '127.0.0.1', port '5432', dbname 'osm_local');
```

User MAPPING

```
CREATE USER MAPPING FOR user
SERVER fdw_pg_server_osm_local
OPTIONS (user 'user', password 'user');
```

Fremdtabelle (Foreign Table) anlegen

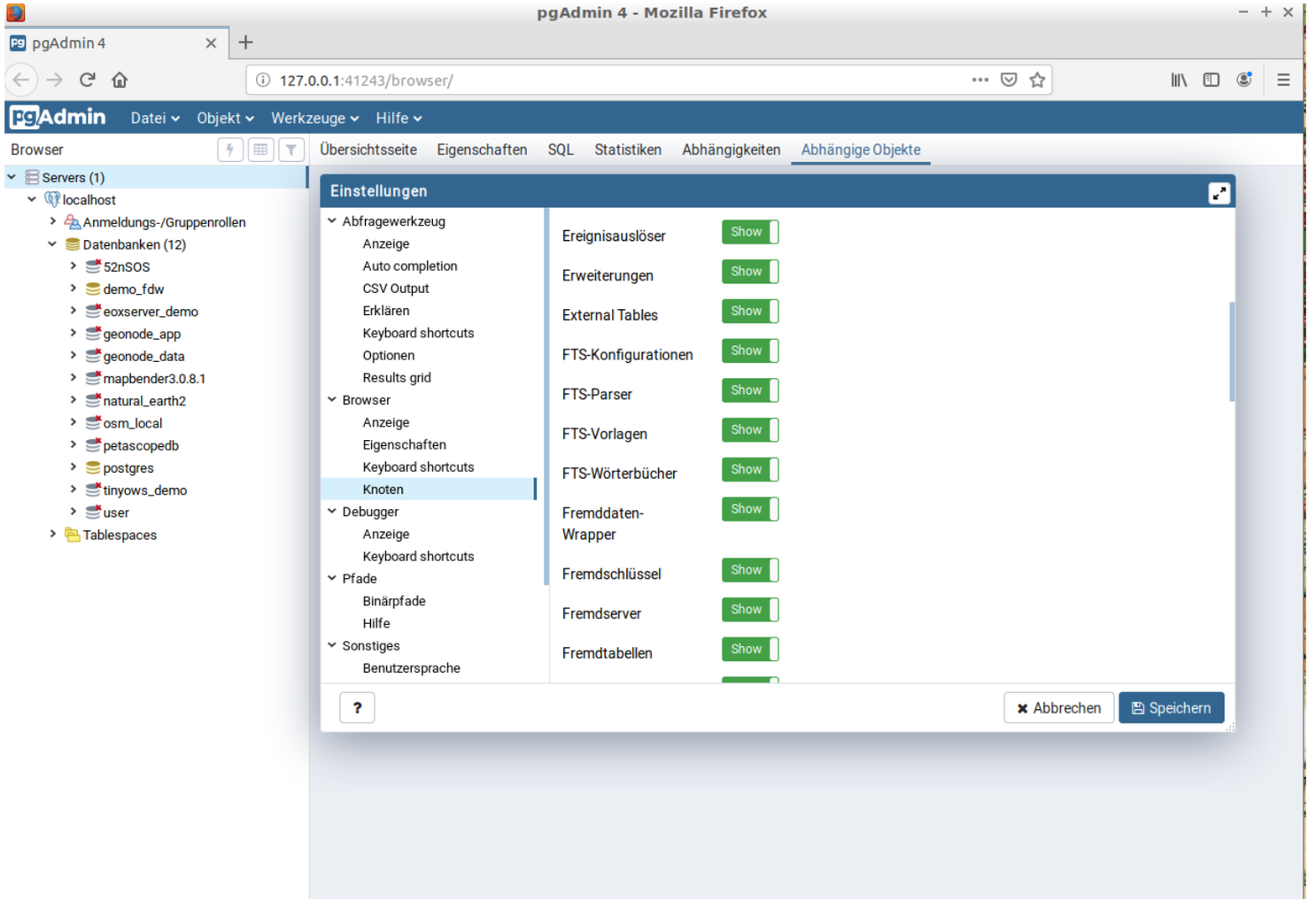
Gesamtes Schema importieren

- LIMIT TO (tablename, tablename)
- EXCEPT (tablename, tablename)

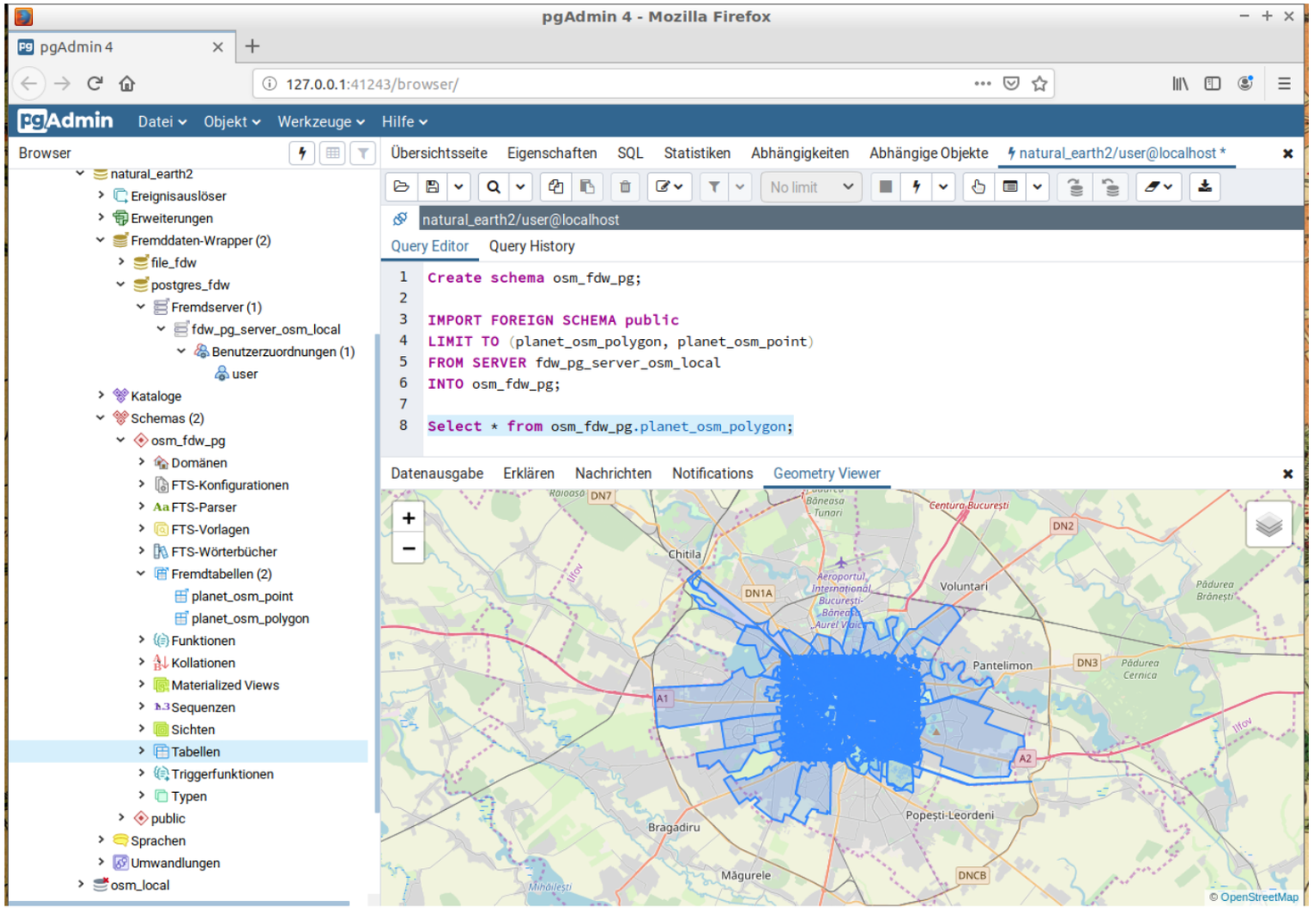
```
Create schema osm_fdw_pg;

