RPostgreSQL

Jan-Philipp Kolb

10 Mai, 2017

Die Nutzung von RPostgreSQL

Figure 1:

PostgreSQL installieren

- Installation Windows
- Installation Linux

PG admin installieren

pgAdmin

pgAdmin ist eine Open-Source-Software zur Entwicklung und Administration von PostgreSQL-Datenbanken und davon abgeleiteten Datenbanken wie EnterpriseDB Postgres Plus Advanced Server oder Greenplum. Eine graphische Benutzeroberfläche erleichtert die Administration von Datenbanken. Der Editor für SQL-Abfragen enthält ein graphisches EXPLAIN, mit dessen Hilfe sich performantere Abfragen erstellen lassen. Durch eine native Anbindung an PostgreSQL ermöglicht das GUI den Zugriff auf die gesamte PostgreSQL-Funktionalität.

Figure 2:

- PGadmin
- Tutorial zur Nutzung von PGadmin

Figure 3:

Neue Datenbank anlegen

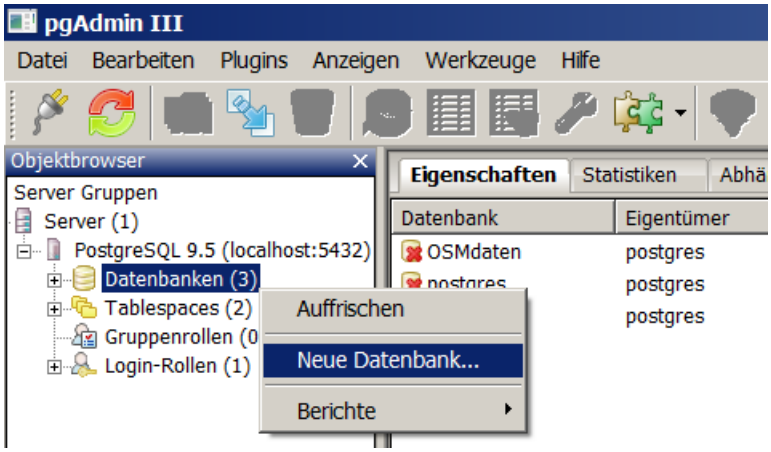

Figure 4:

Eine neue Datenbank

- Unter Linux kann man auch in der Kommandozeile einen neuen Nutzer anlegen:

```
sudo -u postgres createuser Japhilko
```

- und auch eine neue Datenbank anlegen:

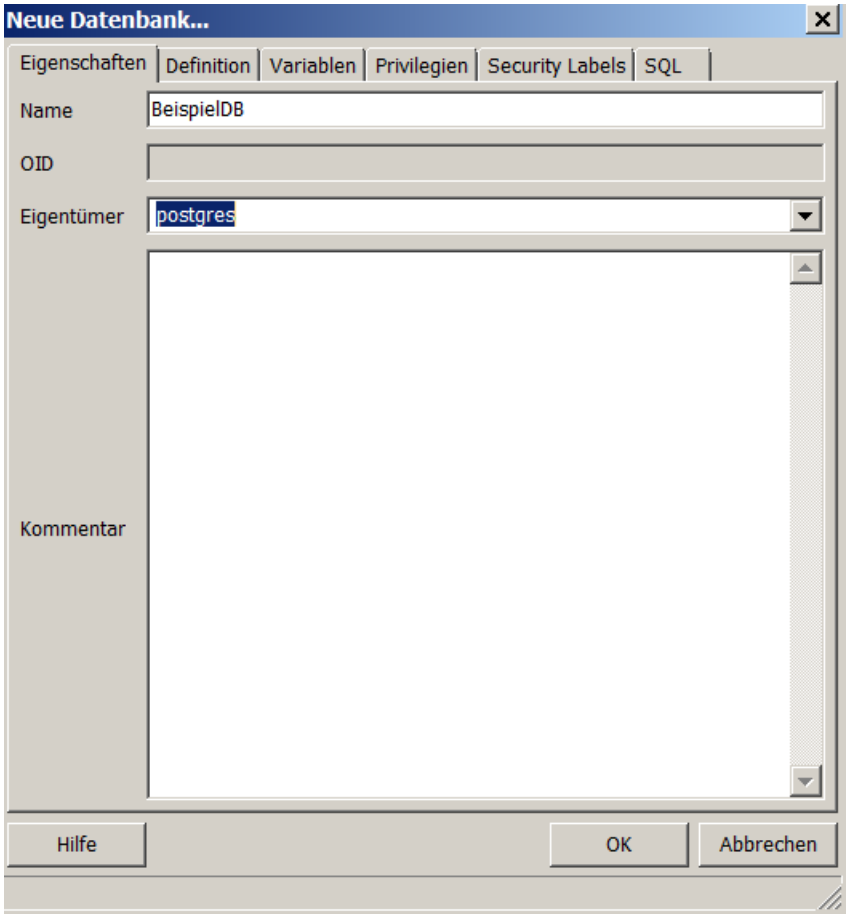
Neue Datenbank...
Eigenschaften
Definition
Variablen
Privilegien
Security Labels
SQL
Name
BeispielDB
OID
Eigentümer
postgres
Kommentar
Hilfe
OK
Abbrechen

