

# PostgreSQL unter Debian Linux

---

PostgreSQL  
Because **elephants** are just  
plain better than dolphins.



Einführung für PostgreSQL 7.4 unter Debian Linux (Stand 30.04.2008)  
von Moczon T. und Schönfeld A.

## Inhalt

1. Installation .....	2
2. Anmelden als Benutzer „postgres“ .....	2
2.1 Anlegen eines neuen Benutzers .....	2
2.2 Anlegen einer neuen Datenbank.....	2
3. Lokale Anmeldung an der Datenbank .....	2
4. Anmeldung an der Datenbank durch ein entferntes System.....	3
5. Datenbankfunktionen .....	4
5.1 Version ermitteln.....	4
5.2 Anlegen einer Tabelle.....	4
5.2.1 Regulärer Ausdruck für die Postleitzahl festlegen .....	5
5.2.2 View für Adressen anlegen.....	5
5.2.3 Regeln festlegen .....	5
5.3 Vererbung von Tabellen .....	5
5.4 Der COPY-Befehl.....	7
5.4.1 Ausgabe auf dem Bildschirm .....	7
5.4.2 Ausgabe in eine Datei (als Benutzer „postgres“) .....	7
5.4.3 Import der Datensätze aus einer Datei .....	7
6. Quellen .....	7

# PostgreSQL unter Debian Linux

---

## 1. Installation

Zuerst installieren wir PostgreSQL 7.4 über den Debian Packet Manager:

```
# apt-get install postgresql
```

## 2. Anmelden als Benutzer „postgres“

Bevor wir uns als „postgres“ anmelden können müssen wir uns zuerst als Superuser anmelden:

```
# su
```

Anschließend wechseln wir zum Benutzer „postgres“:

```
# su - postgres
```

### 2.1 Anlegen eines neuen Benutzers

Die Installation von PostgreSQL bringt eine Reihe von neuen Befehlen mit. Neue Benutzer können mit dem Befehl „createuser“ und (falls gewünscht) mit einem Passwort durch den Parameter „-P“ angelegt werden. Hier ein Beispiel zum Anlegen des Benutzers „gr6“:

```
postgres@pc11:~$ createuser gr6 -P
Geben Sie das Passwort des neuen Benutzers ein: rdf0608
Geben Sie es noch einmal ein: rdf0608
Soll der neue Benutzer Datenbanken erzeugen dürfen? (j/n) j
Soll der neue Benutzer weitere neue Benutzer erzeugen dürfen? (j/n) j
CREATE USER
postgres@pc11:~$
```

### 2.2 Anlegen einer neuen Datenbank

Zur Erstellung einer neuen Datenbank gibt es den Befehl „createdb“. In folgendem Beispiel erstellen wir die Datenbank „rdf“ anhand der Vorlage „template1“:

```
postgres@pc11:~$ createdb -T template1 rdf
CREATE DATABASE
postgres@pc11:~$
```

## 3. Lokale Anmeldung an der Datenbank

Nun stellen wir eine Verbindung mit der zuvor angelegten Datenbank „rdf“ unter dem Standardbenutzer „postgres“ her:

```
postgres@pc11:~$ psql -d rdf
Willkommen bei psql 7.4.19, dem interaktiven PostgreSQL-Terminal.
Geben Sie ein: \copyright für Urheberrechtsinformationen
                \h für Hilfe über SQL-Anweisungen
                \? für Hilfe über interne Anweisungen
                \g oder Semikolon, um eine Anfrage auszuführen
                \q um zu beenden

rdf=#
```

# PostgreSQL unter Debian Linux

---

Möchten wir eine Übersicht der vorhandenen Datenbanken, so erhalten wir diese durch Eingabe von „\l“ (kleines „L“):

```
rdf=# \l
          Liste der Datenbanken
  Name      | Eigentümer | Kodierung
-----+-----+-----
 rdf        | postgres  | UNICODE
 template0  | postgres  | UNICODE
 template1  | postgres  | UNICODE
(3 Zeilen)

rdf=#
```

## 4. Anmeldung an der Datenbank durch ein entferntes System

Zuerst müssen wir am Server die Anmeldung ermöglichen. Hierzu bearbeiten wir die Datei „pg\_hba.conf“:

```
# nano /etc/postgresql/7.4/main/pg_hba.conf
```

Unten fügen wir nun eine neue Zeile mit folgenden Daten hinzu (in diesem Beispiel für den Client 192.168.6.1 mit dem Subnetz 255.255.255.0):

```
host      all      all      192.168.6.1      255.255.255.0      md5
```

Die Reihenfolge hierbei ist: *Host, Datenbank, User, IP-Adresse, IP-Maske* und *Auth.-Methode*.