IMPORT FOREIGN SCHEMA public
LIMIT TO (planet_osm_polygon, planet_osm_point)
FROM SERVER fdw_pg_server_osm_local
INTO osm_fdw_pg;
```

FDW pgAdmin Einstellungen



FDW mit PostgreSQL



Tabellen können nach Belieben genutzt werden, als wären Sie in einer Datenbank.

```
SELECT count(*)
FROM
  ne_10m_admin_1_states_provinces_shp p
  osm_fdw_pg.planet_osm_point o,
WHERE
  ST_Distance(o.way, p.the_geom) = 0
  AND amenity = 'cafe'
  AND p.name = 'Bucharest';

count
---------
174
```

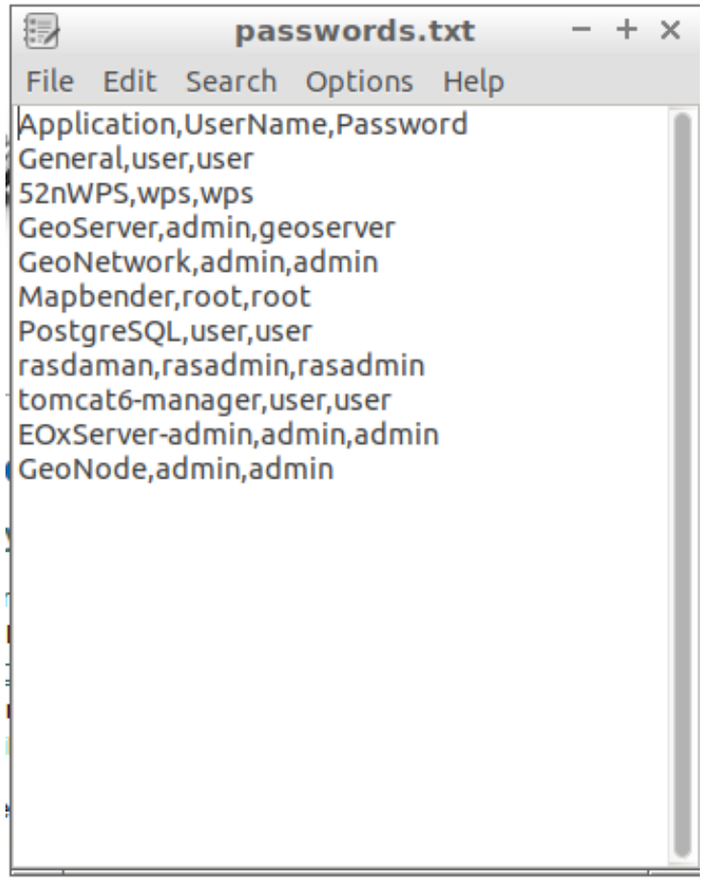
file_fdw für den Zugriff auf Textdateien

https://www.postgresql.org/docs/current/file-fdw.html

```
CREATE EXTENSION file_fdw;

/home/user/Desktop/passwords.txt

CREATE SERVER fdw_server_file
FOREIGN DATA WRAPPER file_fdw;
```



file_fdw für den Zugriff auf Textdateien

Importierte Datei in Tabelle passwords

```
CREATE FOREIGN TABLE passwords (
  application varchar,
  username varchar,
  password varchar
) SERVER fdw_server_file
OPTIONS (
 filename '/home/user/Desktop/passwords.
 format
 'csv',
 header 'true'
);