Figure 5:

```
sudo -u postgres createdb -E UTF8 -O Japhilko offlgeoc
```

Wie bekomme ich Daten in die Datenbank

```
install.packages("RPostgreSQL")
```

```
library("RPostgreSQL")
```

```
citation("RPostgreSQL")
```

Datenbank mit R verbinden

```
pw <- {"1234"}
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, dbname = "BeispielDB",
                 host = "localhost", port = 5432,
                 user = "postgres", password = pw)
rm(pw) # removes the password

dbExistsTable(con, "BeispielDB")
```

Daten an Datenbank schicken

```
data(mtcars)
df <- data.frame(carname = rownames(mtcars),
                 mtcars,
                 row.names = NULL)
df$carname <- as.character(df$carname)
rm(mtcars)

dbWriteTable(con, "cartable",
             value = df, append = TRUE, row.names = FALSE)
```

- eine Abfrage machen:

```
df_postgres <- dbGetQuery(con, "SELECT * from cartable")
```

- die beiden Tabellen müssten gleich sein

```
identical(df, df_postgres)
```

Anwendung - Geodaten in die Datenbank migrieren

- Zunächst muss für die Datenbank die postgis Erweiterung installiert werden:

```
CREATE EXTENSION postgis;
```

- Der Anfang mit PostGIS
- PostGIS und R

Programm zum Import der OpenStreetMap Daten in PostgreSQL- osm2pgsql

- Ausschnitte der OpenStreetMap Daten können bei der Geofabrik heruntergeladen werden
- Nutzung von osm2pgsql
- Läuft unter Linux deutlich besser
- so könnte bspw. ein Import in PostgreSQL aussehen:

```
osm2pgsql -c -d osmBerlin --slim -C  -k  berlin-latest.osm.pbf
```

```
osm2pgsql -s -U postgres -d offlgeoc /home/kolb/Forschung/osmData/data/saarland-latest.osm.pbf
```

```
osm2pgsql -s -U postgres -d offlgeocRLP -o gazetteer /home/kolb/Forschung/osmData/data/rheinland-pfalz-l
```

Mögliche Abfragen

So bekommt man alle administrativen Grenzen:

```
SELECT name FROM planet_osm_polygon WHERE boundary='administrative'
```

- mehr als eine Spalte auswählen

```
df_postgres <- dbGetQuery(con, "SELECT name, admin_level FROM planet_osm_polygon WHERE boundary='adminis
```

Eine Abfrage zu administrativen Grenzen (ein spezielles Level)

```
df_adm8 <- dbGetQuery(con, "SELECT name, admin_level FROM planet_osm_polygon WHERE boundary='administra
```

Mögliche Abfragen

```
df_hnr <- dbGetQuery(con, "SELECT * FROM planet_osm_line, planet_osm_point
WHERE planet_osm_line.name='Nordring' AND planet_osm_line.highway IN ('motorway','trunk','primary')
AND planet_osm_point.name='Ludwigshafen' AND planet_osm_point.place IN ('city', 'town')
ORDER BY ST_Distance(planet_osm_line.way, planet_osm_point.way)")
```

```
df_hnr <- dbGetQuery(con, "SELECT * FROM planet_osm_line, planet_osm_point
WHERE planet_osm_line.name='Nordring' AND planet_osm_point.name='Ludwigshafen'
ORDER BY ST_Distance(planet_osm_line.way, planet_osm_point.way)")
head(df_hnr)
```

```
df_ <- dbGetQuery(con, "SELECT * FROM planet_osm_line, planet_osm_point
WHERE planet_osm_line.name='Nordring' AND planet_osm_point.name='Ludwigshafen'
ORDER BY ST_Distance(planet_osm_line.way, planet_osm_point.way)")
head(df_hnr)
```

```
colnames(df_)
```

```
table(df_$name)
```

Adresse in einem Ort

```
df_sipp <- dbGetQuery(con, "SELECT * FROM planet_osm_line, planet_osm_point
WHERE planet_osm_line.name='Rechweg' AND planet_osm_point.name='Sippersfeld'
ORDER BY ST_Distance(planet_osm_line.way, planet_osm_point.way)")
head(df_sipp)
```

PostgreSQL and Leaflet

```
install.packages("plot3D")
```

```
library(plot3D)
library(RPostgreSQL)
```

Links

- osm2pgsql
- Andrew Whitby - Roll-your-own geocoding with OpenStreetMap Nominatim on Amazon EC2
- OpenStreetMap Nominatim Server for Geocoding
- Getting Started With PostGIS
- PostGIS geocode
- Nominatim installation
- Wie bekommt man OSM Daten
- PostgreSQL