Postgresql

Michael Dienert

10. Dezember 2007

Inhaltsverzeichnis

1 Übersetzen und Installieren

1.1 Installationsort der Programme

2 Einrichten einer Testdatenbank

Die im Folgenden beschriebenen Schritte zur Einrichtung einer Datenbank sind immer dann alle durchzuführen, wenn Postgresql selbst kompiliert und installiert wird.

Wird Postgresql mit einem Paketmanager (z.b. **apt**, Advanced Package Tool von Debian/Ubuntu) installiert, wird der Datenbankserver vollständig konfiguriert. D.h. das Datenbankcluster wird automatisch angelegt und der Datenbankserver (postmaster) wird automatisch gestartet. Es werden auch Start- und Stopp-Skripte für das automatische Starten und Beenden beim Betriebssystemstart und -stop erzeugt.

Ausserdem wird ein Benutzer `postgres` angelegt. Diesem Benutzer gehört das Datenbankcluster und damit ist er zunächst der einzige berechtigte Datenbankbenutzer.

Installiert man postgresql mit `fink` auf einem Mac, muss man das Datenbankcluster selbst initialisieren und auch den Server `postmaster` manuell starten und stoppen. Der Beutzer `postgres` wird bei der Installation angelegt.

Je nachdem wie **postgresql** installiert wurde, sind nun die folgenden Konfigurationsschritte auszuführen.

2.1 Das Datenbank-Cluster

Postgres wird zwar oft als *Datenbank* bezeichnet, es ist aber in Wirklichkeit ein *relationales Datenbank Management System* (RDBMS). Die eigentliche *Datenbank* besteht dabei aus Tabellen, die Postgres in speziellen Datenstrukturen im Speicher und auf der Festplatte ablegt.

Dabei wird der Speicherort auf der Festplatte als *Datenbank-Cluster* bezeichnet. Das ist ein spezielles Verzeichnis.

Dieses Datenbank-Cluster muss vor dem Erstellen der ersten Datenbank *initialisiert* werden. Dabei ist zu beachten, dass der Datenbank-Server auch die nötigen Rechte besitzt, um in dieses Verzeichnis schreiben zu können.

Das Rechteproblem löst sich aber ganz von selbst, wenn das Cluster-Verzeichnis dem selben Benutzer gehört, der anschliessend auch den Postgres-Server startet. Im Regelfall ist das der bei der Installation angelegte Benutzer `postgres`.

Als Benutzer wird das Cluster im Heimatverzeichnis von `postgres` [1] (oder einem beliebigen anderen Benutzer) so angelegt (nachdem man das Verzeichnis data/ erzeugt hat):

```
initdb -D ~/data
```

Das Verzeichnis `~/data` darf man selbstverständlich auch anders nennen.

2.2 Den Datenbank-Server starten und stoppen

2.2.1 Serverstart

Ist das Cluster angelegt, kann der Server gestartet werden:

```
postmaster -D ~/data -i &
```

Die Option `-D` gibt das Cluster an, `-i` startet den Server so, dass er auch auf TCP/IP Anfragen von entfernten Rechnern antwortet (i = Internet Domain).

Wichtig ist dass der Benutzer, der den Server startet auch Besitzer des Clusters ist.

Steht das Cluster im Heimatverzeichnis eines Benutzers, muss auch dieser den Server betreiben.

2.2.2 Server stoppen

Möchte man die Arbeit mit dem Postmaster beenden, kann man den Prozess wie folgt stoppen:

```
pg_ctl stop -D ~/data
```

[1] Als Ort für das Heimatverzeichnis von postgres bietet sich z.b. /var/lib/postgresql/ an.

2.3 Weitere Benutzer anlegen

Mit

```
createuser benutzername
```

lassen sich weitere Benutzer registrieren.

Postgres erlaubt zunächst (in der Default-Konfiguration) nur Benutzer, die auch in der Benutzerdatenbank (z.b. /etc/passwd) des verwendeten Betriebssystems stehen. D.h. die Authentifizierung erfolgt zunächst auf Ebene des Betriebssystems (login).

Um sich selbst als berechtigten Datenbankbenutzer anzulegen, meldet man sich also als Unix-Benutzer `postgres` auf einer Konsole an. Anschliessend kann man für sein eigenes Benutzerkonto auch einen entsprechenden postgresql-Benutzer anlegen.

2.4 Eine Datenbank erzeugen

Sobald der Server läuft, kann man als berechtigter Benutzer eine Datenbank erzeugen:

```
createdb emaille
```

Dazu muss man sich nicht am Server anmmelden: derjenige, dem das Cluster und der Serverprozess (postmaster) gehört hat automatisch **alle Rechte**.

