

postgres Basics

<https://computingforgeeks.com/how-to-install-postgresql-13-on-ubuntu/>

Shell Zugang

psql Cheat Sheet: <https://tomcam.github.io/postgres/>

```
thommie@db2b:~$ sudo su postgres
```

```
\l
```

listet alle Datenbanken

- \c verbindet zu einer Datenbank
- \dt zeigt die Tabellen nach einer Verbindung
- \d und \d+ zeigt die Spalten einer Tabelle
- \du Benutzerrollen

Datenbank und User anlegen

```
create database sqlmail;  
  
create user sqlmail with encrypted password 'xxxxxxxxxxxxx';  
  
grant all privileges on database sqlmail to sqlmail;
```

Datenbank duplizieren

WER greift gerade auf die DB zu?

```
SELECT username, datname, state FROM pg_stat_activity WHERE  
username='discourse2';
```

Alle aktiven Verbindungen zu dieser DB trennen

```
SELECT pg_terminate_backend(pid)  
FROM pg_stat_activity  
WHERE datname = 'discourse2' AND leader_pid IS NULL;
```

Danach kann man die Datenbank duplizieren, indem man die Quelldatenbank als Template benutzt

```
CREATE DATABASE newdb WITH TEMPLATE originaldb OWNER dbuser;
```

Schemata

<https://www.postgresqltutorial.com/postgresql-administration/postgresql-schema/> Das Standard Schema ist public

Arbeiten an der Datenbank

dt listet alle Tabellen in einer Datenbank auf

Nach "\c discourse2": alle Tabellen in einer DB löschen

```
DO $$ DECLARE
  r RECORD;
BEGIN
  FOR r IN (SELECT tablename FROM pg_tables WHERE schemaname =
current_schema()) LOOP
    EXECUTE 'DROP TABLE ' || quote_ident(r.tablename) || ' CASCADE';
  END LOOP;
END $$;
```

q beendet die Verbindung

Benutzeraktionen

\du listet alle User

neuen User anlegen ("ROLE")

```
postgres=# CREATE ROLE discourse2;
CREATE ROLE
postgres=#
```

Passwort setzen

```
ALTER ROLE [username] WITH PASSWORD 'xxxx';
```

GRANT Statement

nur login

```
ALTER ROLE discourse2 LOGIN;
```

Weitere Privilegien:

```
GRANT privilege_list | ALL  
ON table_name  
TO role_name;
```

privilege_list kann `SELECT ' ', INSERT ' ', UPDATE ' ', DELETE ' ', TRUNCATE ' '` etc. sein. Mit ALL geht alles (lesen, schreiben, löschen usw.

```
GRANT ALL ON DATABASE [dbname] TO [username];  
  
postgres=# GRANT ALL ON DATABASE keycloak TO keycloak;  
GRANT
```

Löschen eines Users

```
drop user IF EXISTS dendrite;
```

Tabellen-Aktionen

Alle Tabellen löschen

```
DO $$  
DECLARE  
    r RECORD;  
BEGIN  
    FOR r IN (SELECT tablename FROM pg_tables WHERE schemaname = 'public')  
    LOOP  
        EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(r.tablename) || '  
CASCADE';  
    END LOOP;  
END $$;
```

check

```
SELECT tablename FROM pg_tables WHERE schemaname = 'public';
```

Daten-Aktionen

```
do $$ declare  
    r record;  
begin  
    for r in (select tablename from pg_tables where schemaname = 'my-schema-
```

```
name') loop
    execute 'drop table if exists ' || quote_ident(r.tablename) || '
cascade';
end loop;
end $$;
```

Komplette Datenbank löschen

```
drop DATABASE onlyoffice;
```

Historie der Kommandos in psql

\s command history

beenden

\q DB shell beenden

Dump einer Datenbank als sql file

- Login auf postgres Server
- sudo nach postgres
- Wechsel ins Home Dir von postgres (wg. Schreibrechten für sql File)

```
root@db2b:/home/thommie# sudo su postgres
postgres@db2b:/home/thommie$ cd ~
postgres@db2b:~$ pwd
/var/lib/postgresql
```

mit psql \l Namen aller DBs auflisten

```
postgres@db2b:~$ psql
psql (13.6 (Ubuntu 13.6-1.pgdg20.04+1))
Type "help" for help.
postgres=# \l
```

Danach die Datenbanken dumpen:

```
pg_dump pixelfed>> pixelfed_tokoeka.sql
```

Datenbank löschen

```
$ dropdb your_database
```

Verbindungstest zum postgresql Server

Test einer postgres Verbindung: pg_isready ist Teil von postgresql-client

```
pg_isready -d <db_name> -h <host_name> -p <port_number> -U <db_user>
```

Praktisch:

```
root@docker2:/var/discourse# pg_isready -d Discoursedev -h 10.10.10.18 -p 5432 -U Discourse
10.10.10.18:5432 - Verbindungen werden angenommen
root@docker2:/var/discourse#
```

Import der Datenbanken

```
psql -U postgres -W -d keycloak -f keycloak_tokoeka.sql
```

Templates und UTF8 encoding

Standardmässig werden neue Datenbanken mit der Zeichenkodierung SQL_ASCII angelegt. Das passt meistens, aber nicht immer. Mit dieser Methode werden DBs mit UTF8 Kodierung angelegt:

<https://www.shubhamdipt.com/blog/how-to-change-postgresql-database-encoding-to-utf8/>

```
postgres=# UPDATE pg_database SET datistemplate = FALSE WHERE datname = 'template1';
postgres=# DROP DATABASE template1;
postgres=# CREATE DATABASE template1 WITH TEMPLATE = template0 ENCODING = 'UTF8';
postgres=# UPDATE pg_database SET datistemplate = TRUE WHERE datname = 'template1';
postgres=# \c template1;
You are now connected to database "template1" as user "postgres".
template1=# VACUUM FREEZE;
```

Neu angelegte DBs auf Basis von template1 haben danach utf8 encoding

From:

<https://wiki.netzwissen.de/> - **netzwissen.de Wiki**

Permanent link:

<https://wiki.netzwissen.de/doku.php?id=postgres&rev=1746981980>

Last update: **11/05/2025 - 16:46**