Select * from passwords;
```

application character varying	username character varying	password character varying
General	user	user
52nWPS	wps	wps
GeoServer	admin	geoserver
GeoNetwork	admin	admin
Mapbender	root	root
PostgreSQL	user	user
rasdaman	rasadmin	rasadmin
tomcat6-manager	user	user
EOxServer-admin	admin	admin
GeoNode	admin	admin

Zugriff auf eine Oracle Datenbank über oracle_fdw

Laden der Erweiterung oracle_fdw

Foreign Server anlegen

User Mapping anlegen

Foreign Table anlegen

Viel Spaß!

Laden der Erweiterung oracle_fdw

```
CREATE EXTENSION oracle_fdw;
```

Foreign Server anlegen

```
CREATE EXTENSION oracle_fdw;
CREATE SERVER oradb
    FOREIGN DATA WRAPPER oracle_fdw
    OPTIONS (dbserver '//dbserver.mydomain.com:1521/ORADB');
GRANT USAGE ON FOREIGN SERVER oradb TO pguser;
```

User MAPPING

```
CREATE USER MAPPING FOR pguser SERVER oradb
OPTIONS (user 'orauser', password 'orapwd');
```

Foreign Table

```
CREATE FOREIGN TABLE oratab (
  id        integer           OPTIONS (key 'true')  NOT NULL,
  text      character varying(30),
  floating  double precision  NOT NULL
) SERVER oradb OPTIONS (schema 'ORAUSER', table 'ORATAB');
```

Direkte Nutzung der Tabellen aus ORACLE

WHERE-Bedingung wird an ORACLE weitergegeben

Information über die angefragten Spalten werden an ORACLE weitergegeben

EXPLAIN-Unterstützung

FDW und OGR

Aus PostgreSQL auf zahlreiche Vektordatenformate via OGR FDW zugreifen

- OGR FDW von Paul Ramsey
- ogr_fdw selbst kompilieren oder OSGeoLive 13.0 nutzen
- Unterstützt zahlreiche Formate wie Geopackage, WFS, OSM, ESRI Shape

Installation

Download & kompilieren

Pakete finden sich unter

https://packages.ubuntu.com/source/bionic/pgsql-ogr-fdw

https://packages.debian.org/sid/postgresql-10-ogr-fdw

```
sudo apt-get install postgresql-10-ogr-fdw
```

OGR_FDW - unterstützte Formate

Terminal öffnen

```
/usr/lib/postgresql/10/bin/ogr_fdw_info -f

Supported Formats:
-> "OGR_GRASS" (readonly)
-> "PCIDSK" (read/write)
-> "netCDF" (read/write)
-> "JP2OpenJPEG" (readonly)
-> "PDF" (read/write)
-> "MBTiles" (read/write)
-> "EEDA" (readonly)
-> "ESRI Shapefile" (read/write)
-> "MapInfo File" (read/write)
-> "UK .NTF" (readonly)
-> "OGR_SDTS" (readonly)
-> "S57" (read/write)
-> "DGN" (read/write)
-> "OGR_VRT" (readonly)
-> "REC" (readonly)
-> "Memory" (read/write)
-> "BNA" (read/write)
-> "CSV" (read/write)
```

Aus PostgreSQL auf zahlreiche Vektordatenquellen über OGR FDW zugreifen

Verbindung zu natural_earth2 ESRI SHP

- Laden der Erweiterung ogr_fdw
- Foreign Server anlegen
- ~~Foreign User anlegen~~
- Foreign Table anlegen
- Viel Spaß!

ogr_fdw-Erweiterung laden

```
CREATE EXTENSION ogr_fdw;
```

ogr_fdw_info

```
cd /usr/lib/postgresql/10/bin/
user@user-VirtualBox:/usr/lib/postgresql/10/bin$ ./ogr_fdw_info -s /home/user/dat
Layers:
ne_10m_geography_marine_polys
ne_10m_geography_regions_points
ne_10m_urban_areas
ne_10m_populated_places
ne_10m_admin_0_countries
ne_10m_geography_regions_polys
ne_10m_admin_1_states_provinces_shp
ne_10m_geography_regions_elevation_points
ne_10m_lakes
ne_10m_ocean
ne_10m_rivers_lake_centerlines
ne_10m_land
```

FDW Shape-Verzeichnis

