

PostgreSQL-Installation

Thomas Karsten, Christoph Dalitz
Hochschule Niederrhein

Version 1.9, 09. Oktober 2006

Inhaltsverzeichnis

1 Überblick und Bezugsquellen	1
1.1 Benötigte Software	1
1.2 Bezugsquellen	2
1.3 Installationsüberblick	2
2 Installation des Datenbank-Servers PostgreSQL	3
2.1 Installation unter Linux	3
2.1.1 Binärpakete	3
2.1.2 Quellpakete	4
2.2 Installation unter MacOS X	4
2.3 Installation unter Windows	5
2.3.1 Installation der nativen Win32-Version	5
2.3.2 Installation unter Cygwin	5
2.3.3 Umlaute unter Windows	7
3 Einrichten des Datenbank-Servers	7
3.1 Anlegen des Datenbank-Administrators (entfällt unter Windows)	7
3.1.1 Account anlegen unter Linux	7
3.1.2 Account anlegen unter MacOS X	8
3.2 Einrichten des Datenbank-Administrators	8
3.3 Einen Datenbank-Cluster erstellen	9
3.4 Starten des Datenbank-Servers	9
3.4.1 Manuelles Starten	9
3.4.2 Automatisches Starten (Linux)	10
3.4.3 Automatisches Starten (MacOS X)	10
3.4.4 Testen der Installation	11
3.5 Erstellen einer Datenbank und eines Benutzers	12
3.6 Zugriffssteuerung	12
3.7 Zugang zum Datenbank-Server	13

4 Die Perl-Module DBD-Pg und DBI	13
5 Die C++-Schnittstelle libpqxx	14
A Anhang	i
A.1 Das Programm tar	i
A.2 Das Programm checkinstall	i

Version und Änderungen

Version	Datum	Autor	Änderung
1.0	2003.05.15	Karsten	Ersterstellung
1.1	2004.01.08	Dalitz	Ergänzung Umlaute unter Windows
1.2	2004.07.14	Dalitz	Ergänzung MacOS X. URL für Überblick Windows-Portierungen
1.3	2004.07.15	Dalitz	Ergänzung Systemstart unter MacOS X
1.4	2004.10.26	Dalitz	Ergänzung Lokalisierung und Readline
1.5	2004.12.07	Dalitz	Korrektur Startscript für MacOS X
1.6	2005.02.28	Dalitz	Korrektur Lizenz und Tippfehler "PASSWORD"
1.7	2005.10.04	Dalitz	Installation nativer Win32 Version ergänzt
1.8	2005.11.22	Dalitz	Encoding bei "create database" ergänzt, Korrektur Script für automatischen Start beim Linux-Systemstart
1.9	2006.10.09	Dalitz	Korrektur einzelner Befehle

1 Überblick und Bezugsquellen

PostgreSQL unterliegt der „BSD License“ und ist deshalb kostenlos.

Bei den meisten Linux-Distributionen (z.B. SuSE, RedHat) wird PostgreSQL mitgeliefert, ist also auf den beigelegten CDs / DVDs enthalten. In diesem Fall ist PostgreSQL bereits kompiliert und muss nur noch mit dem entsprechenden Paketmanager (z.B. YaST für SuSE, rpm für RedHat) in das System installiert werden.

1.1 Benötigte Software

Wir benötigen bei dieser Installation drei Pakete, die für PostgreSQL wichtig und für die Ausführung der Datenbanken-Praktika notwendig sind:

- PostgreSQL
Datenbank-Management-System
- DBI
abstrakte Datenbankschnittstelle für die Programmiersprache Perl
- DBD-Pg
PostgreSQL-Treiber für die Perl-DBI-Schnittstelle

1.2 Bezugsquellen

Wenn man keine aktuelle Version von PostgreSQL besitzt, kann man die neueste Version über das Internet beziehen. Dort gibt es sowohl die vorkompilierten Binär- (Dateiname: `paketname.rpm`) als auch die Quellpakete (Dateiname: `paketname.tar.gz`).

Die Binärpakete werden wie in Punkt 2.1.1 beschrieben mit `rpm` installiert. Auch hier müssen alle eventuellen Abhängigkeiten zwischen Paketen vor der Installation aufgelöst werden.

Die Quellpakete werden mit dem Befehl `tar` entpackt. Weitere Beschreibungen dazu befinden sich im Punkt 2.1.2.

- PostgreSQL: <http://www.postgresql.org>
- Perl DBD-Pg: <http://search.cpan.org>
- Perl DBI: <http://search.cpan.org>

Wer den Datenbank-Server auch mit C++ ansprechen möchte, benötigt zusätzlich noch das `libpqxx`-Paket, das die C++-Schnittstelle für PostgreSQL darstellt. Dieses Paket wurde mit der Version 7.3 ausgelagert, ist jedoch optional und für die Datenbanken-Veranstaltung nicht erforderlich.

- `libpqxx`: <http://gborg.postgresql.org/project/libpqxx>

Diese Installationsanweisung beschreibt hauptsächlich das Aufspielen der Quellpakete, angefangen bei der Kompilation der Quellen bis zur Konfiguration des Programmes.

Sowohl die Binärpakete als auch die gepackten tar-Archive können z.B. im Verzeichnis `/usr/local/src` abgespeichert werden.

