Der Start mit PostgreSQL

von Stefan Kunick

Inhaltsverzeichnis

Vorwort..... 6
Über die Datenbank PostgreSQL..... 7
Installation der Datenbank auf der LINUX-Host..... 8
Installation einer neueren Version aus dem Internet..... 9
Installation auf einer Host mit SuSE-LINUX..... 12
Installation mit der SuSE-Distribution..... 12
Der User postgres (SuSE)..... 14
Installation auf einer Host mit Red Hat/Fedora-LINUX..... 16
Installation mit der Red Hat/Fedora-Distribution..... 16
Der User postgres (Red Hat/Fedora)..... 18
Installation auf einer Host mit Mandrake-LINUX..... 21
Installation mit der Mandrake-Distribution..... 21
Der User postgres (Mandrake 10 community)..... 23
Weiterführende Arbeiten..... 25
Automatisches Login..... 32
Die Datenbank beenden..... 33
Die Konfigurationsdatei pg_hba.conf..... 35
Die Konfigurationsdatei postgres.conf..... 38
Die Konfigurationsdatei pg_ident.conf..... 39
Das Programm postmaster..... 40
Die Parameter zum Programm postmaster..... 41
Arbeiten mit der Datenbank..... 43
Datenbanken anlegen..... 43
Das Programm createdb..... 43
Datenbanken löschen..... 45
Backup der Datenbank..... 45
Das Programm pg_dump..... 46
Das Programm pg_dumpall..... 50
Restore der Datenbank..... 52
Mit dem Programm psql..... 52
Das Programm pg_restore..... 53
Plattenplatz optimieren..... 55
Routine Reindexing..... 55
Migration der Datenbank / Einführung einer neuen Version..... 56
Write-Ahead Logging..... 57
Datenbank Aktivitäten überwachen..... 58
Version 8 – Postgres und Windows..... 59
Installation..... 59
Das Programm psql konfigurieren..... 68
Arbeiten mit der Datenbank..... 70
Deinstallation..... 71

Client-Software und andere Zugriffsmöglichkeiten ... 72
Installation der Programme psql und unixODBC auf einem LINUX-Host ... 73
SuSE-Distribution ... 73
Red Hat/Fedora Distribution ... 78
Mandrake Distribution ... 80
Der ODBC-Treiber für Windows-Systeme ... 81
Das Programm psql ... 82
Allgemeines zum Programm psql ... 82
Die Terminaloptionen des Programms psql ... 85
Verwalten von Benutzern, Gruppen und Rechte ... 88
Benutzer verwalten ... 88
SQL-Befehle ... 88
Programm createuser ... 89
Programm dropuser ... 91
Gruppe verwalten ... 92
Zugriffsrechte/Privilegien verwalten ... 92
Weitere Dienstprogramme ... 94
createlang ... 95
droplang ... 96
ecpg ... 97
initdb ... 99
initlocation ... 100
ipcclean ... 100
pg_config ... 101
pg_ctl ... 102
pg_resetxlog ... 104
postgres ... 105
Wichtige Punkte beim Datenbankentwurf ... 109
Datentypen ... 109
Views ... 111
Fremdschlüssel ... 112
Schemas ... 113
Funktionen ... 114
allgemeines ... 114
SQL-Funktionen ... 115
PL/pgSQL-Funktionen ... 116
Trigger ... 121
Constraints ... 124
Zugriff mit Programmiersprachen und Programmen ... 125
Mit C/C++/DOT.NET auf die Datenbank zugreifen ... 125
C-Zugriff von Unix ... 125
Embedded SQL ... 134
Der DOT.NET-Zugriff mit der Schnittstelle Npgsql ... 146
Mit Java auf die Datenbank zugreifen ... 162
Serienbriefe mit OpenOffice ... 165
Zugriff einrichten ... 165

Serienbrief verfassen 169
Daten in Postgres übernehmen 171
Anhang A - SQL-Befehle 173
Anhang B - Verzeichnisbaum unter LINUX 178
Anhang C - weitere Möglichkeiten zur Administration der Datenbank 179
Administration über das Programm telnet 179
SuSE-LINUX und telnet 180
Red Hat/Fedora LINUX und telnet 182
Mandrake 10 und telnet 185
Mit Telnet arbeiten 187
Administration über das Programm EMS PostgreSQLManager 188
Administration über das Programm pgAdminIII 189
Anlage D – die Funktionen von PostgreSQL 191
Mathematische Funktionen 191
Trigonometrische Funktionen 193
SQL-Zeichenkettenfunktionen und -operatoren 194
Andere Zeichenkettenfunktionen 195
SQL-Funktionen und -Operatoren für binäre Datenbanken 196
Andere Funktionen für binäre Daten 197
Datentyp-Formatierungsfunktionen 198
Mustervorlagen 199
Mustervorlagen für die Formatierung von Datum/Zeit 199
Mustervorlagen für die Formatierung von numerischen Werten 201
Funktionen für Datum/Zeit 202
Geometrische Funktionen 203
Geometrische Typumwandlungsfunktionen 204
Sitzungsinformationsfunktionen 205
Aggregatfunktionen 206
Anhang E – die Zeichensätze 207
Anhang F - Liste der Dienstprogramme 208
Anhang G – Fehlercode beim embedded SQL 209
Quellen 213

Diese Seite bleibt aus technischen Gründen frei

Vorwort

Weshalb das Buch entstand:
Durch meine Beschäftigung mit JAVA, C, C++ und LINUX befasste ich mich mit der Datenbank PostgreSQL näher. Da es zu diesem Zeitpunkt relativ wenig deutsche Literatur über diese Datenbank gab, schrieb ich den Leitfaden „Die ersten Schritte in der Administration mit der PostgreSQL-Datenbank“. Mit der Zeit wurde der Leitfaden immer umfangreicher und rückte auch mehr von der Administrationsseite ab.

Hinweis über das Buch:

1. Dieses Buch beschreibt den Einstieg in den Umgang mit der PostgreSQL-Datenbank. Diese Beschreibung wendet sich an die Leser, die bisher wenig mit Linux gearbeitet haben. Das Buch eignet sich auch für die Gelegenheitstäter, die sich in großen Abständen mit dem Thema befassen. Sie erhalten übrigens die Orginalunterlagen von der Web-Seite www.postgresql.org (sie sind hier in Englisch abgefasst), oder auch in Ihrer Linux-Distribution (teilweise mit deutscher Beschreibung).
2. Der Installations- und Einrichtungsvorgang ist aus Sicht der SuSE-Linux Professional Distribution 8.0 beschrieben. Auf andere Distributionen wird aber eingegangen. Bei den nachfolgenden und vorhergehenden Versionen kann es Abweichungen geben.
3. Der Leser installiert und bedient die Software auf eigenes Risiko. Der Autor lehnt jede Haftung ab.
4. Über mögliche Anregungen und Hinweise würde ich mich sehr freuen (eMail: stefan@kunick.org).
5. Die in diesem Buch verwendeten Soft- und Hardwarebezeichnungen sind in vielen Fällen auch eingetragene Warenzeichen; sie werden ohne Gewährleistung der freien Verwendbarkeit benutzt. Ich richte mich im Wesentlichen nach der Schreibweise der Hersteller. Die Wiedergabe von Waren- und Handelsnamen berechtigt nicht zur Annahme, dass solche Namen (im Sinne der Warenzeichen und Markenschutz-Gesetzgebung) als frei zu betrachten sind.
6. Das Buch darf kostenlos weitergegeben werden.

Dank an:
Erika Aupperle, Cornelia Boenigk und Janny Kunick für die Durchsicht des Buches

Waiblingen, den 23.02.2005 **Stefan Kunick**

Über die Datenbank PostgreSQL

PostgreSQL ist eine relationale Datenbank, die auf der POSTGRES Version 4.2 basiert. Sie wurde an der Universität von Kalifornien am Berkeley Computer Science Institut entwickelt. Das Projekt POSTGRES DBMS begann im Jahr 1986. 1994 fügten Andrew Yu und Jolly Chen den SQL Interpreter hinzu. Die Version Postgres95 wurde im Web veröffentlicht und war der Nachfolger des POSTGRES Berkeley Codes. Als Open-Source-Programm steht es frei verfügbar unter der Adresse http://www.postgresql.org im Internet. Es unterstützt die SQL-Standards 92 und 99.

Die Größe einer Datenbank ist unbegrenzt und eine Tabelle kann bis zu 64 Terra-Byte groß werden. Bei der Anzahl der Datensätze gibt es keine Grenzen (der verfügbare Plattenplatz einmal ausgenommen). Die Anzahl der Spalten ist auf 1600 begrenzt und jede Spalte nimmt maximal Daten bis zu einem Gigabyte auf.

Zusätzlich kann man auch Bilder in der Datenbank abspeichern. Ein weiterer Pluspunkt der Datenbank sind die Funktionen. Mit ihnen kann man direkt SQL-Befehle auf der Datenbank ausführen.

Installation der Datenbank auf der LINUX-Host

Verschiedene LINUX-Distributoren, wie z.B. SuSE, Mandrake und Red Hat bieten Ihnen auf Ihren CD's, bzw. DVD's auch eine Postgres-Version an. Das Verfahren ist in der Anfangsphase im Umgang mit der Datenbank recht einfach und Sie kommen hier auch ohne einen Download vom Internet aus. Leider sind diese Versionen nicht immer die aktuellsten. Ein Punkt gegen eine Installation von den Datenträgern der LINUX-Distributoren ist der, dass die Programme dann im Verzeichnis /usr/bin abgelegt werden. Bei einem möglichen Update auf eine höhere Version (die Sie sich vielleicht aus dem Internet heruntergeladen haben) müssen Sie dann die Programme aus diesem Verzeichnis entfernen.

(**Anmerkung des Autors:** ab der Version 8.0 benötigen Sie das Programm CYGWIN, bei einer Installation unter Windows nicht mehr).

Die Installation der Datenbank läuft in mehreren Stufen ab. Dazu können Sie folgende Wege einschlagen:

Stufe	**LINUX-Distribution egal**	**SuSE-Distribution**	**Red Hat/Fedora-Distribution**	**Mandrake-Distribution**	**Windows**
Installation der Serversoft-ware	Siehe Kapitel: Installation einer neuen Version aus dem Internet	Siehe Kapitel: Installation mit der SuSE-Distribution	Siehe Kapitel: Installation mit der Red Hat-Distribution	Siehe Kapitel: Installation mit der Mandrake-Distribution	Siehe Kapitel: Installation unter Windows
User postgres	Siehe Kapitel: Der User postgres (SuSE)	Siehe Kapitel: Der User postgres (SuSE)	Siehe Kapitel: Der User postgres (Red Hat)	Siehe Kapitel: Der User postgres (Mandrake))	
Datenbank einrichten	Siehe Kapitel: Weiterführende Arbeiten				

Installation einer neueren Version aus dem Internet

Die hier beschriebenen Arbeitsschritte müssen Sie entweder als Benutzer root oder mit Administrator Rechten durchführen (auf dem PC hatte ich LINUX mit den Softwarepaketen für eine Arbeitsstation installiert und dem KDE-Desktop).

Schritt	Inhalt
1	Über ***Yast*** (oder bei anderen LINUX-Distributionen ähnliches Programm (bei Fedora z.B. ***Hinzufügen/Entfernen von Applikationen***)) folgende Softwarepakete installieren: - gcc - cpp - make **ANMERKUNG zu SuSE 8.0:** Hier finden Sie die Pakete unter den Rubriken ***Entwicklung/Programmiersprachen c und c++*** und ***Entwicklung/Tools/Building***) **ANMERKUNG zu SuSE 8.2:** Sie finden die Pakete unter ***c/c++ Compiler und Werkzeuge*** und ***Entwicklung erweitert***. Aus dem ***Paket Entwicklung erweitert*** installieren Sie noch die verschiedenen Librarys (z.B. Readline) **ANMERKUNG zu SuSE 9.1:** Setzen Sie den Filter bei der Paketauswahl auf Selektionen. Dann finden Sie den Compiler unter dem Eintrag ***C/C++Compiler und Werkzeuge*** (Es reichen die Pakete: gcc, readline-devel (Filter auf Paketgruppen setzen und unter Entwicklung, Bibliotheken, c und c++ suchen), zlib-devel (suchen wie bei readline-devel) und make). **ANMERKUNG zu RedHat 9/Fedora:** Sie installieren hier die Entwicklungstools aus dem Bereich der Softwareentwicklung **ANMERKUNG zu Mandrake 10 comunity:** Sie installieren hier die Pakete ***gcc-3.3.2-6mdk***, ***gcc-c++-3.3.2-6mdk***, ***automake-1.4-23.p6.mdk*** und ***zlib-devel-1.2.1-2mdk*** (mit der Library Readline gab es allerdings Probleme, sie war dort leider nicht vorhanden). Das Programm zur Softwareinstallation finden Sie wie folgt: ***Start-Taste*** in der ***Taskleiste***, ***System***, ***Einstellungen***, ***Paketierung***, ***Software installieren***).

Schritt	Inhalt
2	Wenn Sie die Software von einer CD in ein Verzeichnis kopieren, dann mounten Sie die CD über den Befehl ***mount /media/cdrom***. Die Daten der CD finden Sie im Verzeichnis / cdrom (vorher allerdings in das CD-Rom-Laufwerk einlegen). Bei den neueren Versionen ist das teilweise nicht mehr nötig. **ANMERKUNG zu Mandrake:** Sie müssen die CD nicht mounten. Sie finden die Daten im Verzeichnis /mnt/cdrom.
3	Mit dem Befehl ***mkdir /usr/local/src/Postgres*** legen Sie ein neues Verzeichnis für die Quellen an
4	Kopieren Sie die gz-Datei (z. B. Postgresql-7.4.3.tar.gz, die Zahlenkombination 7.4.3 steht hier für die Version 7.4.3, bei neueren Versionen ändert sie sich) in das neu angelegte Verzeichnis (z.B. mit dem cp-Befehl)
5	Wenn Sie im Arbeitsschritt 2 eine CD gemountet haben, so können Sie jetzt die CD mit dem Befehl ***umount /media/cdrom*** unmounten (bei Mandrake nicht notwendig). Anschließend entnehmen Sie die CD.
6	Jetzt entkomprimieren Sie die Datei mit dem Befehl ***gunzip postresql-7.4.3.tar.gz***
7	Danach packen Sie die notwendigen Dateien aus. Geben Sie den Befehl ***tar xf postgresql-7.4.3.tar*** ein.
8	Wechseln Sie nun in das Verzeichnis postgresql-7.4.3 mit dem Befehl ***cd postgresql-7.4.3***
9	Als nächstes konfigurieren Sie den Source-Tree. Er enthält die Optionen, mit denen Sie PostgreSQL übersetzen. Sie geben dazu den Befehl ***./configure*** ein (sollten Sie die Library Readline nicht installiert haben, so können Sie diese Library auch weglassen. Geben Sie dazu den Befehl ***./configure --without-readline*** ein. Diese Library ist für die Befehls-History zuständig). Wenn alles funktioniert hat, erhalten Sie die Datei config.status. Wenn Sie möchten, dann können Sie auch die Meldungen in Deutsch ausgeben lassen. Sie benötigen nur den Parameter –enable-nls='de'.
10	Mit dem Build-Prozess erzeugen Sie nun die ausführbaren Dateien. Der Prozess kann längere Zeit den Rechner in Anspruch nehmen. Das Kommando lautet ***gmake***. Ob das Programm erfolgreich durchlief sehen Sie durch die Meldung: ***All of PostgreSQL is successfully made. Ready to install***.

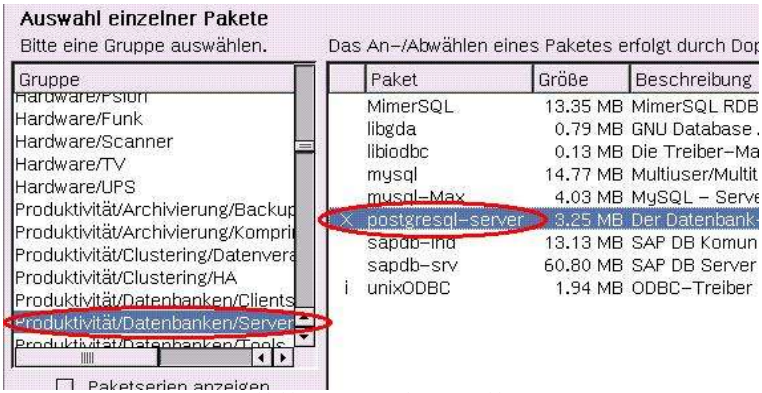
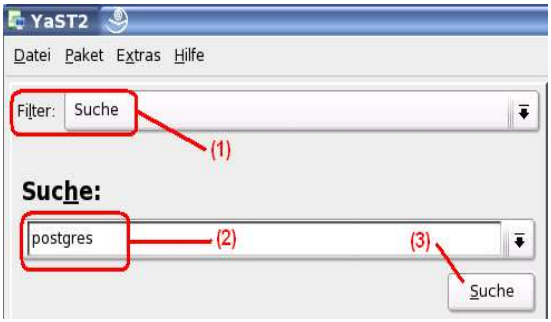
Schritt	Inhalt
11	Mit einem Regession-Test testen Sie den soeben erzeugten Build. Diese Aktion können Sie nicht als Benutzer root durchführen. Vergeben Sie dazu die notwendigen Rechte an einen anderen Benutzer. Die Befehle sehen so aus: 1. ***chmod -R a+w src/test/regress*** 2. ***chmod -R a+w contrib/spi*** 3. ***su <anderer Benutzer>*** 4. ***gmake check*** (Sie erhalten dann die Meldung ***All 93 tests passed***) 5. wieder als Administrator anmelden
12	Mit dem Befehl ***gmake install*** installieren Sie PostgreSQL in das Verzeichnis, das in der Datei ***configure*** angegeben wurde (Standard: /usr/local/pgsql/bin, /usr/local/pgsql/lib ...., siehe Eintrag ***ac_default_prefix,*** **Vorsicht:** es gibt mehrere Einträge). Wenn Sie die Datei mit dem Editor ***vi*** bearbeiten, dann können Sie nach den Einträgen mit ***/ac_default_prefix*** suchen. (**TIPP:** Rechte für die Anwender überprüfen und vergeben)
13	Setzen Sie nun die Umgebungsvariablen für die Shared Libraries (**TIPP:** am besten im Startskript vermerken. Z.B. in der Datei /etc/profile (dann gelten die Werte für alle)). Das Vorgehen kann bei anderen Linux Distributionen ggf. abweichen. Die Befehle lauten: 1. ***LD_LIBRARY_PATH=/usr/local/pgsql/lib*** 2. ***export LD_LIBRARY_PATH***
14	Passen Sie danach die Umgebungsvariable PATH an . 1. ***PATH=/usr/local/pgsql/bin:$PATH*** 2. ***export PATH***
15	Die Umgebungsvariable MANPATH ist für die Manpages wichtig. Die Anweisungen lauten: 1. ***MANPATH=/usr/local/pgsql/man:$MANPATH*** 2. ***export MANPATH*** (der Tipp vom Schritt 14 gilt hier übrigens auch) **ANMERKUNG:** Wenn Sie auf die Umgebungsvariablen später nochmals zugreifen möchten, müssen Sie sie dann neu setzen (siehe Kapitel Weiterführende Arbeiten). Das ist dann der Fall, wenn Sie sich als normaler Anwender angemeldet haben und die Installation als User „su“ durchführen und auch die Umgebungsvariablen unter User „su“ gesetzt haben.
16	Nun geht es je nach Distribution entweder im Kapitel *Der User postgres (SuSE), Der User postgres (RedHat/Fedora), Der User postgres (Mandrake)* weiter.

Installation auf einer Host mit SuSE-LINUX

Installation mit der SuSE-Distribution

Wie schon am Anfang beschrieben, benötigen Sie die Professional Version von Linux. Sie enthält auch die entsprechende Software. Sie können natürlich auch alternativ eine Version aus dem Internet herunterladen und auf dem Rechner installieren.

Schritt	Inhalt / Aktion
1	Rufen Sie als erstes das YaST2-Kontrollzentrum auf. Wenn Sie den KDE-Desktop einsetzen drücken Sie hier die ***Start***-Taste, ***Einstellungen***, ***Yast menu*** und dann ***Yast Kontrollzentrum***.
2	Unter der Rubrik ***Software*** (linke Seite) finden Sie das ICON ***Software installieren/löschen***. Bitte das Programm ***Software installieren/löschen*** mit einem Doppelklick aufrufen.  *Abbildung 1 SuSE YaST*

Schritt	Inhalt / Aktion
3	Als nächstes installieren Sie die „Server-Anwendung“ auf ihrem Linux-Host. Aus dem Paket ***Produktivität/Datenbanken/Server*** wählen Sie das Paket ***postgresql-server*** aus (drücken Sie hier einfach den Knopf ***Aus/Abwählen***).  *Abbildung 2 SuSE Pakete auswählen* **ANMERKUNG zu neueren Versionen**: Stellen Sie den Filter auf Suchen (siehe (1))ein und geben Sie in dem Eingabefeld den Begriff postgres (siehe (2)) ein. Wenn Sie dann den Knopf ***Suche*** drücken, erhalten Sie eine komplette Trefferliste aller Postgres-Pakete.  *Abbildung 3 SuSE optimale Suche nach Paketen*
4	Soll auf der Linux - Host nur die Datenbank laufen dann können Sie schon die Auswahl abschließen und die geforderte CD einlegen.

Schritt	Inhalt / Aktion
5	Wenn die Software installiert ist, gehen Sie weiter zu dem Kapitel *Der User postgres (SuSE)*.

Der User postgres (SuSE)

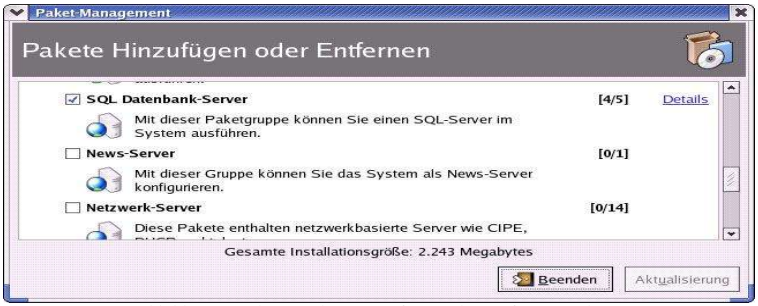
Schritt	Inhalt / Aktion
1	Überprüfen Sie, ob der Anwender postgres schon in der Benutzerverwaltung vorhanden ist (drücken Sie die Start-Taste, System, Konfiguration, ***KUser*** (Benutzerverwaltung) (In den neueren SuSE-Versionen müssen Sie das Programm KUser noch installieren. Sie finden es im Paket kdeadmin3kde). (**TIPP:** das Programm Yast geht auch)  *Abbildung 4 SuSE Benutzerverwaltung* **ANMERKUNG zu SuSE 9.1:** Sollten Sie eine Version aus dem Internet installieren, dann ist der User postgres noch nicht vorhanden.

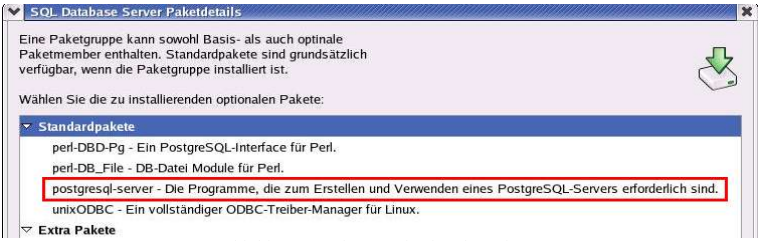
Schritt	Inhalt / Aktion
2	Als Nächstes überprüfen Sie noch einige Einstellungen des Benutzers ***postgres*** und vergeben ihm auch ein Passwort! Drücken Sie dazu den Knopf ***Edit*** in der Menueleiste. Anschließend erhalten Sie den Zugriff auf die Benutzereigenschaften. Über den Knopf ***Passwort setzen*** ändern Sie das Passwort ab.  *Abbildung 5 SuSE Benutzer anlegen*
3	Zum Schluss überprüfen Sie noch, ob der Anwender zur Gruppe ***daemon*** gehört. Über den Karteireiter Gruppen erreichen Sie die Seite.  *Abbildung 6 SuSE Benutzer und Gruppe*
4	Alle folgenden Schritte finden Sie im Kapitel *Weiterführende Arbeiten*.

Installation auf einer Host mit Red Hat/Fedora-LINUX

Installation mit der Red Hat/Fedora-Distribution

Für die Datenbank verwendete ich die Version 9 von Red Hat (die Version mit den 3 CD's. Sie können die Version unter www.linuxiso.org auch herunterladen). Red Hat hat diese Produktlinie unter dem Namen Fedora ausgegliedert. Von der o.g. Homepage können Sie sich die Software auch downloaden. Alternativ bietet Ihnen der Zeitschriftenhandel die Distribution auch an (Kostenpunkt ca. 10 Euro). Von der Gestaltung her hat sich nicht viel geändert.

Schritt	Inhalt / Aktion
1	Drücken Sie die Start-Taste (mit dem roten Hut). Danach geht es weiter mit den Menüpunkten ***Systemeinstellungen*** und ***Hinzufügen / Entfernen von Applikationen***
2	Im Fenster ***Pakete Hinzufügen oder Entfernen*** finden Sie den Eintrag ***SQL Datenbank-Server***. Wählen Sie diesen Punkt aus.  *Abbildung 7 Red Hat Pakete Hinzufügen oder Entfernen Red Hat* Bei Fedora finden Sie die Pakete unter der Rubrik ***Server*** und dem Eintrag ***PostgreSQL Datenbank***.

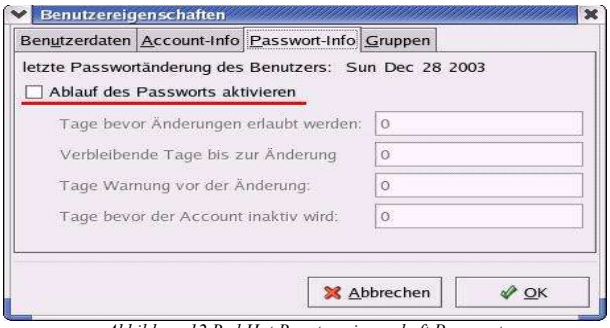
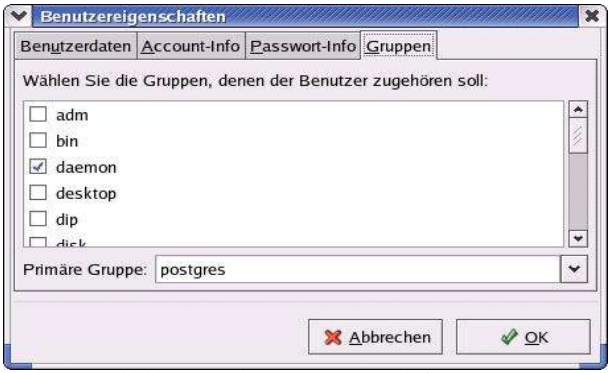
Schritt	Inhalt / Aktion
3	Wenn Sie im vorherigen Schritt auf den Eintrag ***Details*** drücken, sehen Sie im Detail, welche Software mit den Standardpaketen installiert wird.  *Abbildung 8 Red Hat Paketdetails Red Hat*
4	Danach schließen Sie die Auswahl ab, drücken den Knopf ***Aktualisierung*** und legen die geforderten CD's ein.
5	Wenn die Software installiert ist, gehen Sie weiter zu dem Kapitel *Der User postgres (Red Hat/Fedora).*

Der User postgres (Red Hat/Fedora)

Wie auch bei SuSE-Distribution , überprüfen wir nun die Einstellungen des Benutzers ***postgres***.

Schritt	Inhalt / Aktion
1	Drücken Sie die Start-Taste (mit dem roten Hut). Danach geht es weiter mit den Menüpunkten ***Systemeinstellungen*** und ***Benutzer und Gruppen.***
2	Verändern Sie zuerst die Einstellungen über die Darstellung der Benutzer und Gruppen. Sie finden den Punkt im Menüpunkt ***Präferenzen (Systembenutzer und -gruppen filtern).*** Wählen Sie anschließend den Benutzer ***postgres*** aus und drücken den Knopf Eigenschaften.  *Abbildung 9 Red Hat Benutzerverwaltung*

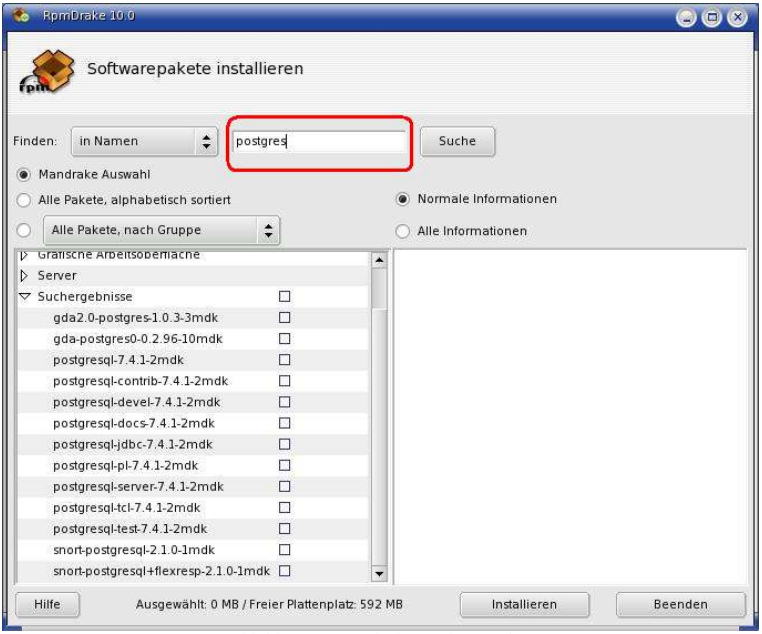
Schritt	Inhalt / Aktion
3	Als Nächstes können Sie unter der Registrierkarte ***Benutzerdaten*** dem Anwender ein Passwort vergeben.  *Abbildung 10 Red Hat Benutzereigenschaften Benutzerdaten*
4	Unter der Registrierkarte Account-Info entsperren Sie den Benutzer-Account und prüfen, ob eine Verfallszeit vorgesehen ist.  *Abbildung 11 Red Hat Benutzereigenschaften Account-Info*

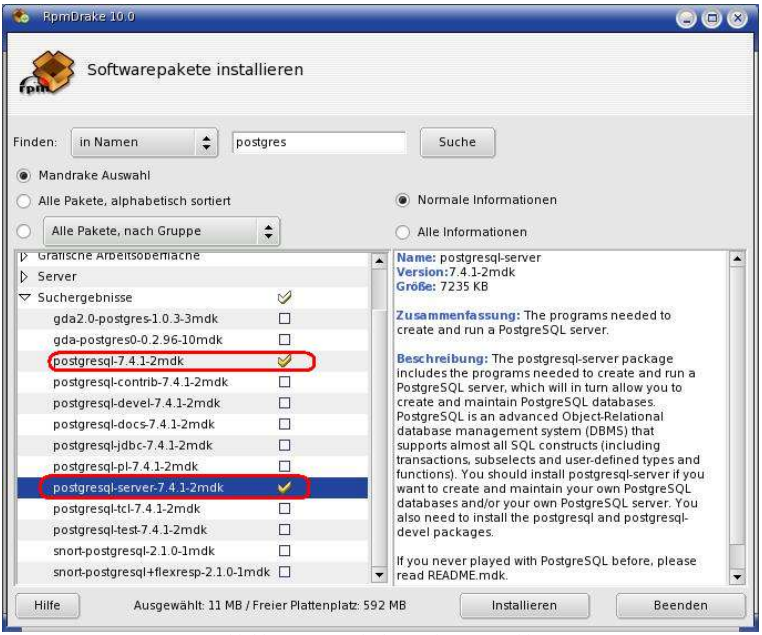
Schritt	Inhalt / Aktion
5	Wechsel Sie danach auf die Karteikarte Passwort-Info. Beachten Sie hier den Ablauf des Passwortes.  *Abbildung 12 Red Hat Benutzereigenschaft Passwort*
6	Zuletzt überprüfen Sie noch die Gruppenzugehörigkeit. Mit einem Kreuz bei dem Eintrag ***daemon*** können Sie dann die Datenbank als „Dienst“ laufen lassen. Danach geht es weiter im Kapitel *Weiterführende Arbeiten*  *Abbildung 13 Red Hat Benutzereigenschaft Gruppen*

Installation auf einer Host mit Mandrake-LINUX

Installation mit der Mandrake-Distribution

Für die Datenbank verwendete ich die Version 10 community. Die CD's finden Sie auch unter der Homepage www.linuxiso.org. Wenn Sie Mitglied im Mandrakeclub sind, dann können Sie das Betriebssystem auch von der Homepage von Mandrake herunterladen.

