

Einführung in die Verwaltung von Geodaten in der PostgreSQL Datenbank mit PostGIS



Charlotte Toma

- FOSS Academy - die Akademie für Freie und Open Source Software



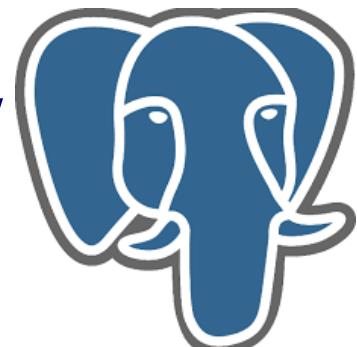
- Das Motto der FOSS Academy ist "Aufwind durch Wissen"
- Aktuelles Schulungsprogramm <https://www.foss-academy.eu/termine>
- FOSS Academy ist ein Schulungsinstitut der



WhereGroup

Datenbankserver PostgreSQL

- Objektrelationales Datenbankmanagementsystem
- Am weitesten entwickelter OpenSource Datenbankserver
- Aktive Entwicklung seit mehr als 20 Jahren
 - <http://www.postgresql.org/about/history>
 - <http://www.postgresql.org/community/contributors/>
- Weltweiter Anwenderkreis, u.a. Firmen wie BASF, Zalando, Heroku, Skype, Afilias und Fujitsu
- Kommerzieller Support
 - https://www.postgresql.org/support/professional_support/
 - Professional Services - Europe



PostGIS ist ein räumlicher "Aufsatz" zur Speicherung und Verwaltung von Geodaten in PostgreSQL