Weitere Informationen finden Sie unter:

<http://www.postgresql.org/docs/7.4/static/client-authentication.html>

In der Datei „postgresql.conf“ muss nun noch der Parameter „tcpip\_socket“ bearbeitet werden:

```
# nano /etc/postgresql/7.4/main/postgresql.conf

tcpip_socket = true
```

Zuletzt muss PostgreSQL durch den Benutzer „postgres“ (s.oben) neu gestartet werden:

```
# /etc/init.d/postgresql-7.4 restart
```

# PostgreSQL unter Debian Linux

---

Am entfernten System muss (soweit nicht vorhanden) noch die Client-Software installiert werden:

```
# apt-get install postgresql-client-7.4
```

Die Verbindung zum Server mit der IP 192.168.6.2 lässt sich nun folgendermaßen herstellen:

```
# psql --host=192.168.6.2 --dbname=rdf --username=gr6 --password
Passwort: rdf0608
Willkommen bei psql 7.4.19, dem interaktiven PostgreSQL-Terminal.
```

```
Geben Sie ein: \copyright für Urheberrechtsinformationen
               \h für Hilfe über SQL-Anweisungen
               \? für Hilfe über interne Anweisungen
               \g oder Semikolon, um eine Anfrage auszuführen
               \q um zu beenden
```

```
rdf=#
```

## 5. Datenbankfunktionen

### 5.1 Version ermitteln

Die Version kann mit dem Befehl „*SELECT version();*“ ermittelt werden:

```
rdf=> SELECT version();
                                             version
-----
PostgreSQL 7.4.19 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.1.2
20061115 (prerelease) (Debian 4.1.1-21)
(1 Zeile)

rdf=>
```

### 5.2 Anlegen einer Tabelle

Hier ein Beispiel zum Anlegen der Tabelle Adressen mit einem Primary-Key der bei 1000 startet:

```
rdf=# CREATE SEQUENCE seq_id START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE
rdf=# CREATE TABLE Adressen (
rdf(#      id          BIGINT  NOT NULL
rdf(#                               PRIMARY KEY
rdf(#                               DEFAULT NEXTVAL('seq_id'),
rdf(#      name       VARCHAR  NOT NULL,
rdf(#      vorname    VARCHAR  NOT NULL,
rdf(#      strasse     VARCHAR  NOT NULL,
rdf(#      plz         CHAR(5)  NOT NULL,
rdf(#      ort         VARCHAR  NOT NULL
rdf(# );
HINWEIS: CREATE TABLE / PRIMARY KEY erstellt implizit einen Index
»adressen_pkey« für Tabelle »adressen«
CREATE TABLE
rdf=#
```

Die Einträge werden wie üblich über den SQL-Befehl „*INSERT*“ eingefügt.

# PostgreSQL unter Debian Linux

---

## 5.2.1 Regulärer Ausdruck für die Postleitzahl festlegen

PostgreSQL ermöglicht es auch reguläre Ausdrücke festzulegen. Hier ein Beispiel für die Postleitzahl:

```
rdf=# ALTER TABLE Adressen ADD CONSTRAINT plz_check CHECK (plz ~ '^[0-9][0-9][0-9][0-9]$');
ALTER TABLE
rdf=#
```

## 5.2.2 View für Adressen anlegen

Hier ein Beispiel zum Anlegen eines neuen Views „view\_adressen“:

```
rdf=# CREATE VIEW view_adressen AS
rdf=#     SELECT name, vorname, strasse, plz, ort FROM Adressen ORDER BY
name;
CREATE VIEW
rdf=#
```

Abgefragt werden kann der View später folgendermaßen:

```
SELECT * FROM view_adressen;
```

## 5.2.3 Regeln festlegen

Hier eine Beispielregel die das Löschen von Einträgen aus der Tabelle „Adressen“ verhindert:

```
CREATE RULE dont_delete AS ON DELETE TO Adressen DO INSTEAD NOTHING;
```

Nachfolgender Befehl hätte somit keine Funktion mehr:

```
DELETE FROM Adressen WHERE name='Sepp';
```

Hier auch noch ein Beispiel zum Entfernen der Regel:

```
DROP RULE dont_delete ON Adressen;
```