Schritt	Inhalt / Aktion
1	Drücken Sie die Start-Taste (mit dem gelben Stern). Danach geht es weiter mit den Menüpunkten ***System, Einstellungen, Paketierung*** und ***Software installieren.***
2	Um die Suche nach den Softwarepaketen etwas zu vereinfachen, geben Sie in diesem Fenster postgres ein und drücken den Knopf Suche.  *Abbildung 14 Mandrake - Pakete suchen*

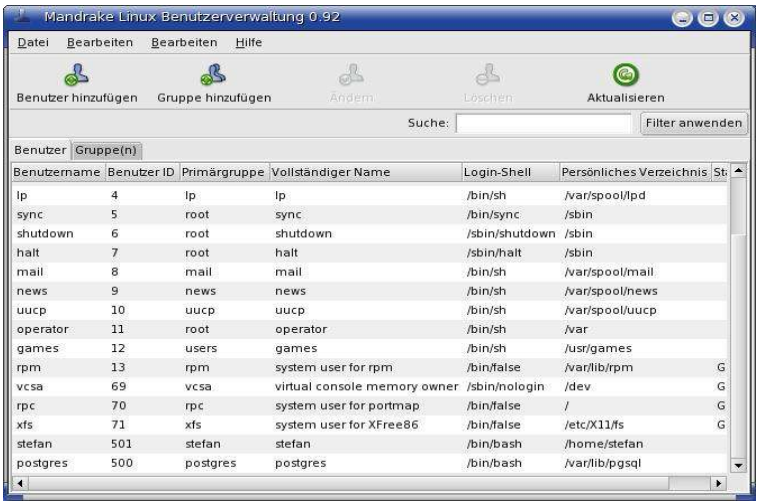
Schritt	Inhalt / Aktion
3	Danach erhalten Sie eine Trefferliste. Aus dieser Liste wählen wir auf jeden Fall den Eintrag postgresql-server-7.4.1-2mdk aus (enthält die Server-Software). Zur Verwaltung der Datenbank auf der Host ist auch Client-Software sinnvoll. Deshalb selektieren Sie auch das Softwarepaket postgresql-7.4.1-2mdk. Am Ende drücken Sie den Knopf Installieren.  *Abbildung 15 Mandrake - Pakete auswählen*
4	Im Anschluß legen Sie die angeforderte CD ein.
5	Nach der Installation gehen Sie weiter zu dem Kapitel *Der User postgres (Mandrake 10 community).*

ANMERKUNG: Die beiden Verzeichnisse /var/lib/pgsql und /var/lib/pgsql/data sind schon angelegt und gehören dem User postgres (siehe Kapitel **Weiterführende Arbeiten**)

Der User postgres (Mandrake 10 community)

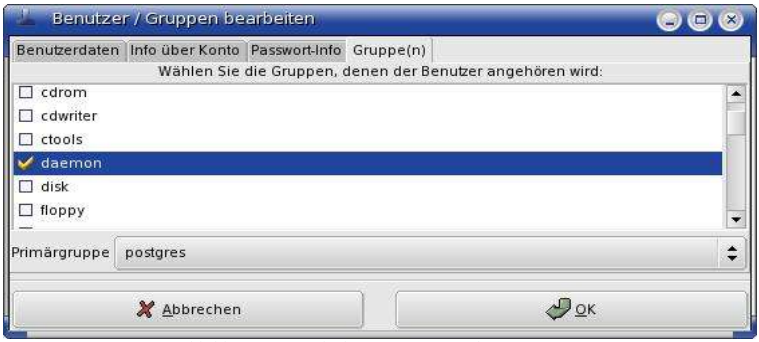
Wie auch bei den anderen Distributionen überprüfen wir nun die Einstellungen des Benutzers ***postgres***. Bei dieser Version müssen Sie allerdings den User neu anlegen. Es gibt mit dem Programm Probleme mit der Benutzer-ID. Sie können hier keinen User anlegen, der eine ID unter 500 hat. Wenn Sie eine ID unter 500 haben möchten, verwenden Sie hier den Konsolen-Befehl ***adduser -u postgres***.

Schritt	Inhalt / Aktion
1	Drücken Sie die Start-Taste (in der Taskleiste). Danach geht es weiter mit den Menüpunkten ***System***, ***Einstellung***, ***Sonstiges*** und ***User Administration.*** **ANMERKUNG:** Wenn Sie die Software von der Mandrake CD installiert haben, ist der Benutzer schon angelegt. Er ist nur noch nicht der Gruppe daemon zugeordnet.
2	Wenn Sie alle Benutzer ansehen möchten, entfernen Sie die Filterung des Systemkennzeichens (Menueleiste ***Bearbeiten***, ***Systemkennzeichen filtern***). Drücken Sie hier dann den Knopf ***Benutzer hinzufügen***.



Benutzername	Benutzer ID	Primärgruppe	Vollständiger Name	Login-Shell	Persönliches Verzeichnis	St
lp	4	lp	lp	/bin/sh	/var/spool/lpd	
sync	5	root	sync	/bin/sync	/sbin	
shutdown	6	root	shutdown	/sbin/shutdown	/sbin	
halt	7	root	halt	/sbin/halt	/sbin	
mail	8	mail	mail	/bin/sh	/var/spool/mail	
news	9	news	news	/bin/sh	/var/spool/news	
uucp	10	uucp	uucp	/bin/sh	/var/spool/uucp	
operator	11	root	operator	/bin/sh	/var	
games	12	users	games	/bin/sh	/usr/games	
rpm	13	rpm	system user for rpm	/bin/false	/var/lib/rpm	G
vcsa	69	vcsa	virtual console memory owner	/sbin/nologin	/dev	G
rpc	70	rpc	system user for portmap	/bin/false	/	G
xfs	71	xfs	system user for XFree86	/bin/false	/etc/X11/fs	G
stefan	501	stefan	stefan	/bin/bash	/home/stefan	
postgres	500	postgres	postgres	/bin/bash	/var/lib/pgsql	

Abbildung 16 Mandrake Benutzerverwaltung

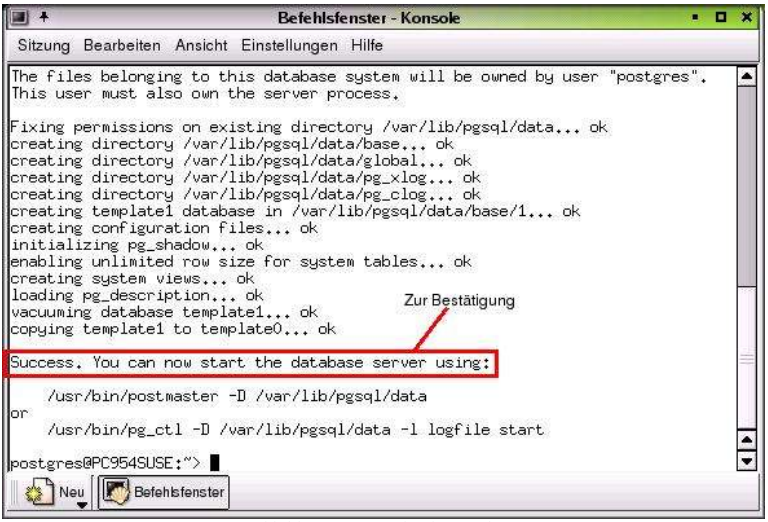
Schritt	Inhalt / Aktion
3	Kreuzen Sie im Karteireiter ***Gruppen*** den Eintrag ***daemon*** an.  *Abbildung 17 Mandrake Benutzerverwaltung Gruppen*
4	Im Konto ***Info über Konto*** überprüfen Sie nur noch die Einträge (das Benutzerkonto soll z.B. nicht ablaufen).  *Abbildung 18 Mandrake Benutzerverwaltung Info über Konto*

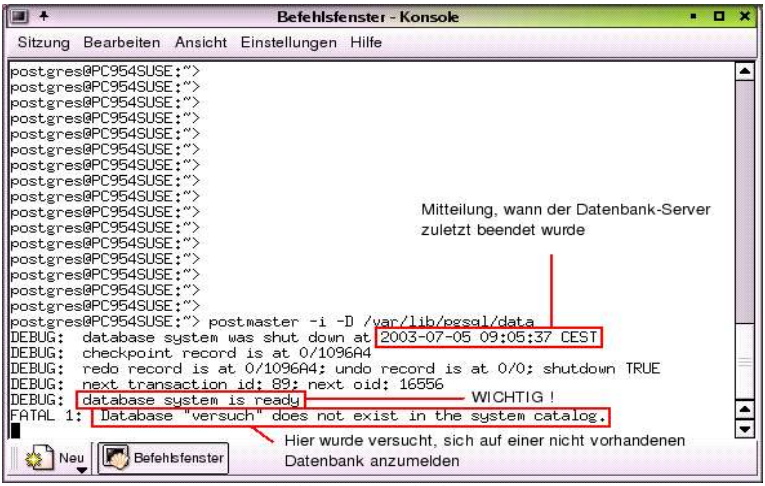
Schritt	Inhalt / Aktion
5	Hier noch den Eintrag **Passwort soll ablaufen** überprüfen  *Abbildung 19 Mandrake Benutzerverwaltung Passwort-Info*

Weiterführende Arbeiten

Die hier beschriebenen Arbeitsschritte sind alle unabhängig von der entsprechenden LINUX-Distribution. Wir richten hier die Datenbank ein.

Schritt	Inhalt / Aktion
1	Melden Sie sich als Anwender ***root*** an (von der Konsole mit dem Befehl ***su*** geht es auch) und wechseln Sie in das Verzeichnis ***/var/lib/pgsql***. Legen Sie dort das Verzeichnis ***data*** an (**INFO:** eine Übersicht des Dateibaumes finden Sie im Anhang). **ANMERKUNG:** Sie können natürlich auch ein anderes Verzeichnis verwenden. Später können Sie auch das ***data***-Verzeichnis in ein anderes Verzeichnis verschieben. Beim Programmstart geben Sie dann das geänderte Verzeichnis mit an. (Sie können auch ein über NFS gemapptes Verzeichnis verwenden, es belastet dann aber das Netz und die Kapazität des Netzes ist auch entscheidend). Übrigens, wenn Sie sich als Anwender postgres anmelden, dann können Sie ohne Probleme in Ihrem Home-Directory das Verzeichnis ***data*** anlegen und sparen sich den nächsten Arbeitsschritt.

Schritt	Inhalt / Aktion
2	Mit dem Befehl chown wechseln Sie den Besitzer des Verzeichnisses ***/var/lib/pgsql*** und ***/var/lib/pgsql/data***. Der genaue Syntax lautet: ***chown -c postgres pgsql*** (aus dem Verzeichnis /var/lib heraus) bzw. ***chown -c postgres data*** (aus dem Verzeichnis /var/lib/pgsql)**.**
3	Melden Sie sich anschließend als User ***postgres*** an (geht auch über die Konsole mit dem Befehl ***su postres*** vom aktuell angemeldeten Benutzer aus) und starten Sie die Datenbank mit dem Befehl: ***initdb -D /var/lib/pgsql/data***  *Abbildung 20 Datenbank einrichten* Nähere Erläuterungen zum Start der Datenbank erhalten Sie auf den folgenden Seiten.

Schritt	Inhalt / Aktion
4	Nun können Sie die Datenbank starten mit der Anweisung: ***postmaster -i -D /var/lib/pgsql/data*** (mit diesem Prozess läuft dann die Datenbank, beim Start des Systems muss dieser Befehl dann immer eingegeben werden) **WICHTIG:** zusätzlich mit der Option -i starten, damit es mit der TCP/IP-Verbindung klappt (die Option -i muß vor dem -D kommen). Wenn alles in Ordnung ist erhalten Sie die Meldung ***database system is ready*** (siehe unten). Weitere Informationen über das Programm postmaster finden Sie im Kapitel postmaster.  *Abbildung 21 Datenbank starten*
5	Als Nächstes legen Sie eine Datenbank an. Den dazu notwendigen Befehl geben Sie über die Konsole ein (hier sollten Sie als Benutzer postgres angemeldet sein). Er lautet: ***createdb -E LATIN1 [NAME der Datenbank]*** (z.B. createdb -E LATIN1 test) **VERWEIS:** Siehe auch im Anhang A SQL-Befehle. **WICHTIG:** Mit der Option -E LATIN1 haben Sie später keine Probleme mit den Umlauten.
6	Testweise können Sie jetzt die Datenbank aufrufen (Sie müssen noch als Anwender postgres angemeldet sein, sonst erhalten Sie kein Zugriffsrecht auf die Datenbank). Die Verbindung zur eben angelegten Datenbank nehmen Sie über den Befehl ***psql -d [NAME der Datenbank]*** auf. Das Programm beenden Sie mit der Anweisung ***\q***. Eine nähere Beschreibung des Programms psql erhalten Sie im Kapitel psql.

Schritt	Inhalt / Aktion
7	Im unserem Fall lautet das Data - Verzeichnis des SQL-Servers /var/lib/pgsql/data. In ihm finden Sie die wichtigsten Konfigurationsdateien. Um z.B. von anderen Computern auf die Datenbank unserer LINUX-Host zugreifen zu können, müssen Sie die Datei pg_hba.conf bearbeiten (siehe extra Kapitel)! Eine weitere wichtige Konfigurationsdatei ist die Datei postgresql.conf. Sie enthält z.B. die Anzahl der maximal möglichen Verbindungen und ob z.B. eine TCP/IP Verbindung möglich ist (siehe auch später). In der Datei postmaster.pid finden Sie unter anderem die Process-ID des zurzeit gestarteten Programmes postmaster (**WICHTIG:** um die Datenbank zu beenden).
8	**TIPP:** Damit Sie nicht immer das Datenverzeichnis der Datenbank eingeben müssen empfiehlt sich der Einsatz der Systemvariablen PGDATA. Ich habe dazu die Variable PGDATA in die Datei etc/profile eingetragen und anschließend exportiert. Nach einem erneuten abarbeiten der Datei (z.B. bei einem Systemstart) kennt dann das System die Variable. Bitte prüfen Sie vorher, ob die Variable nicht schon in einem anderen Skript gesetzt wurde. Übrigens, wenn Sie eine Version aus dem Internet installiert haben, dann können Sie auch die anderen Variablen hier berücksichtigen (PATH, MANPATH) (**ANMERKUNG:** Bei SuSE 9.1 wird empfohlen die Einträge in der Datei /etc/profile.local vorzunehmen).

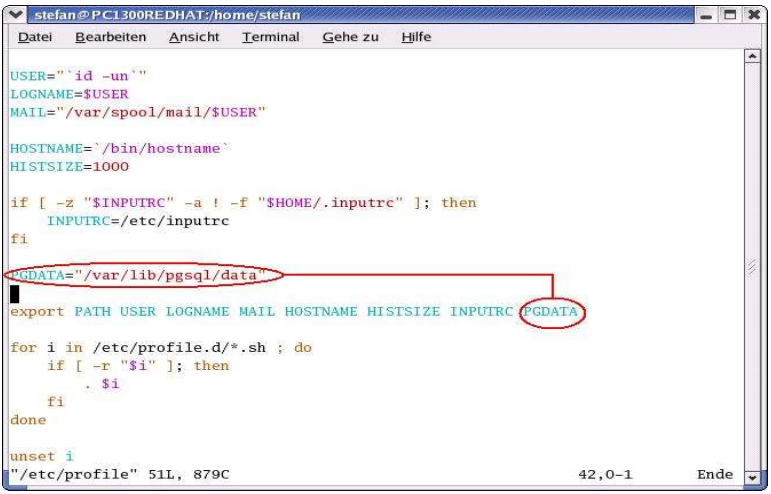


Abbildung 22 PGDATA

Die Datenbank kann man übrigens auch automatisch beim Hochfahren des Systemes starten. Beachten Sie dabei, dass dann der Benutzer ***postgres*** kein Passwort enthalten darf (**VORSICHT!**)! Um die Datenbank zu starten, bearbeiten Sie im Verzeichnis /etc die Datei rc.local (**ANMERKUNG:** Diese Datei hat die gleichen Aufgaben wie die Datei AUTOEXEC.BAT bei den Betriebssystem Windows 98/95). Das Prinzip wurde verbessert. Beim Systemstart durchläuft LINUX verschiedene Runlevels. In jedem Runlevel startet das System verschiedene Skripts in unterschiedlichen Verzeichnissen (siehe auch die README-Datei im Verzeichnis /etc/rc.d). Sie finden die Verzeichnisse für die einzelnen Runlevels unter dem Verzeichnis /etc/rc.d). Die Nummer des Runlevels ist mit im Verzeichnisnamen kodiert (rc[Runlevel].d). Erst ab dem Runlevel 3, wenn das Netzwerk gestartet ist, macht es Sinn die Datenbank zu starten. Beachten Sie, dass nur Skriptdateien berücksichtigt werden, die mit S oder K beginnen und eine zweistellige Zahl enthalten (das S steht für Start, das K für Kill). Kopieren Sie sich die Datei /etc/init.d/skeleton in das entsprechende Verzeichnis und bearbeiten die Datei (**besser:** Sie kopieren die Datei in das Verzeichnis ***/etc/init.d*** und legen einen symbolischen Link in das Verzeichnis. Der Link in dem Verzeichnis darf bzw. muss dann einen anderen Namen haben. Mit dem Befehl ***ln -s <quelle> <ziel>*** legen Sie den Link an. Mit dem Befehl **ls -l** sehen Sie Ihre „Quelldatei“).
ANMERKUNG: das Verzeichnis rc.d ist nur ein Link auf ein anderes Verzeichnis. Beim Systemstart ruft LINUX alle Programme und Skripts als Superuser auf. Als Superuser, oder als anderer Benutzer, ist es allerdings nicht zulässig das Programm postmaster zu starten. Deshalb rufen Sie einen Daemon-Prozess auf und geben als Anwendernamen des Benutzers postgres an. Das vereinfachte Skript sieht dann wie unten gezeigt aus (Sie können in das Verzeichnis /etc/rc.d/rc5.d eine Verknüpfung legen) (Übrigens: Das Byte zur Ausführung der Datei nicht vergessen! In der ersten Zeile der Datei sollte auch der Eintrag ***#! /bin/sh*** vorhanden sein):

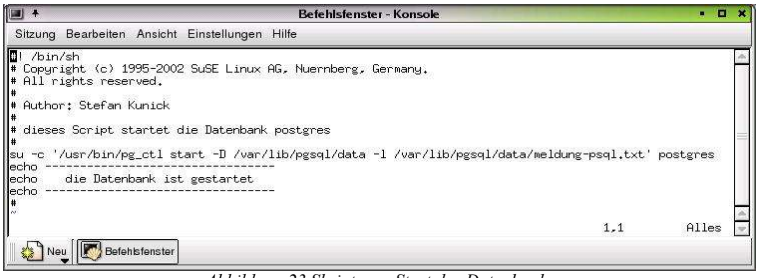


Abbildung 23 Skript zum Start der Datenbank

Die wichtigste Zeile in dem Skript ist die Anweisung ***su -c '/usr/bin/pg_ctl start -D /var/lib/pgsql/data' postgres***. (**ANMERKUNG:** Sollte es Probleme mit dem Login von postgres geben, schreiben Sie anstelle postgres -l postgres. Wenn Sie eine Version aus dem Internet auf der Host installiert haben, dann finden Sie die Datei im Verzeichnis ***/usr/local/pgsql/bin***). Zusätzlich können Sie die Meldungen beim Start auch noch in eine Datei umleiten. Der Daemon pg_ctl startet die Datenbank. Wenn Sie vor dem Datenverzeichnis noch die Option ***-l logfile*** hinzufügen, dann werden alle Meldungen der Datenbank in der Logfile mit protokolliert. Damit auch weitere Anwender über TCP/IP auf die Host zugreifen können, ändern Sie in der Datei /var/lib/pgsql/data/postgresql.conf den Eintrag ***#tcpip_socket = false*** auf:

tcpip_socket = true

ANMERKUNG: Sie können die Datenbank entweder über das Programm ***postmaster*** starten, oder mit ***pg_ctl*** (siehe auch unter dem Kapitel Weitere Dienstprogramme). Sie können auch die Befehle in die Datei ***/etc/rc.d/rc.local*** einbauen. Auch die Datei ***contrib/start-scripts/linux*** (befindet sich im Verzeichnis der Version aus dem Internet, nach dem Entpacken der Datein). Besonders wichtig ist die Einstellung bezüglich des TCP/IP-Zugriffs. Übergeben Sie die Information entweder per Parameter dem Programm postmaster, oder vermerken Sie es in der Datei postgresql.conf. Das System hat für Sie schon eine erste Datenbank (Name ***template1***) angelegt. Sie dient Ihnen später als Kopiervorlage für weitere Datenbanken.

Wenn Sie bei der Red Hat/Fedora Distribution die Software von der CD installieren, können Sie die Datenbank starten. Rufen Sie das Programm ***Dienste*** auf (***Start-Taste, Systemeinstellungen, Servereinstellungen***). Wie auch bei der SuSE-Distribution finden Sie die entsprechende Start-Datei im Verzeichnis /etc/rc.d/rc5.d (S85postgresql). Über das Programm ***Dienste*** können Sie den Programmstart veranlassen.

Abbildung 24 Dienst-Konfiguration

Automatisches Login

Legen Sie im Home-Verzeichnis des Users die Datei ***.pgpass*** an. In ihr tragen Sie dann den User-Namen und die Passwörter ein, die dann als default-Wert für den Zugriff auf die Datenbank verwendet werden. Das Format der Einträge sieht dann so aus:

hostname:port:datenbank:benutzername:passwort

Anstelle eines Feldes könne Sie auch das Wildcartzeichen * einsetzen. Den Einträgen mit **:** und **** stellen Sie das Zeichen \ voran. Achten Sie auch auf die passenden Zugriffsrechte (mit dem Befehl ***chmod 0600 .pgpass*** ändern). Sind die Zugriffsrechte weniger strikt, dann wird die Datei ignoriert.

Die Datenbank beenden

Wenn Sie den Rechner herunterfahren oder neu starten, dann sollten Sie vorher die Datenbank beenden. Wie auch beim Start der Datenbank, gibt es eine Shell zum beenden der Datenbank (Den Prozess mit dem kill-Befehl zu beenden, kann zu Problemen führen, die Prozess ID finden Sie in der Datei /var/lib/pgsql/data/postmaster.pid, oder über den Befehl ps -A). Am besten Sie verwenden hier die die Shell pg_ctl im Verzeichnis /usr/bin (sehen Sie sich die Datei interessehalber einmal im Editor an, die Datei kann auch in einem anderen Verzeichnis liegen).

Der Syntax ist wie folgt aufgebaut:

pg_ctl stop [-W] [-D DATADIR] [-s] [-m SHUTDOWN-MODE]

Parameter	Inhalt
-W	Optional - Wartet nicht bis alle Operationen ausgeführt sind
-D	Optional - Das Datenverzeichnis ANMERKUNG: Im Skript ist es als Optional angegeben, es ist aber besser man gibt es an.
-s	Optional - gibt nur Fehler aus, keine zusätzlichen Informationen
-m	Optional - Der Shutdown-Modus (siehe auch weitere Tabelle)

SHUTDOWN -MODE	Inhalt
smart	Ende, nachdem sich alle Clientes abgemeldet haben. Neue Verbindungen lässt der Postmaster nicht mehr zu.
fast	Alle Transaktionen werden noch ausgeführt, danach trennt das Programm alle Verbindungen.
immediate	Beendet den Postmaster ohne Rücksicht auf Verluste.

Auf dem nächsten Screenshot sehen Sie die Meldungen beim Beenden der Datenbank.

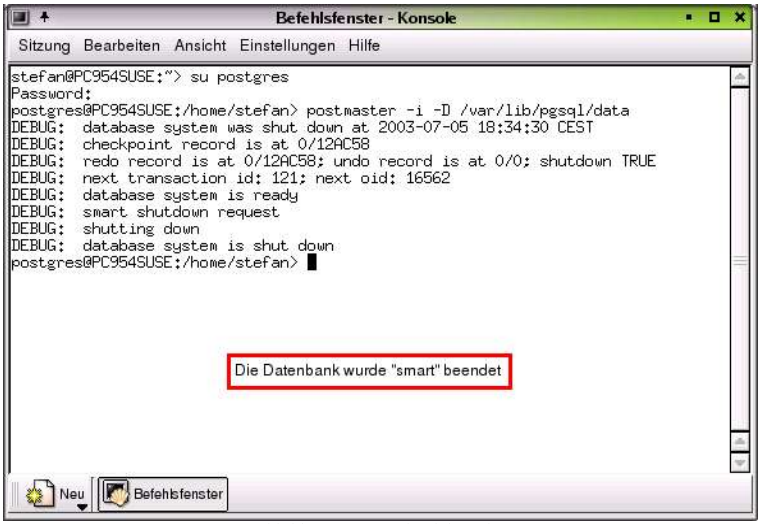


Abbildung 25 Datenbank beenden

WICHTIG: Damit die Datenbank auf jeden Fall beendet wird, sollten Sie ein Shell-Skript in das Verzeichnis /etc/rc.d/rc0.d und /etc/rc.d/rc6.d legen (am besten ein Link). Es gelten dieselben Bedingungen wie beim Start. Der Befehl sieht dann z.B. so aus:

su -c '/usr/bin/pg_ctl stop -D /var/lib/pgsql/data -m smart' postgres

oder bei der Installation aus dem Internet

su -c '/usr/local/pgsql/bin/pg_ctl stop -D /var/lib/pgsql/data -m smart' postgres

Die Konfigurationsdatei pg_hba.conf

Ohne diese Datei kann kein anderer LINUX-Host, bzw. Windows-PC auf Ihre Datenbank zugreifen. Sie enthält alle wichtigen Informationen. Diese Datei befindet sich im Verzeichnis PGDATA (z.B. /var/lib/pgsql/data). Folgende Punkte regelt die Konfigurationsdatei:

1.) Welcher Host (bzw. welches Netzwerk) darf sich anmelden
2.) Welche Zugriffsberechtigung hat jeder Host
3.) Welcher Host darf auf welche Datenbank zugreifen

In dieser Steuerdatei werden die Kommentarzeilen mit einem #-Zeichen an der ersten Stelle gekennzeichnet. Leere Zeilen ignoriert das Programm. In einer Konfigurationseinstellung trennen Sie Parameter durch mehrere Leerzeilen oder dem Tab-Zeichen. Jede Einstellung enthält eine IP-Adresse, die Datenbank und den Zugriffsberechtigungstyp. 3 unterschiedliche Möglichkeiten haben Sie (eine Erklärung der Informationen [DBNAME finden Sie in einer weiteren Tabelle):

Einstellung	Inhalt
host	Hier geben Sie die IP-Adressen der Host oder des Netzwerkes ein (im Beispiel unten das Feld host), die sich auf der Datenbank anmelden dürfen. Das Format sieht so aus: [host] [DBNAME] [IP_ADDRESS] [ADDRESS_MASK] [AUTH_TYPE] Die Information ADDRESS_MASK kann auch Platzhalter enthalten. Möchten Sie zum Beispiel allen IP-Adressen aus dem Bereich 192.168.x.x den Zugriff auf die Datenbank erlauben, so geben Sie die ADDRESS_MASK 255.255.0.0 vor. Die Zahl Null steht hier als Platzhalter, an dieser Stelle wird die IP-Adresse nicht geprüft! In neueren Versionen können Sie anstelle der IP-Adresse und der Adressmaske auch nur die CIDR-ADDRESSE eingeben. Die Zahl nach dem Schrägstrich steht für die Anzahl von Bits (von rechts nach links) in der IP-Adresse, die für Host reserviert sind (z.B. 127.0.0.1/32 für eine komplette Adresse und 60.0.0.0/24 für einen Host in einem A-Netzwerk. **WICHTIG:** Sie dürfen hier nur IP-Adressen eingeben und keine Host-Namen!

Einstellung	Inhalt
hostssl	Der Eintrag ist mit der Einstellung host identisch. Die Verbindung wird aber nur über eine sichere SSL IP- Connection aufgenommen. Mit der Einstellung host sind neben den „unsicheren“ auch die sicheren Verbindungen möglich.
local	Über diese Einstellung sind Verbindungen nur über den local UNIX domain socket möglich.