2.4.1 Das Datenbankcluster sichern

Das Datenbankcluster ist z.b. das Verzeichnis: `/var/lib/postgresql/8.1/`. Bei Rechnern, die mit rembo/myshn bei jedem Start neu synchronisiert swerden, werden also Änderungen im Dateisystem und damit auch im Datenbankcluster bei jedem Neustart gelöscht.

Nun kann der Benutzer postgres im Stammverzeichnis von `data/` mit folgendem Kommando ein Archiv des Clusters erzeugen:

```
tar -czf data.tgz data/
```

Dieses Archiv kann sich nun ein normaler Benutzer in sein Heimatverzeichnis kopieren.

Nach dem nächsten Recnerstart muss der Benutzer `postgres` das Archiv wieder ins Stammverzeichnis von `data/` kopieren und wie folgt extrahieren:

```
tar -xzf data.tgz
```

Sehr wichtig ist, dass dieser Befehl im richtigen Verzeichnis ausgeführt wird!

2.5 Der Datenbank-Client

Sobald der Server läuft und eine Datenbank erzeugt ist, kann man sich mit einem Client-Programm mit dem Server verbinden:

```
psql emaille
```

Bei der Arbeit mit dem Client ist folgendes ganz wichtig:

Im Gegensatz zur z.B. Bash, hat die Eingabetaste (Return) hier die selbe Bedeutung wie in einem Editor: sie fügt einfach nur einen Zeilenumbruch ein. Erst wenn man ein SQL-Statement mit einem **Semikolon** abschliesst und dann Return betätigt, wir *alles seit dem letzten Semikolon* zum Server gesandt.

Mit `\h` erhält man eine Hilfe zu psql.

Möchte man SQL-Befehle komfortabel editieren, gelangt man mit `\e` in einen Editor. Der Standardeditor ist hier der unvermeidliche **vi**. Möchte man einen anderen Editor, muss man vor dem Start von psql mit (z.B.)

```
export EDITOR=geany
```

die Umgebungsvariable EDITOR mit seinem Wunscheditor setzen.

Egal welchen Editor man nimmt, der editierte SQL-Befehl wird nach dem Speichern und Beenden des Editors von psql zum Datenbankserver geschickt. Dabei arbeitet der Editor im /tmp-Verzeichnis.

Achtung: Auch im Editor muss man ein SQL-Statement mit einem Semikolon abschliessen!

Möchte man die Struktur einer Tabelle oder der Datenbank anzeigen lassen, hilft `\d` (Display) weiter.

3 Aufgaben

- Entwirf eine Mini-Datenbank, die die Emaille-Adressen von Personen speichert. Dabei soll eine Person auch mehrere Emaille-Adressen besitzen dürfen.
- Erzeuge die beiden Tabellen der Datenbank mit dem Postgresql-Client psql. Informiere Dich vorher, wie mit dem Befehl CREATE TABLE auch Primär- und Fremdschlüssel angelegt werden.
- Trage einige Testdatensätze in die Tabellen ein (INSERT INTO ...)
- Entwirf eine Abfrage, die eine Adressenliste ausgibt.

4 Musterlösung

Die Datenbank enthält folgende Tabellen:

$$person(\underline{pid}, nachname, vorname)$$

$$emaille(\underline{emaille}, \underbrace{\underline{pid}}_{Fremdschluessel})$$

Die beiden Tabellen können z.B. mit folgenden SQL-Anweisungen erzeugt werden, die man am besten in einer Textdatei speichert und anschliessend mit dem psql-Client einlesen kann:

```
/*zuerst loeschen wir eine evtl. vorhandene Datenbank 'emaille'*/
DROP DATABASE emaille;

/*und legen sie neu an*/
CREATE DATABASE emaille;

/*jetzt koennen wir uns mit der db 'emaille' verbinden*/
\connect emaille

/*in der nun leeren db werden tabellen erzeugt*/
CREATE TABLE person(
   pid int not null,
   nachname text,
   vorname text,
   PRIMARY KEY (pid)
);

CREATE TABLE emaille(
   emaille text,
   pid int,
   PRIMARY KEY (pid, emaille),
   FOREIGN KEY (pid) REFERENCES person
      ON UPDATE CASCADE
      ON DELETE CASCADE
);

/*zum schluss geben wir ein paar testdatensaetze ein*/
INSERT INTO person VALUES (1, 'neumann', 'alfred');
INSERT INTO emaille VALUES ('alfred@mad.org', 1);

/*und formulieren eine abfrage ueber die beiden relationen*/
SELECT nachname, vorname, emaille as mehladresse
   FROM person INNER JOIN emaille ON person.pid = emaille.pid;
```

Sind diese SQL-Anweisungen in der Datei emaille.sql gespeichert, kann man die Datei mit folgender Kommandozeile als Ganzes von psql ausführen lassen:

```
psql -f emaille.sql postgres
```

Die Datenbank `postgres` existiert immer und wird benötigt, um sich mit dem RDBMS verbinden zu können.