1.3 Installationsüberblick

Folgende Schritte werden im weiteren Textverlauf behandelt, um die gesamten benötigten Pakete zu installieren:

- Installation des Datenbank-Servers PostgreSQL (Punkt 2)
- Anlegen des Datenbank-Administrators `postgres` im System (Punkt 3.1)
- Erstellen einer ersten Datenbank mit `initdb` (Punkt 3.3)
- Anlegen eines Benutzers in der Datenbank (Punkt 3.5)
- Installation der Perl-Module DBI und DBD-Pg (Punkt 4)
- Installation der C++-Schnittstelle `libpqxx` (Punkt 5, für die Datenbankveranstaltung nicht notwendig)

2 Installation des Datenbank-Servers PostgreSQL

2.1 Installation unter Linux

Es gibt zwei Alternativen für die Installation der Pakete.

- Zum Einen kann man die Programme aus den Binärpaketen installieren (Punkt 2.1.1). Diese Methode ist die einfachere, jedoch ist die Installation gegebenenfalls vom System abhängig. D.h. es müssen eventuell weitere Pakete hinzu installiert werden.
- Zum Anderen können die Programme aus den Quelldateien erst kompiliert und dann installiert werden (Punkt 2.1.2). Diese Methode ist die kompliziertere der beiden. Hier sind die Programme aber nicht so abhängig von anderen Paketen, weil sie entsprechend der aktuellen Bibliotheksversionen auf dem lokalen Rechner kompiliert werden.

Anmerkung: Im Folgenden werden die entsprechenden Kommandozeilenbefehle zur Installation mit angegeben. Das Zeichen vor der Kommandozeile (\$ oder #) stellt den Prompt der Shell dar. Dabei steht \$ für den Prompt des normalen Benutzers und # für den Prompt von `root`.

2.1.1 Binärpakete

RedHat 8.0 Die Linux-Distribution von RedHat Version 8.0 liefert alle wichtigen Pakete zur Installation von PostgreSQL mit. Auf der 2. CD im Verzeichnis RedHat/RPMS sind das folgende Dateien:

- PostgreSQL Version 7.2.2: `postgresql-7.2.2-1.i386.rpm`
- DBD-Pg Version 1.13: `perl-DBD-pg-1.13-5.i386.rpm`
- DBI Version 1.30: `perl-DBI-1.30-1.i386.rpm`

Das jeweilige Paket wird als `root` durch den Befehl

```
# rpm -i dateiname
```

installiert.

Bei den Binärpaketen müssen eventuelle Abhängigkeiten aufgelöst werden. Sollten bei der Ausführung der Installation mit `rpm` Fehlermeldungen wegen nicht installierter Pakete auftreten, so sollten diese nachinstalliert werden.

SuSE 8.0 Bei der Linux-Distribution von SuSE lässt sich PostgreSQL mit dem Installationstool YaST sehr einfach installieren. Dort befindet sich PostgreSQL Version 7.2 in der Serie `ap` als Pakete `postgresql`, `postgresql-server` und `postgresql-libs`. Diese Pakete müssen ausgewählt und können dann von YaST installiert werden. Die Perl-Schnittstellen sind jedoch nicht enthalten und sind wie in Punkt 1.2 und Punkt 4 zusätzlich in das System einzubinden.

Debian Die Debian-Distribution liefert einen Paketmanager mit, der Pakete gleichzeitig installieren und zusätzlich auch Abhängigkeiten auflösen kann.

Die von uns benötigten Pakete werden mit dem folgenden Befehl installiert:

```
# apt-get install postgresql-client \  
                    postgresql-dev \  
                    postgresql-doc \  
                    libdbi-perl \  
                    libdbd-pg-perl
```

2.1.2 Quellpakete

Bevor Sie PostgreSQL kompilieren können, müssen natürlich die Entwicklungswerkzeuge (*gcc*, *make*, ...) installiert sein, was bei den meisten Distributionen standardmäßig der Fall ist. Zusätzlich ist die Installation der *Readline* Bibliothek dringend zu empfehlen, weil sonst in *psql* kein Kommandozeilen-Editieren möglich ist. Dazu ist die Installation der *readline*-Runtime Bibliotheken nicht ausreichend, es müssen auch die *readline*-Header Files installiert sein (prüfen Sie, ob die Datei */usr/lib/readline/readline.h* vorhanden ist).

Für die Installation von PostgreSQL müssen wir erst einmal das tar-Archiv entpacken. Dies geschieht wie in Abschnitt A.1 angegeben. Damit wird das Verzeichnis */usr/local/src/postgresql-7.3.2* erstellt und alle Quelldateien werden dort abgespeichert.

Anmerkung: Mit den Quellen befinden sich gleichzeitig auch Hilfedateien im jeweiligen Quellverzeichnis. Diese haben meist den Namen *README* oder *INSTALL* und sollten vor dem Kompilierungsvorgang und der Installation gelesen werden.

Als Nächstes wechseln wir in dieses Verzeichnis und führen dort den Befehl *configure* aus, der das Makefile erstellt und entsprechend zusätzlicher Angaben auch konfigurieren kann.

```
$ ./configure
```

Anmerkung: PostgreSQL-Versionen unter 7.3 haben die Lokalisierungsmöglichkeit defaultmäßig ausgeschaltet. Um sie für diese älteren PostgreSQL-Versionen zu aktivieren, verwenden Sie folgende *configure*-Option:

```
$ ./configure --enable-locale
```

Jetzt müssen die Quelldateien kompiliert werden (dies kann als normaler Benutzer geschehen) um dann in die entsprechenden Verzeichnisse installiert werden zu können (als *root*). Dies geschieht mit folgenden Befehlen:

```
$ make
$ su
Password:
# make install
```

Anmerkung: Anstatt den Befehl *make install* zum Installieren zu benutzen, kann auch das Programm *checkinstall* (siehe Abschnitt A.2) verwendet werden. Dadurch ist es möglich, die Installation wieder rückgängig zu machen (Deinstallation) und den Systemzustand vor der Installation wieder herzustellen.