Erklärung:

Feld	Inhalt
DBNAME	Es enthält den Namen einer Datenbank. Die Information ***all*** steht für alle Datenbanken. Mit ***sameuser*** gestatten Sie den Zugriff auf die Datenbank, die denselben Namen wie der Anwender hat, der sich zurzeit auf der Datenbank anmeldet.
AUTH_TYPE	Die Möglichkeiten sehen so aus: - **trust** => jeder gültige Benutzername wird akzeptiert (**Anmerkung:** Gefährlich, Anwender anderer Hosts können sich ohne Passwort als User postgres auf der Datenbank anmelden!) - **password** => hier gibt der Anwender ein Passwort ein. Der Anwender muss nicht unbedingt als gültiger Benutzer der Host angelegt sein. Wenn das Argument nicht verwendet wird, dann sieht das Programm in der Tabelle pg_shadow nach. - **md5** => ist wie password aufgebaut, der Anwendername ist verschlüsselt (notwendig für Clients deren Versions-Nr. unter der Version 7.2 liegt). - **ident** => hier meldet sich der Anwender unter seinem Betriebssystemnamen auf der Datenbank an. Auch bei diesem Verfahren ist Vorsicht geboten. Auf dem Server müssen Sie den Dämon identd laufen lassen. - **krb4** => Kerberos V4 Zugriffsberechtigung, bei TCP/IP-Verbindungen, funktioniert nicht bei local UNIX-domain sockets. - **krb5** => Kerberos V5 Zugriffsberechtigung (siehe auch krb4) - **reject** => Diese Einstellung bedeutet, dass diese IP-Adresse zurückgewiesen wird
AUTH_ ARGUMENT	Ist teilweise noch ein Benutzername, der noch mit übergeben wird (oder bei AUTH_TYPE=ident sameuser.)

ANMERKUNG: AUTH_ARGUMENT enthält nur aus drucktechnischen Gründen eine Zeilenschaltung

Einige Beispiele:

```
# TYP DATENBANK BENUTZER IP-ADRESSE IP_MASKE        METHODE
local all       all                               trust
```

Jeder lokale Benutzer kann sich auf dem lokalen System (über Unix-Domain-Sockets) auf jeder Datenbank unter jedem Namen anmelden.

```
# TYP DATENBANK BENUTZER IP-ADRESSE IP_MASKE        METHODE
host  versuch   all      60.1.2.0   255.255.255.0   trust
```

Anmeldung auf der Datenbank ***versuch*** von jeder Host aus dem Netz 60.1.2.x. Der Benutzername ist der Anmeldename vom Betriebssystem.

```
# TYP DATENBANK BENUTZER IP-ADRESSE IP_MASKE        METHODE
host  versuch   stefan   60.1.2.3   255.255.255.255 md5
```

Der Benutzer Stefan meldet sich von der Host 60.1.2.3 auf der Datenbank Versuch an, wenn er das richtige Passwort hat.

```
# TYP DATENBANK BENUTZER IP-ADRESSE IP_MASKE        METHODE
host  versuch   all      60.0.0.0   255.0.0.0       ident sameuser
```

Alle Anwender aus dem Netz 60.x.x.x können sich auf der Datenbank ***versuch*** anmelden. Der Anmeldename ist der Name den Ident für die Verbindung ermittelt hat.

```
# TYP DATENBANK BENUTZER IP-ADRESSE IP_MASKE        METHODE
host  versuch   all      60.0.0.0   255.0.0.0       ident abc
```

Alle Anwender aus dem Netz 60.x.x.x können sich auf der Datenbank ***versuch*** anmelden, wenn sie die Ident-Prüfung bestehen. Näheres entnehmen Sie der Beschreibung der Datei pg_ident.conf

TIPP: Um bei Änderungen im laufenden Betrieb die Datenbank nicht herunter- und wieder hochzufahren, geben Sie den Befehl
pg_ctl reload [-D DATADIR] ein.

Die Konfigurationsdatei postgres.conf

Auch in dieser Datei befinden sich wichtige Informationen für die Datenbank. Sie legen dort z.B. Verbindungsinformationen (Port, maximale Anzahl von Usern), Verzeichnis von weiteren Konfigurationsdateien und internen Einstellungen (shared Memory usw.) fest. Bei der Windows-Version bearbeiten Sie diese Zeile, damit die anderen PCs auf die Datenbank zugreifen können:

```
#listen_addresses='localhost'
```

Entfernen Sie das Kommentarzeichen und ersetzen Sie localhost gegen einen Stern. Dadurch kann jeder auf die Datenbank zugreifen (siehe unten).

```
listen_addresses='*'
```

Weitere wichtige Punkte sind zum Beispiel die Port-Nr (Standardwert: port = 5432) und maximale Anzahl von Benutzern (Standardwert: max_connections = 100).

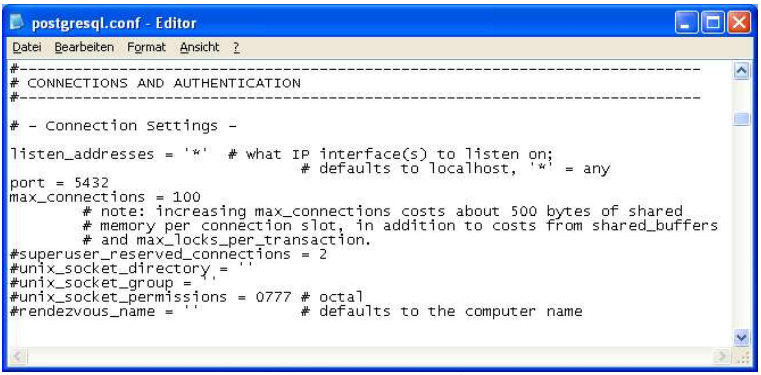


Abbildung 26 postgresql.conf - Inhalt

Die Konfigurationsdatei pg_ident.conf

Diese Datei ist für das Ident-Authentifizierungssystem wichtig. Hier prüft die Datenbank den Benutzernamen des Betriebssystems und ermittelt dann die erlaubten Datenbankbenutzernamen. In dieser Datei tragen Sie neben den Anmeldenamen auch „Map-Namen“. In der Konfigurationsdatei pg_hba.conf tragen Sie zu diesen „Map-Namen“ die Datenbanken ein (***Hinweis:*** Der Dämon identd muss auf dem System laufen). Ein Beispiel:

Datei pg_hba.conf

```
# TYP  DATENBANK  BENUTZER  IP-ADRESSE  IP_MASKE     METHODE
host   versuch    all       60.0.0.0    255.0.0.0    ident abc
```

Zu dem „Map-Namen“ abc finden Sie in der Datei pg_ident.conf die entsprechenden Benutzer. An der Datenbank können sich alle Leute anmelden, die unter dem Begriff abc eingetragen sind.

Datei pg_ident.conf

Unter dem Eintrag Ident-Benutzer führen Sie die Anwender mit ihren UNIX-Namen. Unter der Rubrik PG-Benutzer finden Sie die erlaubten Benutzernamen der UNIX-User für die Datenbank. Für einen Unix-Benutzer sind auch mehrere PG-Benutzernamen möglich.

```
# MAPNAME      IDENT-BENUTZER        PG-BENUTZER
abc            stefank               stefan
abc            janina                janina
```

Das Programm postmaster

Das Programm ist das Herz der Datenbank. Läuft es nicht, so kann keiner auf die Daten zugreifen. Das Programm startet einen separaten Serverprozess und überwacht die Verbindung. Weiterhin überwacht es auch die Kommunikation zwischen den Serverprozessen. In der Grundeinstellung läuft das Programm im Vordergrund und gibt verschiedene Meldungen in die Standard-Ausgabe aus. In der Praxis läuft das Programm aber im Hintergrund. Ein gestartetes Programm Postmaster verwaltet nur einen Datenbankcluster. Sie können aber mehrere Postmaster-Programme gleichzeitig laufen lassen. Das Programm können Sie entweder als User ***postgres*** starten, oder auch als ein anderer PostgreSQL-Benutzer.

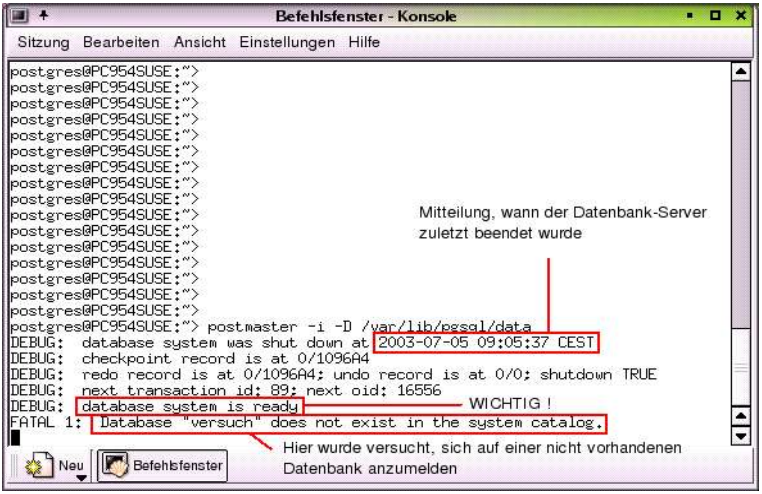


Abbildung 27 Postmaster

Die Parameter zum Programm postmaster

Parameter	Pflicht	Inhalt
-A 0|1	nein	Hilfe zur Fehlersuche, ist nur verfügbar, wenn es bei Compilerlauf eingestellt wurde. In der Grundeinstellung ist es an.
-B [NBUFFERS]	nein	Enthält die Anzahl der shared Buffers. Die Grundeinstellung ist 64
-c [NAME]=[VALUE]	nein	Setzen eines Runtime-Parameters
-d [1-5]	nein	Die verschiedenen Debug Levels. Sie können Werte von 1 bis 5 einstellen. Je höher der Debug-Level ist, umso umfangreichere Meldungen gibt das Programm aus.
-D [DATADIR]	ja	Dieser Parameter enthält das Database Verzeichnis.
-F	nein	Den fsync ausschalten
-h [HOSTNAME]	nein	Der Hostname oder die IP-Adresse auf der die Datenbank läuft
-i	nein	Er ermöglicht eine TCP/IP Verbindung (wichtig bei ODBC!) auf die Datenbank. Wenn mehrere Anwender auf die Datenbank über ODBC zugreifen, ist der Parameter Pflicht.
-k [DIRECTORY]	nein	
-l	nein	Er ermöglicht SSL Verbindungen auf die Datenbank
-N [max. Verb.]	nein	Er steht für die maximale Anzahl von erlaubten Verbindungen (default 32)
-o [OPTIONS]	nein	Er gibt OPTIONEN an den Datenbank-Server weiter
-p [PORT]	nein	Der offenen Port für die Verbindungsaufnahme mit der Datenbank. Per Default ist der Port 5432 vorgesehen.
-S	nein	Startet die Datenbank ohne Ausgabe von Meldungen. Das Programm läuft dann im Hintergrund

Zusätzlich gibt es noch 2 Parameter für die Entwickler der Datenbank.

Parameter	Pflicht	Inhalt
-n	nein	Das shared Memory wird nach einem Absturz nicht reinitialisiert.
-s	nein	Sendet das Signal SIGSTOP an alle Backend-Server, wenn einer abstürzt

Umgebungsvariablen

PGCLIENTENCODING
PGDATA (und wenn weitere vergeben wurden, z.B. PGDATA2)
PGDATASTYLE
PGPORT
TZ

Arbeiten mit der Datenbank

Datenbanken anlegen

Es gibt zwei verschiedene Möglichkeiten eine Datenbank anzulegen (von Haus aus haben Sie die Datenbank template1):

1. Mit Hilfe eines SQL-Befehles (siehe auch Anhang (CREATE DATABASE Datenbankname OWNER Eigentümer)
2. Über das Programm createdb. Die genaue Syntax sieht wie folgt aus: ***createdb [-O Owner] Datenbankname***
   Die Option -O steht für die Zuweisung des neuen Eigentümers (siehe Bild unten).

Abbildung 28 Datenbanken anlegen - createdb

Das Programm createdb

Wie schon oben aufgeführt, legen Sie mit diesem Programm neue Datenbanken an. Die Syntax sieht so aus:

createdb [OPTIONEN] [DBNAME] [DESCRIPTION]

Ihnen stehen folgende Optionen zur Verfügung (***ANMERKUNG:*** Die Kurzform der Optionen steht in eckigen Klammern. Wenn Sie mehrere Optionen einsetzen möchten, dann trennen Sie die Optionen durch ein Leerzeichen).

Option	***Inhalt***
`--location=PATH [-D]`	Das alternative Verzeichnis für die Datenbank
`--encoding=ENCODING [-E]`	Der Zeichensatz der Datenbank (Stichwort Umlaute)

Option	*Inhalt*
`--owner=OWNER [-O] (großes O)`	Der Eigentürmer der Datenbank.
`--template=TEMPLATE [-T]`	Die Template-Datenbank zum Kopieren.
`--echo [-e]`	Hier sehen Sie die SQL-Befehle, mit denen die Datenbank angelegt wird
`--quiet [-q]`	Es wird keine Meldung ausgegeben
`--help`	Die Hilfe für das Programm

Auch für die Verbindung zu anderen Datenbankservern gibt es noch Optionen:

Option	*Inhalt*
`--host=HOSTNAME [-h]`	Der Name der Host
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Abbildung 29 createdb - ein Beispiel

Sollten Sie eventuell Probleme mit dem Plattenplatz haben, so können Sie die neue Datenbank auch in einem anderen Verzeichnis ablegen. Gehen Sie dabei wie unten gezeigt vor:

Schritt	Inhalt
1	Weisen Sie der Variablen PGDATA2 ein Verzeichnis zu (z.B. ***PGDATA2=/opt/datenbank/daten***). Halten Sie diese Information auch in anderen Dateien fest (z.B. /etc/profile oder im Start-Skript). Den Namen der Variablen können Sie übrigens Beliebig vergeben. Wenn PGDATA2 schon benutzt wird, dann vergeben Sie die Variable PGDATA3.
2	Exportieren Sie die Information mit: ***export PGDATA2***
3	Danach richten Sie den Speicherplatz mit dem Befehl ***initlocation PGDATA2*** ein (kein $ vor dem PGDATA2)
4	Führen Sie einen Restart des Datenbank-Servers durch.
5	Danach legen Sie die Datenbank an. Der Befehl sieht so aus: ***CREATE DATABASE [name] WITH LOCATION = 'PGDATA2'*** Das Programm ***createdb*** hat für diesen Fall die Option ***-D***.

Datenbanken löschen

Wie auch beim Anlegen einer Datenbank haben Sie zwei Möglichkeiten dazu:

1. Mit Hilfe eines SQL-Befehles (siehe auch Anhang (DROP DATABASE Datenbankname))
2. Über das Programm dropdb. Die Syntax lautet so: ***dropdb Datenbankname***

Backup der Datenbank

Eine Sicherung der Datenbank gehört zu den elementarsten Aufgaben des Systemverwalters. Dazu haben Sie die Programme ***pg_dump*** und ***pg_dumpall***. Beide rufen Sie über die Konsole auf. Für einzelne Tabellen können Sie auch den SQL-Befehl copy einsetzen.

Das Programm pg_dump

Mit dem Programm sichern Sie eine Ihrer Datenbanken. Die Ausgabe erfolgt in eine Skriptdatei (enthält Text und SQL-Befehle um die Datenbank so wiederherzustellen, wie sie zum Zeitpunkt der Sicherung war) oder eine andere Archivdatei. Das Programm blockiert keine anderen Benutzer und erstellt konsistente Sicherungsdateien. Während der Sicherung sollten Sie unbedingt die Meldungen lesen. Geben Sie bei dem Befehl keinen Datenbanknamen an, dann nimmt das Programm den Wert aus der Umgebungsvariable PGDATABASE. Ist auch der Wert leer, dann nimmt das Programm die aktuelle Datenbank.

Aufbau:

pg_dump [OPTIONEN] Datenbank

Ihnen stehen folgende Optionen zur Verfügung (***ANMERKUNG:*** Die Kurzform der Optionen steht in eckigen Klammern. Wenn Sie mehrere Optionen einsetzen möchten, dann trennen Sie die Optionen durch ein Leerzeichen).

Option	***Bedeutung***
`--data-only` `[-a]`	Sichert nur die Daten, nicht das Schema (die Datendefinitionen)!
`--blobs` `[-b]`	Die Lage Objekts werden auch gesichert
`--clean` `[-c] (kleines c)`	Zuerst werden die Datenbankobjekte gelöscht und dann erzeugt. Diese Option hilft Ihnen bei reinem Textformat. Bei den anderen Formaten können Sie diese Option angeben, wenn Sie pg_restore ausführen.
`--create` `[-C] (großes C)`	Als Erstes legt das Programm die Datenbank an und meldet sich dann bei ihr an. Diese Möglichkeit hilft Ihnen nur bei der Sicherung von Textdateien. Bei den anderen Formaten können Sie diese Option angeben, wenn Sie pg_restore ausführen.
`--inserts` `[-d]`	Hier sichern Sie die Daten als INSERT-Befehle und nicht als COPY. Leider läuft mit dieser Einstellung die Wiederherstellung langsamer. Das Archiv lässt sich aber besser auf andere Datenbankprodukte portieren.

Option	*Bedeutung*

`--column-inserts`
`--attribute-inserts`
`[-D]`
	Die Ausgabe erfolgt in Form von INSERT-Befehlen mit vorgegebenen Spaltennamen (INSERT INTO Tabelle (Spalte, ...) VALUES ...). Leider wirkt sich das negativ auf die Geschwindigkeit des Restores aus.

`--file` `[-f]`	Die Ausgabe erfolgt in eine vorgegebene Datei. Ohne diese Angabe wird die Standardausgabe verwendet.
`--format` `[-F]`	Hier können Sie das Ausgabeformat der Sicherung wählen: p = die Daten als Text und SQL-Anweisungen für die Struktur t = Ausgabe als ein tar-Archiv. Kann später als Eingabedatei für pg_restore verwendet werden. Schemaelemente können umgeordnet oder auch ausgelassen werden. Sie können auch eingrenzen, welche Daten bei der Wiederherstellung geladen werden sollen. c = Erzeugt ein „Custom“ Archiv, kann als Eingabe für pg_restore verwendet werden. Diese Möglichkeit ist am flexibelsten, da man Daten und Schemaelemente umsortieren kann. Außerdem ist in der Grundeinstellung eine Komprimierung der Informationen vorgesehen.
`--ignore-version` `[-i]`	Die Version zwischen dem Programm und dem Datenbankserver wird ignoriert (wichtig: bei früheren Versionen). ACHTUNG: Wenn es nicht funktioniert, dann sagen Sie nicht, Sie seien nicht gewarnt worden.
`-n namespace`	
`--oids` `[-o]`	Gibt auch den Objekt Identifier jeder Tabelle (OIDs) aus. Das ist wichtig, wenn Ihre Anwendung diese Informationen benötigt (siehe Fremdschlüssel, Constraints).
`--no-owner` `[-O] (großes O)`	Gibt keine Befehle aus, die dafür sorgen, dass die Objekteigentümer mit der ursprünglichen Datenbank übereinstimmen. Diese Option unterbindet aber nicht alle Verbindungsversuche mit der Datenbank, sondern nur solche, die dazu dienen die Eigentumsverhältnisse wiederherzustellen. Diese Option hilft Ihnen nur beim reinen Textformat. Bei den anderen Formaten können Sie diese Optionen angeben, wenn Sie pg_restore ausführen.

Option	*Bedeutung*
`--no-reconnet [-R]`	überholt (Version 7.4.6)
`--schema`	Sichert nur den Inhalt des Schemas
`--schema-only [-s]`	Sichert nur das Schema (Definitionen), nicht die Daten.
`-- superuser=Username [-S Username]`	Eingabe des Superusernamens um den Trigger abzuschalten. Das ist nur dann wichtig, wenn –disable-triggers eingesetzt wird.
`--table=Tabelle [-t]`	Sichert nur eine einzelne Tabelle. Haben Sie in verschiedenen Schemas Tabellen mit dem gleichen Namen, dann werden alle gesichert. Möchten Sie nur eine bestimmte Tabelle sichern, dann nehmen Sie die Parameter –schema und –table. ACHTUNG: Zurzeit wird für diese Option keine Garantie übernommen.
`--verbose [-v]`	Die Fortschrittsmeldungen werden in die Standardfehlerausgabe geschrieben.
`--no-privileges --no-acl [-x]`	Verhindert die Sicherung der Zugriffsberechtigungen (siehe grant/revoke Anweisungen).
`-X use-set-session-authorization`	überholt in der Version 7.4.6
`--disable-triggers [-X disable-triggers]`	Diese Option ist dann relevant, wenn Sie nur die Daten sichern. Sollten Sie die Daten wieder zurück laden, dann greifen die Prüfungen und Trigger nicht. Diese Option müssen Sie als Superuser ausführen.
`--compress=0..9 [-Z]`	Hier stellen Sie die Stufe der Komprimierung ein. Zurzeit unterstützt nur das „Custom“ Archivformat diese Option.
`--host=HOSTNAME [-h]`	Der Name der Host
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Umgebungsvariablen:

PGDATABASE
PGHOST
PGPORT
PGUSER

Beispiele:

Eine einfach Sicherung in eine Datei:

pg_dump [Datenbank] > [der Dateiname, in die Sie die Sicherung umleiten]

Große Datenbanken können Sie auch komprimieren. Leiten Sie die Ausgabe an das Programm gzip weiter. In der Praxis sieht es so aus:

pg_dump [Datenbank] | gzip > [Name der Datei.gz]

Wenn Sie alle Datenbanken sichern möchten, dann nehmen Sie das Programm ***pg_dumpall***.

Das Programm pg_dumpall

Mit diesem Programm sichern Sie einen kompletten Datenbankcluster. Im Falle eines Restores nehmen Sie das Programm psql. Im Gegensatz zum Programm pg_dump sichern Sie hier auch globale Objekte! Leider gibt es hier auch eine Einschränkung: „large Objects“ werden leider nicht gesichert (Stand Version 7.4.6). Setzen Sie hier das Programm pg_dump ein. Das Programm meldet sich mehrere Male am Datenbankserver an. Damit Sie das Passwort nicht jedesmal eingeben müssen, hinterlegen Sie es in der Datei ***$HOME/.pgass***.

Aufbau:

pg_dumpall [OPTIONEN]

Ihnen stehen folgende Optionen zur Verfügung (***ANMERKUNG:*** Die Kurzform der Optionen steht in eckigen Klammern. Wenn Sie mehrere Optionen einsetzen möchten, dann trennen Sie die Optionen durch ein Leerzeichen).

Option	***Bedeutung***
`--data-only [-a]`	Sichert nur die Daten, nicht das Schema (die Datendefinitionen)!
`--clean [-c] (kleines c)`	Zuerst werden die Datenbankobjekte gelöscht und dann erzeugt. Diese Option hilft Ihnen bei reinem Textformat. Bei den anderen Formaten können Sie diese Option angeben, wenn Sie pg_restore ausführen.
`--inserts [-d]`	Hier sichern Sie die Daten als INSERT-Befehle und nicht als COPY. Leider läuft mit dieser Einstellung die Wiederherstellung langsamer. Das Archiv lässt sich aber besser auf andere Datenbankprodukte portieren.
`--column-inserts --attribute-inserts [-D]`	Die Ausgabe erfolgt in Form von INSERT-Befehlen mit vorgegebenen Spaltennamen (INSERT INTO Tabelle (Spalte, ...) VALUES ...). Leider wirkt sich das negativ aus die Geschwindigkeit des Restores aus.
`--globals-only [-g]`	Nur die globalen Objekte werden gesichert (Anwender und Gruppen), keine Datenbanken.

Option	*Bedeutung*
`--ignore-version [-i]`	Die Version zwischen dem Programm und dem Datenbankserver wird ignoriert (wichtig: bei frühreren Versionen). ACHTUNG: Wenn es nicht funktioniert, dann sagen Sie nicht Sie seien nicht gewarnt worden).
`--oids [-o]`	Gibt auch den Objekt Identifier jeder Tablle (OIDs) aus. Das ist wichtig, wenn Ihre Anwendung diese Informationen benötigt (siehe Fremdschlüssel, Constraints).
`--schema-only [-s]`	Sichert nur das Schema und keine Daten.
`--verbose [-v]`	Die Fortschrittsmeldungen werden in die Standardfehlerausgabe geschrieben.
`--no-privileges --no-acl [-x]`	Verhindert die Sicherung der Zugriffsberechtigungen (siehe grant/revoke Anweisungen).
`--host=HOSTNAME [-h]`	Der Name der Host
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Umgebungsvariablen:

PGHOST
PGPORT
PGUSER

Beispiel:

alle Datenbanken sichern:

pg_dumpall > sicher.out

Um diese Datenbank wiederherzustellen (die Datenbank, an der Sie sich

gerade angemeldet haben, ist unerheblich, die Skriptdatei enthält alle Befehle, um die gesicherte Datenbank zu erzeugen und sich dann mit ihr zu verbinden):

psql -f sicher.out template1

Restore der Datenbank

Für die Wiederherstellung stehen Ihnen 2 Möglichkeiten zur Verfügung:

1. Das Programm psql
2. Das Programm pg_restore

Mit dem Programm psql

Mit dem psql-Befehl können Sie Ihre Daten aus der Sicherungskopie wieder in die Datenbank zurück spielen. Die Syntax siehe wie folgt aus:

psql -d [Datenbank] < [der Dateiname Ihrer Sicherungsdatei]

Bitte beachten Sie, dass das psql-Kommando keine Datenbank neu anlegt! Komprimierte Dateien stellen Sie so wieder her (ggf. muss die Datenbank vorher noch angelegt werden):

cat [Name der Sicherungsdatei.gz] | gunzip | psql [Name der Datenbank]

Anmerkung: Über den Weg Backup und Restore können Sie auch Daten von einem Server zu einem anderen übertragen. Der Befehl sieht dann so aus:

pg_dump -h [Host1] [Datenbank] | psql -h [Host2] [Datenbank]

Das Programm pg_restore

Mit diesem Programm stellen Sie eine Datenbank wieder her. Die Ausgangsdatei muss vom Programm pg_dump erstellt worden sein. Wenn Sie keinen Dateinamen angeben, dann nimmt das Programm die Standardeingabe.

Aufbau:

pg_restore [OPTIONEN] [Dateiname]

Ihnen stehen folgende Optionen zur Verfügung (***ANMERKUNG:*** Die Kurzform der Optionen steht in eckigen Klammern. Wenn Sie mehrere Optionen einsetzen möchten, dann trennen Sie die Optionen durch ein Leerzeichen).

Option	***Bedeutung***
`--data-only [-a]`	Sichert nur die Daten, nicht das Schema zurück!
`--clean [-c] (kleines c)`	Zuerst werden die Datenbankobjekte gelöscht und dann erzeugt.
`--create [-C]`	Legt die Datenbank an, bevor sie zurück kopiert wird. (Wenn Sie diese Option verwenden, dann legt das Programm die in der Option -d übergebene Datenbank nur an und kopiert die Daten aus in Archiv in die Datenbank).
`--inserts [-d]`	Das Programm verbindet sich mit der Datenbank und überträgt die Daten in diese Datenbank.
`--file=dateiname [-f]`	Name der Ausgangsdatei
`--format=Format [-F]`	Das Format der Archivdatei t=Tar-Archiv c=“Custom“ Format, ggf. komprimiert, Daten und Schemaelemente können umsortiert werden.
`--ignore-version [-i]`	Die Versionsprüfung wird ignoriert
`--index=Index [-I] (großes i)`	Der in der Definition benannte Index wird wieder hergestellt.

Option	*Bedeutung*
`--list [-l] (kleines L)`	Gibt den Inhalt des Archivs wieder.
`--use-list=Datei [-L Datei]`	Nur die in der Datei aufgeführten Objekte werden wiederhergestellt. Durch ein Semikolon am Anfang können Sie Zeilen auskommentieren.
`--orig-order [-N]`	Das Programm stellt die Objekte in der Reihenfolge her, in der man sie angelegt hat.
`--host=HOSTNAME [-h]`	Der Name der Host
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Plattenplatz optimieren

Bei den Update- oder Delete-Befehlen werden unter Umständen nicht sofort die alten Daten entfernt. Mit dem SQL-Befehl VACUUM bereinigen Sie den Plattenplatz. Sie können Ihn entweder gezielt bei einer Tabelle anwenden, oder auch auf die ganze Datenbank. Seit der Version 7.2 können Sie den VACUUM-Befehl neben den normalen Operationen wie Select, Insert, Update und Delete einsetzen. Wenn Sie etwas an einer Tabellenstruktur ändern, darf er nicht laufen! Setzen Sie ihn am Besten in den Zeiten geringer Aktivitäten ein. Verwenden Sie zur Komprimierung nur den VACUUM-Befehl, nicht den VACUUM FULL-Befehl. Den VACUUM FULL-Befehl benötigen Sie, wenn Sie die meisten Einträge in einer Tabelle gelöscht haben. Überprüfen Sie auch immer den Verbrauch an Plattenplatz. (z.B. mit dem Befehl ***contrib/dbsize*** aus dem Programm psql). Siehe auch Programm Vaccumdb.

Routine Reindexing

Die Datenbank kann leider die B-tree Indexseiten nicht richtig wieder verwenden. Deshalb müssen Sie von Zeit zu Zeit die Seiten wieder neu aufbauen. Diese Aktionen erledigen Sie mit dem Befehl ***REINDEX***. **ACHTUNG:** Wenn Sie den Befehl bei Systemtabellen und -Indexen anwenden möchten, dann starten Sie das Postgres im Single-Use Mode (siehe Programm postgres)!

Migration der Datenbank / Einführung einer neuen Version

Bei einem größeren Versionswechsel müssen Sie die Datenbank neu aufbauen. Als Faustformel gilt hierbei: Ändert sich die Zahl nach dem ersten Punkt in der Versionsnummer, so ist die Datenbank mitbetroffen (z.B. von 7.3 auf 7.4). Bei einem Versionswechsel nach dem zweiten Punkt tauschen Sie nur die Programme aus (z.B. von 7.3.3 auf 7.3.4).

Bei einem Releasewechsel gehen Sie mit der Installation der Programme so vor, wie im Kapitel **Installation einer neuen Version aus dem Internet** beschrieben (ändern Sie aber das Zielverzeichnis ab, nicht unbedingt /usr/local/pgsql). Danach pflegen Sie noch die entsprechenden Variablen. Sie können übrigens auch ohne Probleme zwei unterschiedliche Versionen auf einem Host laufen lassen. Sie müssen nur mit verschiedenen Portnummern arbeiten.