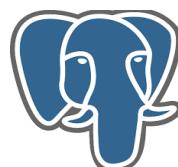
PostGIS

GEOS

PROJ4

LibXML

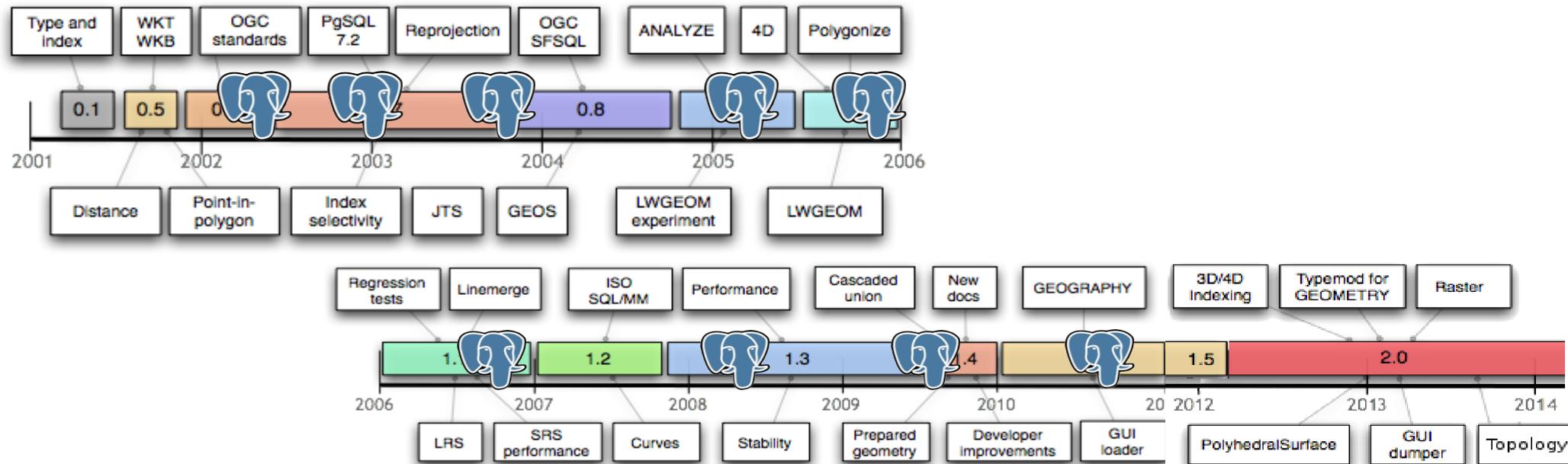
GDAL



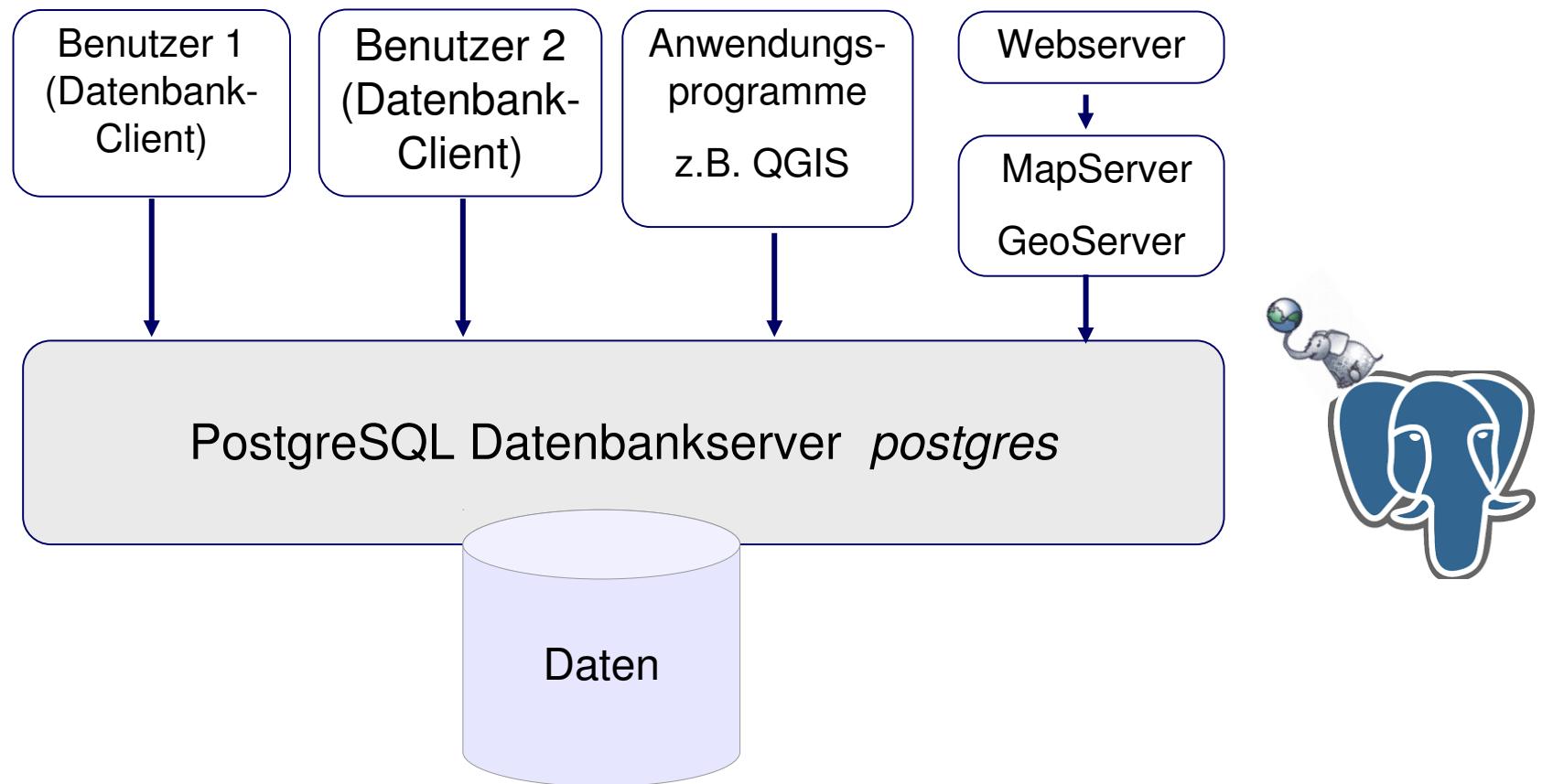
PostgreSQL

PostGIS Geschichte

- Entwicklung ab 2001 - Version 0.1 (eigener Type geometry, WKT, einige Funktionen area(), length(), räumlicher Index, JDBC)
- 2003 Version 0.8 unter Nutzung der GEOS Bibliothek, konform zu "Simple Features for SQL" Spezifikation
- 3. April 2012: Version 2.0 (Meilenstein und Major Release - mit vielen Neuerungen)



Systemarchitektur: PostgreSQL verwendet ein Client/Server-Modell



Sourcen und Systemvoraussetzungen

- Einsatzfähigkeit unter allen UNIX-kompatiblen Plattformen, MAC OS X, Solaris, Windows
- Für Windows liegt ein Installer vor (Installation von PostGIS über den PostgreSQL StackBuilder)
- Sourcen und Installationsanleitung finden sich unter
 - <http://www.postgresql.org/download>
 - <http://www.postgis.org/download/>

PostGIS auf OSGeo-Live



- <http://live.osgeo.org>
- GIS Software Kollektion
- 60 Open Source GIS Anwendungen
- Beispieldaten
- Dokumentationen
- Basiert auf Lubuntu
- Bootfähige DVD, USB-Stick oder virtuelle Maschine
- ISO zum Download unter
<http://adhoc.osgeo.osuosl.org/livedvd/docs/en/download.html>

Datenbank Clients und Hilfsprogramme

- **Clients zur Bearbeitung der Daten und Verwaltung der PostgreSQL-Datenbank**

- psql – kommandozeilenbasierter Datenbank-Client
- ***pgAdmin III*** und pgAdmin 4 – grafischer Datenbank-Client, verfügbar für Windows, Mac und UNIX-Systeme
- phppgAdmin – Webbasierter Datenbank-Client
- ***QGIS DB Manager***, gvSIG CE Database Treiber
- Weitere Datenbank-Clients, z.B. Navicat



- **PostgreSQL-Hilfsprogramme**

- Client Applications/Utilities, z.B. createlang, createdb, dropdb, pg_dump
- Server Applications/Utilities, z.B. postgres, pg_ctl, initdb

Datenbank anlegen über psql

- Datenbank anlegen mit dem Hilfsprogramm **createdb**

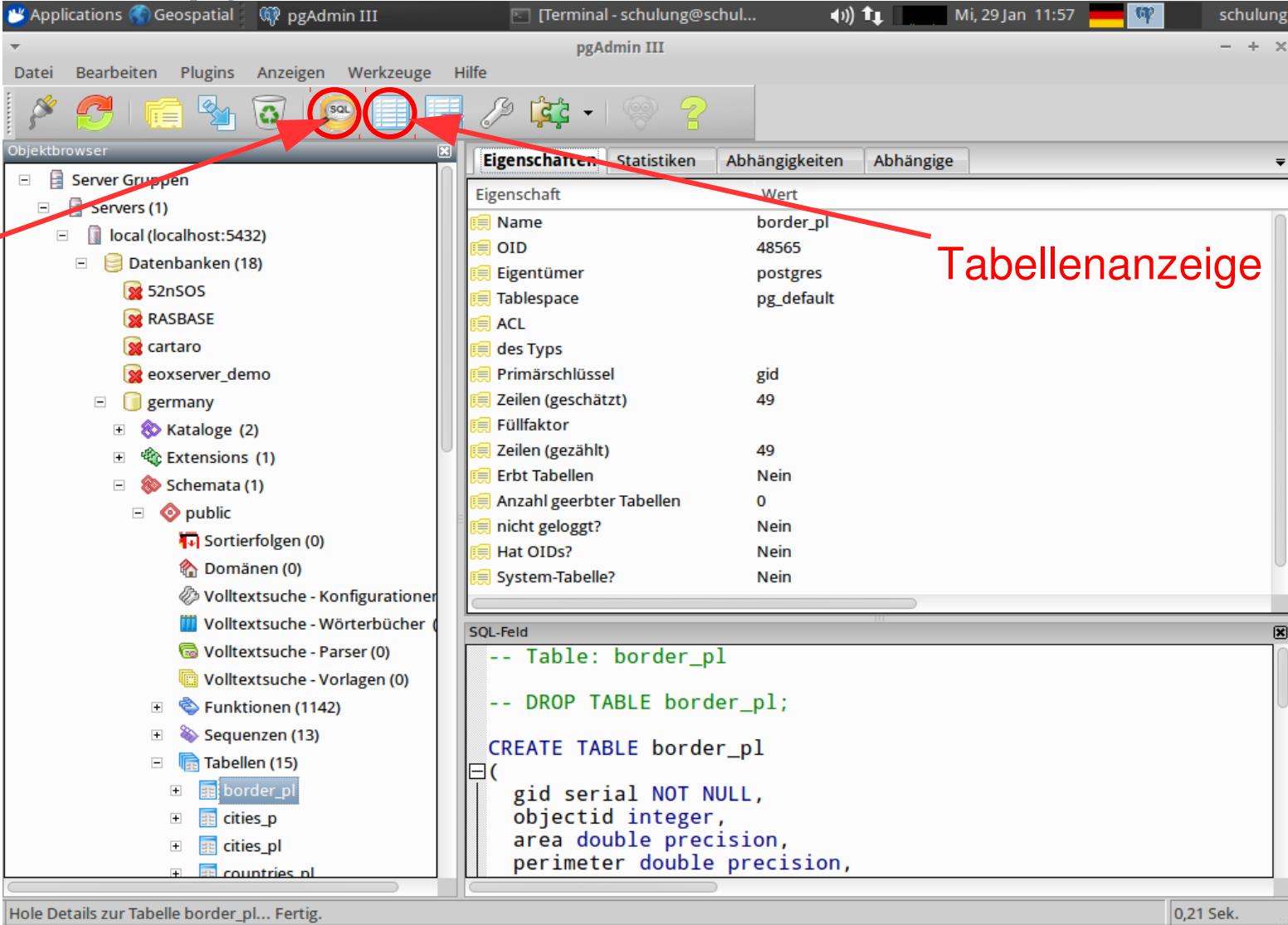
```
$ createdb [-U <user>] [-E <encoding>] [-T <template>] <dbname>  
$ createdb -U postgres -E UTF8 -T postgis_template dbschulung
```