Damit ist die Installation des Datenbank-Management-Systems PostgreSQL abgeschlossen. Standardmäßig befinden sich alle installierten Dateien im Verzeichnis */usr/local/pgsql*.

Die ausführbaren Dateien sind im Unterverzeichnis *bin/*, die Header-Dateien im Unterverzeichnis *include/* abgelegt. Im Pfad */usr/local/pgsql/doc/html* findet sich die PostgreSQL-HTML-Dokumentation.

Abschließend kann das Verzeichnis */usr/local/src/postgresql-7.3.2* und sein gesamter Inhalt gelöscht werden.

2.2 Installation unter MacOS X

Unter MacOS X empfiehlt sich eine Installation aus den Sourcen. Dazu muss allerdings zuvor folgende frei verfügbare Software installiert sein:

- *Apple Developer Tools* ("Xcode-Tools"), die auf den MacOS X CD's dabei sind bzw. bei einem vorinstallierten System aus dem Verzeichnis */Programme/Installers/Xcode Tools* nachinstallierbar sind. Wer sie nicht findet, kann Sie auch kostenlos von der Apple Homepage beziehen.

- *readline* Bibliothek, die sich am einfachsten über *Fink* (<http://fink.sourceforge.net/>) installieren lässt, z.B. von der Kommandozeile mit `apt-get install readline` oder über den grafischen *FinkCommander*.

Das weitere Vorgehen ist wie im Abschnitt 2.1.2 beschrieben, allerdings müssen beim `configure`-Schritt die folgenden Optionen angegeben werden, damit die *readline*-Bibliothek im *Fink*-Verzeichnisbaum (unter `/sw`) gefunden wird. Die Befehlsfolge zum Kompilieren und Installieren ist also

```
./configure --with-includes=/sw/include --with-libraries=/sw/lib
make
sudo make install
```

2.3 Installation unter Windows

Um den PostgreSQL-Server unter Windows zu betreiben gibt es zwei Optionen:

- a) Verwendung der nativen Win32-Version (erst ab PostgreSQL 8.0)
- b) Verwendung der Unix-Emulation *Cygwin* (alle PostgreSQL-Versionen)

Methode a) ist einfacher und für einen Produktivbetrieb vorzuziehen. Zu Testzwecken kann die Methode b) aber auch sinnvoll sein, wenn der Produktivbetrieb später auf einem Unix-System erfolgen soll. Insbesondere haben steht mit Methode b) auch unter Windows eine Umgebung zur Verfügung, die der Linux-Umgebung im Praktikum sehr ähnlich ist.

Um Datenbankprogramme mit der PostgreSQL Bibliothek *libpq* zu schreiben, können Sie bei Methode a) die Win32-Portierung des `gcc` namens *MingW32* verwenden (kostenlos erhältlich von <http://www.mingw.org/>). Bei Methode b) können Sie den in *Cygwin* enthaltenen Compiler `gcc` verwenden. Beide Compiler können zusammen mit der IDE *Dev-C++* verwendet werden (kostenlos erhältlich von <http://www.bloodshed.net/devcpp.html>), die standardmäßig bereits den *MingW32* enthält.

2.3.1 Installation der nativen Win32-Version

Seit PostgreSQL Version 8.0 gibt es eine native Portierung auf Windows, die nicht die *Cygwin*-Umgebung benötigt. Der von der Postgres-Homepage <http://www.postgresql.org/> erhältliche Installer bietet optional an, neben der Softwareinstallation auch folgende Schritte durchzuführen:

- Anlegen eines Betriebssystem-Accounts, unter dem der Serverprozess läuft
- Einrichten des Serverprozesses als Windows-„Dienst“
- Anlegen eines Datenbank-Clusters mit einem zugehörigen DBA-Account (dies ist kein Betriebssystem-Account, sondern ein Datenbank-Account)

Wenn Sie diese Schritte vom Installer durchführen lassen, können Sie die entsprechenden Kapitel in dieser Installationsanleitung überspringen und mit Abschnitt 3.5 fortfahren.

2.3.2 Installation unter Cygwin

Der Datenbank-Server PostgreSQL ist in der *Cygwin*-Distribution bereits enthalten und kann mit Hilfe des *Setup*-Programmes installiert werden. *Cygwin* kann kostenlos unter folgenden Adressen heruntergeladen werden:

- <http://www.cygwin.com>
- <http://sources.redhat.com/cygwin>

Dort wird die Setup-Datei heruntergeladen, die dann alle weiteren zur Installation notwendigen Schritte durchgeht.

Nach dem Start der `setup.exe` kann man die Distribution vom Internet installieren („Install from Internet“). Das darauf folgende Fenster zeigt alle Mirrors an, wo Cygwin heruntergeladen werden kann. Nachdem ein Mirror ausgewählt wurde, werden alle in dieser Version verfügbaren Pakete dargestellt. Für die Durchführung der Praktika sind folgende Pakete notwendig:

Section	Pakete
Admin	cygrunsrv
Archive	unzip, zip
Base	alle Pakete
Database	postgresql
Devel	binutils, ctags, gcc, gcc2, gdb, make, mktemp
Doc	cygwin-doc,man,texinfo
Interpreters	gawk,perl
Libs	w32api,libz2,zlib
Shells	ash,bash,sh-utils
Text	groff,less,texinfo
Utils	bzip2,cygutils,diff,file,time

Anmerkung: Pakete werden an- oder abgewählt, indem auf die runden Symbole geklickt wird. Nach dieser Auswahl werden die Pakete heruntergeladen und installiert (als `root`-Verzeichnis für die Cygwin-Distribution benutzt man am besten das Verzeichnis `C:\Cygwin`).