Wenn Sie zwei Versionen auf einem Host laufen lassen, dann können Sie die Datenbank in einem Zug umstellen:

pg_dumpall -p 5432 | psql -d template1 -p 6543

Eine weitere Möglichkeit der Migration der Daten sieht so aus:

Schritt	Aktion
1	Wechseln Sie in das Home-Verzeichnis vom User postgres (z.B. / var/lib/pgsql)
2	Führen Sie eine Datensicherung durch: ***pg_dumpall > backup***
3	Fahren Sie danach die Datenbank herunter: ***pg_ctl stop -D / var/lib/pgsql/data***
4	Jetzt sollten Sie sich nochmals vergewissern, dass Sie bei den nächsten Schritten nicht versehentlich die alten Programmversionen aufrufen.
5	Legen Sie ein neues Data-Verzeichnis an: z.B. /var/lib/pgsql/data740
6	Mit dem Befehl ***initdb -D /var/lib/pgsql/data740*** richten Sie nun die neue Datenbank ein.
7	Bearbeiten Sie die Einstellungen in den Konfigurationsdateien (pg_hba.conf und postgresql.conf).
8	Starten Sie nun die Datenbank (z.B. mit dem Programm ***postmaster -D /var/lib/pgsql/data740***)

Schritt	Aktion
9	Im letzten Schritt spielen Sie das Backup vom Anfang wieder ein: ***psql template1 < backup***

Write-Ahead Logging

Die Datenbank Postgres zeichnet mit der Standardtechnik Write-Ahead Logging (WAL) die Transaktionen auf. Das Prinzip bedeutet, dass die Änderungen in die Datendateien (die die Tabellen und Indexe enthalten) erst geschrieben werden dürfen, wenn die Änderungen in dem Log aufgezeichnet wurden und die Logeinträge sich auf einem Speichermedium befinden. Dadurch schreibt man jede Datenseite nicht nach jeder Transaktion auf die Festplatte zurück, denn im Falle eines Absturzes stellt man die Datenbank mit Hilfe des Log wieder her (Fachbegriff Vorwärts-Wiederherstellung, auch REDO genannt). Die noch nicht abgeschlossenen Transaktionen werden aus den Datenseiten entfernt (Rückwärts-Wiederherstellung, UNDO, zurzeit noch nicht implementiert). In der Praxis führt das Verfahren zu einer erheblichen Reduzierung der Schreibvorgänge auf der Festplatte. Seit der Version 7.1 hat Postgres dieses Verfahren. Von Seiten des Administrators ist keine Aktion notwendig, außer den zusätzlichen Speicherplatz zu überprüfen. Die Dateien finden Sie in dem Verzeichnis $PGDATA/pg_xlog (nicht zu verwechseln mit dem Verzeichnis pg_log, hier liegen die normalen LOG-Mitteilungen).

Checkpoints in der Datei pg_control bedeuten, dass alle Informationen vor dem Punkt in die Datendateien übernommen wurden. Die Checkpoints erzeugt der Datenbankserver in regelmäßigen Abständen. Auch Sie können mit dem SQL-Befehl CHECKPOINT einen Checkpoint auslösen. Im Fall eines Recovery durchläuft das Programm die Datei pg_control vom Ende her und löst die REDO Operationen aus.

Datenbank Aktivitäten überwachen

Als Administrator haben Sie dazu verschiedene Möglichkeiten:

1. Sie lassen den Postmaster-Prozess im Vordergrund laufen und können so die wichtigsten Meldungen sehen.
2. Eine Log-Datei protokolliert alle wichtigen Ereignisse mit (**Anmerkung:** Mit dem tail-Befehl können Sie sich die letzten Zeilen der Datei ansehen).
3. Mit dem ps - Kommando sehen Sie auch alle Aktivitäten auf dem Host. Die Anweisung sieht so aus:

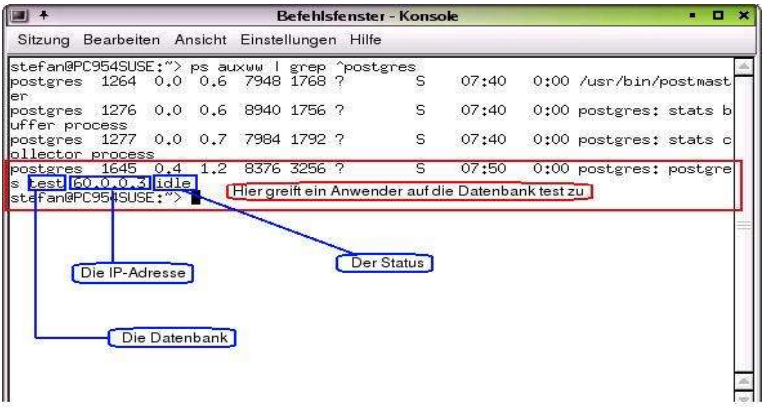


Abbildung 30 Datenbank Aktivitäten überwachen

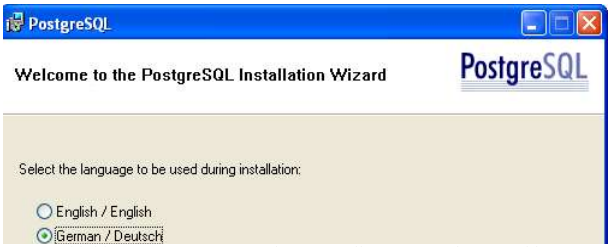
ps auxww | grep ^postgres .

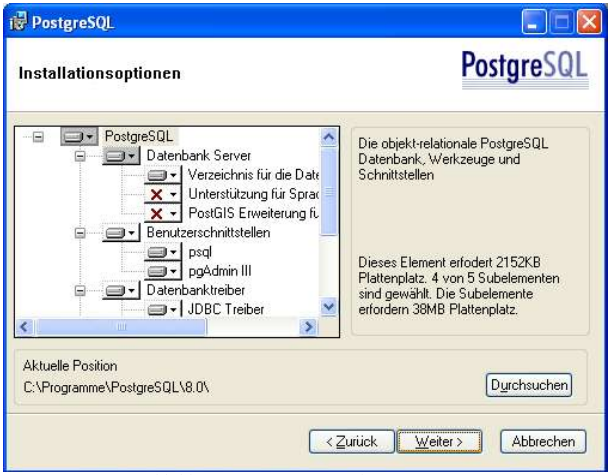
ANMERKUNG: Auch mit dem SQL-Befehl ***select * from pg_stat_activity;*** lassen sich die Aktivitäten anzeigen.

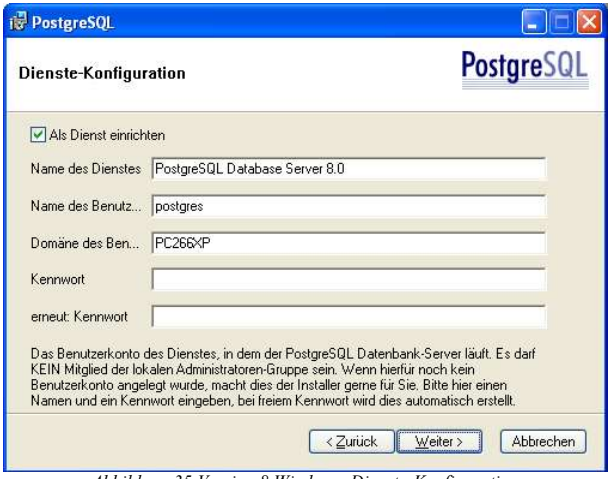
Version 8 – Postgres und Windows

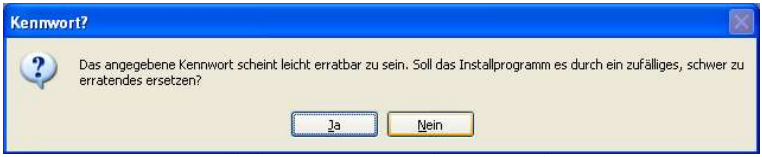
Installation

Mit der neuen Version 8.0 ist es nun auch möglich, ohne CYGWIN die Datenbank auf einem Windows-Rechner zu betreiben. Ich habe die neue Version (zurzeit noch als RC1-Version) auf einem Windows 2003 - Server installiert. Laut Homepage (pgFoundry http://pgfoundry.org/projects/pginstaller.) lässt sich die Datenbank auf Windows NT/2000/XP und 2003 installieren (für Windows 95/98/Me benötigen Sie immer noch CYGWIN). **WICHTIG:** Als Filesystem ist NTFS Pflicht! Es gibt übrigens eine extra japanische Version. Sie erkennen Sie an dem Zusatz ja.

Schritt	Inhalt / Aktion
1	Anmelden am PC mit Administrationsrechten. Starten Sie den Vorgang mit dem Explorer durch einen Doppelklick auf die MSI-Datei. Nehmen Sie hier die Datei ***postgresql-8.0.msi***.
2	 *Abbildung 31 Version 8 Windows, Installation, Sprache auswählen* Wählen Sie hier die Sprache aus und drücken den Knopf ***Start***.
3	 *Abbildung 32 Version 8 Windows, Installation* Auch hier die Informationen durchlesen, der Empfehlung Folge leisten und dann den Button **Weiter** drücken.

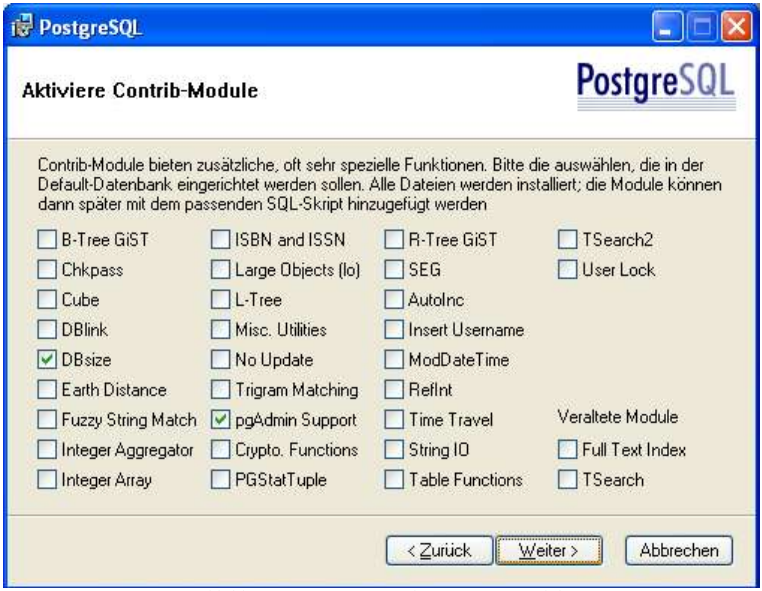
Schritt	Inhalt / Aktion
4	 *Abbildung 33 Version 8 Windows, Installation, Hinweise* Lesen Sie sich die Installationshinweise noch durch und drücken den Knopf ***Weiter***.
5	 *Abbildung 34 Version 8 Windows, Optionen* Hier können Sie noch einige Optionen abwählen bzw. verändern. Alle wichtigen Konfigurationsdateien finden Sie dann im Verzeichnis DATA (unterhalb von ..\8.0).

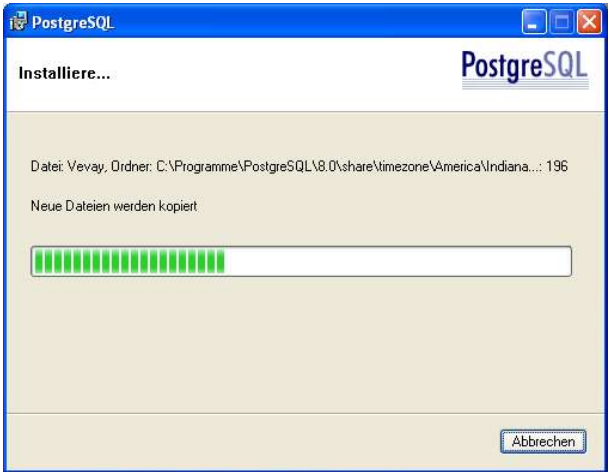
Schritt	Inhalt / Aktion
6	 *Abbildung 35 Version 8 Windows, Dienste-Konfiguration* Wie schon bei der Unix-Version, empfiehlt es sich unter Windows die Datenbank auch als Dienst zu betreiben. Das Installationsprogramm legt Ihnen den Benutzer automatisch neu an. Sie füllen einfach nur die Felder aus (**ANMERKUNG:** Aus Sicherheitsgründen ohne Adminrechte). Vergeben Sie auch ein Passwort für das Benutzerkonto. Wenn Sie ein einfaches Passwort vergeben, z.B. der Name des Benutzerkontos, dann weist Sie das Installationsprogramm später darauf hin.
7	 *Abbildung 36 Version 8 Windows, User postgres* Wenn Sie beim Schritt 6 einen neuen User vorgesehen haben, fragt das Programm noch mal nach.

Schritt	Inhalt / Aktion
8	 *Abbildung 37 Version 8 Windows, Password* Ggf. kann es bei der Vergabe des Passworts ein Problem geben (siehe auch Schritt 6, zu kurz, keine Zahlen usw.). Hier können Sie ein zufälliges Passwort vergeben.
9	 *Abbildung 38 Version 8 Windows, Service Rechte* Zur Bestätigung, dass alles funktioniert, erhalten Sie diese Meldung.

Schritt	Inhalt / Aktion
10	 *Abbildung 39 Version 8 Windows, database cluster* In diesem Schritt legen Sie nun den Superuser der Datenbank an (der User mit den Adminrechten beim Programm psql). Verwechseln Sie diesen User nicht mit dem User postgres im Schritt 6! Stellen Sie noch die Werte für die Felder ***Locale*** und ***Encoding*** ein. Wichtig ist auch noch das Passwort für den Superuser der Datenbank und auch die Haken im Eintrag ***Datenbank-Cluster initialisieren*** und ***Verbindung auf allen Netzadressen annehmen, nicht nur auf Localhost***. Stellen Sie bitte auch noch das Encoding auf ***LATIN1***.
11	 *Abbildung 40 Version 8 Windows, remote Zugriff* Bitte den Hinweis beachten und später berücksichtigen.

Schritt	Inhalt / Aktion
12	 *Abbildung 41 Version 8 Windows, prozedurale Sprachen* Belassen Sie hier den Vorschlag **PL/pgsql.** Wenn Sie später Funktionen auf der Datenbank entwickeln möchten, ist die prozedurale Sprache notwendig.

Schritt	Inhalt / Aktion
13	 *Abbildung 42 Version 8 Windows, weitere Module* Hier können Sie noch zusätzliche Module aktivieren. Belassen Sie es bei den Vorschlägen und drücken Sie den Knopf ***Weiter***.

Schritt	Inhalt / Aktion
14	 *Abbildung 43 Version 8 Windows, Ende der Installation* Nun ist alles eingestellt und fertig für die Installation. Drücken Sie die ***Weiter***-Taste.
15	 *Abbildung 44Version 8 Windows, Dateien kopieren* Danach kopiert das Programm alle notwendigen Dateien.

Schritt	Inhalt / Aktion
16	 *Abbildung 45 Version 8 Windows, Installation vollständig* Am Ende erhalten Sie diese Mitteilung. Mit dem Druck auf den Knopf ***Beenden*** ist die Installation fertig.

17	Damit alle anderen Hosts auf die Datenbank zugreifen können, ändern Sie noch die Zeile in der Datei postgresql.conf ab (über: **Start-Taste, Programme, PostgreSQL 8.0, Konfigurationsdateien, postgesql.conf bearbeiten**)
	`#listen_addresses='localhost'`
	in
	`listen_addresses='*'`

18	Die Datei pg_hba.conf noch bearbeiten (über: **Start-Taste, Programme, PostgreSQL 8.0, Konfigurationsdateien, pg_hba.conf bearbeiten**) (siehe auch Schritt 11).

Die wichtigen Konfigurationsdateien wie postgres.conf und pg_hba.conf finden Sie im \Data-Verzeichnis. Da die Dateien aus einer UNIX-Umgebung kommen, haben Sie in der Beta-Version die Zeilenschaltung von UNIX mit übernommen (Bei der RC1-Version ist das nicht mehr der Fall).

Das Programm psql konfigurieren

Wenn Sie das erste Mal das Programm psql starten, dann erhalten Sie folgende Meldung:

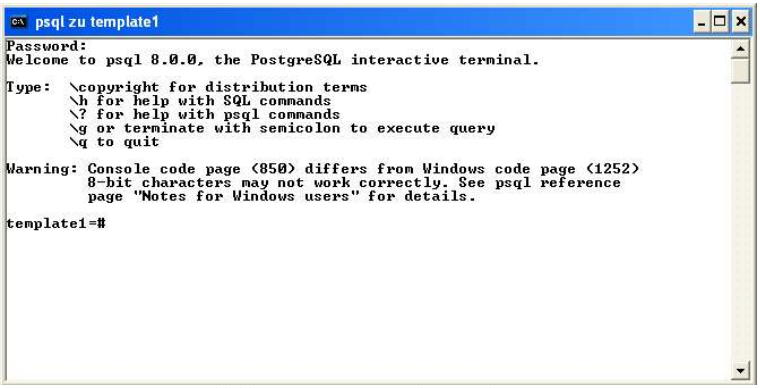


Abbildung 46 Version 8 Windows, psql Hinweis

Das Problem können Sie auch recht einfach beheben. Rufen Sie den Command-Prompt auf (***Start***-Taste, ***Ausführen*** und dann ***CMD*** eingeben). Arbeiten Sie danach die unten gezeigten Schritte ab.

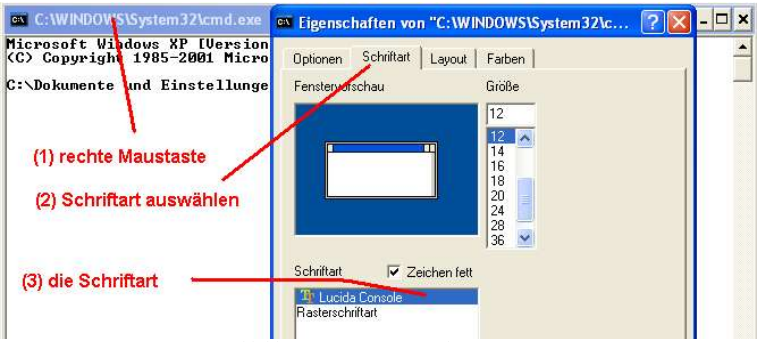


Abbildung 47 Version 8 Windows, CMD

Damit Sie nicht jedes Mal die Einstellung vornehmen müssen, stellen Sie ein, dass die Eigenschaften für jedes Fenster gelten.

Abbildung 48Version 8 Windows, CMD Eigenschaften übernehmen

Vor dem Start vom Programm psql ändern Sie noch die Codepage

Abbildung 49 Version 8 Windows, Codepage

Die Angaben können Sie natürlich auch in eine Batch-Datei schreiben.

Arbeiten mit der Datenbank

Den Dienst der Datenbank können Sie recht einfach beenden. Auch das Programm pgAdmin III befindet sich schon auf dem PC. Das Programm **psql** finden Sie unter dem Eintrag **psql to template1**.

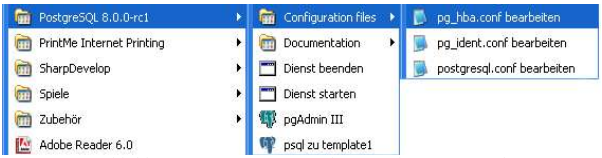


Abbildung 50 Version 8.0 Windows - Arbeiten mit psql

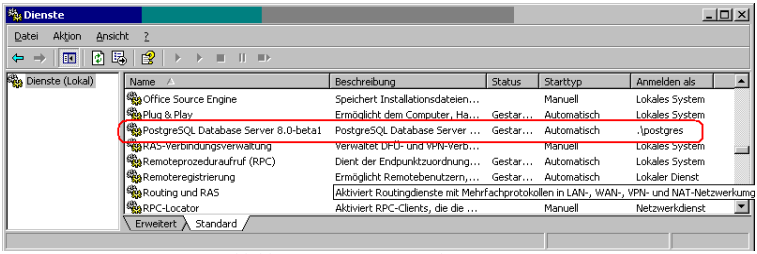
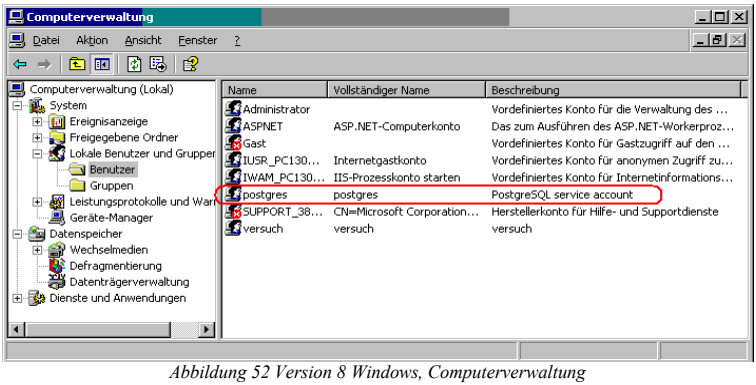
TIPP: Über den NET-Befehl können Sie auch die Datenbank beenden und wieder neu starten.

```
NET START <<Name des Dienstes, lt. Windows Dienste in der System-
steuerung  (z.B.: pgsql-8.0-beta2.dev3)>>
```
==> Datenbank starten

```
NET STOP <<Name des Dienstes, lt. Windows Dienste in der System-
steuerung  (z.B.: pgsql-8.0-beta2.dev3>>
```
==> Datenbank beenden

ANMERKUNG: In den etwas neueren Versionen können Sie die wichtigsten Konfigurationendateien bequem aus dem Menüsystem aufrufen und mit dem Programm NOTEPAD bearbeiten

Deinstallation

Schritt	Inhalt / Aktion
1	Prüfen Sie, welche Daten für Sie auf der Datenbank noch wichtig sind und sichern Sie die Informationen.
2	Beenden Sie den Dienst der Datenbank (z.B. über die Systemsteuerung, Verwaltung und Dienste)  *Abbildung 51 Version 8 Windows, Dienste*
3	Über den Eintrag ***Software***, in der ***Systemsteuerung***, finden Sie die Datenbank (wenn Sie die Software über die MSI-Datei installiert haben). Wählen Sie Postgres aus und drücken Sie deinstallieren.
4	Entfernen Sie noch den User Postgres aus der Benutzerverwaltung des  *Abbildung 52 Version 8 Windows, Computerverwaltung* Computers, soweit er angelegt wurde.

Client-Software und andere Zugriffsmöglichkeiten

Nach dem nun die Datenbank läuft, möchten Sie natürlich auch mit verschiedenen Programmen auf die Daten zugreifen. Hier erhalten Sie einen Auszug der Möglichkeiten.

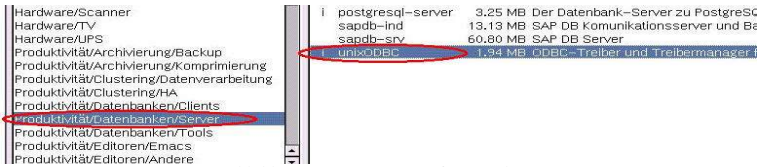
	Schnittstellen	**Programme / Möglichkeiten**
Der Postgres Datenbankserver auf Ihrem LINUX-Host	unixODBC	Verschiedene Programme, die unixODBC nutzen.
	Windows ODBC	Mögliche Programme wie z.B.: - ACCESS - OpenOffice - jedes andere Programm, das Datenbanken über ODBC nutzt
	psql	Das Standardprogramm, mit dem Sie über SQL-Befehle alles auf Ihrem Postgres Datenbankserver machen können.
	JDBC (org.postgresql.Driver)	Ihr JAVA-Programm
	libpg (C-Library)	Ihr C-Programm
	Libpqxx (C++-Library)	Ihr C++-Programm
	Npgsql.dll	Ihr C#-Programm

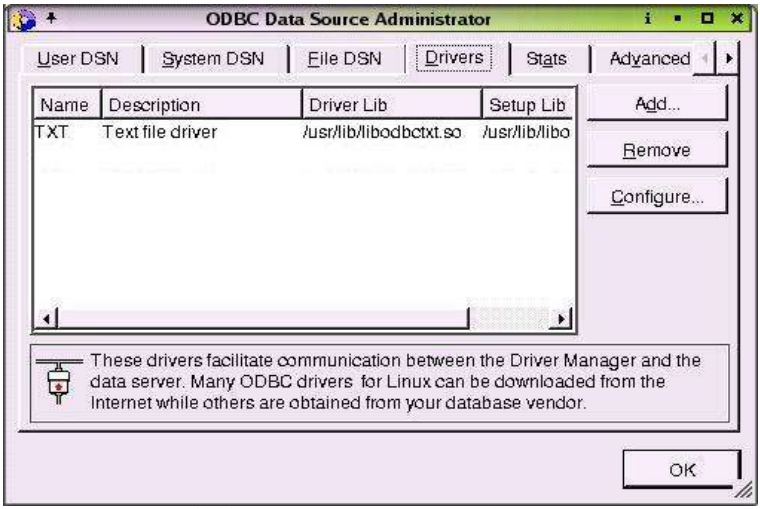
ANMERKUNG: Zurzeit gibt es noch Probleme mit dem unixODBC-Treiber. UnixODBC delegierte die Aufgabe an die Entwicklergruppe von Postgres zurück. Unter der SuSE 8.0 Version hatte ich keine Probleme. Bei der SuSE 8.2-Version fehlten die entsprechenden Dateien. In neueren SuSE-Versionen sind sie aber wieder vorhanden. Bei der Red Hat 9 Distribution fehlen bei unixODBC auch die grafischen Konfigurationsprogramme.

Installation der Programme psql und unixODBC auf einem LINUX-Host

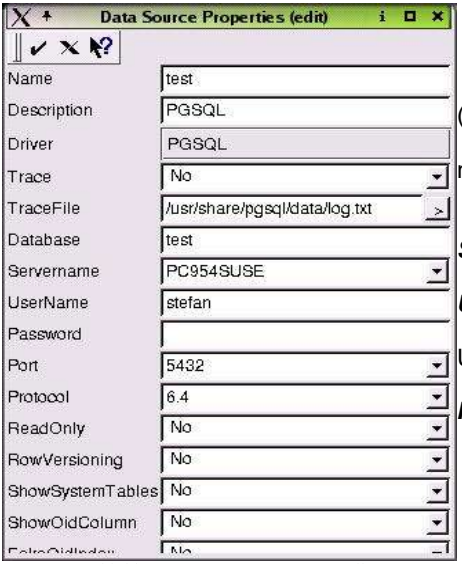
SuSE-Distribution

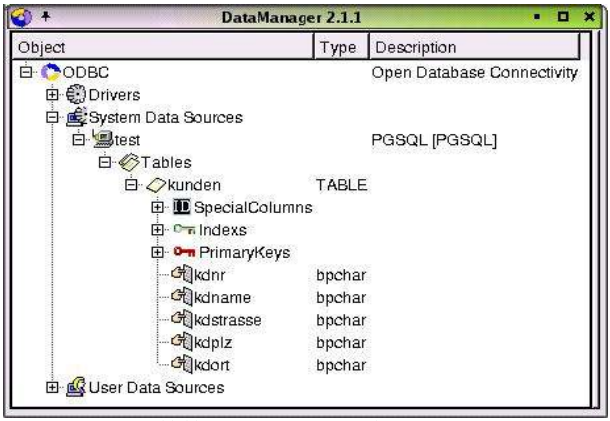
Damit Sie über andere Programme auf Ihre Datenbank zugreifen können, müssen Sie die Client-Software auf Ihrem LINUX-Host installieren. Das Programm ***psql*** ermöglicht Ihnen nur die Ausführung von SQL-Statements. Im hier gezeigten Beispiel sehen Sie wie Sie unixODBC auf Ihrem LINUX-Host installieren. Das Programm können Sie übrigens auch gleich mit dem Server installieren. Im folgenden Beispiel sehen Sie die Installation unter der SuSE-Distribution. Alternativ können Sie die entsprechenden Programme natürlich auch wie im Kapitel **Installation einer Version aus dem Internet** beschrieben installieren.

Schritt	Inhalt / Aktion
1	Rufen Sie als erstes das YaST2-Kontrollzentrum auf. Wenn Sie den KDE-Desktop einsetzen, drücken Sie hier die ***Start***-Taste, ***Einstellungen***, ***Yast menu*** und dann ***Yast Kontrollzentrum***.
2	Unter der Rubrik ***Software*** (linke Seite) finden Sie das ICON ***Software installieren/löschen***. Bitte das Programm Software installieren/löschen mit einem Doppelklick aufrufen (siehe auch Installation der Datenbank). Über dieses Programm installieren Sie folgende Softwarepakete:  *Abbildung 53 unixODBC-Softwarepakete* 1. ***unixODBC*** aus dem Paket ***Produktivität/Datenbanken/Server*** (siehe Bild oben) (nur bei ODBC!) 2. ***postgressql-odbc*** aus dem Paket ***Produktivität/Datenbanken/Clients*** (nur bei ODBC!) 3. ***unixODBC-gui-qt*** aus dem Paket ***Produktivität/Datenbanken/Tools*** (nur bei ODBC!) 4. ***postgresql*** aus dem Paket ***Produktivität/Datenbanken/Tools*** (wenn das Programm psql gewünscht wird)

Schritt	Inhalt / Aktion
3	Melden Sie sich nun an Ihrem Host ab und melden Sie sich an dem Host als Anwender ***root*** wieder an. Wenn Sie nur das Programm ***psql*** benötigen ist hier schon die Installation beendet.
4	Sie rufen danach das Programm ***ODBCConfig*** auf, in dem Sie über die ***Start***-Taste, ***Büroprogramme***, ***Datenbanken*** auf den Programmeintrag kommen.
5	Rufen Sie in dem Programm die Karteikarte ***Drivers*** auf und drücken den Button ***Add***.  *Abbildung 54 unixODBC-Data Source Administrator*

Schritt	Inhalt / Aktion
6	In dem nächsten Window arbeiten Sie folgende Punkte ab:  *Abbildung 55 unixODBC-Treiber* 1. ***Name*** vergeben: z.B. PGSQL 2. ***Description*** vergeben: z.B. für den POSTgre Server 3. Als ***Driver*** verwenden Sie die Datei ***/usr/lib/libodbcpsql.so*** 4. Als ***Setup***-Datei nehmen Sie die Datei ***/usr/lib/libodbcpsqlS.so*** 5. Im Eintrag ***FileUsage*** lassen Sie die 1 stehen ***ANMERKUNG:*** Die Dateien libodbcpsql.so und libodbcpsqlS.so können sich auch in einem anderen Verzeichnis befinden (siehe z.B. bei der Installation einer neuen Version aus dem Internet - Anhang C). In der SuSE 8.2 Version gab es ähnlich lautende Dateien, Sie sollten nur die oben gezeigten Dateien verwenden.
7	Danach rufen Sie die Karteikarte ***System DSN*** auf und drücken den Button ***Add***

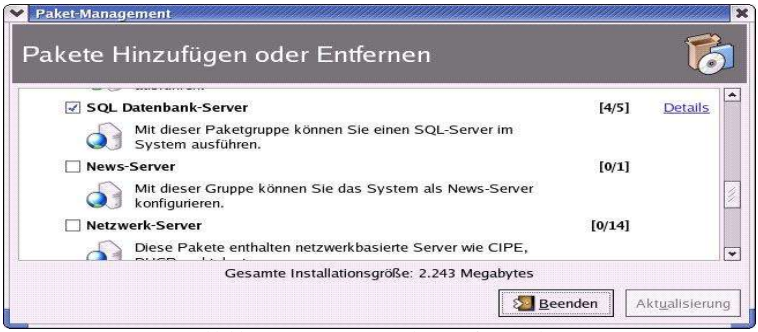
Schritt	Inhalt / Aktion
8	Bearbeiten Sie im Window Data Source Properties folgende Punkte:  *Abbildung 56 unixODBC-Datenquelle* 1. ***Namen*** vergeben 2. ***Description*** vergeben 3. Auswahl des Treibers (***Driver***) 4. ***Trace*** und ***TraceFile*** sind nicht unbedingt notwendig 5. Eingabe der ***Database*** 6. Eingabe des ***Servername***ns 7. Eingabe eines ***UserNamens*** 8. Das ***Password*** für den User noch eingeben 9. Die Port-Nr. in der Zeile ***Port*** noch überprüfen Danach alles wie gewohnt abspeichern und das Programm verlassen.