```
CREATE SERVER myserver
  FOREIGN DATA WRAPPER ogr_fdw
  OPTIONS (
    datasource '/home/user/data/natural_earth2/',
    format 'ESRI Shapefile' );
```

Fremdtabelle erstellen zu Shapedatei

```
CREATE FOREIGN TABLE ne_10m_populated_places (
  fid bigint,
  geom Geometry(Point,4326),
  scalerank integer,
  natscale integer,
  labelrank integer,
  featurecla varchar,
  name varchar,
  ...
  pop2010 real,
  pop2015 real,
  pop2020 real,
  pop2025 real,
  pop2050 real,
  cityalt varchar
) SERVER myserver
OPTIONS (layer 'ne_10m_populated_places');
```

Encod?ng

Meldung: 'utf-8' codec can't decode byte 0xed in position 1: invalid continuation byte

```
CREATE SERVER myserver_latin1
  FOREIGN DATA WRAPPER ogr_fdw
  OPTIONS (
    datasource '/home/user/data/natural_earth2/',
format 'ESRI Shapefile',
config_options 'SHAPE_ENCODING=LATIN1');


Set client_encoding to UNICODE;
```

Untersuchen der Daten über ogr_fdw_info

```
/usr/lib/postgresql/10/bin/ogr_fdw_info -s /home/user/feature_city.osm
Layers:
  points
  lines
  multilinestrings
  multipolygons
  other_relations
```

ogr_fdw_info schreibt die SQL für Sie

```
 /usr/lib/postgresql/10/bin/ogr_fdw_info -s /home/user/feature_city.osm -l points

CREATE SERVER myserver_osm
  FOREIGN DATA WRAPPER ogr_fdw
  OPTIONS (
    datasource '/home/user/feature_city.osm',
    format 'OSM' );

CREATE FOREIGN TABLE points (
  fid bigint,
  geom Geometry(Point,4326),
  osm_id varchar, name varchar,
  barrier varchar,  highway varchar,
  ref varchar,  address varchar,
  is_in varchar,  place varchar,
  man_made varchar,
  other_tags varchar
) SERVER myserver_osm OPTIONS (layer 'points');
```

FDW und EXPLAIN ANALYZE

```
Total points: 77003


Select count(*) from points where highway = 'traffic_signals';
-------
600

EXPLAIN ANALYZE
Select count(*) from points where highway = 'traffic_signals';

"Aggregate  (cost=1027.50..1027.51 rows=1 width=8) (actual time=2586.355..2586.35
"  ->  Foreign Scan on points  (cost=25.00..1025.00 rows=1000 width=0) (actual ti
"        Filter: ((highway)::text = 'traffic_signals'::text)"
"Planning time: 34.801 ms"
"Execution time: 2637.603 ms"
```

OGR_FDW WFS Support

lesende Unterstützung

```
CREATE SERVER myserver_wfs_qgis_server
  FOREIGN DATA WRAPPER ogr_fdw
  OPTIONS (
    datasource 'WFS:http://localhost/cgi-bin/qgis_mapserv.fcgi?map=/home/user/wor
    format 'WFS',
    config_options 'CPL_DEBUG=ON');
```

WFS Verbindung aufbauen

```
Create schema fdw_wfs_qgis_server;

IMPORT FOREIGN SCHEMA ogr_all
FROM server myserver_wfs_qgis_server
INTO fdw_wfs_qgis_server;

Show client_min_messages;
SET client_min_messages=debug2;

Select * from fdw_wfs_qgis_server.ne_10m_admin_0_countries;
```

FDW eine sehr hilfreiche Verbindung

einfach - schnell

schnell auf und abgebaut

verbindet verschiedene Welten

Lesender und schreibender Zugriff

Vielen Dank

Astrid Emde | WhereGroup | astrid.emde@wheregroup.com