Eine detaillierte Installationsanleitung für die Cygwin-Distribution gibt es im Internet unter:
<http://www.pgsql.info/pg-win.php>

Zusätzlich zu Cygwin wird der IPC-Manager (Inter-Process-Communication-Manager) `CygIPC` von PostgreSQL benötigt. Dieser stellt bestimmte UNIX-Systemaufrufe („Systemcalls“) zur Verfügung, die weder in dem Betriebssystem Windows noch in der aktuellen Cygwin-Distribution (Version 1.13-2) vorhanden sind, von dem Datenbank-Server allerdings genutzt werden. (Jedoch ist es vorgesehen, diese Systemaufrufe einer späteren Cygwin-Distribution hinzuzufügen.)

`CygIPC` ist ebenfalls kostenlos und kann unter folgender Adresse heruntergeladen werden:

<http://www.neuro.gatech.edu/users/cwilson/cygutils/cygic/index.html>

Die Archivdatei `cygipc-x.xx-x.tar.bz2` (wobei hiermit die jeweils aktuellste Version gemeint ist) kann z.B. in das Verzeichnis `$HOME` gespeichert werden.

Nach dem Download wird der IPC-Manager über das Installationsprogramm von Cygwin installiert. Dazu startet man die `setup.exe` und wählt den Punkt „Install from Local Directory“. Nach der Angabe des Verzeichnisses, in dem die `cygipc`-Datei liegt, wird diese installiert.

Das Installieren über das Cygwin-Installationsprogramm hat den Vorteil, dass das Paket zu einem späteren Zeitpunkt wieder deinstalliert werden kann.

Anmerkung: Alternativ kann die Datei `cygipc-x.xx-x.tar.bz2` auch manuell installiert werden. Dazu muss die Datei im Verzeichnis `$CYGWINHOME` gespeichert werden. Jetzt kann der IPC-Manager mit dem Befehl `bunzip2 -c /cygipc-x.xx-x.tar.bz2 | tar xvf -` installiert werden. Diese Methode lässt eine automatische Deinstallation jedoch nicht zu und wird an dieser Stelle deshalb auch nicht empfohlen.

Der IPC-Manager `ipc-daemon` wurde jetzt in das Verzeichnis `/usr/local/bin`, die dazu gehörige Dokumentation in das Verzeichnis `/usr/local/doc/cygipc-x.xx` installiert.

Vor der Initialisierung einer PostgreSQL-Datenbank (Punkt 3.3) und vor jedem Starten des Datenbank-Servers (Punkt 3.4) muss der IPC-Manager `ipc-daemon.exe` gestartet werden. Dies geschieht mit dem Kommando

```
$ ipc-daemon &
```

2.3.3 Umlaute unter Windows

Der SQL-Interpreter `psql` ist eine sogenannte “Konsolen-Anwendung”. Leider verwenden die Windows-“Konsole” ein anderes Encoding als der Rest des Windows-Systems, wie sich leicht durch die folgenden Befehle in einer DOS-Shell überprüfen lässt:

```
echo äöü > bla.txt  
notepad bla.txt
```

Es ist aber möglich, mit dem Befehl `chcp` (change codepage) das Konsolen-Encoding auf den Wert des normalen Windows-Encoding zu ändern. Allerdings muss dann auch der Konsolenfont auf *Lucida* geändert werden, weil der Default Konsolenfont nicht mit dem Windows-Encoding zurechtkommt. Um also Umlaute and der “Konsole” problemlos nutzen zu können, sind zwei Schritte erforderlich:

- Setzen Sie die Codepage mit dem Befehl `cmd.exe /c chcp 1252` (1252 ist die deutsche Windows-Codepage). Damit sich dies auf alle Cygwin-Fenster auswirkt, ergänzen Sie diesen Befehl am besten in `/etc/profile`.
- Setzen Sie den Konsolenfont auf *Lucida Console* (rechter Mausklick auf die Titelleiste des Konsolen-Fensters).

3 Einrichten des Datenbank-Servers

Das Starten des Datenbank-Servers und das Einrichten einer ersten Datenbank wird von keinem Installationstool unterstützt und muss deshalb vom Benutzer selbst vorgenommen werden. Jedoch ist in der Linux-Distribution von SuSE schon einiges vorkonfiguriert. Für die Einrichtung der Datenbank muss ein Datenbank-Administrator im System bestimmt werden. Unter Linux ist dies der Benutzer `postgres`. Unter Windows ist dies der normale Benutzer. Im Folgenden wird, wenn es um beide Systeme geht, nur noch vom Datenbank-Administrator gesprochen.

3.1 Anlegen des Datenbank-Administrators (entfällt unter Windows)

Bevor wir mit dem Einrichten der Datenbank starten, müssen wir erst einen Eigentümer der PostgreSQL-Datenbanken anlegen. Üblicherweise ist dies der Benutzer mit dem Namen `postgres` in der Gruppe `postgres` oder `users`.