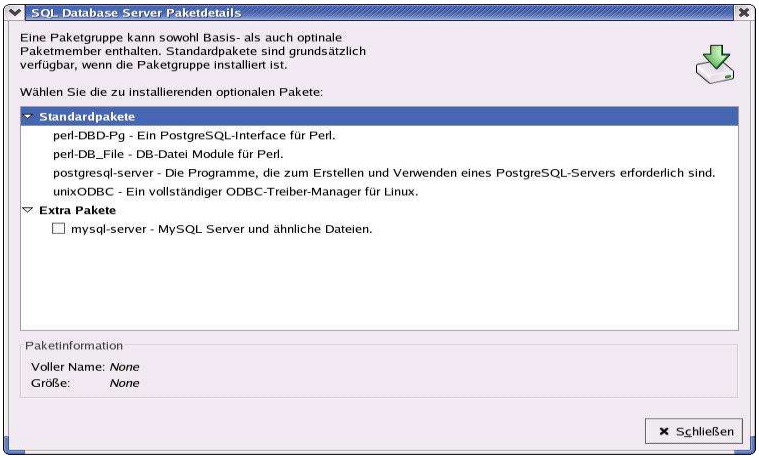
Schritt	Inhalt / Aktion
9	Nun kann man auch mit dem Programm ***DataManager*** (befindet sich direkt neben dem Programm ODBCconfig) auf die Datenbank zugreifen.  *Abbildung 57 unixODBC-Treibermanager*

Anmerkung: Sollte die Verbindung von einem Client auf einen Host nicht möglich sein, so überprüfen Sie unter anderem die Einstellungen der Firewall auf dem Server (z.B. Sie haben die PersonalFirewall im Einsatz).

Red Hat/Fedora Distribution

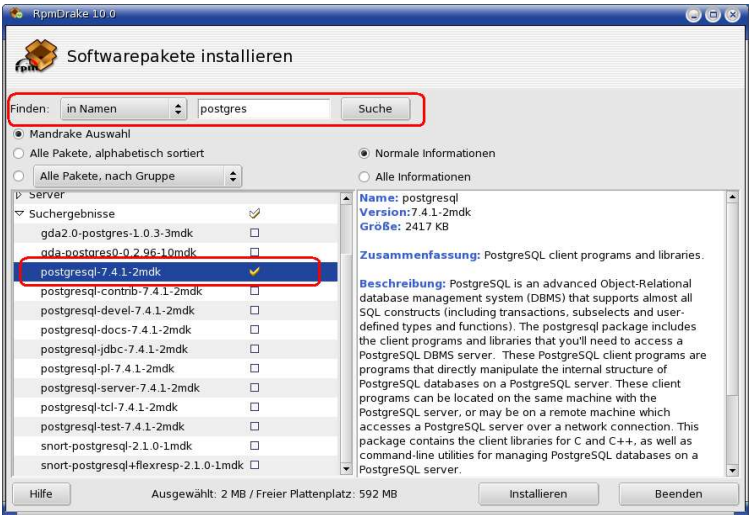
Bei der Linux-Distribution von Red Hat (ohne unixODBC) gehen Sie wie unten beschrieben vor (sie gleicht der Server-Installation und Sie erhalten auch die Server-Programme):

Schritt	Inhalt / Aktion
1	Drücken Sie die Start-Taste (mit dem roten Hut). Danach geht es weiter mit den Menüpunkten ***Systemeinstellungen*** und ***Hinzufügen / Entfernen von Applikationen***
2	Im Fenster ***Pakete Hinzufügen oder Entfernen*** finden Sie den Eintrag ***SQL Datenbank-Server***. Wählen Sie diesen Punkt aus.  *Abbildung 58 unixODBC-Red Hat Softwarepakete*

Schritt	Inhalt / Aktion
3	Wenn Sie im vorherigen Schritt auf den Eintrag ***Details*** drücken, sehen Sie im Detail, welche Software mit den Standardpaketen installiert wird.  *Abbildung 59 unixODBC-Red Hat Softwarepakete*
4	Danach schließen Sie die Auswahl ab und legen die geforderten CDs ein.

Mandrake Distribution

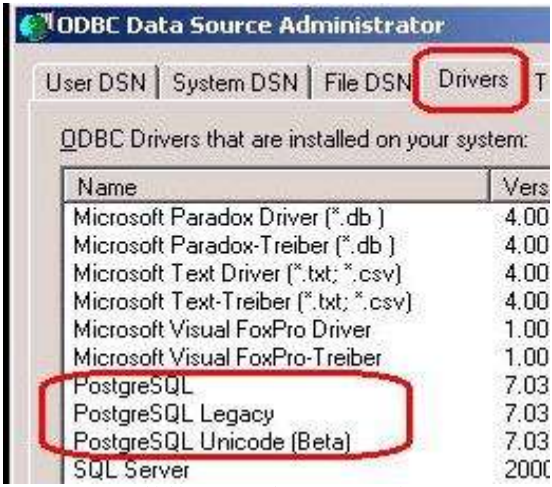
Bei der community Edition gehen Sie wie unten beschrieben vor (sie gleicht der Server-Installation und Sie erhalten auch die Serverkomponenten).

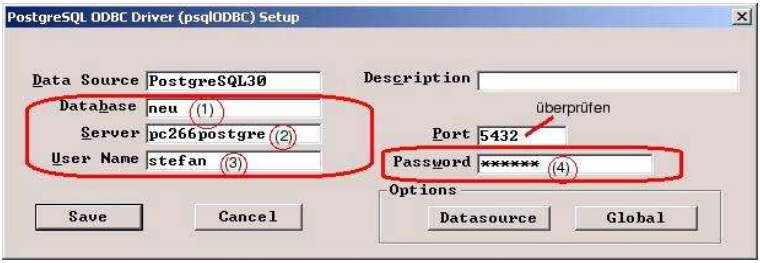
Schritt	Inhalt / Aktion
1	Drücken Sie die Start-Taste (mit dem gelben Stern). Danach geht es weiter mit den Menüpunkten ***System, Einstellungen, Paketierung*** und ***Software installieren.***
2	Melden Sie sich nun mit dem Root-Passwort an.
3	Suchen Sie in dem nun folgenden Fenster nach dem Namen postgres (Knopf ***Suche*** drücken). In der Trefferliste wählen Sie dann den Eintrag ***postgresql-7.4.1-2mdk*** aus. Anschließend drücken Sie den Knopf ***Installieren***.  *Abbildung 60 Mandrake - Installation psql*

UnixODBC müssen Sie leider per Hand installieren. In der community Edition verlangt das Installationsprogramm leider nach einer nicht vorhandenen 4. CD.

Der ODBC-Treiber für Windows-Systeme

Sie finden die neuesten ODBC-Treiber auf der Homepage von PostgreSQL (www.postgresql.org). Über den Explorer rufen Sie die MSI-Datei auf (die Schritte sind hier etwas gepackt dargestellt) (**Anmerkung:** Auch bei dem Windows.NET Server 2003 gibt es keine Probleme).

Schritt	Inhalt
1	Installieren Sie hier die entsprechende MSI-Datei auf dem entsprechenden PC (psqlodbc.msi)
2	Wechseln Sie in der Systemverwaltung in den ***ODBC Data Source Administrator***. Unter der Karteikarte ***Drivers*** sollten Sie dann die PostgreSQL-Treiber finden.  *Abbildung 61 Windows ODBC-Treiber*

Schritt	Inhalt
3	Richten Sie danach entweder unter der ***User DSN*** oder ***System DSN*** den Zugriff auf die Datenbank ein. Sie geben hier die Datenbank **(1)**, den Server **(2)** (IP-Adresse geht auch!), Benutzername **(3)** und Passwort **(4)** ein.  *Abbildung 62 Windows ODBC-Einstellungen*
4	Prüfen Sie z.B. über Access, ob Sie über die nun neu eingerichtete ODBC-Datenquelle auf die Tabellen des Host zugreifen können.

Das Programm psql

Allgemeines zum Programm psql

Mit diesem Programm können Sie verschiedene SQL-Befehle und Abfragen auf der Datenbank ausführen. Wenn Sie das Programm als Anwender ***postgres*** aufrufen, dann befinden Sie sich im Administrator-Modus (Sie erhalten das #-Zeichen als Prompt, sonst sehen Sie den Pfeil nach rechts), ansonsten befinden Sie sich im normalen User-Modus. Der vollständige Programmaufruf sieht so aus:

psql [Option] [dbname] [username]

Die Optionen sind im Text weiter unten beschrieben. Die Information ***dbname*** steht für die Datenbank und der Wert ***username*** beinhaltet den Benutzernamen. Weiterhin finden Sie auch die Beschreibung der Terminaloptionen.

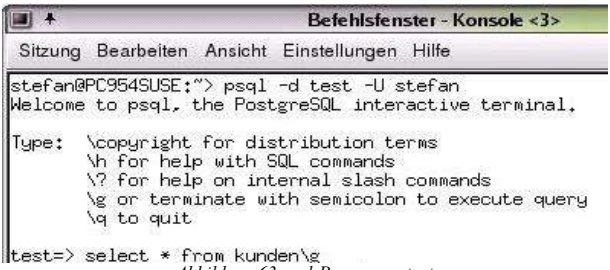


Abbildung 63 psql-Programmstart

WICHTIG: Mit der Zeichenkombination **\q** beenden Sie die Sitzung. Die Ausführung eines SQL-Kommandos veranlassen Sie durch die Buchstabenkombination **\g** (mit einem Semikolon den Befehl abschließen geht auch)! Wenn Sie Benutzer auf der Datenbank anlegen, dann sollten Sie als LINUX-Anwender ***postgres*** das Programm ***psql*** aufrufen.

Parameter	Inhalt
[options]	Die Terminaloptionen können Sie beim Start des Programms mit angeben (z.B. Sie führen eine SQL-Anweisung aus). Folgende Optionen gibt es: **-a** Gibt alle Eingaben aus dem Script aus **-c <query>** Führt eine einzige Abfrage (oder \-Befehl) aus und beendet psql **-d <dbname>** Bei einer bestimmten Datenbank anmelden. Fehlt der Datenbankname, dann wird der Anwendername als Datenbank verwendet. **-e** Gibt das SQL-Kommando als Überschrift aus (z.B. wenn SQL-Anweisungen aus einer Datei ausgeführt werden. **-f <filename>** Führt die Abfrage aus einer Datei aus und beendet das Programm. **-F <string>** Setzt den Datenfeldtrenner (default: "|") (-P fieldsep=) **-h <host>** Beinhaltet den Hostnamen einer Datenbank (**wichtig,** wenn Sie psql von einem anderen Host aufrufen) **-H** Wechselt in den HTML-Tabellen Ausgabemodus **-l** Zeigt die vorhandenen Datenbanken an und beendet anschließend das Programm **-o <filename>** Leitet die Ausgabe der Abfrage in eine Datei oder Pipe um **-p <port>** Enthält die Port-Nr. der Datenbank **-q** Läuft im Hintergrund, gibt keine Meldungen und Abfragen aus **-R <string>** Setzt den Datensatzseparator (default: Zeilenschaltung) (-P recordsep=) **-s** Der „kleine Debug-Modus“. Jede SQL-Anweisung sehen Sie und müssen die Ausführung bestätigen **-t** Zeigt nur die Spalten an (-P tuples_only) **-U <username>** Enthält den Benutzernamen beim Programmstart **-v name=val** Setzen einer psql-Variablen 'name' mit dem Wert 'val' **-V** Zeigt nur die aktuellen Versionsinformationen an **-X** Liest nicht die Konfigurationsdateien beim Start (~/.psqlrc)

Parameter	Inhalt
-d [dbname]	Der Name der Datenbank (**TIPP:** Die Datenbank template1 ist immer vorhanden)
-U [User]	Der Benutzername

Die Terminaloptionen des Programms psql

Sie können die hier gezeigten Befehle nur innerhalb des Programms psql ausführen!

Option	Inhalt
\a	Steuert den Ausgabemodus bei Abfragen, mit bzw. ohne Feldname
\c[onnect] [DBNAME]-[USER]]	Auf einer neuen Datenbank anmelden
\C TITLE	Wird im Zusammenhang mit dem Select-Befehl verwendet. Gibt bei der Ausgabe zuerst den Titel und dann die Feldnamen aus
\c [DIRNAME]	Wechselt das aktuelle Verzeichnis
\copyright	Ausgabe der Copright-Informationen
\d [TABELLE]	Ausgabe der Feldbeschreibungen einer Tabelle, wenn Sie die Tabelle weglassen, sehen Sie alle Tabellen
\da [Suchmuster möglich]	Ausgabe aller Aggregat-Funktionen
\dt [Suchmuster möglich]	Ausgabe alle Tabellen einer Datenbank
\di [Suchmuster möglich]	Ausgabe aller Indexe einer Datenbank
\ds [Suchmuster möglich]	Ausgabe aller Sequenzen einer Datenbank
\dv [Suchmuster möglich]	Ausgabe aller Views einer Datenbank
\dp [Suchmuster möglich]	Ausgabe aller Zugriffsrechte, Rechte sind: arwdRxt

Option	Inhalt
\dS [Suchmuster möglich]	Ausgabe aller Systemtabellen
\dl	Ausgabe aller „großen Objekte“
\dd [Suchmuster möglich]	Nach Kommentaren für Tabellen, Typen, Funktionen oder Operatoren sehen
\df [Suchmuster möglich]	Ausgabe aller Funktionen
\do [Name]	Ausgabe aller Operatoren
\dT [Suchmuster möglich]	Ausgabe aller Datentypen
\du [Suchmuster möglich]	Ausgabe aller Anwender
\e [DATEINAME]	Bearbeiten einer externen Datei im Editor vi
\echo [TEXT]	Schreiben eines Textes in die Standardausgabe
\f [STRING]	Setzen eines Feldtrenners
\g [DATEINAME]	Ausführen eines SQL-Kommandos aus einer Datei und Ausgabe in die Datei des SQL-Kommandos
\h [SQL-Befehl]	Hilfe über die Syntax eines SQL-Befehl anfordern, z.B. ***\h ALTER USER***
\H	Wechselt in den HTML Ausgabemodus (standardmäßig aus)
\i [DATEINAME]	Ausführen eines Kommandos aus einer Datei, Ausgabe des Ergebnisses auf den Bildschirm
\l	Anzeigen aller Datenbanken
\o [DATEINAME]	Ausgabe aller Abfrageergebnisse in eine Datei oder in eine pipe
\p	Ansehen des Abfragepuffers (z.B. nachdem eine Select-Abfrage ausgeführt wurde)
\q	Das Programm psql verlassen
\r	Den Abfragepuffer löschen
\s [FILENAME]	Zeigt die bisher ausgeführten Befehle an. Eine Ausgabe in eine Datei ist möglich
\set [NAME] [VALUE]	Setzen einer internen Variablen

Option	Inhalt
\t	Nur die Zeilen einer SELECT-Anweisung ansehen
\unset [NAME]	Löschen einer internen Variablen
\w [DATEINAME]	Den aktuellen Abfragepuffer in eine Datei schreiben
\x	Wechseln des aktuellen Ausgabemodus. Man kann z.B. bei einer Select-Anweisung die Daten sequentiell oder tabellarisch ausgeben.
\z	Ausgabe der Zugriffsberechtigungen
\! [BEFEHL]	Ein Befehl oder eine Shell ausführen

Abbildung 64 psql-Liste verlassen

ANMERKUNG: Wenn sich die Hilfe über mehrere Zeilen erstreckt, dann können Sie über die Buchstabenkombination ***:q*** (wie beim vi) die Hilfe beenden.

Verwalten von Benutzern, Gruppen und Rechte

Um für die Datenbank Benutzer, Gruppen und Rechte zu verwalten, können Sie als LINUX-Anwender postgres das Programm psql aufrufen. Anschließend verwenden Sie zur Administration SQL-Befehle.

Benutzer verwalten

SQL-Befehle

Aktion	SQL-Befehl
Benutzer anlegen	***CREATE USER [NAME] PASSWORD 'geheim'*** **ANMERKUNG:** das Wort geheim steht nur stellvertretend für das individuelle Passwort des Anwenders. Zwischen dem Wort PASSWORD und 'geheim' befindet sich ein Leerzeichen.
Vorhandenen Benutzer löschen	***DROP USER [NAME]***
Vorhandenen Benutzer ändern	***ALTER USER [NAME] [WITH] [Option, z.b. PASSWORD 'geheim'***, oder VALID UNTIL 'absolute Zeit im Format TT.MM.JJ'] **ANMERKUNG:** Es gibt noch mehrere Optionen.

ANMERKUNG: Mit ***\du*** (Programm psql) können Sie sich alle User anzeigen lassen

Auch von der Kommandozeile aus können sie Benutzer anlegen und löschen. Dazu haben Sie die Programme ***createuser*** und ***dropuser***. Beide Programme senden SQL-Befehle an die Datenbank. Sie setzen auch Superuser-Rechte voraus.

Programm createuser

Wenn noch keine Anwender angelegt sind, dann melden Sie sich als Unix-Anwender postgres an. Unter seiner Benutzerkennung legen Sie die weiteren Anwender an. Das Programm createuser ist wie folgt aufgebaut:

createuser [Optionen] [Benutzername]

Die verschiedenen Optionen des Programmes createuser entnehmen Sie der Tabelle (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	*Inhalt*
`--adduser [-a]`	Der Benutzer kann neue Anwender anlegen
`--no-adduser [-A] (großes A)`	Der Benutzer hat nicht das Recht andere Anwender anzulegen
`--createdb [-d]`	Der Benutzer kann eine neue Datenbank anlegen
`--no-createdb [-D]`	Der Benutzer kann keine neue Datenbank anlegen
`--pwprompt [-P]`	Dem neuen Benutzer auch ein Passwort zuweisen
`--encrypted [-E]`	Das gespeicherte Passwort verschlüsseln
`--unencrypted [-N]`	Das gespeicherte Passwort nicht verschlüsseln
`--sysid=SYSID [-i]`	Dem neuen User auch eine SYSID in der Datenbank zuweisen
`--echo [-e]`	Die Meldungen bei der Aktion mit ausgeben (hier sehen Sie die SQL-Befehle)
`--quiet [-q]`	Die Meldungen bei der Aktion nicht ausgeben
`--help`	Die Hilfe für den Befehl

Wenn Sie keine Option angeben, dann fragt Sie das Programm nach den wichtigsten Optionen (siehe auch Bilder auf der nächsten Seite).

Abbildung 65 createuser ohne Optionen

Abbildung 66 createuser mit Optionen

Auch von Ihrer localem Host aus besteht die Möglichkeit auf anderen Hosts Anwender anzulegen. Dazu haben Sie diese Optionen (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	***Inhalt***
`--host=HOSTNAME [-h]`	Der Name des Hosts
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Wenn Sie für mehrere Optionen die Kurzform wählen, dann nehmen Sie als Trenner das Leerzeichen (z.B. ***createuser -h pc1300linux -D -A ralf***)

ANMERKUNG: Im Programm psql sehen Sie die angelegten Benutzer mit dem Befehl \du. Im unten gezeigten Beispiel wurde dem Benutzer janny die SYSID 555 zugewiesen und das Recht neue Benutzer anzulegen. Das Attribut „create database“ fehlt dagegen.

Abbildung 67 createuser - Benutzer abfragen

Programm dropuser

Nachdem Sie nun Benutzer angelegt haben, möchten Sie auch Datenbank-Anwender löschen. Dafür haben Sie das Gegenstück zum Programm createuser - dropuser -. Der Befehlsaufbau sieht so aus:

dropuser [OPTION]... [USERNAME]

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	*Inhalt*
`--echo [-e]`	Zeigt die SQL-Befehle an, die an die Datenbank gesendet werden.
`--interactive [-i]`	Abfrage, bevor gelöscht wird
`--quiet [-q]`	Gibt keine Meldungen aus
`--host=HOSTNAME [-h]`	Der Name des Hosts
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Abbildung 68 dropuser - Beispiel

Gruppe verwalten

Verschiedene Benutzer kann man in einer Gruppe zusammenfassen. Wie auch bei den Anwendern verwalten Sie die Gruppe über SQL-Befehle.

Aktion	SQL-Befehl
Gruppe anlegen	***CREATE GROUP [NAME]***
Einen oder mehrere Anwender einer Gruppe zuordnen	***ALTER GROUP [NAME] ADD USER USERNAME1, ...*** **ANMERKUNG:** Die 3 Punkte stehen für weitere Anwender. Verwenden Sie das Komma als Trennzeichen.
Einen oder mehrere Anwender aus einer Gruppe löschen	***ALTER GROUP [NAME] DROP USER USERNAME1, ...*** **ANMERKUNG:** Die 3 Punkte stehen für weitere Anwender. Verwenden Sie das Komma als Trennzeichen.

Zugriffsrechte/Privilegien verwalten

Bei der Anlage einer neuen Datenbank wird ihr ein Owner (es ist der Anwender der die Datenbank erstellt hat) zugeordnet. Der Owner kann zurzeit nicht geändert werden. Erst durch die Erteilung von Zugriffsrechten können die anderen Anwender auf die Datenbank zugreifen. Mit dem SQL-Befehl GRANT verwalten Sie die Zugriffsrechte. Folgende Rechte gibt es:

Aktion	das dazugehörige Recht	Objekt
Lesen	Select	Tabelle
Hinzufügen	Insert	Tabelle
Schreiben	Update	Tabelle
Löschen	Delete	Tabelle
Anlegen	Create	Datenbank
Anlegen von temporären Tabellen	Temporary / Temp	Datenbank

Um einem Anwender ein Recht zu erteilen, ist der Befehl GRANT wie folgt aufgebaut:

```
GRANT [RECHT] ON
[DATABASE Datenbankname | TABLE Tabellenname]
TO [BENUTZERNAME]
```

Wenn es geklappt hat, dann gibt das Programm zur Bestätigung nochmals das Recht aus. Um einer Gruppe ein Recht zu erteilen, ist der Befehl GRANT so aufgebaut:

```
GRANT [RECHT] ON
[DATABASE Datenbankname | TABLE Tabellenname]
TO GROUP [GRUPPENNAME]
```

Geben Sie als Anwendernamen PUBLIC an, dann erteilen Sie jedem Anwender in der Datenbank das entsprechende Recht. Das Recht ALL beinhaltet übrigens alle Rechte.

Neben dem Zuerkennen von Rechten existiert der Befehl REVOKE um einem Anwender auch Rechte zu entziehen. Möchten Sie z.B. allen Anwendern alle Rechte auf einer Datenbank entziehen, dann geben Sie den Befehl REVOKE ALL ON [DATENBANK] FROM PUBLIC ein. Die Befehle DROP, GRANT, REVOKE usw. können im Bezug auf die Datenbank nur vom Owner ausgeführt werden und dem Owner auch nicht entzogen, bzw. anderen Benutzer zuerkannt werden. Der Owner einer Tabelle kann sich aber selber Rechte entziehen.

Weitere Dienstprogramme

clusterdb

Das ist ein Hilfsprogramm mit dem Sie schon geclusterte Tabellen neu clustern. Die Tabellen erhalten den selben Index, der beim letzten Durchlauf verwendet wurde. Tabellen, die noch nicht geclustert wurden, berührt das Programm nicht. Das Programm nutzt den SQL-Befehl CLUSTER und das Programm psql.

Aufbau:

clusterdb [Option]

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	*Bedeutung*
`--all` `[-a]`	Das Programm läuft über alle Datenbanken
`--dbname Datenbank` `[-d] Datenbank`	Das Programm bearbeitet nur die vorgegebene Datenbank
`--echo` `[-e]`	Hier sehen Sie alle Befehle, die an den Server gehen
`--quiet` `[-q]`	Gibt keine Meldungen aus
`--table Tabelle` `[-t Tabelle]`	Bearbeitet nur die vorgegebene Tabelle
`--host=HOSTNAME` `[-h]`	Der Name des Host
`--port=PORT` `[-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME` `[-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password` `[-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

createlang

Mit diesem Befehl installieren Sie auf einer Datenbank eine neue prozedurale Sprache (Siehe auch Funktionen und Trigger). Sie können nur Sprachen installieren, die von Postgres unterstützt werden. Alternativ funktioniert es auch mit dem SQL-Befehl ***CREATE LANGUAGE [Sprache]***. Der Weg über das Programm createlang ist aber einfacher. Die Syntax sieht so aus:

Aufbau:

createlang [Verbindungsoptionen] langname [dbname]

oder

createlang [Optionen] –list | -l dbname

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	***Bedeutung***
`langname`	Der Name der Sprache
`--dbname [-d]`	Name der Datenbank. Wenn nichts angegeben wird, dann greift das Programm auf die Datenbank zu, die den Systemnamen des gerade angemeldeten Benutzers hat.
`--echo [-e]`	Gibt die SQL-Befehle mit aus
`--list [-l]`	Zeigt die auf der Datenbank installierten Sprachen an
`-L Verzeichnis`	Der Name des Verzeichnis in dem sich der Interpreter befindet
`--host=HOSTNAME [-h]`	Der Name des Host
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden

Option	*Bedeutung*
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Umgebungsvariablen
PGDATABASE
PGHOST
PGPORT
PGUSER = Default connection parameters

Bsp.: ***createlang pltcl template1***

droplang

Mit diesem entfernen Sie aus der Datenbank eine prozedurale Programmiersprache (das Gegenstück zu createlang). Es werden auch Sprachen entfernt, die nicht von der Postgres Distribution unterstützt werden.

Aufbau:

droplang [Optionen] langname [dbname]

oder

droplang [Optionen] –list | -l dbname

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	*Bedeutung*
`langname`	Der Name der Sprache
`--dbname [-d]`	Name der Datenbank. Wenn nichts angegeben wird, dann greift das Programm auf die Datenbank zu, die den Systemnamen des gerade angemeldeten Benutzers hat

Option	*Bedeutung*
`--echo [-e]`	Gibt die SQL-Befehle mit aus
`--list [-l]`	Zeigt die auf der Datenbank installierten Sprachen an
`--host=HOSTNAME [-h]`	Der Name des Host
`--port=PORT [-p]`	Die Port-Nr. der Datenbank
`--username=USERNAME [-U] (großes U)`	Der Benutzername, mit dem Sie sich auf dem Datenbankserver anmelden
`--password [-W] (großes W)`	Das Passwort zu dem Benutzer, mit dem Sie sich auf dem Datenbankserver anmelden

Bsp: ***dropland pltcl dbname***

ecpg

Dieses Programm ist für C/C++-Programmierer interessant. Mit dem Programm konvertieren Sie embedded SQL-Befehle in spezielle C-Funktionen. Die Ausgabedatei übersetzen Sie danach mit Ihrem C-Compiler. Die Eingabedatei sollte die Dateierweiterung **.pgc** haben. Ist das nicht der Fall, dann hängt das Programm an die Eingabedatei die Erweiterung **.c** an. Sie können die Eingabedatei auch mit der Option -o überschreiben.

Aufbau:

ecpg [Optionen] Datei

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	*Bedeutung*
`-c`	Erzeugt automatisch C-code aus den SQL-Anweisungen (betrifft das EXEC SQL TYPE)

Option	*Bedeutung*
`-C mode (großes C)`	Ein Kompatibilitäts-Modus. Gültige Modis sind: INFORMIX oder INFORMIX_SE
`-D symbol`	Definiert ein C Präprozessor Symbol
`-i`	Berücksichtigt auch die vom System eingebundenen Dateien
`-I Verzeichnis (großes I)`	Hier weisen Sie zusätzlich ein Verzeichnis zu, dass Include-Dateien enthält
`-o`	Eingabedateiname, wird überschrieben
`-t`	Schaltet autocommit bei Transaktionen an (Ausnahme: Die Anweisung befindet sich in einem expliziten Transaktionsblock). Als Grundeinstellung gilt: Wenn EXEC SQL COMMIT angegeben wurde, werden die Aktionen sofort eingetragen
`-v (kleines v)`	Gibt zusätzliche Informationen über die Version und Include-Verzeichnisse aus
`--help`	Die Hilfe zu dem Programm
`--version`	Gibt nur die Version des Programmes ecpg aus

initdb

Mit diesem Programm richten Sie einen neuen PostgreSQL-Datenbankcluster ein. Der Datenbankcluster beinhaltet eine Sammlung von Datenbanken, die von einer einzelnen Serverinstanz verwaltet wird. Neben den Konfigurationsdateien legt das Programm auch die Datenbank template1 an.

Aufbau:

initdb [Optionen] --pgdata | -D Verzeichnis

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	***Bedeutung***
--pgdata=Verzeichnis [-D]	Das Verzeichnis für den Datenbankcluster
--encoding=Zeichensatz [-E]	Wählt den Zeichensatz für die Datenbank template1 (die „Kopiervorlage“ für spätere Datenbanken). Grundeinstellung ist der Zeichensatz SQL_ASCII
--locale=locale	Setzen von Werten für den Datenbankcluster. Wenn nichts angegeben wird, dann übernimmt das Programm die Informationen aus der Umgebung, in der das Programm ausgeführt wird
--lc-collate=locale	wie locale, spezielle Kategorie
--lc-ctype=locale	wie locale, spezielle Kategorie
--lc-messages=locale	wie locale, spezielle Kategorie
--lc-monetary=locale	wie locale, spezielle Kategorie
--lc-numeric=locale	wie locale, spezielle Kategorie
--lc-time=locale	wie locale, spezielle Kategorie
--username	Der Name des Datenbank-Superusers (meistens der User Postgres)
--pwpromt	Das Passwort
--debug [-d]	Gibt Debugmeldungen aus

Option	*Bedeutung*
-L Verzeichnis	Gibt an, wo die Eingabedateien zu finden sind. Ist normalerweise nicht notwendig
--noclean [-n]	Wenn das Programm abbricht, löscht es alle bis dahin erzeugten Dateien. Diese Option verhindert das

Umgebungsvariablen

PGDATA

initlocation

Im Kapitel Datenbanken anlegen wurde auch schon kurz auf das Programm eingegangen. Wenn der Platz auf einer Platte zu knapp wird, dann legen Sie auf einer anderen Platte einen neuen Bereich an. Vergeben Sie auch für den weiteren Speicherbereich eine neue Umgebungsvariable (z.B. PGDATA2).