-U <user>	Datenbankbenutzer
-E <encoding>	Zeichensatzkodierung (UTF8,LATIN1)
-T <template>	Datenbank, die als Vorlage dienen soll
<dbname>	Name der zu erstellenden Datenbank

PostgreSQL Client pgAdmin III

Geospatial > Databases > pgAdmin

SQL
Abfrage-
werkzeug



The screenshot shows the pgAdmin III interface. The left pane is the Object Browser displaying database structures. The right pane shows the properties of a selected table named 'border_pl'. The bottom pane contains the SQL code for creating and dropping the table.

Eigenschaften (Properties) for border_pl:

Eigenschaft	Wert
Name	border_pl
OID	48565
Eigentümer	postgres
Tablespace	pg_default
ACL	
des Typs	
Primärschlüssel	gid
Zeilen (geschätzt)	49
Füllfaktor	
Zeilen (gezählt)	49
Erbt Tabellen	Nein
Anzahl geerbter Tabellen	0
nicht geloggt?	Nein
Has OIDs?	Nein
System-Tabelle?	Nein

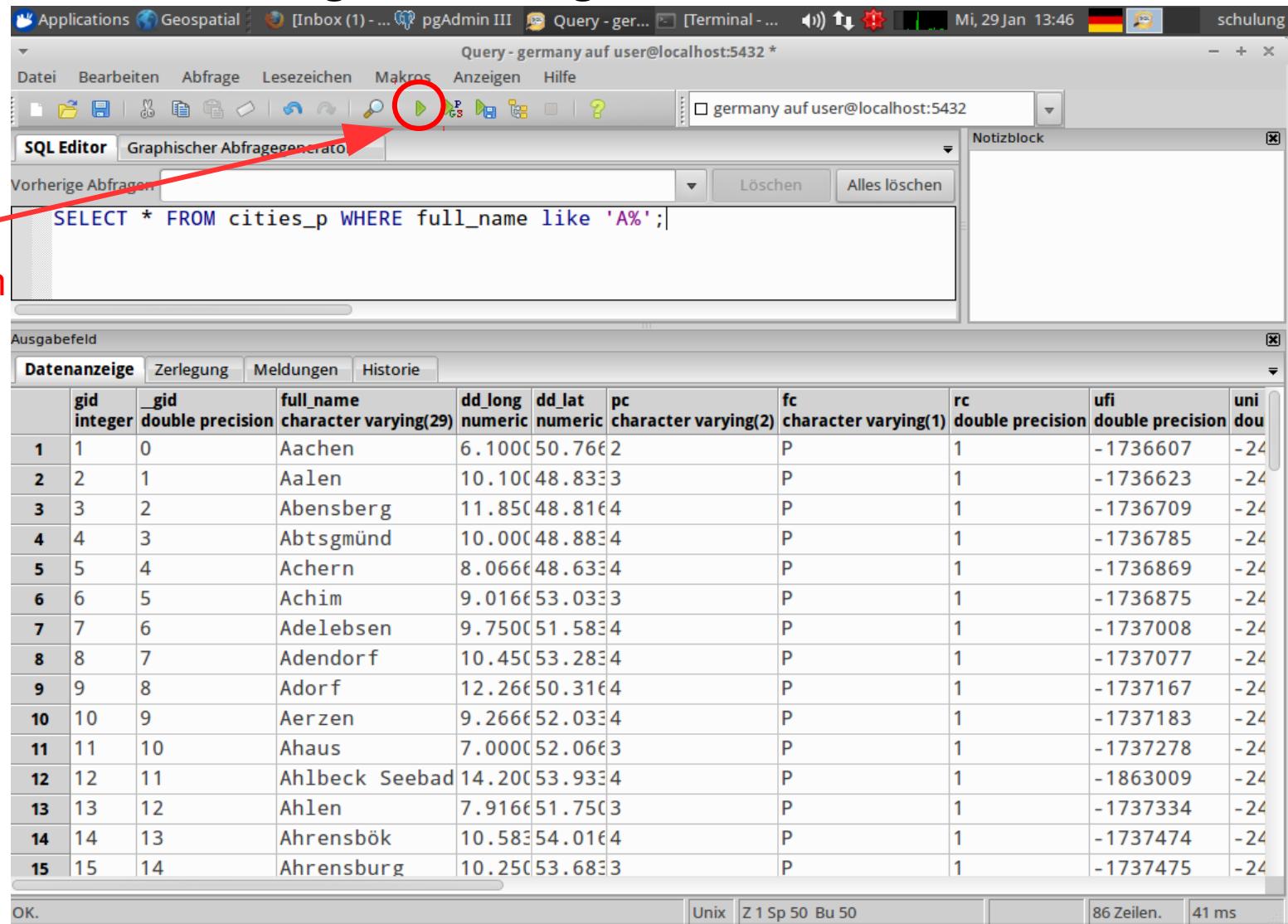
SQL-Feld (SQL Field):

```
-- Table: border_pl
-- DROP TABLE border_pl;
CREATE TABLE border_pl
(
    gid serial NOT NULL,
    objectid integer,
    area double precision,
    perimeter double precision,
```

pgAdmin III – SQL Abfragewerkzeug

Geospatial > Databases > pgAdmin

SQL-Abfragen
ausführen



The screenshot shows the pgAdmin III interface. In the top bar, there's a toolbar with various icons. One of the icons is a green triangle pointing right, which is highlighted with a red circle and a red arrow pointing from the 'SQL-Abfragen ausführen' text above. Below the toolbar is the 'SQL Editor' tab, which is active. Inside the editor, there's a text input field containing the SQL query: 'SELECT * FROM cities_p WHERE full_name like 'A%';'. To the right of the editor is a large table titled 'Ausgabefeld' (Output Field) showing the results of the query. The table has columns for 'gid', 'full_name', 'dd_long', 'dd_lat', 'pc', 'fc', 'rc', 'ufi', and 'uni'. The first 15 rows of the table are shown, listing various German cities starting with 'A'. At the bottom of the pgAdmin window, there are several status bars and buttons.