3.1.1 Account anlegen unter Linux

Anmerkung: Bei vielen Linux-Distribution (z.B. Debian, SuSE) wird bereits bei der Installation der Binärpakete automatisch ein Benutzer mit dem Namen `postgres` erzeugt. Des Weiteren ist hier auch schon sein Home-Verzeichnis `/var/lib/pgsql` vorhanden. Nun ist lediglich noch das Passwort durch den `root`-Benutzer anzupassen (siehe unten).

Das Anlegen des Benutzers `postgres` geschieht als `root` mit folgendem Befehl:

```
# useradd -d /home/postgres -p postgres postgres
```

Der Benutzer `postgres` wurde erstellt und hat als Login-Passwort `postgres`. Aus Sicherheitsgründen sollte dies jedoch nachträglich geändert werden:

```
# passwd postgres
```

Als Home-Verzeichnis wurde `/home/postgres` ausgewählt, das der `root`-Benutzer noch erstellen und die Eigentumsrechte dieses Verzeichnis an den Benutzer `postgres` übergeben muss:

```
# mkdir /home/postgres
# chown postgres /home/postgres
```

Alle weiteren Schritte werden nicht als `root` sondern als `postgres` ausgeführt:

```
# su - postgres
```

3.1.2 Account anlegen unter MacOS X

Unter MacOS X können Sie den `postgres` Account wie jeden normalen Account über *Systemeinstellungen/Benutzer* anlegen. Dann wird der Benutzer allerdings auch in der Anmeldemaske angeboten und in sein Homeverzeichnis `/Users/postgres` wird allerlei Kram kopiert.

Wenn Sie das nicht wollen, können Sie den Benutzer auch über den grafischen *Netinfo Manager* (in *Programme/Dienstprogramme*) oder von der Kommandozeile mit `niutil` anlegen (dabei `sudo` nicht vergessen). Bei Verwendung des Netinfo Manager kann man den `www`-Account als Muster duplizieren. Wichtig ist dabei, dass man eine neue User-ID vergibt (z.B. 401) und als Login-Shell `/bin/bash` statt `/bin/false` einträgt, denn sonst kann man mit diesem Account nicht arbeiten! Anschliessend ist das Passwort zu setzen mit

```
sudo passwd postgres
```

Ferner muss als `root` bzw. mit `sudo` das im Netinfo Manager gewählte Homeverzeichnis angelegt werden und mit `chown` dem `postgres`-User zugewiesen werden.

Die weiteren Schritte erfolgen als User `postgres`, der mit `su - postgres` angenommen werden kann.

3.2 Einrichten des Datenbank-Administrators

Als ersten Schritt fügt der Datenbank-Administrator (`postgres` unter Linux und MacOS X) seiner Login-Datei `.profile`, die sich im Home-Verzeichnis befindet, folgendes hinzu:

```
# damit PostgreSQL Programme gefunden werden
PATH=$PATH:/usr/local/pgsql/bin
export PATH
# damit Manpages gefunden werden
MANPATH=$MANPATH:/usr/local/pgsql/man
export MANPATH
# Variablen für PostgreSQL-Server
PGUSER=$USER
PGDATA=$HOME/data
export PGUSER PGDATA
```

Abschließend wird die Datei `.profile` in der aktuellen Shell ausgeführt, um die neu hinzugefügten Einstellungen zu übernehmen. Ob dies erfolgreich geschehen ist, kann mit dem darauf folgenden Befehl überprüft werden.

```
$ . .profile
$ echo $PGDATA
```

3.3 Einen Datenbank-Cluster erstellen

Um die Datenbank einzurichten benötigen wir ein Datenbankverzeichnis. In unserem Linux-Beispiel ist das `/home/postgres/data`, das wir jedoch zuvor mit dem Benutzer `postgres` (Linux) bzw. mit dem normalen Benutzer unter Windows erstellen müssen:

```
$ mkdir $PGDATA          # entfällt bei SuSE
$ initdb --locale=de_DE -D $PGDATA
```

Die Lokalisierungseinstellung `--locale=de_DE` ist erforderlich für eine korrekte Sortierung und Konversion (*upper, lower*) von Umlauten und kann nur auf der Serverseite zum Zeitpunkt von `initdb` eingestellt werden! Siehe dazu den Abschnitt *Localization* im PostgreSQL Administrator's Guide.

Anmerkung: In der Cygwin-Distribution muss vor dem Start des Datenbank-Servers der `ipc-daemon` aktiv sein (Punkt 2.3.2).

Nach erfolgreicher Initialisierung des Datenbank-Clusters erscheinen folgende Zeilen auf dem Bildschirm:

```
Success. You can now start the database server using:
    /usr/local/pgsql/bin/postmaster -D /home/postgres/data
or
    /usr/local/pgsql/bin/pg_ctl -D /home/postgres/data -l logfile start
```

3.4 Starten des Datenbank-Servers

Es gibt zwei verschiedene Methoden, um den Datenbank-Server zu starten: entweder manuell als Datenbank-Administrator (ohne `root`-Rechte unter Linux) oder automatisch beim Systemstart.

Das manuelle Starten ist einfacher. Jedoch muss dies nach jedem Neustart des Systems geschehen und kann auf die Dauer lästig werden.

Das automatische Starten des Datenbank-Servers PostgreSQL ist bequemer, allerdings ist die Konfiguration dafür etwas komplizierter.

3.4.1 Manuelles Starten

Bei dieser Methode startet der Datenbank-Administrator den Datenbank-Server manuell mit Hilfe des oben angegebenen Befehls:

```
$ pg_ctl -D $PGDATA -l logfile start
```

Anmerkung: In der Cygwin-Distribution muss vor dem Start des Datenbank-Servers der `ipc-daemon` aktiv sein (Punkt 2.3.2).