## 5.3 Vererbung von Tabellen

Um die Vererbung von Tabellen zu erklären, nehmen wir am besten ein kleines Beispiel. Wir leiten die Tabelle „APerson“ von der vorhandenen „Adressen“ ab:

```
rdf=# CREATE TABLE APerson (
rdf=#     geb date
rdf=# ) INHERITS(Adressen);
CREATE TABLE
```

Nun fragen wir erstmal die Tabelle „Adressen“ ab:

```
rdf=# SELECT * FROM Adressen;
 id | name      | vorname  | strasse      | plz  | ort
-----+-----+-----+-----+-----+-----
 1001 | Moczon    | Thomas   | Strasse      | 90489 | Nürnberg
 1002 | Schoenfeld | Alexander | Musterstr. 11 | 90617 | Puschendorf
(2 Zeilen)
```

# PostgreSQL unter Debian Linux

---

Nun fügen wir einen neuen Datensatz in die Tabelle „APerson“ ein:

```
rdf=# INSERT INTO APerson (name, vorname, strasse, plz, ort, geb) VALUES
('Mustermann', 'Max', 'Musterweg 11', 12345, 'Musterstadt', NOW());
```

Wenn wir nun die Tabelle „APerson“ abfragen sehen wir Max Mustermann:

```
rdf=# SELECT * FROM APerson;
 id | name      | vorname | strasse   | plz | ort      | geb
-----+-----+-----+-----+-----+-----+-----
--
 1003 | Mustermann | Max     | Musterweg 11 | 12345 | Musterstadt | 2008-04-30
(1 Zeile)
```

Fragen wir nun erneut die Tabelle „Adressen“ ab, so sehen wir folgendes:

```
rdf=# SELECT * FROM Adressen;
 id | name      | vorname | strasse   | plz | ort
-----+-----+-----+-----+-----+-----
 1001 | Moczon    | Thomas  | Strasse   | 90489 | Nürnberg
 1002 | Schoenfeld | Alexander | Musterstr. 11 | 90617 | Puschendorf
 1003 | Mustermann | Max     | Musterweg 11 | 12345 | Musterstadt
(3 Zeilen)
```

Wie wir sehen können ist nun auch Max Mustermann in dieser Tabelle zu finden, obwohl wir ihn ja eigentlich nur in der Tabelle „APerson“ hinzugefügt hatten. Dies ist der Fall, da wir die Tabelle abgeleitet haben.

Möchten wir nun nur die Einträge sehen die zur Tabelle „Adressen“ gehören, so erreichen wir dies durch das Schlüsselwort „ONLY“:

```
rdf=# SELECT * FROM ONLY Adressen;
 id | name      | vorname | strasse   | plz | ort
-----+-----+-----+-----+-----+-----
 1001 | Moczon    | Thomas  | Strasse   | 90489 | Nürnberg
 1002 | Schoenfeld | Alexander | Musterstr. 11 | 90617 | Puschendorf
(2 Zeilen)
```

# PostgreSQL unter Debian Linux

---

## 5.4 Der COPY-Befehl

Der COPY-Befehl ermöglicht ein einfaches Ex- und Importieren von Datensätzen, allerdings kann dies nicht jeder Benutzer (erfolgreich getestet mit Benutzer „*postgres*“)

### 5.4.1 Ausgabe auf dem Bildschirm

Hier ein Beispiel, um die Datensätze der Tabelle „*Adressen*“ mit einem Komma getrennt auszugeben:

```
rdf=> COPY APerson TO STDOUT WITH DELIMITER ',';
1003;Mustermann;Max;Musterweg 11;12345;Musterstadt
```

### 5.4.2 Ausgabe in eine Datei (als Benutzer „*postgres*“)

Auch der Export in eine Datei ist möglich (Format wie unter 5.4.1):

```
rdf=# COPY APerson TO '/tmp/test.txt' WITH DELIMITER ',';
COPY
```

### 5.4.3 Import der Datensätze aus einer Datei

Editiert man nun die unter 5.4.2 erstellte Datei kann man diese zu den Datensätzen hinzufügen:

```
rdf=> COPY person FROM '/tmp/test.txt' WITH DELIMITER ',';
```

## 6. Quellen

<http://www.postgresql.org>

<http://www.postgresql.org/docs/7.4/static/>

[http://pgug.de/documents/Chemnitzer\\_Linuxtage/2006/pgsql-2006-03-04.php](http://pgug.de/documents/Chemnitzer_Linuxtage/2006/pgsql-2006-03-04.php)