gid	full_name	dd_long	dd_lat	pc	fc	rc	ufi	uni
1	Aachen	6.1000	50.7662		P	1	-1736607	-24
2	Aalen	10.1000	48.8333		P	1	-1736623	-24
3	Abensberg	11.8500	48.8164		P	1	-1736709	-24
4	Abtsgmünd	10.0000	48.8834		P	1	-1736785	-24
5	Achern	8.0666	48.6334		P	1	-1736869	-24
6	Achim	9.0166	53.0333		P	1	-1736875	-24
7	Adelebsen	9.7500	51.5834		P	1	-1737008	-24
8	Adendorf	10.4500	53.2834		P	1	-1737077	-24
9	Adorf	12.2666	50.3164		P	1	-1737167	-24
10	Aerzen	9.2666	52.0334		P	1	-1737183	-24
11	Ahaus	7.0000	52.0663		P	1	-1737278	-24
12	Ahlbeck Seebad	14.2000	53.9334		P	1	-1863009	-24
13	Ahlen	7.9166	51.7503		P	1	-1737334	-24
14	Ahrensbök	10.5833	54.0164		P	1	-1737474	-24
15	Ahrensburg	10.2500	53.6833		P	1	-1737475	-24

Tipp:
F5 drücken!

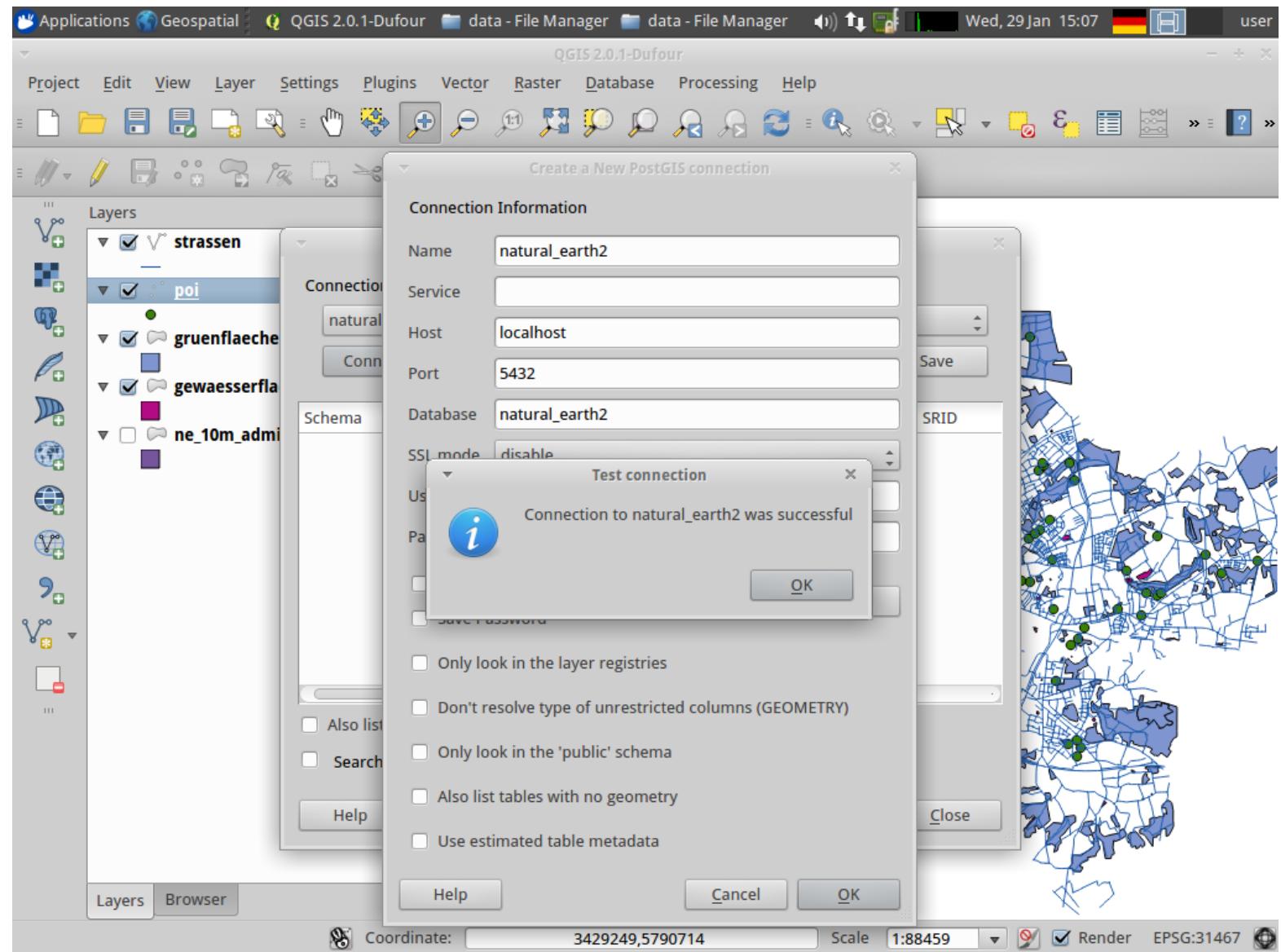
QGIS PostGIS Features

- Visualisierung
- Abfragen über den Abfragemanager
- QGIS DB Manager
- Digitalisierung
- Offline Digitalisierung
- 3D Visualisierung über Plugin
- PostGIS Raster visualisieren per drag & drop aus dem DB Manager
- Plugin PGVS – konkurrierendes Editieren von PostGIS Ebenen