3.4.2 Automatisches Starten (Linux)

Die Konfiguration des Systems zum automatischen Starten des Datenbank-Servers PostgreSQL kann nur der `root`-Benutzer durchführen.

Um das System so zu konfigurieren, dass PostgreSQL beim Systemstart hochgefahren wird, schreiben wir uns ein kleines entsprechendes Script namens `postgresql` und kopieren es zu den anderen Systemscripten ins Verzeichnis `/etc/rc.d/init.d/`.

```
#!/bin/bash
# postgresQL - Script zum automatischen Starten

export PGDATA=~postgres/data
export PGSQL=/usr/local/pgsql
case "$1" in
  start)  echo Starting PostgreSQL
          su postgres -c "$PGSQL/bin/pg_ctl -D $PGDATA -l \
            ~postgres/logfile start"
          ;;
  stop)   echo Shutting down PostgreSQL
          su postgres -c "$PGSQL/bin/pg_ctl stop"
          ;;
  restart) $0 stop
           $0 start
           ;;
  reload) su postgres -c "$PGSQL/bin/pg_ctl -D ~postgres/data \
            -l ~postgres/logfile reload"
           ;;
  *)      echo "Usage: $0 {start | stop | restart | reload}"
           ;;
esac
```

Anmerkung: Bei dem Binärpaket der Linux-Distribution von SuSE existiert das entsprechende Script namens `postgresql`. Hier muss man nur noch den entsprechenden Link für den jeweiligen Runlevel setzen (siehe unten).

Andernfalls gibt es meistens ein Script namens `skeleton`, das mit den obigen Angaben vervollständigt werden kann. Dies hat auch den Vorteil, dass es an das Erscheinungsbild der Distribution angepasst ist.

Nachdem wir die Datei `postgresql` für `root` ausführbar gemacht haben, legen wir zwei Links im Verzeichnis des aktuellen Runlevels ab (in diesem Beispiel ist das der Runlevel 3). Wir wechseln also in das Verzeichnis `/etc/rc.d/rc3.d/` (RedHat).

```
# chmod 755 postgresQL
# cd /etc/rc3.d/
# ln -s ../init.d/postgresQL S50postgresQL
# ln -s ../init.d/postgresQL K50postgresQL
```

Anmerkung: Den aktuellen Runlevel zeigt das Kommando `runlevel` an.

Mit dieser Einstellung wird der Datenbank-Server beim Starten des Runlevel 3 automatisch hochgefahren.

3.4.3 Automatisches Starten (MacOS X)

Dienste, die unter MacOS X beim Systemstart gestartet werden sollen, müssen in `/Library/StartupItems` eingetragen werden. Als Muster kann man einen der Apple Systemdienste in `/System/Library/StartupItems`

nach `/Library/StartupItems` kopieren und nach seinen Wünschen anpassen, wobei die Kommandos zum Starten und Stoppen des vorherigen Abschnitts 3.4.2 verwendet werden können.

Anmerkung: Unter <http://www.entropy.ch/software/macosx/postgresql/> gibt es ein MacOS X Installer-Paket, das bereits ein entsprechendes StartupItem in `/Library` installiert. Wenn Sie dieses Pakte verwenden, so müssen Sie unbedingt anschließend das Startscript `/Library/StartupItems/PostreSQL/PostgreSQL` anpassen an die Pfade Ihrer Installation und die Option `"-i"` beim Starten entfernen, wenn Sie keinen Interzuzugriff auf Ihre Datenbank wünschen.

Das Startscript sollte dabei als erstes `/etc/rc.common` sourcen. Dieses Script lädt die Variablen aus `/etc/hostconfig`, so dass durch Ändern dieser Datei das Starten von PostgreSQL leicht an- bzw. ausgeschaltet werden kann. Das Script kann z.B. wie folgt aussehen:

```
#!/bin/sh

. /etc/rc.common

PGDATA=/Users/postgres/data
PGROOT=/usr/local/pgsql

StartService ()
{
    if [ "${POSTGRES:=-YES-}" = "-YES-" ]; then
        ConsoleMessage "Starting PostgreSQL database server"
        su postgres -c "$PGROOT/bin/pg_ctl start -D $PGDATA \
            -l $PGDATA/logfile"
    fi
}

StopService()
{
    ConsoleMessage "Stopping PostgreSQL database services"
    su postgres -c "$PGROOT/bin/pg_ctl stop -D $PGDATA"
}

RestartService ()
{
    StopService
    StartService
}

RunService "$1"
```

Damit die if-Zweige durchlaufen werden, muss zusätzlich in der Datei `/etc/hostconfig` der folgende Eintrag ergänzt werden:

```
POSTGRES=-YES-
```

Wenn Sie PostgreSQL irgendwann nicht mehr beim Systemstart starten wollen, dann ändern Sie diesen Eintrag einfach in `POSTGRES=-NO-`.

3.4.4 Testen der Installation

Um die Installation zu testen, starten wir den Datenbankenserver wie in 3.4 angegeben entweder manuell oder automatisch.

Den automatischen Startvorgang unter Linux testen wir als `root` mit dem oben erstellten Script wie folgt:

```
# /etc/init.d/postgresql start
```

Der automatische Startvorgang unter MacOS X kann nach dem Eintrag in *Library/StartupItems* als Benutzer `root` (eine Root-Shell bekommt man mit `sudo bash`) getestet werden mit dem Befehl:

```
# SystemStarter start PostgreSQL
```

Durch diesen Befehl wird der PostgreSQL-Server natürlich nur dann gestartet, wenn Sie tatsächlich in */etc/hostconfig* die Variable `POSTGRES` auf “-YES-” gesetzt haben.