Aufbau
initlocation Verzeichnis

Gehen Sie z.B. so vor:

1. initlocation /opt/pg/data2
2. Restart des Datenbankservers
3. createdb -D /opt/pg/data dbname

ipcclean

Das Programm ist die „Müllabfuhr“ für das Programm Postmaster. Sollte das Programm abgestürzt sein, dann entfernt es belegte Speicherbereiche und Semaphoren. Ein Restart bewirkt übrigens auch das Gleiche. Das Programm war eine Notlösung, bevor der Postmaster selber die Bereiche aufräumen konnte.

pg_config

Das Programm gibt verschiedene Informationen über die installierte Postgres-Version aus.

Aufbau:

pg_config [OPTIONEN]

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	***Bedeutung***
`--bindir`	Das Verzeichnis der ausführbaren Dateien
`--includedir`	Ausgabe des Ortes der C-Headerdateien für die Client-Programme
`--includedir-server`	Ausgabe des Ortes der C-Headerdatei für die Server-Programme
`--libdir`	Ausgabe des Ortes der Objekt-Dateien (Bibliotheken)
`--pkglibdir`	Ausgabe des Ortes der dynamisch ladbaren Module.
`--configure`	Ausgabe der Optionen, die dem configure-Skript übergeben wurden, als Sie Postgres übersetzten
`--version`	Ausgabe der Versions-Nr. der Datenbank

pg_controldata

Hier erhalten Sie die Informationen, die vom Programm initdb erzeugt wurden. Das Programm können Sie nur unter der Benutzerkennung ausführen, mit der Sie den Datenbankcluster angelegt haben (Programm initdb).

Aufbau:

pg_controldata [Verzeichnis | Umgebungsvariable PGDATA]

pg_ctl

Ist ein Utility zum Starten, Beenden, Restart oder Statusabfrage des PostgeSQL Servers (Programm postmaster).

Aufbau:

Start des Servers:
pg_ctl start [-w] [-s] [-D Verzeichnis] [-l Datei] [-o Option] [-p Path]

Beenden des Servers:
pg_ctl stop [-W] [-s] [-D Verzeichnis] [-m s[mart] | f[ast] | i[mmdediate]]

Restart des Servers:
pg_ctl restart [-w] [-s] [-D Verzeichnis] [-m s[mart] | f[ast] | i [mmediate]]
[-o Option]

Konfigurationsdateien erneut laden:
pg_ctl reload [-s] [-D Verzeichnis]

Den aktuellen Status anzeigen:
pg_ctl status [-D Verzeichnis]

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	***Bedeutung***
`-D`	Das Verzeichnis mit den Datenbankdateien. Hier können Sie auch die Umgebungsvariable PGDATA einsetzen
`-l (kleines L)`	Der Name der Logdatei. Wenn Sie die Attribute mit dem umask-Befehl auf 077 setzen, so können andere auch auf die Datei zugreifen. Normalerweise ist das nicht erlaubt
`-m Modus`	Die Modis smart, fast, oder immediate. Verwenden Sie hier den ersten Buchstaben der Modis

Option	*Bedeutung*
`-o Option`	Hier können Sie direkt an das Programm Postmaster Optionen weitergeben
`-p`	Das Verzeichnis, in dem sich das Programm Postmaster befindet. Meistens ist es mit dem Verzeichnis identisch, in dem sich auch das Programm pg_ctl befindet
`-s`	Gibt nur Fehler aus, keine Informationen
`-w (kleines W)`	Zeit, die das Programm warten soll, bis es die Datenbank startet oder beendet (mehr als 60 geht nicht)
`-W (großes W)`	Wartet nicht, bis das Herunterfahren oder der Start abgeschlossen. Das ist die Grundeinstellung beim Start oder Restart

Beispiele:
pg_ctl start
pg_ctl -o „-p 5432“ start

Umgebungsvariable
PGDATA

pg_resetxlog

Löscht den Write-Ahead-Log (WAL) und bei Bedarf auch Einträge in der Datei pg_control. Die Funktion ist notwendig, wenn diese Dateien beschädigt sind. Sie sollten dieses Programm nur als letzten Ausweg nehmen, wenn der Server wegen dieser Datei nicht mehr startet.
Da nach dieser Aktion ggf. noch inkonsistente Daten vorhanden sind, sollten Sie die Daten mit pg_dump sichern, initdb ausführen und einen reload duchführen. Das Programm können Sie nur als der Benutzer ausführen, der den Cluster initialisierte. ACHTUNG: Das Programm dürfen Sie nicht ausführen, wenn der Server läuft. Sollte der Server abgestürzt sein, dann entfernen Sie die Sperrdatei. Achten Sie unbedingt darauf, dass kein anderes Postgres-Programm mehr läuft.

Aufbau:
pg_restxlog [OPTIONEN] Verzeichnis

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	*Bedeutung*
`-f`	Wenn sich in der Datei pg_control ungültige Daten befinden, dann werden sie durch plausible Informationen ersetzt (VORSICHT: danach sofort sichern und wiederherstellen).
`-n`	Gibt nur die Daten aus, die aus pg_contol stammen. An der Datenbank verändert das Programm nichts.

Option	*Bedeutung*

`-o oid`
`-x xid`
`-l ID, Seq.`
	Hier setzen Sie die nächste Transaktionsnummer, bzw. die WAL-Startadresse. Die nächste Transaktionsnummer ermitteln Sie aus dem Verzeichnis pg_clog. Nehmen Sie hier den höchsten numerischen Datennamen, addieren eine 1 und multiplizieren mit 1048576 (VORSICHT: Die Dateinamen sind hexadezimal). Wenn zum Beispiel 0011 der größte Eintrag in pg_clog ist, dann können Sie -x 0x1200000 verwenden. (Die fünf Nullen am Ende ergeben den richtigen Multiplikationsfaktor.) Die WAL-Startadresse sollte numerisch größer sein als jede Datei, die gegenwärtig im Verzeichnis pg_xlog unter dem Datenverzeichnis existiert. Die Adressen sind auch in hexadezimaler Form und haben zwei Teile. Wenn zum Beispiel der größte Eintrag in pg_xlog 000000FF0000003A ist, dann können Sie -l 0xFF,0x3B verwenden. Es gibt keinen einfachen Weg, die nächste OID, die größer ist als die größte in der Datenbank, zu ermitteln, aber glücklicherweise ist es nicht entscheidend, die nächste OID richtig zu bestimmen.

postgres

Das Programm führt den PostgresSQL Server im Einzelbenutzer Modus aus (während der Initialisierung (Programm initdb, oder Behebung von Problemen). Normalerweise wird es nicht direkt ausgeführt. Dafür gibt es die Mehrbenutzerversion (siehe Programm Postmaster).

Aufbau

postgres [Optionen] Datenbank

oder

postgres [Optionen] [halb interne Optionen]

Ihnen stehen folgende Optionen zur Verfügung (in den eckigen Klammern finden Sie die Kurzform der Option):

Option	***Bedeutung***
`-A 0|1`	Hilfe zur Fehlersuche, ist nur Verfügbar, wenn es bei Compilerlauf eingestellt wurde. In der Grund-einstellung ist es an
`-B Buffer`	Anzahl der Buffer für den Serverprozess. Per Grundeinstellung sind es 64 Buffer. Jeder hat eine Größe von 8 kB
`-c Name=Value`	Parameter für die Laufzeit-Konfiguration. Der Parameter kann mehrfach gesetzt werden
`-d Debug-Level`	Der Debug-Level. Gültige Werte sind von 1 bis 5
`-D Verzeichnis`	Das Datenverzeichnis
`-e`	Europäische Datumsangaben, also im Format TMJ
`-E`	Ausgabe aller Befehle, bevor sie ausgeführt werden
`-F`	Schaltet die Funktion fsync aus. Sie können den Wert auch in der Datei postgresql.conf setzen. VORSICHT: Es steigert die Leistung, aber ein Datenverlust ist beim Systemabsturz eher möglich
`-o Dateiname`	Alle Serverlog Meldungen kommen in diese Datei. Wenn das Programm unter dem Programm postmaster läuft, wird diese Option ignoriert und die Ausgabe erfolgt auf die Standard Fehlerausgabe vom Programm postmaster
`-N`	Schaltet das Neue-Zeile-Zeilen als Trennzeichen für Befehle aus. Wenn Sie diese Option einsetzen, haben Sie Probleme um die Anwendung zu beenden. Geben Sie 2 mal das EOF Zeichen ein (STRG+D)
`-P`	Die Systemindexe werden beim Durchsuchen und der Aktualisierung von Systemtabellen ignoriert. Verwenden Sie diese Optionen wenn Sie den Befehl REINDEX auf Systemtabellen/-indexe anwenden
`-s`	Zeitangaben und andere Informationen werden nach jedem Kommando ausgegeben

Option	*Bedeutung*
`-S`	Hier legen Sie fest, wie viel Speicher für interne Sortier- und Hash-Vorgänge das Programm nutzen darf, bevor die Informationen auf temporäre Dateien ausgelagert werden. Geben Sie den Wert in Kilobyte an (Grundeinstellung ist 1024 Kilobyte). Dieser Wert steht jeder einzelnen Abfrage oder jedem einzelnen Sortiervorgang zur Verfügung

Die nun folgende halbinterne Option hängen Sie an den Parameter -f an!

halb interne Option	*Bedeutung*
`s`	Schaltet sequenzielle Scans ab
`i`	Schaltet Index Scans ab
`m`	Schaltet Merge-Verbunde ab
`n`	Schaltet Nested-Loop-Verbunde ab
`h`	Schaltet Hash-Verbunde ab

halb interne Option	*Bedeutung*
`-i`	Fällt in der Version 7.4.6 weg
`-O`	Die Struktur der Systemtabellen kann geändert werden. Das Programm initdb nutzt diese Option
`-p Datenbank`	Der Prozess wurde vom Postmaster gestartet und verwendet eine vorgegebene Datenbank
`-t pa | pl | e`	Gibt die Zeit aus, die jede Abfrage an die Systemmodule benötigt. Die Option kann aber nicht im Zusammenhang mit der Option -s eingesetzt werden. Die Bedeutung der Optionen im Einzelnen: pa=Parser, pl=Planner, e=executor
`-v Protokoll`	Legt die Versions-Nr. des Protokolls fest, die für diese Sitzung verwendet wird

halb interne Option	*Bedeutung*
-W Sekunden	Die Anzahl von Sekunden, die der Prozess ruht. Als Entwickler können Sie die Zeit nutzen, um einen Debugger zu starten

Umgebungsvariable:

PGDATA

Wichtige Punkte beim Datenbankentwurf

Datentypen

Von anderen Datenbanken sind Ihnen auch einige Datentypen bekannt. In der unten gezeigten Tabelle finden Sie die einzelnen Datentypen. Als Anwender können Sie übrigens auch eigene Datentypen anlegen.

Name	***Alias***	***Beschreibung***
bigint	int8	8-Byte-Ganzzahl mit Vorzeichen
bigserial	serial8	selbstzählende 8-Byte-Ganzzahl
bit		Bitkette mit fester Länge
bit varying(*n*)	varbit(*n*)	Bitkette mit variabler Länge
boolean	bool	Boole'scher (logischer) Wert (wahr/falsch)
box		Rechteck
bytea		binäre Daten, mit variabler Länge
character varying(*n*)	varchar(*n*)	Zeichenkette mit variabler Länge
character(*n*)	char(*n*)	Zeichenkette mit fester Länge
cidr		IP-Netzwerkadresse
circle		Kreis in der Ebene
date		Kalenderdatum (Jahr, Monat, Tag)
double precision	float8	Fließkommazahl mit doppelter Präzision
inet		IP-Hostadresse
integer	int, int4	4-Byte-Ganzzahl mit Vorzeichen
interval(*p*)		Zeitspanne
line		Gerade in der Ebene (nicht voll implementiert)
lseg		endliche Strecke in der Ebene
macaddr		MAC-Adresse
money		Geldbetrag
numeric [(*p*, *s*)]	decimal [(*p*, *s*)]	exakte Zahl mit wählbarer Präzision

Name	***Alias***	***Beschreibung***
path		offener oder geschlossener Pfad in der Ebene
point		geometrischer Punkt in der Ebene
polygon		Polygon
real	float4	Fließkommazahl mit einfacher Präzision
smallint	int2	2-Byte-Ganzzahl mit Vorzeichen
serial	serial4	selbstzählende 4-Byte-Ganzzahl (z.B. lfd. Nr. beim Einfügen in die Datenbank
text		Zeichenkette mit variabler Länge
time [(*p*)] [without time zone]		Tageszeit
time [(*p*)] with time zone	timetz	Tageszeit mit Zeitzone
timestamp [(*p*)] without time zone	timestamp	Datum und Zeit
timestamp [(*p*)] [with time zone]	timestamptz	Datum und Zeit mit Zeitzone

TIPP: Wenn Sie sich den Aufbau einer Tabelle aus dem Programm psql ansehen möchten, dann geben Sie den Befehl ***\d <Tabellenname>*** ein (mit ***\dt*** sehen Sie alle Tabellen in der Datenbank).

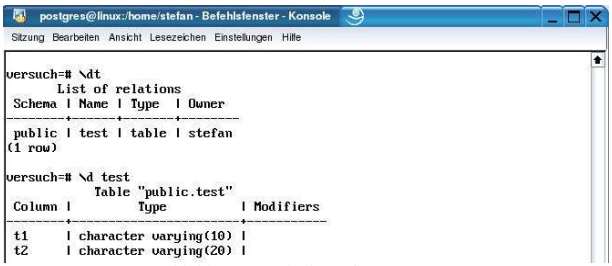


Abbildung 69 Tabellenaufbau anzeigen

Views

Mit Views (Sichten) können Sie z.B. bei umfangreichen oder komplizierten Select-Abfragen Arbeit bei der Eingabe des SQL-Befehls sparen. Zusätzlich können Sie auch anderen Anwendern eine Schnittstelle auf die Tabelle zur Verfügung stellen (z.B. eine View auf eine Tabelle, die Daten zur Lohn- und Gehaltsabrechnung enthält. Einige Anwender sollen nur auf Name, Anschrift und Geburtsdatum zugreifen können. Sie können auch mehrere Tabellen zu einer View zusammenfassen). Eine View legen Sie mit dem CREATE Befehl an, siehe Beispiel unten:

Aktion	***Inhalt***
View anlegen	CREATE VIEW personalview AS SELECT persnr, persgeschlecht, persvorname, persnachname, persstrasse, persplz, persort, persgeburtsdatum FROM personal
View anwenden	SELECT * FROM personalview where persnr=4711
View löschen	DROP personalview

Fremdschlüssel

Fremdschlüssel sind wichtig um die referentielle Integrität der Daten zu erhalten. Z.B. eine Datenbank im Verkauf enthält unter anderem 2 Tabellen für die Bestellungen. In einer Tabelle speichern wir die „Kopfdaten der Bestellung“ (Kunden-Nr. Verkäufer, Datum usw.) und in der anderen Tabelle die „Bestellpositionen“.

Aktion	Inhalt
Beim Tabellen anlegen	CREATE TABLE bestellkopf (bestellnr varchar(15) primary key, bestellvknr integer, bestellkunde varchar(10)); CREATE TABLE bestellposition (bestellnr varchar(15) references bestellkopf (bestellnr), artikelnr varchar(10), menge integer); **WICHTIG:** Mit der Angabe references stellen Sie eine Verbindung zwischen den beiden Tabellen her. Nach dem Zieldatenfeld (hier bestellnr) können Sie noch die Optionen ON DELETE RESTRICT (verhindert löschen), ON DELETE CASCADE (wenn der Bestellkopf gelöscht wird, werden auch die Bestellpositionen gelöscht). Der Fremdschlüssel muss immer entweder ein Primärschlüssel oder ein Unique Constraint in der anderen Tabelle sein! Im oben gezeigten Beispiel müssen Sie zuerst die Bestellpositionen und dann den Bestellkopf löschen.
Nachträglich ändern	Mit dem Alter-Befehl geht es leider nicht. Sie müssen die Tabelle neu anlegen (unter einem neuen Namen) und dann die Daten hineinkopieren (Stichwort: SQL-Befehl SELECT INTO). Im Anschluss die alte Tabelle löschen und dann die neue umbenennen.

Schemas

Zurzeit hat unsere Datenbank noch eine recht flache Struktur. Wir können z.B. alle Tabellen und Funktionen, die mit der Abteilung Verkauf zu tun haben, in einem Schema ablegen. Im Gegensatz zu der Datenbank sind die Schemas nicht strikt von einander getrennt. Wenn Sie kein Schema anlegen, dann greifen alle auf das Schema public zu.

Aktion	Inhalt
Schema anlegen	CREATE SCHEMA name [AUTHORIZATION benutzername]; Mit AUTHORIZATION können Sie als DB-Administrator für jemand anderes anlegen, lassen Sie es weg, dann wird es mit dem aktuell angemeldeten Benutzernamen gleichgesetzt.
Zugriff auf eine Tabelle in einem Schema	Schema.tabelle oder datenbank.schema.tabelle.
Schema löschen	DROP SCHEMA name (geht nur wenn es leer ist, ansonsten: DROP SCHEMA name CASCADE)
Schema-Suchpfad	Mit der Anweisung **SHOW search_path** wird die aktuelle Einstellung ausgegeben. Voreingestellt sieht es so aus: **$user, public.** Als erstes wird das Schema des Users durchsucht (soweit vorhanden) und dann das public. Mit der Anweisung **SET search_path TO name,public;** ändern wir die Einstellung vom Suchpfad.
Tabelle von einem Schema zu einem anderen Schema verschieben	Nur über den Weg Neuanlage der Tabelle im Zielschema und die Daten kopieren.

Funktionen

allgemeines

Bei der Datenbank PostgreSQL gibt es schon von Haus aus verschiedene Funktionen, siehe auch Anlage D Funktionen. Bei Bedarf können Sie auch bei dieser Datenbank eigene Funktionen schreiben. Die Funktionen können Sie einfach in SQL-Befehle einbauen (z.B. Sie möchten wissen wie viele Datensätze sich in der Tabelle Kunden befinden, dann geben Sie den Befehl SELECT COUNT(*) FROM KUNDEN ein, die Funktion COUNT (siehe Anlage D Aggregatfunktion) zählt die Anzahl der Datensätze). Sie können auch zusätzlich eigene Funktionen hinzufügen. Ein wesentlicher Vorteil der Funktionen ist der, dass sie direkt auf dem Server ausgeführt werden. PostgreSQL bietet Ihnen dazu 4 Arten von Funktionen:

Art	***Inhalt***
Mit SQL geschriebene Funktion	Hier hinterlegen Sie eine beliebige Liste von SQL-Befehlen. Die Funktion können Sie dann z.B. über den SELECT-Befehl aufrufen. Nähere Informationen können Sie dem Punkt SQL-Funktionen entnehmen.
Mit prozeduralen Sprachen	Mit prozeduralen Sprachen wie PL/pgSQL, PL/Tcl, PL/Perl und PL/Python. Ggf. müssen Sie noch die Srache auf der Datenbank installieren. Geben Sie dazu den Befehl ***createlang plpgsql <<Name der Datenbank>>*** ein (plpgsql steht für die Sprache PL/pgSQL, führen Sie diesen Befehl von der Kommandozeile aus). Mit ***droplang plpgsql <<Name der Datenbank>>*** löschen Sie übrigens die Sprache.
Interne Funktionen	Es sind interne C-Funktionen, die statisch in das Programm eingebunden wurden.
C-Funktionen	Diese Funktionen sind dynamisch ladbare Objekte und werden bei Bedarf geladen.

SQL-Funktionen

Eine SQL-Funktion ist recht einfach aufgebaut. Sie enthält 3 Teile:

Teil	Inhalt
Kopfteil	***CREATE OR REPLACE FUNCTION <<Name>>(<<Parameter>>) RETURNS <<Art>> AS '*** Bei den Parametern geben Sie keine Variablen an, sondern nur die Datentypen (zum Beispiel integer, text usw.). Innerhalb der SQL-Befehle verwenden Sie die Parameter-Nr. und setzen der Nummer ein $-Zeichen voran. Die Funktion hat keinen Rückgabewert. Dann setzen Sie als Art den Wert void ein.
Inhalt / Aktion	Der oder die SQL-Befehle.
Endteil	***' LANGUAGE SQL;***

Hier ein Muster (***TIPP:*** Verwenden Sie immer CREATE OR REPLACE im Kopfteil, Sie müssen dann die Funktion bei einer Änderung nicht vorher löschen):

```
CREATE OR REPLACE FUNCTION Zusammen(text, text) RETURNS text AS '
SELECT $1 || $2;
' LANGUAGE SQL;
```

Speichern Sie die Anweisung in einer Datei ab und starten das Programm psql. Mit dem psql Kommando \i [Dateiname] legen Sie die Funktion an. Die Funktion aufrufen geht so:

```
test=# select zusammen('das ','geht');
 zusammen
----------
 das geht
(1 row)

test=# _
```

Abbildung 70 einfache SQL-Funktion

Anstelle einer Tabelle können Sie auch in einer Select-Abfrage eine Funktion angeben.

PL/pgSQL-Funktionen

Da mit der Zeit der Umfang der SQL-Befehle nicht ausreichte, entwickelte man eine weitere Sprache, die auf den SQL-Befehlen aufbaute. Bevor Sie PL/pgSQL-Funktionen in einer Datenbank implementieren, sollten Sie sich vergewissern, dass die Sprache auch auf der Datenbank installiert wurde. ANMERKUNG/TIPP: Möchten Sie Variablen Texte zuweisen, oder auch Funktionen, dann verwenden Sie das Hochkomma zwei Mal (z.B. ***RETURN "WICHTIG"***; oder ***a := "MÜNCHEN"***).

Teil	Inhalt
Kopfteil	***CREATE OR REPLACE FUNCTION <<Name>> (<<Parameter>>) RETURNS <<Art>> AS '*** Bei den Parametern geben Sie keine Variablen an, sondern nur die Datentypen (zum Beispiel integer, text usw.). Innerhalb der SQL-Befehle verwenden Sie die Parameter-Nr. und setzen der Nummer ein $-Zeichen voran. Hat die Funktion keinen Rückgabewert, dann setzen Sie als Art den Wert void ein.
Deklarationen	***DECLARE*** << die einzelnen Variablen und der Datentyp>> Die Syntax der Deklaration sieht so aus: Name [CONSTANT] Type [NOT NULL] [:= Wert]; Die Information CONSTANT steht für einen konstanten Wert. Weisen Sie einer Variablen den Wert NULL zu, die allerdings mit NOT NULL definiert wurde, dann erhalten Sie einen Laufzeitfehler. Schon während der Deklaration können Sie den Variablen einen Wert zuweisen (siehe nachfolgendes Beispiel), oder auch einen Bezug auf den Eingabeparameter nehmen. Record-Variablen sind Platzhalter. Sie nehmen die Struktur einer von einem SELECT- oder FOR-Befehl zugewiesenen Zeile an.
Inhalt / Aktion	Der oder die SQL-Befehle. Dieser Teil muss mit dem Begriff ***BEGIN*** anfangen und mit dem Begriff ***END;*** beendet werden. Vor dem Ende sollten Sie aber noch den Rückgabewert festlegen.
Endteil	***' LANGUAGE 'plpgsql';***

Die Funktion können Sie recht einfach mit einem Editor erstellen. Ein Import in die Datenbank verläuft aus dem Programm psql so:

\i Dateiname

Hier ein Muster:

```
CREATE OR REPLACE FUNCTION mwst_voll(real) RETURNS real AS '
DECLARE
 betrag ALIAS FOR $1;
 steuer REAL := 1.16;
BEGIN
 return betrag * steuer;
END;
' LANGUAGE SQL;
```

Einen Auszug verschiedener Möglichkeiten entnehmen Sie der unten gezeigten Tabelle:

Möglichkeit	Befehle
Zuweisung von Werten	Variable := Wert; z.B.: `mwst := 16;`
Ausführen von Ausdrücken ohne Rückgabewert	PERFORM abfrage; mit dieser Möglichkeit führen Sie Anfragen aus und verwerfen das Ergebnis. `PERFORM anfrage;`
Ausführen von dynamischen Abfragen	EXECUTE SQL-Befehl; diese Möglichkeit bietet sich an, wenn Sie einen SQL-Befehl aus festen Werten und Variablen zusammensetzen. Z.B.: `EXECUTE „DELETE FROM KUNDEN WHERE KDID=“ || $1;`
Ergebnisstatus mit GET DIAGNOSTICS	Mit der Methode GET DIAGNOSTICS haben Sie eine Möglichkeit die Auswirkungen eines Befehls abzufragen. Zurzeit gibt es die Statuswerte ROW_COUNT (Anzahl der Zeilen vom letzten SQL-Befehl) und RESULT_OID (enthält die OID der letzten Zeile von einem INSERT-Befehl). Die Anweisung sieht so aus: `GET DIAGNOSTICS iAnzahlZeilen := ROW_COUNT;`

Möglichkeit	Befehle
Ergebnisstatus mit FOUND	Hier haben Sie die zweite Möglichkeit. FOUND ist ein Bool-Wert. SELECT INTO, UPDATE, INSERT, DELETE und FETCH setzen FOUND auf wahr, wenn mindestens eine Zeile zurückgegeben wurde, ansonsten enthält FOUND falsch. Genauso verhält es sich mit FOR, wenn die Schleife einmal durchlaufen wurde. FOUND ist übrigens eine lokale Variable.

Bedingungen	Es gibt mehrere Ausführungen:
	- IF ... THEN ... END IF;
	`IF mwst <> 0 THEN`
	` betrag := betrag * (100+mwst);`
	`END IF;`
	- IF ... THEN ... ELSE ... END IF;
	`IF zahl < 10 THEN`
	` zahl := zahl + 10;`
	`ELSE`
	` zahl := 100;`
	`END IF;`
	- IF ... THEN ... ELSE IF
	`IF demo_satz.geschlecht = „m“ THEN`
	` geschlecht := „Man“;`
	`ELSE`
	` IF demo_satz.geschlecht = „f“ THEN`
	` geschlecht := „FRAU“;`
	` END IF;`
	`END IF;`
	- IF ... THEN ... ELSEIF ... THEN ... ELSE
	`IF zahl = 0 THEN`
	` ergebnis := „NULL“;`
	`ELSEIF zahl > 0 THEN`
	` ergebnis := „positiv“;`
	`ELSEIF zahl < 0 THEN`
	` ergebnis := „negativ“;`
	`END IF;`

Rückkehr aus einer Funktion	Dies erreichen Sie mit dem Befehl RETURN <<Ausdruck>>. Der Begriff void steht für keinen Rückgabewert.
Kommentarzeile	Geben Sie hier vor den Kommentaren zwei Bindestriche ein. Z.B.: `-- das ist ein Kommentar`
Schleife verlassen	Dies können Sie mit dem Befehl EXIT erreichen.

Möglichkeit	Befehle

Einfache Schleife	Eine Möglichkeit ist der LOOP-Befehl. Die Schleife können Sie dann z.B. mit dem EXIT-Befehl
	`LOOP`
	`  ...`
	`END LOOP`

Schleife mit Abbruchbe-dingung	Mit der While Schleife können Sie eine Abbruchbedingung einbauen.
	`WHILE zahl > 10 LOOP`
	`  ...`
	`END LOOP`

FOR-Schleife	In der FOR-Schleife können Sie sowohl hochzählen als auch herunter (REVERSE) zählen. Der Aufbau sieht so aus (übrigens anstelle von des Wortes to gibt es hier 2 Punkte):
	FOR variable IN [REVERSE] Ausdruck1 .. Ausdruck2 LOOP
	...die Anweisungen
	END LOOP;
	`FOR i IN 1..10 LOOP`
	`  ...`
	`END LOOP`
	
	`FOR i IN REVERSE 10..1 LOOP`
	`  ...`
	`END LOOP`

Möglichkeit	Befehle

Cursor	Möchten oder müssen Sie Anfrageergebnisse Zeile für Zeile abarbeiten, dann verwenden Sie in diesem Fall einen Cursor. Definieren Sie die Cursorvariable mit dem Datentyp ***refcursor***, oder auch so: Name ***CURSOR [(Argumente)] FOR Anfrage;*** Bei mehreren Argumenten trennen Sie die Liste durch Kommas. Hier haben Sie einige Beispiele:
	`DECLARE`
	`curs1 refcursor; // der Cursor ist ungebunden, jede Anfrage kann verwendet werden`
	`curs2 CURSOR FOR SELECT * FROM KUNDEN; // der Cursor ist gebunden`
	`curs3 CURSOR (key integer) IS SELECT * FROM KUNDEN WHERE kdid=key; // ist auch gebunden, die Information key wird durch einen Integer-Parameter ersetzt.`
	Um die Zeilen abzurufen, öffnen Sie mit dem Befehl OPEN den Cursor. Bei einem ungebundenen Cursor haben Sie zwei Möglichkeiten:
	OPEN <<ungebundener Cursor>> FOR SELECT ...;
	OPEN <<ungebundener Cursor>> FOR EXECUTE <<Zeichenkette>>;
	Bei dem gebundenen Cursor geht es noch einfacher:
	OPEN curs2;
	OPEN curs3(4711); // hier übergeben wir einen Parameter
	Mit der Anweisung FETCH übertragen Sie eine Zeile aus dem Cursor in eine oder mehrere Variablen. Der Aufbau sieht so aus:
	FETCH curs1 INTO a;
	FETCH curs2 INTO a, b, c;
	Mit ***CLOSE*** beenden Sie die Abfrage: ***CLOSE curs1;***
	Der Rückgabewert einer Funktion kann übrigens auch ein Cursor sein.

Fehler und Meldungen	Mit der Anweisung ***RAISE*** erzeugen Sie entweder Einträge im Serverlog, senden Mitteilungen an den Client, oder brechen die aktuelle Transaktion mit einem Fehler ab. Der Aufbau sieht wie folgt aus: ***RAISE level 'Format' [, Variable [, ...]];*** Die Level haben folgende Auswirkungen: ***DEBUG*** und ***LOG*** - Meldung mit unterschiedlicher Priorität in den Serverlog schreiben ***NOTICE*** und ***WARNING*** – Eintrag in den Serverlog und an den Client senden (mit unterschiedlicher Priorität) ***EXCEPTION*** – Transaktion abbrechen und Fehlermeldung erzeugen

Trigger

Trigger reagieren auf Ereignisse. Ein Ereignis ist zum Beispiel, wenn Sie einen Datensatz in eine Tabelle einfügen. Hier können Sie beispielsweise das Datum der Neuanlage eines Datensatzes in einem Datenfeld festhalten. Den Trigger können Sie entweder vor dem Ereignis (INSERT, UPDATE oder DELETE) oder auch nach dem Ereignis aktiv werden lassen. Der Trigger ruft dann eine Funktion auf.