QGIS PostGIS

*Geospatial >
DesktopGIS > QGIS*

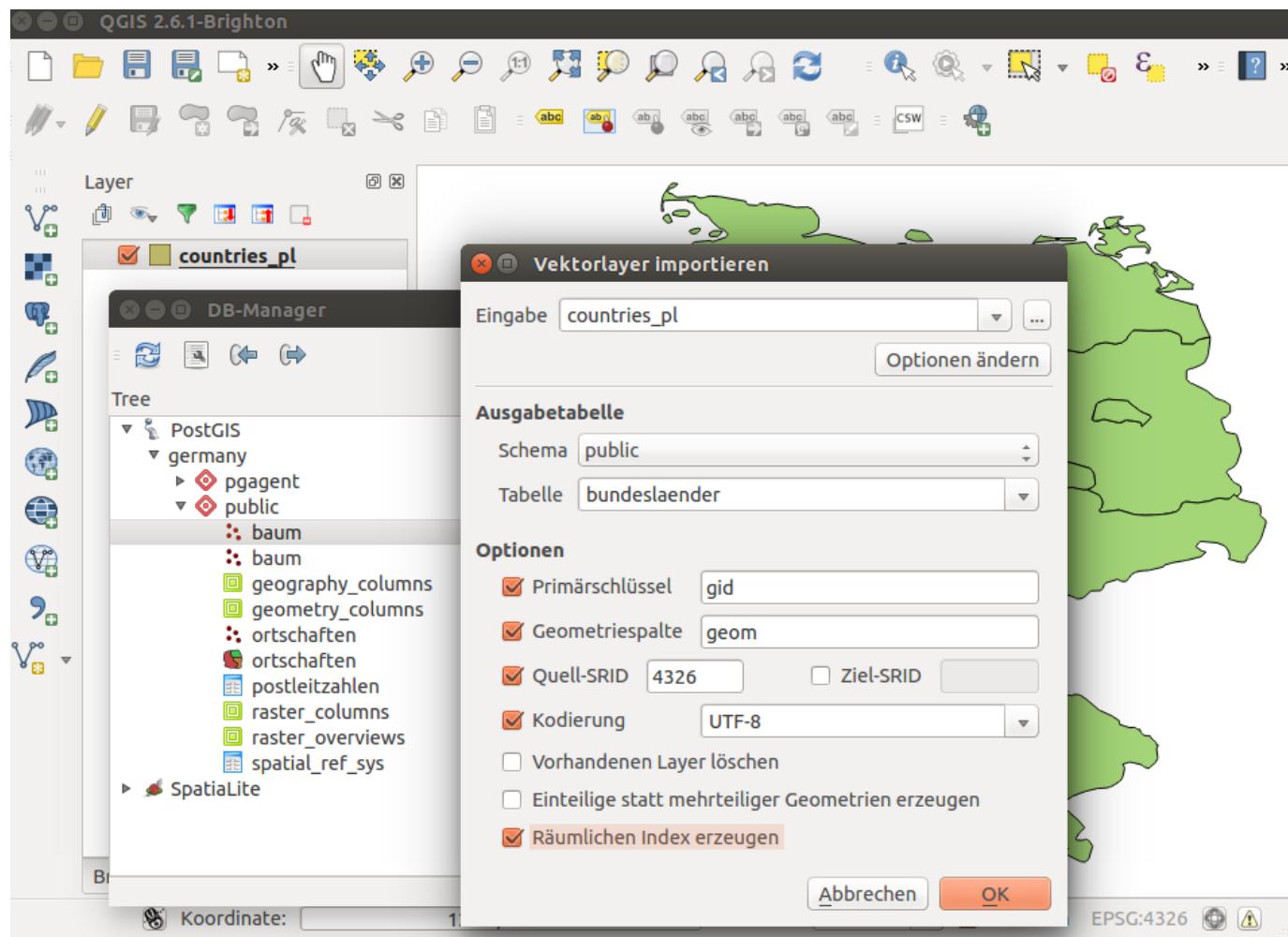
*Layer > Add Layer >
Add PostGIS Layer >
Connections
„NaturalEarth“ > Edit*



Geodatenimport über QGIS DB-Manager

- Einfacher Import / Export
- Visualisierung der Tabelle
- Anzeige der Geodaten
- Anzeige Tabellenstruktur
- Index Erstellung
- Wartung

Database > DB Manager > PostGIS



SQL (Structured Query Language) Syntax

- Standardsprache für den Zugriff auf Datenbanken (ANSI und ISO-Standard)
- Beispiel: Zeige mir alle Einträge aus den Feldern nutzungstyp und id der Tabelle nutzung, bei denen das Feld nutzungstyp den Eintrag Grünland enthält

```
SELECT nutzungstyp, id  
FROM nutzung  
WHERE nutzungstyp = 'Grünland' ;
```

SELECT	= Schlüsselwort/Keyword
nutzungstyp	= Name/Bezeichner
Grünland	= Attribut

- jeder SQL-Befehl schließt mit einem Semikolon ab

Data Definition Language = DDL

- Teil der Sprache SQL über den Datenbanken erstellt und Datenstrukturen geändert werden können
- Beispiel: DB-Struktur anlegen und ändern

```
CREATE DATABASE schulung;
```

```
CREATE TABLE baum (
    gid serial ,
    baumart varchar
);
```

```
ALTER TABLE baum ADD COLUMN nutzung varchar;
ALTER TABLE baum RENAME nutzung TO nutzungstyp;
ALTER TABLE baum ADD CONSTRAINT pk_gid PRIMARY KEY (gid);
```

```
DROP TABLE baum;
```

Achtung – Groß-/Kleinschreibung in PostgreSQL: PostgreSQL interpretiert alle Namen/Bezeichner als Kleinbuchstaben, ansonsten sind diese in Hochkommata anzugeben!
Tabellen- und Feldnamen in Kleinbuchstaben sind einfacher zu handhaben.