Nach erfolgreichem Start von PostgreSQL erzeugen wir als Datenbank-Administrator (z.B. `postgres`) mit folgendem Kommando eine PostgreSQL-Sitzung:

```
$ psql template1
```

Ab hier sollte der Benutzer mit der Datenbank verbunden sein. Sollte es zu keiner Fehlermeldung kommen, wurde PostgreSQL erfolgreich installiert und eingerichtet!

3.5 Erstellen einer Datenbank und eines Benutzers

Nach den oben genannten Schritten sind wir in der Datenbank `template1` eingeloggt. Das ist eine Datenbank, die beim Initialisieren des Datenbank-Clusters mit `initdb` automatisch angelegt wurde. In diesem Datenbank-Cluster können wir weitere Datenbanken anlegen, die dann alle in `template1` definierten Objekt übernehmen. Deshalb sollte in `template1` nicht direkt gearbeitet werden.

Daher legen wir eine neue Datenbank namens `dbs` an. In dieser Datenbank erstellen wir einen neuen Benutzer namens `dummy`:

```
template1=# CREATE DATABASE dbs ENCODING 'latin1';
CREATE DATABASE
template1=# \c dbs
You are now connected to database dbs.
dbs=# CREATE USER dummy;
CREATE USER
dbs=#
```

Die Angabe des Encodings bewirkt, dass die Daten im Server in diesem Encoding gespeichert werden, und dass eine automatische Konvertierung zum Client möglich ist, z.B. in `psql` mit dem Befehl `\encoding utf8`. Auf neueren Systemen wird meistens Unicode verwendet, so dass Sie statt `latin1` als Encoding `utf8` angeben sollten.

3.6 Zugriffssteuerung

Die Zugriffsrechte der Datenbankbenutzer auf die Datenbank wird in der Datei `pg_hba.conf` im Verzeichnis `$PGDATA` festgelegt. Diese Datei kann der Datenbank-Administrator entsprechend modifizieren.

Als Default-Werte sind schon zwei Einträge vorgenommen worden. Diese erlauben sowohl jedem lokalen als auch jedem fremden Benutzer den Zugriff auf jede Datenbank von PostgreSQL.

Um diese Zugriffsrechte zumindest etwas einzuschränken, ist das Feld `METHOD` anzupassen. Die wichtigsten Attribute sind hier:

- `trust` für uneingeschränkten Zugriff
- `md5` für Zugriff mit verschlüsseltem Passwort
- `reject` für die Unterbindung eines Zugriffes

Anmerkung: Die Änderungen der Zugriffsrechte müssen der laufenden Datenbank manuell mitgeteilt werden. Dies geschieht mit dem Aufruf `pg_ctl reload`. Dadurch liest PostgreSQL seine Konfigurationsdateien neu ein. Ohne diesen manuellen Aufruf bleiben die Änderungen am System bis zum Neustart des Datenbank-Servers unwirksam.

Demzufolge können die Zugriffsmethoden für alle Benutzer mindestens auf `md5` gesetzt werden. Das erfordert bei jedem Login in die Datenbank die Eingabe eines Passwortes oder das Setzen der Umgebungsvariablen `$PGPASSWORD` mit dem entsprechenden Wert.

Um das Passwort zu setzen, muss entweder der Datenbank-Administrator oder der entsprechende Benutzer im Datenbanksystem folgenden Befehl anwenden (vor der Änderung der Konfiguration in `md5`):

```
db=# ALTER USER dummy PASSWORD 'dummy' ;
```

Jetzt hat der Benutzer `dummy` sein Passwort und muss es bei jedem Login neu eingeben (falls der Eintrag im `METHOD`-Feld nicht `trust` ist).

Der Zugriff über eine Netzwerkverbindung sollte für einen „Stand-Alone-Rechner“ generell unterbunden werden. Dazu darf die Variable `tcpip_socket` in der Datei `postgresql.conf` nicht gesetzt werden. Zur weiteren Vergabe der Zugriffsrechte sei hier auf die PostgreSQL-Dokumentation verwiesen.

3.7 Zugang zum Datenbank-Server

In der Datenbank sollte man zum einfachen Arbeiten nicht als Datenbank-Administrator angemeldet sein (aus Sicherheitsgründen). Deswegen ist es vorteilhaft, sich als regulärer Benutzer einzuloggen.

Der „normale Benutzer“ im System fügt seiner Pfadvariablen `PATH` am besten das Verzeichnis `/usr/local/pgsql/bin` hinzu. Somit kann ab hier Zugang zum Datenbank-Server über folgendes Kommando erreicht werden:

```
$ psql db dummy
```

Anmerkung: Natürlich muss vorher der Datenbank-Server entweder vom Benutzer `postgres` oder vom System gestartet worden sein (siehe Punkt 3.4). Wie eine Datenbank namens `db` eingerichtet und der Benutzer `dummy` erstellt wird, wird im Punkt 3.5 beschrieben.

4 Die Perl-Module DBD-Pg und DBI

DBI (DataBase Interface) ist die abstrakte Bibliothek, um mit der Programmiersprache Perl eine Verbindung zu einer Datenbank herstellen zu können.

Um tatsächlich PostgreSQL mit Perl ansprechen zu können, muss zusätzlich zur abstrakten Bibliothek der spezielle Datenbank-Treiber (DBD – DataBase Driver) für PostgreSQL in das Perl-Programm eingebunden werden.