Wenn Sie mit PL/pgSQL Triggerprozeduren schreiben, dann haben Sie noch mehrere zusätzliche Variablen:

Name	Inhalt
NEW	Datentyp RECORD, enthält die neue Tabellenzeile bei INSERT/UPDATE-Operationen auf Zeilenebene.
OLD	Wie NEW, enthält aber die alte Tabellenzeile bei DELETE/UPDATE-Operationen.
TG_NAME	Datentyp NAME, die Variable enthält den Namen des ausgeführten Triggers
TG_WHEN	Datentyp TEXT, beinhaltet die Zeichenkette BEFORE oder AFTER, abhängig von der Definition des Triggers.
TG_LEVEL	Datentyp TEXT, steht für die Zeichenkette ROW oder STATEMENT, hängt vom Trigger ab.
TG_OP	Datentyp TEXT, ist die Zeichenkette, die angibt, aus welcher Operation der Trigger ausgelöst wurde.
TG_RELID	Datentyp OID, enthält die OID der Tabelle, die den Trigger ausgelöst hat.
TG_RELNAME	Datentyp NAME, diese Variable enthält den Namen der Tabelle, die den Trigger ausgelöst hat.
TG_NARGS	Datentyp INTEGER, beinhaltet die Anzahl der Argumente, die der Triggerprozedur vom Befehl CREATE TRIGGER übergeben wurden.
TG_ARGV[ ]	Ein Array aus Texten, enthält die Argumente vom CREATE TRIGGER-Befehl. Der erste Eintrag hat den Index 0.

Der Rückgabewert einer Triggerfunktion muss entweder den NULL-Wert, oder einen Record- bzw. Zeilentypwert haben. Er gleicht der Struktur der

Tabelle, die den Trigger ausgelöst hat. Gibt ein BEFORE-Trigger einen NULL-Wert zurück, dann löst der Triggermanager weitere Operationen für die Zeile nicht mehr aus. Gibt die Funktion dagegen einen anderen Wert zurück, dann laufen die Aktionen weiter. Kommt bei einer NEW-Aktion ein Zeilenwert zurück, der sich vom ursprünglichen Wert unterscheidet, dann verändert sich die Zeile, die eingefügt oder aktualisiert wird. Einzelne Werte können Sie direkt in NEW ersetzen und NEW zurückgeben.

Einen Trigger realisieren Sie in 2 Schritten:

Schritt	Inhalt/Aktion
Vorab zur Info	Die Tabelle ist so aufgebaut: - vnr: varchar(5) - vname: varchar(30) - datum: date, das Tagesdatum, an dem der Datensatz geändert wurde - benutzer: varchar(30), der Benutzer, der zuletzt den Datensatz geändert hat

Funktion anlegen (Schritt 1)	Soweit notwendig, legen Sie eine neue Funktion an:
	`CREATE OR REPLACE FUNCTION Versuch_Datum_geaendert() RETURNS trigger AS '`
	`BEGIN`
	`-- Prüfe die Felder vnr und vname gefüllt sind`
	`IF NEW.vnr IS NULL THEN`
	`RAISE EXCEPTION ''Feld vnr nicht gefüllt'';`
	`END IF;`
	`IF NEW.vname IS NULL THEN`
	`RAISE EXCEPTION ''Feld vname nicht gefüllt'';`
	`END IF;`
	`-- Datum und Benutzer zuweisen`
	`NEW.datum := ''now'';`
	`NEW.benutzer := current_user;`
	`RETURN NEW;`
	`END;`
	`' LANGUAGE plpgsql;`

Trigger anlegen (Schritt 2)	Legen Sie nun den Trigger an:
	`CREATE TRIGGER Versuch_Datum_geaendert BEFORE INSERT OR UPDATE ON versuch`
	`FOR EACH ROW EXECUTE PROCEDURE Versuch_Datum_geaendert();`

Führen Sie nun einen kleinen Versuch durch, lassen Sie bei einer Insert-Operation ein Feld leer. Sie erhalten anschließend diese Meldung:

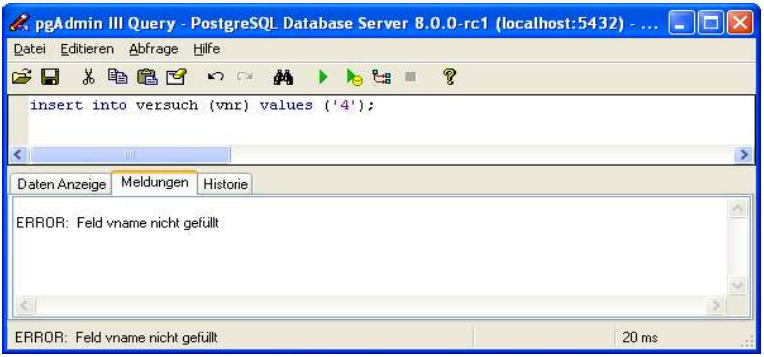


Abbildung 71 Trigger - Insert

Ein Update läuft dagegen ohne Probleme durch:

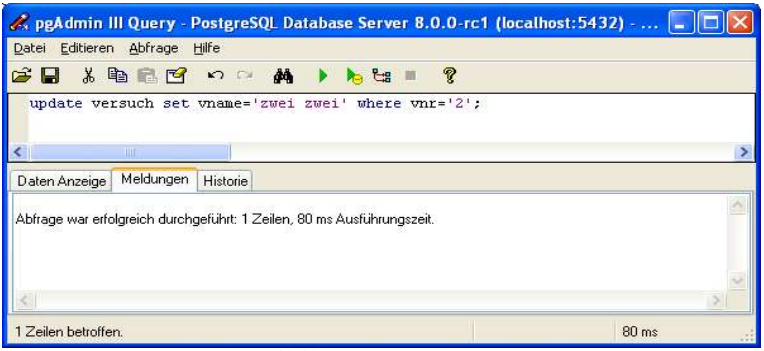


Abbildung 72 Trigger - Update

Constraints

Oftmals haben Sie das Problem, dass z.B. bei einer Insert-Anweisung Datenfelder unberücksichtigt bleiben (der Preis ist null usw.). Mit Contraints verhindern Sie das. So prüfen Sie mit dem Check-Constraints, ob ein Feld gefüllt ist.

Aktion	*Inhalt*
Check-Constrain anlegen	Bei der Neuanlage der Tabelle: **CREATE TABLE artikel (artikel_nr varchar(15),** **artikel_name varchar(25),** **artikel_preis numeric CHECK (artikel_preis>0));** Um eine Fehlermeldung übersichtlicher zu gestalten, schreiben Sie nach dem Wort CONSTRAIN die Fehlermeldung, siehe Beispiel: **CREATE TABLE artikel (artikel_nr varchar(15),** **artikel_name varchar(25),** **artikel_preis numeric CONSTRAIN positiver_preis (artikel_preis>0));** Eine Prüfung auf den Wert NULL sieht so aus: **CREATE TABLE artikel (artikel_nr varchar(15),** **artikel_name varchar(25) NOT NULL,** **artikel_preis numeric);** Geben Sie dagegen nur NULL an, dann bedeutet das, dass auch der Wert NULL zulässig ist. Mit dem Constraint UNIQUE verhindern Sie z.B. die doppelte Vergabe einer Artikel-Nr.
Constraint hinzufügen	Das funktioniert mit dem Befehl ALTER. Die Syntax sieht so aus: **ALTER TABLE [Tabelle] ADD CHECK ([Feld] <> '');** **ALTER TABLE [Tabelle] ADD CONSTRAIN doppelte_artikel_nr UNIQUE ([Feld]);** Da das NOT-NULL-Constraint nicht als Tabellen-Constraint geschrieben werden kann, führen Sie die Anweisung so aus: **ALTER TABLE [Tabelle] ALTER COLUMN [Feld] SET NOT NULL;**
Constraint entfernen	Alle Constraints, außer NOT-NULL-Constraints können Sie so entfernen: **ALTER TABLE [Tabellenname] DROP CONSTRAINT [Name];** Das NOT NULL so: **ALTER TABLE [Tabellenname] ALTER COLUMN [Feld] DROP NOT NULL;**

Zugriff mit Programmiersprachen und Programmen

Mit C/C++/DOT.NET auf die Datenbank zugreifen

Mit den Programmiersprachen C, C++ und DOT.NET können Sie auf die Informationen der Datenbank zugreifen.

C-Zugriff von Unix

Von Unix/LINUX funktioniert es recht einfach mit der Bibliothek libpg. Diese Bibliothek enthält verschiedene C-Funktionen. Der Compilerlauf sieht so aus:

gcc -o <<Programm>> <<Programm, incl. Erweiterung>> -L/usr/lib -lpq

Der Parameter -L/usr/lib steht für den Namen des Verzeichnisses, in dem sich die Bibliothek libpg befindet. In der nachfolgenden Tabelle finden Sie einen Auswahl der Möglichkeiten die Ihnen die Bibliothek bietet.

Punkt	***Inhalt/Auszug aus Programmen***
Die Header-Datei	`#include "/usr/include/pgsql/libpq-fe.h"`

Anmelden auf der Datenbank, mit Pqsetdb	Alle notwendigen Informationen übergeben Sie entweder der Funktion Pqsetdb (ist ein Makro, mit Null-Zeigern für Name und Passwort), oder der Funktion PQsetdbLogin:
	`char *pghost, // Name oder IP-Adresse des Hosts mit der Datenbank`
	`*pgport, // die Port-Nr. der Datenbank`
	`*pgoptions, // Konfigurationsoption, geht an den Server`
	`*pgtty; // eine Datei, ein TTY für Debug-Informationen vom Server`
	`char *dbName; // Name der Datenbank`
	`PGconn *conn;`
	`PGresult *res;`
	`...`
	`pghost = "PC954SUSE";`
	`pgport = "5432";`
	`pgoptions = "";`
	`pgtty = "";`
	`dbName = "neu";`
	`conn = Pqsetdb(pghost,pgport,pgoptions,pgtty,dbName);`

Punkt	*Inhalt/Auszug aus Programmen*

Anmelden auf der Datenbank mit PqsetdbLogin	In diesem Beispiel benötigen Sie noch zusätzlich den Benutzernamen und ein Passwort:
	`char *pghost, // Name oder IP-Adresse des Hosts mit der Datenbank`
	`*pgport, // die Port-Nr. der Datenbank`
	`*pgoptions, // Konfigurationsoption, geht an den Server`
	`*pgtty; // eine Datei, ein TTY für Debug-Informationen vom Server`
	`char *dbName; // Name der Datenbank`
	`char *login; // Name des Anwenders auf der Datenbank`
	`char *pwd; // das Passwort des Anwenders`
	`...`
	`pghost = "PC954SUSE";`
	`pgport = "5432";`
	`pgoptions = "";`
	
	`pgtty = "";`
	`dbName = "test";`
	`login = "postgres";`
	
	`/* Zugriff auf die Datenbank */`
	`conn = PQsetdbLogin (pghost,pgport,pgoptions,pgtty,dbName,login,pwd);`

Verbindungs-status abfragen	Über Funktion Pgstatus lässt sich die Information abfragen:
	`if (PQstatus(conn) == CONNECTION_BAD)`
	`{`
	`printf("Verbindungsfehler:%s\n",PQerrorMessage (conn));`
	`}`

SQL-Befehl ausführen, Funktionen aufrufen	Im hier gezeigten Beispiel tragen wir einen Datensatz in eine Tabelle ein (über Pqexec können Sie aber auch Funktionen aufrufen):
	`PGresult *res;`
	`...`
	`res = PQexec(conn,"insert into kunden (KDNR,KDNAME) values ('9999','weg')");`

Punkt	*Inhalt/Auszug aus Programmen*

Prüfen, ob der SQL-Befehl erfolgreich ausgeführt wurde	Die Abfrage kann gleich nach dem SQL-Befehl kommen:
	`PGresult *res;`
	`...`
	`if (!res || PQresultStatus(res)!=PGRES_COMMAND_OK)`
	`{`
	`  printf("FEHLER beim insert\n");`
	`}`
	
	WICHTIG: Neben PGRES_COMMAND_OK gibt es natürlich auch noch andere Werte:
	– PGRES_EMPTY_QUERY: Eine leere Zeichenkette wurde an den Server gesendet
	– PGRES_TUPLES_OK: Die Anfrage wurde erfolgreich ausgeführt
	– PGRES_COPY_OUT: Copy-Datentransfer vom Server gestartet (ausgehender)
	– PGRES_COPY_IN: Copy-Datentransfer zum Server gestartet (eingehender)
	– PGRES_BAD_RESPONSE: Die Antwort des Servers wurde nicht verstanden
	– PGRES_NONTATAL_ERROR: Ein Fehler trat auf
	– PGRES_FATA_ERROR: Ein schwerer Fehler trat auf

Anzahl der betroffenen Datensätze ermitteln	Z.B. der Rückgabewert aus einer SELECT-Anweisung:
	`int  ntuples;`
	`PGresult *res;`
	`...`
	`ntuples = PQntuples(res);`

Anzahl der Spalten aus einer SELECT-Abfrage ermitteln	Diese Information ist wichtig, wenn Sie die Daten auslesen möchten:
	`PGresult *res;`
	`int nFields;`
	`...`
	`nFields = PQnfields(res);`

Spaltenname aus der Abfrage ermitteln	Als Rückgabewert erhalten Sie einen String mit dem Spaltennamen. Die Nummer fängt bei 0 an.
	`PGresult *res;`
	`...`
	`printf("%-15s", PQfname(res,i));`

Feldinhalte abfragen	Das kurze Beispiel ist dem Programm Kundenliste entnommen. Die Variable i steht für die Anzahl der Datensätze und j für die Anzahl der Felder (wichtig i und j beginnen bei 0):
	`PGresult *res;`
	`...`
	`printf("%-15s",PQgetvalue(res,i,j));`

Punkt	*Inhalt/Auszug aus Programmen*

SQL Resultate löschen	Wenn Sie Ihr Ergebnis aus dem SQL-Befehl nicht mehr benötigen:
	`PGresult *res;`
	`...`
	`PQclear(res);`

Fehlermeldung ausgeben	Das hier gezeigte Beispiel finden Sie auch im Programm Daten einfügen.
	`PGresult *res;`
	`...`
	`printf("Fehler bei Insert: %s\n",PQresultErrorMessage(res));`

Verbindung zur Datenbank beenden	Beenden Sie so die Verbindung:
	`PGconn *conn;`
	`...`
	`PQfinish(conn);`

Programm Kundenliste

Das hier gezeigte Programm ist einem Beispiel aus der Postgres-Dokumentation angelehnt:

ANMERKUNG: Bei einigen Programmbeispielen befindet sich die Fortsetzung einer Zeile auf einer 2. Zeile, siehe Beispiel unten:

```
    →       →   //·(pg_database·Zugriff·auf·die·Systemtabellen)¶
    →       →   res·=·PQexec(conn,"DECLARE·mycursor·CURSOR·FOR·SELECT
kdnr,·kdname,·kdort·FROM·kunden");·//·eine·Zeile!¶
```

```
#include <stdio.h>
#include "/usr/include/pgsql/libpq-fe.h"

/*
* Demo-Programm: Ausgabe einer Kundenliste
*/

// Die Verbindung zur Datenbank beenden
void exit_nicely(PGconn *conn)
{
        PQfinish(conn);
        exit(1);
}
```

```
main()
{
        char    *pghost,
                *pgport,
                *pgoptions,
                *pgtty;
        char    *dbName;
        int     nFields;
        int     ntuples;
        int     i,
                j;
        Pgconn   *conn;
        PGresult *res;

        printf("Ausgabe der Kundenliste\n");

/*
* Zugriff auf die Datenbank
*/
        pghost = "PC954SUSE";
        pgport = "5432";
        pgoptions = "";
        pgtty = "";
        dbName = "test";

        /* Zugriff auf die Datenbank */
        conn = PQsetdb(pghost,pgport,pgoptions,pgtty,dbName);
        /* prüfen, ob die Verbindung erfolgreich war */
        if (PQstatus(conn) == CONNECTION_BAD)
        {
                printf("Fehler:%s\n",PQerrorMessage(conn));
        }
        else
        {
                // die Zeilen von der pg_database holen
                // (pg_database Zugriff auf die Systemtabellen)
                res = PQexec(conn,"DECLARE mycursor CURSOR FOR SELECT
kdnr, kdname, kdort FROM kunden");
                if (!res||PQresultStatus(res)!=PGRES_COMMAND_OK)
                {
                        printf("Fehler bei DECLARE CURSOR command
failed\n");
                        PQclear(res);
                        exit_nicely(conn);
                }
                PQclear(res);
                res = PQexec(conn,"FETCH ALL in mycursor");
                if (!res||PQresultStatus(res)!=PGRES_TUPLES_OK)
                {
                        printf("Fehler bei FETCH ALL\n");
                        PQclear(res);
                        exit_nicely(conn);
                }
```

```
                /* zuerst die Feldnamen ausgeben */
                nFields = PQnfields(res);
                for(i=0;i<nFields;i++)
                        printf("%-15s", PQfname(res,i));
                printf("\n");
                /* den Inhalte ausgeben */
                // Anzahl der Datensätze
                ntuples = PQntuples(res);
                printf("\n[Anzahl Einträge:%d]\n",ntuples);
                // den Inhalt der Datenfelder
                for(i=0;i<ntuples;i++)
                {
                        for(j=0;j<nFields;j++)
                        {
                                printf("%-15s",PQgetvalue(res,i,j));
                        }
                        printf(" %d\n",i);
                }
                PQclear(res);
                /* Curosr schließen */
                res=PQexec(conn,"CLOSE mycursor");
                PQclear(res);
        }
        printf("\nENDE\n");
        PQclear(res);
        exit_nicely(conn);
}
```

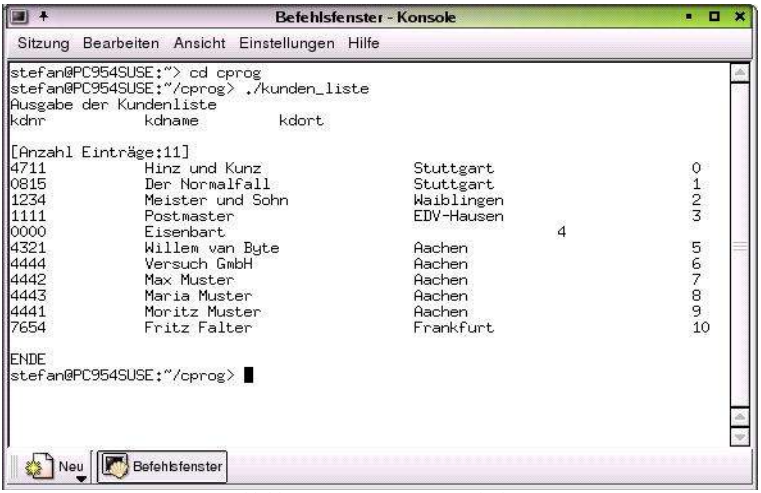


Abbildung 73 C-Programm-Kundenliste

Programm Datensatz einfügen

Im nachfolgenden Beispiel trägt das Programm einen Datensatz in die Tabelle ein. Von diesem Programm existieren 2 Versionen. In einer habe ich einen nicht vorhandenen Spaltennamen angegeben (Spalte kname). In der zweiten Version wurde der Spaltenname richtig geschrieben (Spalte kdname).

```
#include <stdio.h>
#include "/usr/include/pgsql/libpq-fe.h"

/*
* Datensatz eintragen
*/

void Programm_beenden(PGconn *conn)
{
        PQfinish(conn);
        exit(1);
}

main()
{
        char    *pghost,
                *pgport,
                *pgoptions,
                *pgtty;
        char    *dbName;
        char    *login;
        char    *pwd;
        int     nFields;
        int     ntuples;
        int     i,
                j;

        PGconn  *conn;
        PGresult *res;

        printf("Programm Datensatz einfügen\n");

/*
* Zugriff auf die Datenbank
*/
        pghost = "PC954SUSE";
        pgport = "5432";
        pgoptions = "";

        pgtty = "";
        dbName = "test";
        login = "postgres";
```

```
        /* Zugriff auf die Datenbank */
        conn = PqsetdbLogin(pghost,
                pgport,pgoptions,pgtty,
                dbName,login,pwd);

        /* prüfen, ob die Verbindung erfolgreich war */
        if (PQstatus(conn) == CONNECTION_BAD)
        {
                printf("Fehler:%s\n",PQerrorMessage(conn));
        }
        else
        {
                /* start des Transaktionsblockes */
                res = PQexec(conn,"BEGIN");
                if (!res || PQresultStatus(res)!=PGRES_COMMAND_OK)
                {
                        printf("FEHLER beim BEGIN Befehls\n");
                }
                else
                {
                        // löscht PGresults, wenn sie nicht
                        // mehr benötigt werden
                        PQclear(res);
                        res = PQexec(conn,"insert into kunden
(kdnr,kdname,kdort) values ('00071','Karl Berg','München')");
                        if (!res||PQresultStatus(res)
                                !=PGRES_COMMAND_OK)
                        {
                                printf("Fehler bei Insert: %
s\n",PQresultErrorMessage(res));
                                PQclear(res);
                                Programm_beenden(conn);
                        }
                        else
                        {
                                printf("Datensatz eingefügt");
                                PQclear(res);
                                /* die Transaktion festschreiben */
                                res=PQexec(conn,"COMMIT");
                        }
                }
        }
        printf("\nENDE\n");
        PQclear(res);
        Programm_beenden(conn);
}
```

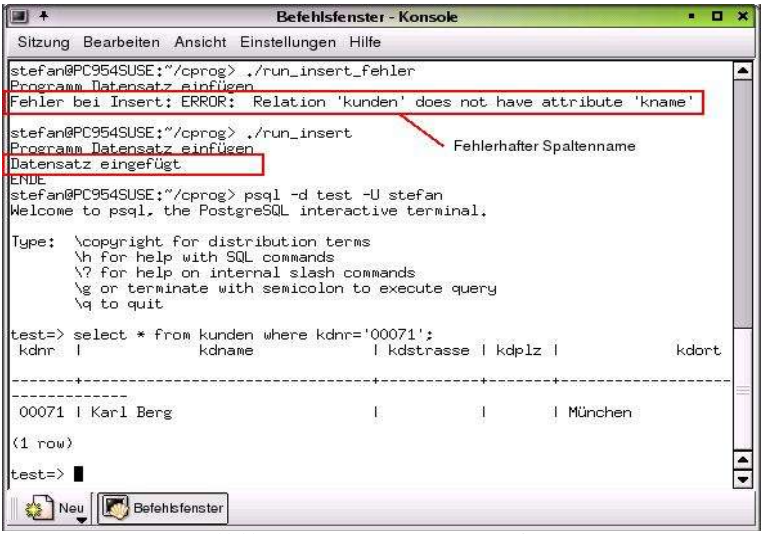


Abbildung 74 C-Programm-Datensatz einfügen

Embedded SQL

In C und C++-Programme können Sie auch SQL-Befehle einbauen. Über das Programm ecpg (siehe auch Kapitel Weitere Dienstprogramme) erhalten Sie dann den C-Code. Danach übersetzen Sie ihn mit Ihrem Compiler. Generell gilt hier: Stellen Sie jedem SQL-Befehl die Anweisung EXE SQL voran. In Ihrem Programm bauen Sie die unten aufgeführten Punkte unbedingt ein:

Punkt	***Inhalt***
Verbindung zum Datenbankserver	***EXEC SQL CONNECT TO Ziel [AS Verbindung] [USER Benutzername]*** Das Ziel beinhaltet diese Möglichkeiten: - dbname[@hostname] [:port] - tcp:postgresql://hostname[:port] [/dbname] [?optionen] - unix:postgresql://hostname[:port] [/dbname] [?optionen] - DEFAULT - Zeichenvariable - Zeichenketten Für den Benutzernamen haben Sie folgene Möglichkeiten: - Benutzername - Benutzername/Passwort - Benutzername IDENTIFIED BY Passwort - Benutzername USING Passwort Beide Informationen können SQL-Namen, Zeichenketten-variablen oder auch Zeichenkonstanten sein. Wenn Sie mehrere Verbindungen in einem Programm verwalten, dann setzen Sie Verbindungsnamen ein.
der SQL-Befehl	***EXEC SQL ....;*** Sie können hier auch mehrere SQL-Befehle angeben. Möchten Sie zum Beispiel die Ergebnisse eines SELECT-Befehles sequentiell abarbeiten, dann sehen Sie sich vom Stammdatenprogramm die Kundenliste an. Hier arbeiten Sie das Ergebnis mit einem Cursor ab.
Änderung festschreiben	***EXEC SQL COMMIT;*** dieser Befehl ist in der Standardeinstellung notwendig. Die Schnittstelle unterstützt auch Autocommit (Option -t beim Programm ecpg, oder mit dem Befehl EXEC SQL SET AUTOCOMMIT TO ON). **Ausnahme:** Sie verwenden einen expliziten Transaktionsblock. Mit dem Befehl EXEC SQL SET AUTOCOMMIT TO OFF schalten Sie den Autocommit-Modus aus.

Punkt	*Inhalt*
Verbindung schließen	***EXE SQL DISCONNECT [Verbindung];*** Die Option Verbindung beinhaltet folgende Werte: - Verbindungsnamen - DEFAULT - CURRENT - ALL
Fehler-behandlung	Funktioniert über den Fehlercode, fragen Sie entweder den Int-Wert der Variablen sqlca.sqlcode ab (siehe auch Programm Fehler), oder mit ***EXEC SQL WHENEVER sqlerror sqlprint;***

In dem nun folgenden Beispiel sehen Sie ein kleines C++-Programm, das in eine Datenbank einen Datensatz einträgt. Nach der Codierung arbeiten Sie 3 Schritte ab:

Schritt	*Inhalt*
1	Erzeugen Sie die C-Datei mit dem Programm ecpg: ***ecp <Dateiname incl Erweiterung>***
2	Übersetzen Sie Ihr C-Programm aus dem Schritt 1: ***g++ -I/<Verzeichnis der Include-Dateien für Postgres> -c <Dateiname incl. Erweiterung .c>***
3	Im letzten Schritt erstellen Sie die ausführbare Datei: ***g++ -o <Name der ausführbaren Datei> <Dateiname aus dem Schritt 2 incl. Erweiterung .o> -L<Verzeichnis der Include-Dateien für Postgres> -lecp***

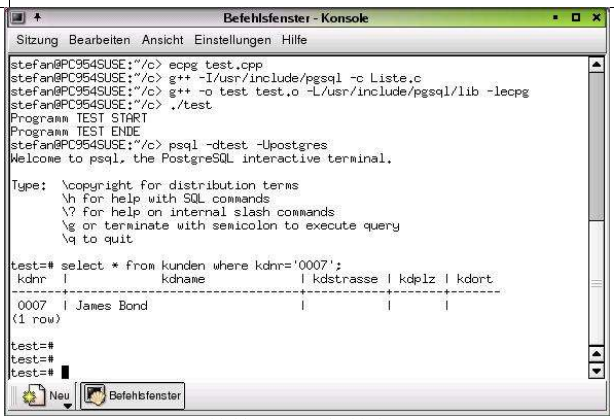


Abbildung 75 Embedded SQL

Das C++-Programm sieht so aus:

```
//
// Testprogramm
//
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
	int iZahl;

	cout << "Programm TEST START" << endl;
	// Verbindung zur Datenbank
	EXEC SQL CONNECT TO test@localhost USER postgres;
	EXEC SQL INSERT INTO KUNDEN (kdnr,kdname) values
('0007','James Bond');
	EXEC SQL COMMIT;
	// wieder abmelden
	EXEC SQL DISCONNECT;
	cout << "Programm TEST ENDE" << endl;
	return 0;
}
```

Das C-Programm von ecpg sieht so aus:

```
/* Processed by ecpg (2.9.0) */
/* These three include files are added by the preprocessor */
#include <ecpgtype.h>
#include <ecpglib.h>
#include <ecpgerrno.h>
#include <sqlca.h>
#line 1 "test.cpp"
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
	int iZahl;

	cout << "Programm TEST START" << endl;
	{ ECPGconnect(__LINE__, "test@localhost" , "postgres" , NULL ,
NULL, 0); }
#line 12 "test.cpp"

	{ ECPGdo(__LINE__, NULL, "insert into KUNDEN ( kdnr  , kdname
) values ( '0007' , 'James Bond' )", ECPGt_EOIT, ECPGt_EORT);}
#line 13 "test.cpp"

	{ ECPGtrans(__LINE__, NULL, "commit");}
#line 14 "test.cpp"

	{ ECPGdisconnect(__LINE__, "CURRENT");}
#line 15 "test.cpp"
```

```
        cout << "Programm TEST ENDE" << endl;
        return 0;
}
```

In unserem Testprogramm haben wir bisher noch keine Fehler abgefangen. Diesen Punkt holen wir nun nach und bauen in das neue Programm einen Fehler ein (eine nicht existierende Tabelle). Das Programm Fehler1.cpp gibt nur die Fehlerursache aus. Dagegen behandeln wir im Programm Fehler2.cpp den Fehler optimaler.

```
//
// Programm Fehler1.cpp
// (falscher Tabellenname)
//
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
        cout << "Programm FEHLER1" << endl;
        cout << "START" << endl;
        // eine sehr einfach Fehlerbehandlung
        EXEC SQL WHENEVER sqlerror sqlprint;
        EXEC SQL CONNECT TO test@localhost USER postgres;
        EXEC SQL INSERT INTO KUDNEN (kdnr,kdname) values
('0007','James Bond');
        EXEC SQL COMMIT;
        EXEC SQL DISCONNECT;
        cout << "ENDE" << endl;
        return 0;
}

//
// Programm Fehler2.cpp
//
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
        cout << "Programm FEHLER2" << endl;
        cout << "START" << endl;
        EXEC SQL CONNECT TO test@localhost USER postgres;
        EXEC SQL INSERT INTO KUDNEN (kdnr,kdname) values
('0007','James Bond');
        if (sqlca.sqlcode==0)
        {
                cout << "Datensatz eingefügt" << endl;
        }
```

```
        else
        {
                cout << "Fehler beim einfügen" << endl;
                cout << "Fehler-Nr.:" << sqlca.sqlcode << endl;
        }
        EXEC SQL COMMIT;
        EXEC SQL DISCONNECT;
        cout << "ENDE" << endl;
        return 0;
}
```

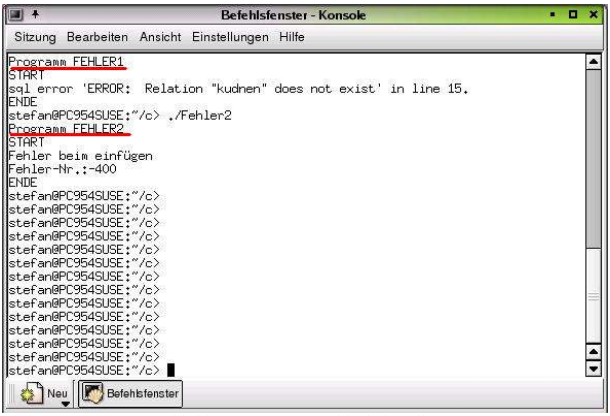


Abbildung 76 ecpg - Fehlerbehandlung

Das struct aus dem Programm Fehler2.cpp sieht so aus:

```
struct sqlca
{
    char sqlcaid[8];
    long sqlabc;
    long sqlcode;
    struct
    {
        int sqlerrml;
        char sqlerrmc[70];
    } sqlerrm;
    char sqlerrp[8];

    long sqlerrd[6];
    /* 0: leer                                          */
    /* 1: OID der aktuellen Zeile, wenn angebracht      */
    /* 2: Anzahl betroffener Zeilen bei INSERT, UPDATE  */
    /*    oder DELETE Befehl                            */
    /* 3: leer                                          */
    /* 4: leer                                          */
    /* 5: leer                                          */
```

```
    char sqlwarn[8];
    /* 0: 'W' wenn mindestens ein weiteres Feld 'W' ist */
    /* 1: wenn 'W' dann wurde mindestens eine Zeichen-  */
    /*    kette beim Speichern in eine Hostvariable     */
    /*    abgeschnitten                                 */
    /* 2: leer                                          */
    /* 3: leer                                          */
    /* 4: leer                                          */
    /* 5: leer                                          */
    /* 6: leer                                          */
    /* 7: leer                                          */

    char sqlext[8];
} sqlca;
```

Die vielen leeren Felder sind für zukünftige Versionen vorgesehen. Ist der Wert in sqlca.sqlcode kleiner als Null, dann haben Sie es hier mit einem schweren Fehler zu tun. Bei einem Fehler der größer Null ist handelt es sich zum Beispiel um den Rückgabewert, wenn eine Tabelle die gewünschte Zeile nicht enthält (siehe auch Anhang). Nach den kleineren Beispielen sehen Sie jetzt ein sehr einfaches Stammdaten-Programm. In dem Beispiel setzen wir auch Host-Variable ein.