Data Manipulation Language = DML

- Teil der Sprache SQL mit dem Daten eingefügt, verändert, abgefragt und gelöscht werden können
- Beispiele: Mehrere oder einzelne Daten einfügen, ändern und aktualisieren

```
INSERT INTO baum (baumart , nutzungstyp) VALUES ('Erle', 'Laubwald');
```

Mehrere Datensätze mit einem INSERT-Befehl einfügen

```
INSERT INTO baum (baumart , nutzungstyp) VALUES  
(‘Erle’, ‘Laubwald’),  
(‘Tanne’, ‘Nadelwald’);
```

Daten ändern/ aktualisieren

```
UPDATE baum SET baumart = ‘Buche’  
WHERE nutzungstyp = ‘Buchenwald’;
```

Daten löschen

```
DELETE FROM baum WHERE gid = 1;
```

Daten abfragen

```
SELECT gid, baumart FROM baum  
WHERE gid > 5  
AND nutzungstyp <> 'Erlenwald'  
ORDER BY gid;
```

Tipp: Abfragen zum Null-Wert funktionieren mit dem folgenden Befehl

[...] WHERE nutzungstyp NOT NULL;

Operatoren **<>** oder **!=** stehen für ungleich

- Beispiel: Ausgabe der Bäume pro Baumart über eine Abfrage/ Sicht
- Sicht (View) – virtuelle Tabelle
- COUNT(*) – Ausgabe der Anzahl
- GROUP BY – zur Gruppierung von Spalten

`CREATE VIEW qry_count AS`

```
SELECT count(baumart) as anzahl,  
       baumart  
    FROM baum  
   GROUP BY baumart  
  ORDER by anzahl;
```

Arbeit mit Schemata

- Organisation der Datenbankobjekte in logischen Gruppen
- Schemata dienen der Strukturierung (z.B. public = alphanumerisch, geo = Geodaten)
- Schema erzeugen (public ist default-Schema)

```
CREATE SCHEMA geo;
```

- Tabelle in einem Schema erzeugen

```
CREATE TABLE geo.bodenarten  
(gid serial, area float8, art varchar);
```

PostGIS Geo-Datenmodell

- Funktions-Präfix **ST_** (spatial type)
- Geometrieobjekte werden als Datentyp **geometry (planar)** oder **geography (spheric)** gespeichert
 - **geometry** enthält meist nur einen Geometriertyp (Punkt, Linie, Polygon, kreisförmige Objekte, Raster u.a.)
 - **GEOMETRYCOLLECTION** kann mehrere Geometriertypen enthalten
- Format der Geometrieobjekte
 - Interne Speicherung als WKB (Well Known Binary)
 - Ausgabe als WKT (Well Known Text) möglich
- Speicherung von Metadaten zu den Feature Tables
 - Pro Feature Table erfolgt ein Eintrag in der Metadatensicht **geometry_columns** bzw. **geography_columns**
(bis PostGIS 1.5 als eigene Tabelle)



- Format der Geometrieobjekte - Beispiele für WKT-Strings
- Weitere Informationen im PostGIS Kapitel **4.1 GIS Objects**

POINT(2572292 5631150)

LINESTRING (2566006 5633207, 2566028 5633215, 2566062 5633227)

MULTILINESTRING ((2566006.4 5633207.9, 2566028.6 5633215.1), (2566062.3 5633227.1, 2566083 5633234.8))

POLYGON ((2568262 5635344, 2568298.5 5635387.6, 2568261.04 5635276.15, 2568262 5635344))

MULTIPOLYGON (((2568262 5635344, 2568298.5 5635387.6, 2568261.04 5635276.15, 2568262 5635344), (2568194.2 5635136.4, 2568199.6 5635264.2, 2568200.8 5635134.7, 2568194.2 5635136.4)))

- **PostgreSQL >= 9.1 und PostGIS >= 2.0**

PostGIS Erweiterung mit Hilfe von CREATE Extension laden

- Laden der PostGIS-Erweiterung
PostGIS Funktionen und Objekt-Definitionen werden geladen

```
CREATE EXTENSION postgis;
```

- Dabei werden auch die Metadatensichten geometry_columns, geography_columns raster_columns und die Tabelle spatial_ref_sys angelegt

- **Tabelle spatial_ref_sys > 3000 EPSG Codes und Definitionen**
- Umrechnungsparameter für Transformation

	srid	auth_name	auth_srid	srtext	proj4text
1	3819	EPSG	3819	GEOGCS["HD1909..."]	+proj=longlat +ellps=bessel...
2	3821	EPSG	3821	GEOGCS["TWD67"...]	+proj=longlat +ellps=aust_S...
3	3824	EPSG	3824	GEOGCS["TWD97"...]	+proj=longlat +ellps=GRS80...
4	3889	EPSG	3889	GEOGCS["IGRS",D...]	+proj=longlat +ellps=GRS80...
5	3906	EPSG	3906	GEOGCS["MGI 190..."	+proj=longlat +ellps=bessel...
6	4001	EPSG	4001	GEOGCS["Unknown..."	+proj=longlat +ellps=airy +...
7	4002	EPSG	4002	GEOGCS["Unknown..."	+proj=longlat +a=6377340....
8	4003	EPSG	4003	GEOGCS["Unknown..."	+proj=longlat +ellps=aust_S...

- Aufbau der **Metadatensicht geometry_columns**
(bis Version 1.5.x Tabelle)

table_catalog	f_table_schema	f_table_name	f_geometry_column	coord_dimension	srid	type
1	public	poi	the_geom	2	31467	POINT
2	public	leitung	the_geom	2	31466	LINESTRING

Erstellen von Geometriespalten

```
CREATE TABLE baum (  
    gid serial PRIMARY KEY,  
    baumart varchar,  
    geom geometry (point,31467)  
);
```

oder Spalte geom nachträglich hinzufügen:

```
ALTER TABLE baum add column geom geometry (point,31467);
```

```
INSERT INTO baum (baumart , geom) VALUES  
('Erle' , ST_GeometryFromText('POINT(3564780.7 5631558.5)', 31467));
```

```
UPDATE baum SET geom = ST_GeometryFromText('POINT(3564850.72 5631672.23)', 31467)  
WHERE gid = 1;
```

- **CSV-Datei (Comma Separated Value) importieren**
 - 1. Tabelle vorbereiten: Tabelle mit entsprechenden Spalten erzeugen

```
CREATE TABLE mytable (gid serial, bezeichnung varchar, x integer, y integer);
```

- 2. Daten laden: CSV-Datei über den Befehl COPY in die Tabelle importieren

```
COPY mytable FROM  
'/data/postgis/osnabrueck/data/osna_pois_csv.csv'  
DELIMITER ','  
CSV  
HEADER  
QUOTE as '\"';
```

Geometrie-Objekt aus Text mit x/y Koordinaten erzeugen

- Beispiel: Spalte x enthält den Rechtswert, Spalte y den Hochwert
- || fügt Zeichenketten zusammen

```
ALTER TABLE mytable ADD COLUMN geom  
geometry(point,31467);
```

```
UPDATE mytable SET geom =  
ST_GeometryFromText ( 'POINT(' || x || ' ' || y || ')' ,  
31467);
```

ST_GeomFromText (text WKT, integer srid)

Anzeige der Geometrie im WKT-Format

- Geometrie wird im WKB-Format (Well Known Binary) gespeichert

```
0101000020EB7A00009A99995976324B41000000B0917B5541
```