Beide Treiber sind für das Praktikum nötig (Praktikumsaufgabe 3) und sollten aus diesem Grund zusätzlich zu PostgreSQL installiert werden.

Vor dem Start der Installation der beiden Perl-Module DBI und DBD-Pg müssen die `tar`-Archive entsprechend Abschnitt A.1 entpackt werden.

Es muss beachtet werden, dass zuerst die abstrakte Datenbankbibliothek DBI installiert werden muss. Weil der spezielle Datenbank-Treiber abhängig von dem Paket DBI ist, kann DBD-Pg nicht vor DBI aufgespielt werden.

Zur Installation von DBI wechseln wir zunächst in das Verzeichnis `/usr/local/src/DBI-1.35`.

In diesem Verzeichnis erstellen wir mit Hilfe des Perl-Scriptes `Makefile.PL` ein Makefile, um danach die Quellen kompilieren zu können. Als `root` wird das Paket DBI schließlich in das System installiert.

Folgende Befehle müssen dazu ausgeführt werden:

```
$ perl Makefile.PL
$ make
$ make test
$ su
Password:
# make install
```

Für das Perl-Modul DBD-Pg wird genau so verfahren wie für die Installation des Moduls DBI.

5 Die C++-Schnittstelle libpqxx

Anmerkung: Dieses Kapitel ist optional. Die mit PostgreSQL 7.3.2 gelieferte Variante ohne die C++-Schnittstelle reicht für die Durchführung der Datenbanken-Praktika aus!

Die C++-Schnittstelle für PostgreSQL ist mit der Version 7.3 ausgelagert worden. Demzufolge muss sie zusätzlich installiert werden. Das geschieht auf ähnliche Art und Weise wie bei den anderen Modulen.

Standardmäßig werden die Header-Dateien durch den Installationsvorgang im Verzeichnis `/usr/local/pqxx/include/pqxx` abgelegt, die Bibliotheken befinden sich im Verzeichnis `/usr/local/pqxx/lib`. Diese Konfiguration lässt sich ändern, indem dem Kommando `configure` entsprechende Werte übergeben werden. Näheres dazu findet sich in der Datei `README`, die man vor der Installation lesen sollte.

```
$ ./configure
$ make
$ make check
$ su
Password:
# make install
```

Anmerkung: Die Shell-Umgebungsvariablen `PGUSER`, `PGDATABASE`, `PGHOST`, `PGPASSWORD` und `PGPORT` müssen vor dem Aufruf des Kommandos `make check` mit den entsprechenden Werten gesetzt werden.

A Anhang

A.1 Das Programm `tar`

`tar` wird verwendet, um mehrere Dateien als ein Archiv abzuspeichern. Dabei werden auch eventuelle Unterverzeichnisse beibehalten. `tar` ist kein Komprimierungsprogramm, die angegebenen Dateien werden quasi nur aneinander angehängt und ergeben letztendlich eine Datei (Archiv). Jedoch besitzt `tar` eine Option, mit der das Archiv nach der Erzeugung automatisch mit dem Programm `gzip` komprimiert wird.

Nachfolgend sind die wahrscheinlich gebräuchlichsten Optionen erklärt.

- Option `x`: extrahiert das Archiv
- Option `z`: ermöglicht das Komprimierungsverfahren
- Option `f`: auf diese Option muss der Name des Archivs folgen
- Option `t`: listet den Inhalt eines Archivs auf
- Option `c`: erstellt ein neues Archiv

Um zum Beispiel das Archiv `postgresql-7.3.2.tar.gz` zu extrahieren, benutzen wir folgende Befehlszeile:

```
$ tar -xzf postgresql-7.3.2.tar.gz
```

Die vollständige Syntax und eine Beschreibung des Programmes können in der Manpage (`man 1 tar`) nachgelesen werden.

A.2 Das Programm `checkinstall`

Die meisten Programme unter Linux besitzen zwar eine Option zum Installieren der Quelldateien (`make install`). Eine Option zum Deinstallieren und zum vollständigen Entfernen aller Dateien des Programmes ist jedoch nur sehr selten vorgesehen. Genau hier kann man das Programm `checkinstall` anwenden.

`checkinstall` „merkt“ sich, in welche Verzeichnisse welche Dateien des zu installierende Programmes hineinkopiert wurden. Diese Informationen werden nach der Installation wahlweise mit dem Paketmanager von Slackware (`installpkg`), RedHat (`rpm`) oder Debian (`dpkg`) abgespeichert und der systemweiten Paketdatenbank hinzugefügt. Dadurch ist es jederzeit möglich, das gesamte Paket wieder zu deinstallieren.

Die Binärversion von `checkinstall` im `rpm`-Format gibt es im Internet unter folgender Adresse:

```
http://asic-linux.com.mx/~izto/checkinstall/
```

Dort folgt man den weiteren Installationshinweisen.

Nach der Installation von `checkinstall` kann jede Installation eines Programmes anstatt mit `make install` mit dem Befehl `checkinstall` durchgeführt werden. Dadurch wird der Installationsvorgang angestoßen und alle Dateien, die bei der Installation ins System integriert werden, werden in einer „Liste“ aufgenommen.

Am Ende einer Installation werden die Informationen aus dieser Liste in ein Paket im entsprechenden Format (Slackware, RedHat oder Debian) abgespeichert und der systemweiten Paket-Datenbank hinzugefügt. Dadurch kann ein installiertes Programm mit dem jeweiligen Paketmanager bei Bedarf wieder restlos entfernt werden.