- ST_AsEWKT zur Anzeige der Geometrie im WKT-Format (Well Known Text) **mit** Ausgabe der SRID (PostGIS eigene Funktion)

```
Select ST_AsEWKT(<geometrycolumn>) FROM <table name>;
```

```
SELECT ST_AsEWKT(geom) FROM baum;
```

```
st_asEWKT
```

```
SRID=31467;POINT(3564780.7 5631558.75) (1 row)
```

Räumliche Funktionen in PostGIS

- Abfrage und Manipulation von Daten über SQL-Interface
- PostGIS übernimmt Aufgaben, die bisher Desktop GIS durchgeführt haben
- PostGIS enthält räumliche Funktionen und Operatoren
- Räumliche Funktionen und Operatoren wurden gemäß der OGC-Spezifikation vollständig implementiert (ab Version 0.8)
- Erweiterung um SQL Multimedia Applications Spatial Specification (z.B. Curves)
- Erweiterung um weitere PostGIS eigene Funktionen (z. B. ST_AsEWKT)
- ab PostGIS 2.0: ,ST_, neue Funktionen, u.a. Funktionen zur Rasterdaten Analyse

Beispiele für räumliche Funktionen

- ST_Extent(geometry) - Ausgabe des Umgebungsrechteckes
- ST_Touches(geometry,geometry) - berühren sich die zwei Geometrien?
- ST_Intersects(geometry,geometry) - schneiden sich die zwei Geometrien?
- ST_Overlaps(geometry,geometry) - überlappen die zwei Geometrien?
- ST_Area(geometry) - Ausgabe der Fläche der Geometrie
- ST_Union(geometry,geometry) - Vereinigen der Geometrien
- ST_ExteriorRing(polygon) - äußerer Ring des Polygons ausgeben
- ST_Centroid(geometry) - Ausgabe des Schwerpunkts

ST_Extent

- ermittelt das Umgebungsrechteck für die angefragten Daten

```
SELECT ST_Extent(geom) FROM laender;
```

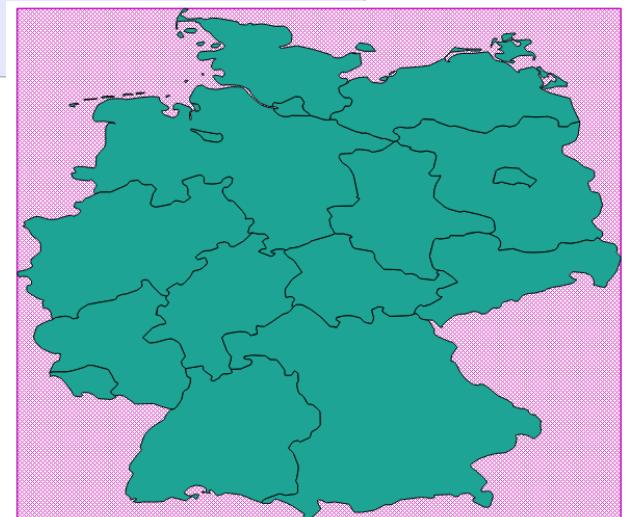
```
-----  
st_extent
```

```
-----  
BOX(5.86416625976 47.27471923,15.0388870 55.0566635131)  
(1 row)
```

```
SELECT ST_Extent(the_geom) FROM  
ne_10m_admin_0_countries;
```

-180 -90, 180 90 (ganze Welt als Ausdehnung)

Database > natural_earth > Schema public



ST_Area

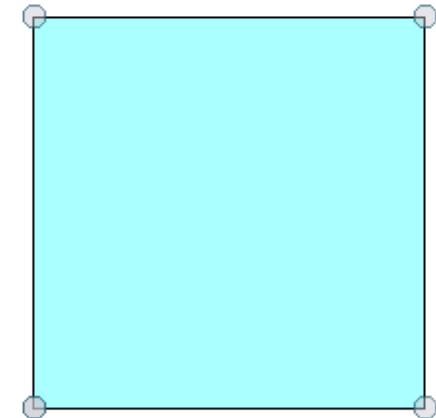
- Ermittelt die Fläche

```
SELECT ST_Area(geom) as area FROM my_polygon;
```

3322654

```
SELECT ST_Area(ST_Transform(the_geom, 25832)) FROM  
ne_10m_admin_0_countries WHERE admin = 'Austria';
```

10.047 → nach Transformation 84267051996, 06

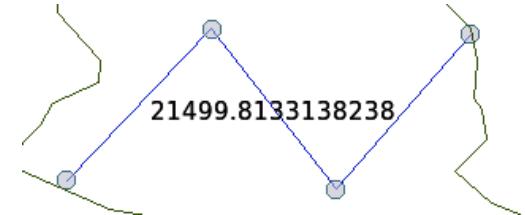


Database > natural_earth > Schema public

Länge ermitteln

- Angabe der Länge in Metern

```
SELECT ST_Length( ST_Transform(the_geom, 31466) ) FROM  
ne_10m_rivers_lake_centerlines;
```



- Angabe der Länge in Kilometern

```
SELECT  
round( ( ST_Length(ST_Transform(the_geom, 31466) ) /1000 )::numeric , 3) ||  
' km' FROM ne_10m_rivers_lake_centerlines;
```



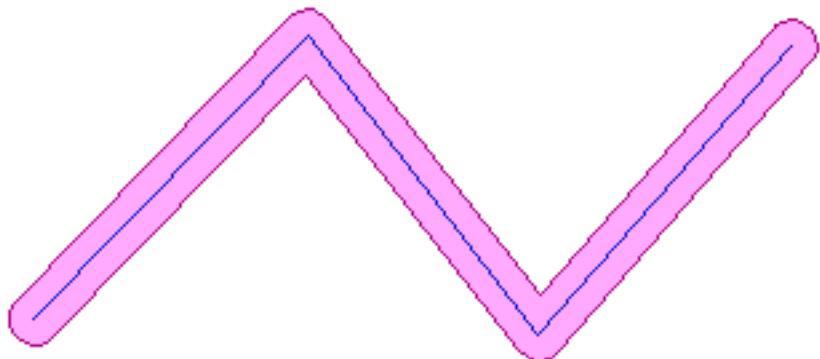
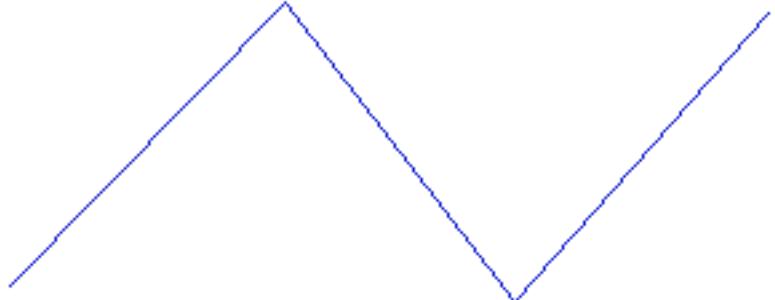
Database > natural_earth > Schema public

Puffer um Geometrie erzeugen

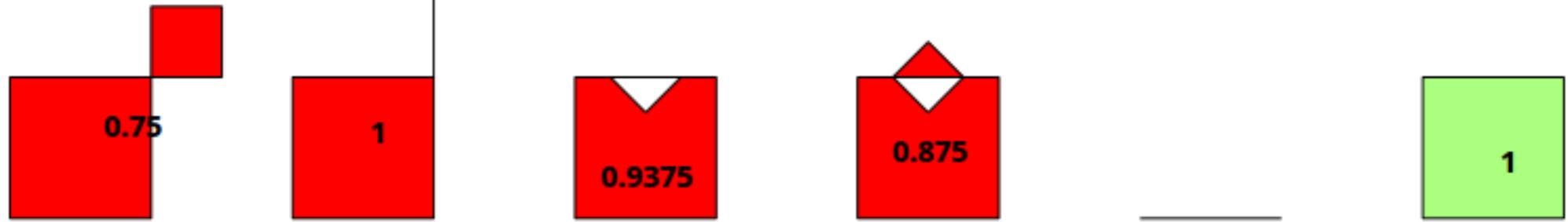
```
SELECT ST_Buffer(the_geom, 0.005) as buffer_50 FROM  
ne_10m_rivers_lake_centerlines;
```

```
CREATE TABLE buffer_line AS  
SELECT ST_Buffer(the_geom,0.005)::geometry(Polygon,4326) AS geom  
FROM ne_10m_rivers_lake_centerlines;
```

```
ALTER TABLE buffer ADD COLUMN gid serial;
```



Datenintegrität



```
SELECT * FROM ne_10m_rivers_lake_centerlines WHERE ST_IsValid(the_geom);
SELECT * FROM ne_10m_rivers_lake_centerlines WHERE ST_IsEmpty(the_geom);
SELECT ST_IsValidReason(the_geom) FROM ne_10m_rivers_lake_centerlines WHERE
not ST_IsValid(the_geom);
SELECT * FROM ne_10m_rivers_lake_centerlines WHERE ST_IsValidDetail(the_geom);
```

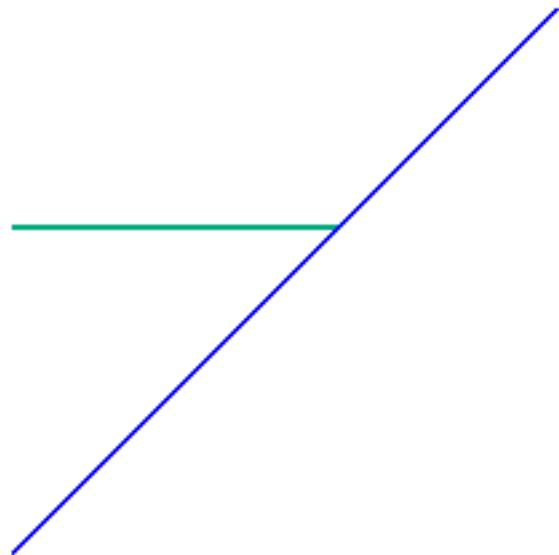
Datenbereinigung

```
SELECT ST_MakeValid(geometry) FROM ne_10m_rivers_lake_centerlines WHERE
ST_IsValid(the_geom) = false;
```

Intersects

```
SELECT ST_Intersects(ST_GeometryFromText('LINESTRING(6 6 ,0 6)',4326) ,  
ST_GeometryFromText('LINESTRING(0 0 , 5 5, 10 10)',4326));
```

t



<https://www.manning.com/books/postgis-in-action-second-edition>

PostGIS in Action

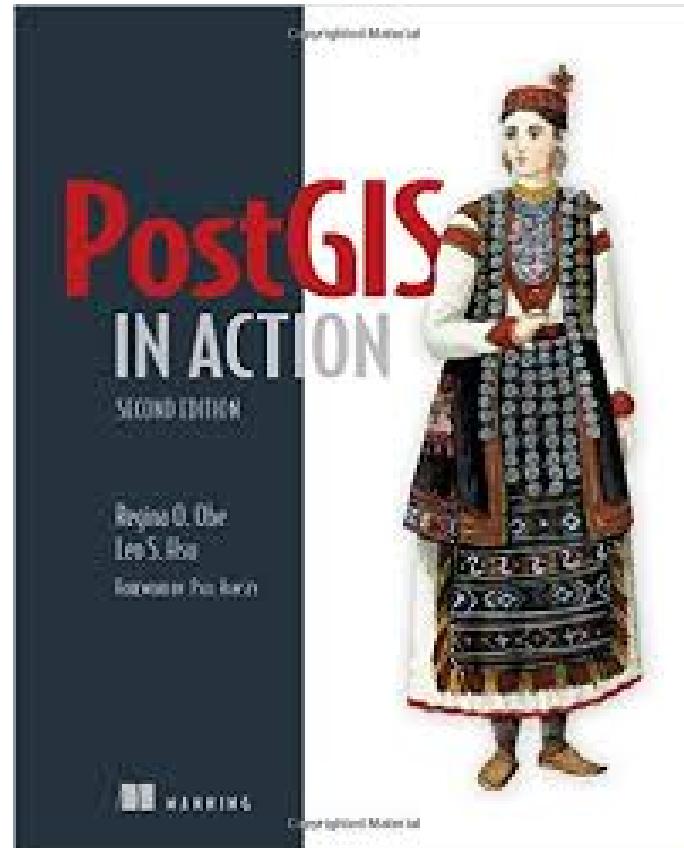
Regina O. Obe und Leo S. Hsu

Vorwort Paul Ramsey

Zweite Auflage 2015

600 Seiten

ISBN 9781617291395



Vielen Dank für Ihre Aufmerksamkeit!

Für Rückfragen stehe ich Ihnen gerne zur Verfügung.

Charlotte Toma

WhereGroup



WhereGroup

charlotte.toma@wheringroup.com

Die Unterlagen finden Sie zum Download unter

https://trac.osgeo.org/osgeolive/wiki/Live_GIS_Workshop_Install