Die Datei Datenbank.h

```
//
// Klasse Datenbank
//
#include <iostream>
using namespace std;

class Datenbank
{
public:
        ~Datenbank();
        void Anmelden();
};
```

Die Datei Datenbank.cpp

```
//
// Klasse Datenbank
//
#include "Datenbank.h"
#include <iostream>
using namespace std;

void Datenbank::Anmelden()
{
        // Verbindung aufnehmen
        EXEC SQL CONNECT TO test@localhost USER postgres;
```

```
}

Datenbank::~Datenbank()
{
        // Verbindung trennen
        EXEC SQL DISCONNECT;
}
```

Die Datei Kunden.h

```
//
// Klasse Kunden
//
#include <iostream>
#include "Datenbank.h"
using namespace std;

class Kunden : public Datenbank
{
public:
        void Kundenliste();
        void Loeschen();
        bool Einfuegen();
        int Lesen(const char *kdnr);
        void setKDNR(char kdnr[6]);
        void setKDNAME(char kdname[30]);
        char* getKDNR();
        char* getKDNAME();

private:
        // aus Platzgründen verwende ich nur diese Felder
        char strKDNR[6];
        char strKDNAME[31];
};
```

Die Datei Kunden.cpp

```
//
// Klasse Kunden
//
#include "Kunden.h"
#include <iostream>
#include <cstring>

void copy(char *strZiel, char *strQuelle, int iStellen)
{
        strncpy(strZiel,strQuelle,iStellen);
        strZiel[iStellen]='\0';
}

void Kunden::Loeschen()
{
        // die Hostvariable
        EXEC SQL BEGIN DECLARE SECTION;
        char skdnr[6]="";
        EXEC SQL END DECLARE SECTION;
```

```
        EXEC SQL BEGIN WORK;
        cout << "Eingabe der Kunden-Nr:";
        cin >> skdnr;
        skdnr[5]='\0';

        EXEC SQL DELETE FROM kunden WHERE kdnr=:skdnr;
        if (sqlca.sqlcode==0)
           cout << "Datensatz gelöscht" << endl;
        else
           if (sqlca.sqlcode==100)
              cout << "Datensatz nicht vorhanden" << endl;
           else
              cout << "Fehler:" << sqlca.sqlcode << endl;
        EXEC SQL COMMIT;
}

bool Kunden::Einfuegen()
{
        bool b;
        // die Hostvariablen, müssen unbedingt vor SQL-Befehl kommen
        EXEC SQL BEGIN DECLARE SECTION;
        char skdnr[6]="";
        char skdname[31]="";
        EXEC SQL END DECLARE SECTION;
        EXEC SQL BEGIN WORK;
        copy(skdnr,strKDNR,5);
        copy(skdname,strKDNAME,30);
        EXEC SQL INSERT INTO KUNDEN (kdnr,kdname)
          values (:skdnr,:skdname);
        if (sqlca.sqlcode==0)
        {
           b=true;
        }
        else
        {
           b=false;
        }
        EXEC SQL COMMIT;

        return b;
}

void Kunden::Kundenliste()
{
        // die Hostvariablen
        EXEC SQL BEGIN DECLARE SECTION;
        char skdnr[5]="";
        char skdname[30]="";
        EXEC SQL END DECLARE SECTION;
        EXEC SQL BEGIN WORK;

        // Cursor Anfang
        EXEC SQL DECLARE mycursor CURSOR FOR
        SELECT kdnr, kdname FROM kunden ORDER BY kdnr;
```

```
        EXEC SQL OPEN mycursor;
        EXEC SQL FETCH NEXT IN mycursor into :skdnr, :skdname;
        cout << "Fehlercode:" << sqlca.sqlcode << endl;
        // durchläuft alle Datensätze
        while (sqlca.sqlcode==0)
        {
                cout << skdnr << '\t' << skdname << endl;
                // nächsten Datensatz lesen
                EXEC SQL FETCH NEXT IN mycursor into :skdnr, :skdname;
                if (sqlca.sqlcode==100)
                        cout << "Ende der Liste" << endl;
        }
        EXEC SQL CLOSE mycursor;
        EXEC SQL COMMIT WORK;
}

void Kunden::setKDNR(char kdnr[6])
{
        strncpy(strKDNR,kdnr,5);
        strKDNR[5]='\0';
}

void Kunden::setKDNAME(char kdname[31])
{
        strncpy(strKDNAME,kdname,30);
        strKDNAME[30]='\0';
}

char* Kunden::getKDNR()
{
        return strKDNR;
}

char* Kunden::getKDNAME()
{
        return strKDNAME;
}

int Kunden::Lesen(const char *kdnr)
{
        int iFehler;
        // die Hostvariablen
        EXEC SQL BEGIN DECLARE SECTION;
        char skdnr[6]="";
        char skdname[31]="";
        EXEC SQL END DECLARE SECTION;
        EXEC SQL BEGIN WORK;

        strncpy(skdnr,kdnr,5);
        skdnr[5]='\0';
        // hier wird nur eine einzelne Zeile abgefragt
        EXEC SQL SELECT kdname INTO :skdname FROM KUNDEN WHERE
kdnr=:skdnr;
        switch (sqlca.sqlcode)
```

```
        {
           case 0:
              skdname[30]='\0';
              setKDNAME(skdname);
              break;
           case -203:
              setKDNAME("mehr Zeilen gefunden");
              break;
           default:
              setKDNAME(sqlca.sqlerrm.sqlerrmc);
              break;
        }
        iFehler = sqlca.sqlcode;
        EXEC SQL COMMIT WORK;

        return iFehler;
}
```

Die Datei Stammdaten.cpp

```
#include "Kunden.h"
#include <iostream>
using namespace std;

Kunden k;

void EingabeS(char *strEin, int iStellen)
{
        // den Eingabepuffer löschen
        cin.clear();
        cin.ignore(cin.rdbuf()->in_avail());
        // Eingabe der Informationen, das CR zählt mit
        iStellen++;
        cin.getline(strEin,iStellen);
        iStellen--;
        strEin[iStellen]='\0';
}

void Eingabe()
{
        char strHKDNR[6];
        char strHKDNAME[31];

        cout << "Eingabe Kunden-Nr.:";
        EingabeS(strHKDNR,5);
        cout << "Eingabe Kunden-Namen:";
        EingabeS(strHKDNAME,30);
        k.setKDNR(strHKDNR);
        k.setKDNAME(strHKDNAME);
        if (k.Einfuegen()==true)
           cout << "Datensatz eingefügt" << endl;
        else
           cout << "Fehler beim einfügen" << endl;
}
```

```
void Lesen()
{
        int iH;
        char strHKDNR[6];

        cout << "Eingabe der Kunden-Nr:";
        EingabeS(strHKDNR,5);
        iH=k.Lesen(strHKDNR);
        if (iH==0)
           cout << k.getKDNAME() << endl;
        else
        {
           cout << "Lesefehler:" << iH << endl;
           cout << k.getKDNAME() << endl;
        }
}

void Menue()
{
        int i=9;

        while(i!=0)
        {
          cout << "0=Ende" << endl;
          cout << "1=Kundenliste ausgeben" << endl;
          cout << "2=Kunde löschen" << endl;
          cout << "3=Kunde erfassen" << endl;
          cout << "4=Kunde lesen" << endl;
          cin >> i;
          switch(i)
          {
            case 1:
                k.Kundenliste();
                break;
            case 2:
                k.Loeschen();
                break;
            case 3:
                Eingabe();
                break;
            case 4:
                Lesen();
                break;
          }
        }
}

int main(int argc, char* argv[])
{
        cout << "Programm Stammdaten" << endl;
        cout << "ANFANG" << endl;
        // die Klasse Datenbank ist hier die Basisklasse und durch
        // die Vererbung erhält die Klasse Kunden die Methode
        // Anmelden. Um eine mehrfache Anmeldung zu vermeiden, reicht
```

```
        // es sie einmal aufzurufen. Eine Alternative ist hier ein
        // Singleton.
        k.Anmelden();
        Menue();
        cout << "ENDE" << endl;
        return 0;
}
```

So sehen die Compiler-Anweisungen aus:

```
ecpg Datenbank.cpp
ecpg Kunden.cpp
g++ -I/usr/include/pgsql -c Kunden.c
g++ -I/usr/include/pgsql -c Datenbank.c
g++ -c Stammdaten.cpp
g++ -o Stammdaten Stammdaten.o Kunden.o Datenbank.o -L/usr/include/pgsql/lib
-lecpg
```

Abbildung 77 ecpg - Programm Stammdaten

ANMERKUNG zu C++: Von CBorg gibt es auch eine C++-Schnittstelle für diese Programmiersprache.

Der DOT.NET-Zugriff mit der Schnittstelle Npgsql

Mit der Npgsql.dll greifen Sie auf die Datenbank zu. Er ist der .Net Data Provider für die Datenbank. Leider ist die Datei zurzeit nur als Beta-Version verfügbar. Sie finden die Schnittstelle über die Homepage von Postgres (oder auch in der Version 8.0 für Windows). Entpacken Sie die Dateien in ein Verzeichnis. Unterhalb des Verzeichnisses Npgsql\bin finden Sie dann die notwendigen DLL's nach Runtimeumgebung geordnet (Mono oder DOT.NET). Für Ihre Anwendung benötigen Sie auf jeden Fall die Dateien Npgsql.dll und Mono.Security.Protocol.Tls.dll. Bei Bedarf auch die anderen Dateien. Aus der unten gezeigten Tabelle entnehmen Sie die wichtigsten Punkte (hier am Beispiel der Programmiersprache C#):

Punkt	*Inhalt*
Namespace	Sparen Sie sich Tipparbeit mit `using Npgsql;`

Verbindung zur Datenbank aufnehmen	Für den Verbindungsaufbau sind die Informationen wie Name/IP-Adresse, Port, User, Passwort und Datenbankname wichtig (siehe auch Programm Kundenliste Postgres). Danach machen Sie ein Open auf die Verbindung.
	
	`NpgsqlConnection conn = new NpgsqlConnection ("Server=60.0.2.1;Port=5432; User Id=stefan; Password=secret; Database=test;");`
	`conn.Open();`

SQL-Befehl ausführen	Verwenden Sie dazu z.B. die Methode NpgsqlCommand (siehe Beispiel unten). Mit ihr können Sie sowohl Datensätze abfragen, als auch andere Aktionen ausführen. `NpgsqlCommand command = new NpgsqlCommand("select * from kunden", conn);`

Ein einzelnes Ergebnis als Rückgabewert aus einer Abfrage erhalten	Wenn Sie nur einen Wert aus einem Kommando Objekt benötigen, dann können Sie die Methode ExecuteScalar() einsetzen (Das unten gezeigte Beispiel finden Sie auch im Programm Kundenliste Postgres).
	
	`NpgsqlCommand command = new NpgsqlCommand("version()", conn);`
	`command.CommandType = CommandType.StoredProcedure;`
	
	`Object result = command.ExecuteScalar();`
	`strH=result.ToString();`
	`labelInfo.Text=strH;`

Punkt	*Inhalt*

Alle Rückgabewerte aus einer Abfrage erhalten	Setzen Sie die Methode NpgsqlCommand.ExecuteReader und das NpgsqlDataReader Objekt ein (siehe auch Programm Kundenliste Postgres).
	
	`....`
	`conn.Open();`
	`NpgsqlCommand command = new NpgsqlCommand("select * from kunden", conn);`
	`try`
	`{`
	`NpgsqlDataReader dr = command.ExecuteReader();`
	`dr.Read();`
	`while(dr.Read())`
	`{`
	`strH="";`
	`for (i = 0; i < dr.FieldCount; i++)`
	`{`
	`strH=strH + dr[i];`
	`strH=strH + "";`
	`}`
	`listBoxKunden.Items.Add(strH);`
	`}`
	`labelInfo.Text="fertig";`
	`}`

Der Einsatz von Parametern in einer Abfrage	Im Beispiel unten sehen Sie die dazu notwendigen Punkte. Besonders wichtig ist, dass Sie dem Parameter auch den entsprechenden DB-Typ zuweisen.
	
	`// Abfrage mit dem Parameter`
	`NpgsqlCommand command = new NpgsqlCommand("select * from kunde where KDNR = :Parameter1", conn);`
	
	`// Den Parameter der Parameter-Collection zuordnen und den Typ festlegen`
	`command.Parameters.Add(new NpgsqlParameter ("Parameter1", DbType.Int32));`
	
	`// Den Parameter mit einem Wert füllen`
	`command.Parameters[0].Value = 4;`
	
	`try`
	`{`
	`// das Kommando ausführen`
	`NpgsqlDataReader dr = command.ExecuteReader();`
	`....`

Punkt	*Inhalt*

Funktionen aufrufen	In dem folgenden Beispiel rufen wir die Funktion Version auf:
	
	`NpgsqlCommand command = new NpgsqlCommand("version()", conn);`
	`command.CommandType = CommandType.StoredProcedure;`
	`Object result = command.ExecuteScalar();`

DataSet und DataGrid	Die dazu notwendigen Befehle lassen sich recht einfach umsetzen. Wie gewohnt greifen Sie über NpgsqlConnection auf die Datenbank zu. Anschließend deklarieren und definieren Sie das DataSet. Der NpgsqlDataAdapter enthält den SQL-Befehl und startet die Abfrage auf der Datenbank. Über die Methode Fill des DataAdapters erhält das DataSet den Inhalt der Abfrage. Danach weisen wir den Inhalt dem Datagrid zu.
	
	`NpgsqlConnection conn = new NpgsqlConnection ("Server=60.0.2.1;Port=5432; User Id=stefan;Database=test;");`
	`conn.Open();`
	`DataSet ds = new DataSet("kunden");`
	`NpgsqlDataAdapter da = new NpgsqlDataAdapter ("select kdnr,kdname,kdstrasse,kdplz,kdort from kunden order by kdnr", conn);`
	`da.Fill(ds);`
	`dataGridKunde.DataSource=ds;`

Punkt	*Inhalt*

DataSet und Datensätze in die Tabelle einfügen.	Dieses Beispiel habe ich der Benutzerdokumentation Npgsql entnommen. Es zeigt recht gut die einzelnen Schritte.
	`NpgsqlConnection conn = new NpgsqlConnection ("Server=60.0.2.1;Port=5432; User Id=stefan;Database=test;");`
	`conn.Open();`
	`DataSet ds = new DataSet();`
	`NpgsqlDataAdapter da = new NpgsqlDataAdapter ("select kdnr,kdname,kdstrasse,kdplz,kdort from kunden", conn);`
	`// den Insert-Befehl zusammenbauen`
	`da.InsertCommand = new NpgsqlCommand("insert into kunden(kdnr,kdname,kdstrasse,kdplz,kdort) " + " values (:strKDNR,:strKDNAME,:strKDSTRASSE,:strKDPLZ,:strKDORT )", conn);`
	`da.InsertCommand.Parameters.Add(new NpgsqlParameter("strKDNR", DbType.String));`
	`da.InsertCommand.Parameters.Add(new NpgsqlParameter("strKDNAME", DbType.String));`
	`da.InsertCommand.Parameters.Add(new NpgsqlParameter("strKDSTRASSE", DbType.String));`
	`da.InsertCommand.Parameters.Add(new NpgsqlParameter("strKDPLZ", DbType.String));`
	`da.InsertCommand.Parameters.Add(new NpgsqlParameter("strKDORT", DbType.String));`
	
	`// die Parameter zusammmensetzen`
	`da.InsertCommand.Parameters[0].Direction = ParameterDirection.Input;`
	`da.InsertCommand.Parameters[1].Direction = ParameterDirection.Input;`
	`da.InsertCommand.Parameters[2].Direction = ParameterDirection.Input;`
	`da.InsertCommand.Parameters[3].Direction = ParameterDirection.Input;`
	`da.InsertCommand.Parameters[4].Direction = ParameterDirection.Input;`
	`da.InsertCommand.Parameters[0].SourceColumn = "kdnr";`
	`da.InsertCommand.Parameters[1].SourceColumn = "kdname";`
	`da.InsertCommand.Parameters[2].SourceColumn = "kdstrasse";`
	`da.InsertCommand.Parameters[3].SourceColumn = "kdplz";`
	`da.InsertCommand.Parameters[4].SourceColumn = "kdort";`
	`da.Fill(ds);`
	`DataTable dt = ds.Tables[0];`

Punkt	Inhalt

	DataRow dr = dt.NewRow();
	dr["kdnr"] = strKDNR;
	dr["kdname"] = strKDNAME;
	dr["kdstrasse"] = strKDSTRASSE;
	dr["kdplz"] = strKDPLZ;
	dr["kdort"] = strKDORT;
	
	dt.Rows.Add(dr);
	
	DataSet ds2 = ds.GetChanges();
	
	da.Update(ds2);
	ds.Merge(ds2);
	ds.AcceptChanges();
	
	MessageBox.Show("Datensatz eingetragen");

ANMERKUNG: Leider gibt es mit Sharpdevelop und der Schnittstelle noch Probleme. Das Programm stürzte jedesmal ab, wenn ich einige Npgsql-Elemente auf dem Formular platzieren wollte.

Hier sehen Sie einige Beispielprogramme:

Programm Kundenliste Postgres

Hier sehen Sie die Daten in der Tabelle Kunden. Über das Programm können Sie aber auch die eingesetzte Postgres-Version abfragen.

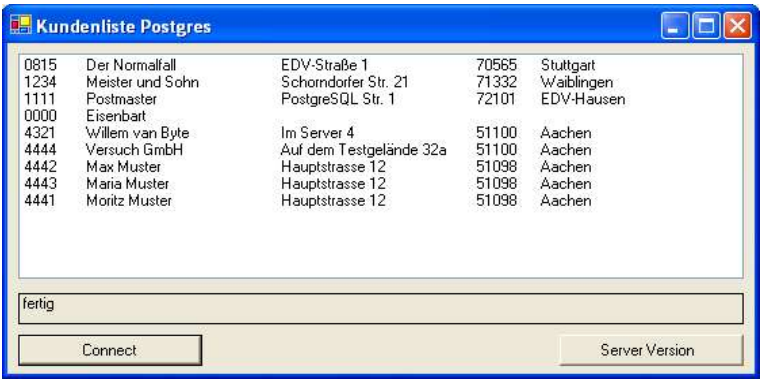


Abbildung 78 C#-Programm-Kundenliste

```
/*
 * Created by SharpDevelop.
 * User: Stefan
 * Date: 31.10.2004
 * Time: 10:12
 *
 * To change this template use Tools | Options | Coding | Edit
Standard Headers.
 */
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Data;
using Npgsql;


namespace pgtest
{
        /// <summary>
        /// Description of MainForm.
        /// </summary>
        public class MainForm : System.Windows.Forms.Form
        {
                private System.Windows.Forms.Label labelInfo;
                private System.Windows.Forms.Button
buttonServerVersion; // ACHTUNG: das ist eine Zeile!!
                private System.Windows.Forms.ListBox listBoxKunden;
                private System.Windows.Forms.Button buttonConnect;
                public MainForm()
                {
                        //
                        // The InitializeComponent() call is required
for Windows Forms designer support.
                        //
                        InitializeComponent();

                        //
                        // TODO: Add constructor code after the
InitializeComponent() call.
                        //
                }

                [STAThread]
                public static void Main(string[] args)
                {
                        Application.Run(new MainForm());
                }

                #region Windows Forms Designer generated code
                ....
                #endregion
                void ButtonConnectClick(object sender,
System.EventArgs e)
```

```
		{
			String strH;
			int i;
			NpgsqlConnection conn = new NpgsqlConnection
("Server=60.0.2.1;Port=5432; User Id=stefan;Database=test;");
			conn.Open();
		NpgsqlCommand command = new NpgsqlCommand("select *
from kunden", conn);
			try
			{
				NpgsqlDataReader dr =
command.ExecuteReader();
				dr.Read();

				while(dr.Read())
				{
					strH="";
					for (i = 0; i < dr.FieldCount;
i++)
					{
						strH=strH + dr[i];
						strH=strH + "";
					}
					listBoxKunden.Items.Add(strH);
				}
				labelInfo.Text="fertig";
			}
			finally
			{
			conn.Close();
			}
		}

		void ButtonServerVersionClick(object sender,
System.EventArgs e)
		{
			// die Postgres Version abfragen
			String strH;
			NpgsqlConnection conn = new NpgsqlConnection
("Server=60.0.2.1;Port=5432; User Id=stefan;Database=test;");
			conn.Open();
			try
			{
				NpgsqlCommand command = new
NpgsqlCommand("version()", conn);
				command.CommandType =
CommandType.StoredProcedure;

				Object result = command.ExecuteScalar();

				strH=result.ToString();
				labelInfo.Text=strH;
```

```
                        }
                        finally
                        {
                        conn.Close();
                        }
                }
        }
}
```

VB.NET-Programm

Nachdem C# Beispiel sehen Sie nun ein VB.NET-Programm:

```
'
' Created by SharpDevelop.
' User: stefan
' To change this template use Tools | Options | Coding | Edit
Standard Headers.
'
Imports System
Imports System.Data
Imports System.Drawing
Imports System.Windows.Forms
Imports Microsoft.VisualBasic
Imports Npgsql

Namespace pg_test

Public Class MainForm
        Inherits System.Windows.Forms.Form
        Private buttonSQLBefehl As System.Windows.Forms.Button
        Private textBoxDatenbank As System.Windows.Forms.TextBox
        Private textBoxPasswort As System.Windows.Forms.TextBox
        Private dataGridSQL As System.Windows.Forms.DataGrid
        Private buttonAnmelden As System.Windows.Forms.Button
        Private textBoxSQLBefehl As System.Windows.Forms.TextBox
        Private labelInfo As System.Windows.Forms.Label
        Private label1 As System.Windows.Forms.Label
        Private labelSQLErgebnis As System.Windows.Forms.Label
        Private labelSQLBefehl As System.Windows.Forms.Label
        Private label3 As System.Windows.Forms.Label
        Private label2 As System.Windows.Forms.Label
        Private groupBox1 As System.Windows.Forms.GroupBox
        Private textBoxName As System.Windows.Forms.TextBox

        ' die Verbindung zur Datenbank
        Private conn as NpgsqlConnection

        Public Shared Sub Main
                Dim fMainForm As New MainForm
                fMainForm.ShowDialog()
        End Sub
```

```
        Public Sub New()
                MyBase.New
                '
                ' The Me.InitializeComponent call is required for
Windows Forms designer support.
                '
                Me.InitializeComponent
                '
                ' TODO : Add constructor code after
InitializeComponents
                '
                labelSQLErgebnis.Visible=false
                dataGridSQL.Visible=false
        End Sub

        #Region " Windows Forms Designer generated code "
        ' This method is required for Windows Forms designer support.
        ' Do not change the method contents inside the source code
editor. The Forms designer might
        ' not be able to load this method if it was changed manually.
        Private Sub InitializeComponent()
                Me.textBoxName = New System.Windows.Forms.TextBox
                Me.groupBox1 = New System.Windows.Forms.GroupBox
                Me.label2 = New System.Windows.Forms.Label
                Me.label3 = New System.Windows.Forms.Label
                Me.labelSQLBefehl = New System.Windows.Forms.Label
                Me.labelSQLErgebnis = New System.Windows.Forms.Label
                Me.label1 = New System.Windows.Forms.Label
                Me.labelInfo = New System.Windows.Forms.Label
                Me.textBoxSQLBefehl = New System.Windows.Forms.TextBox
                Me.buttonAnmelden = New System.Windows.Forms.Button
                Me.dataGridSQL = New System.Windows.Forms.DataGrid
                Me.textBoxPasswort = New System.Windows.Forms.TextBox
                Me.textBoxDatenbank = New System.Windows.Forms.TextBox
                Me.buttonSQLBefehl = New System.Windows.Forms.Button
                Me.groupBox1.SuspendLayout
                CType
(Me.dataGridSQL,System.ComponentModel.ISupportInitialize).BeginInit
                Me.SuspendLayout
                '
                'textBoxName
                '
                Me.textBoxName.Location = New System.Drawing.Point(80,
16)
                Me.textBoxName.Name = "textBoxName"
                Me.textBoxName.Size = New System.Drawing.Size(144, 20)
                Me.textBoxName.TabIndex = 3
                Me.textBoxName.Text = ""
                '
                'groupBox1
                '
                Me.groupBox1.Controls.Add(Me.labelInfo)
                Me.groupBox1.Controls.Add(Me.buttonAnmelden)
```

```
                Me.groupBox1.Controls.Add(Me.textBoxDatenbank)
                Me.groupBox1.Controls.Add(Me.textBoxPasswort)
                Me.groupBox1.Controls.Add(Me.textBoxName)
                Me.groupBox1.Controls.Add(Me.label3)
                Me.groupBox1.Controls.Add(Me.label2)
                Me.groupBox1.Controls.Add(Me.label1)
                Me.groupBox1.Location = New System.Drawing.Point(8, 8)
                Me.groupBox1.Name = "groupBox1"
                Me.groupBox1.Size = New System.Drawing.Size(440, 120)
                Me.groupBox1.TabIndex = 0
                Me.groupBox1.TabStop = false
                Me.groupBox1.Text = "Anmeldeinformation"
                '
                'label2
                '
                Me.label2.Location = New System.Drawing.Point(8, 48)
                Me.label2.Name = "label2"
                Me.label2.Size = New System.Drawing.Size(56, 16)
                Me.label2.TabIndex = 1
                Me.label2.Text = "Passwort"
                '
                'label3
                '
                Me.label3.Location = New System.Drawing.Point(8, 72)
                Me.label3.Name = "label3"
                Me.label3.Size = New System.Drawing.Size(64, 16)
                Me.label3.TabIndex = 2
                Me.label3.Text = "Datenbank"
                '
                'labelSQLBefehl
                '
                Me.labelSQLBefehl.Location = New System.Drawing.Point
(16, 136)
                Me.labelSQLBefehl.Name = "labelSQLBefehl"
                Me.labelSQLBefehl.Size = New System.Drawing.Size(64,
16)
                Me.labelSQLBefehl.TabIndex = 2
                Me.labelSQLBefehl.Text = "SQLBefehl"
                '
                'labelSQLErgebnis
                '
                Me.labelSQLErgebnis.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D
                Me.labelSQLErgebnis.Location = New
System.Drawing.Point(8, 168)
                Me.labelSQLErgebnis.Name = "labelSQLErgebnis"
                Me.labelSQLErgebnis.Size = New System.Drawing.Size
(432, 23)
                Me.labelSQLErgebnis.TabIndex = 5
                '
                'label1
                '
                Me.label1.Location = New System.Drawing.Point(8, 24)
                Me.label1.Name = "label1"
```

```
                Me.label1.Size = New System.Drawing.Size(48, 16)
                Me.label1.TabIndex = 0
                Me.label1.Text = "Name"
                '
                'labelInfo
                '
                Me.labelInfo.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
                Me.labelInfo.Location = New System.Drawing.Point(8,
88)
                Me.labelInfo.Name = "labelInfo"
                Me.labelInfo.Size = New System.Drawing.Size(424, 24)
                Me.labelInfo.TabIndex = 1
                '
                'textBoxSQLBefehl
                '
                Me.textBoxSQLBefehl.Enabled = false
                Me.textBoxSQLBefehl.Location = New
System.Drawing.Point(88, 136)
                Me.textBoxSQLBefehl.Name = "textBoxSQLBefehl"
                Me.textBoxSQLBefehl.Size = New System.Drawing.Size
(232, 20)
                Me.textBoxSQLBefehl.TabIndex = 1
                Me.textBoxSQLBefehl.Text = "select * from kunde"
                '
                'buttonAnmelden
                '
                Me.buttonAnmelden.Location = New System.Drawing.Point
(264, 16)
                Me.buttonAnmelden.Name = "buttonAnmelden"
                Me.buttonAnmelden.Size = New System.Drawing.Size(168,
23)
                Me.buttonAnmelden.TabIndex = 6
                Me.buttonAnmelden.Text = "Anmelden"
                AddHandler Me.buttonAnmelden.Click, AddressOf
Me.ButtonAnmeldenClick
                '
                'dataGridSQL
                '
                Me.dataGridSQL.DataMember = ""
                Me.dataGridSQL.HeaderForeColor =
System.Drawing.SystemColors.ControlText
                Me.dataGridSQL.Location = New System.Drawing.Point(8,
168)
                Me.dataGridSQL.Name = "dataGridSQL"
                Me.dataGridSQL.Size = New System.Drawing.Size(440,
152)
                Me.dataGridSQL.TabIndex = 4
                '
                'textBoxPasswort
                '
                Me.textBoxPasswort.Location = New System.Drawing.Point
(80, 40)
                Me.textBoxPasswort.Name = "textBoxPasswort"
```

```
                Me.textBoxPasswort.PasswordChar =
Microsoft.VisualBasic.ChrW(42)
                Me.textBoxPasswort.Size = New System.Drawing.Size(144,
20)
                Me.textBoxPasswort.TabIndex = 4
                Me.textBoxPasswort.Text = ""
                '
                'textBoxDatenbank
                '
                Me.textBoxDatenbank.Location = New
System.Drawing.Point(80, 64)
                Me.textBoxDatenbank.Name = "textBoxDatenbank"
                Me.textBoxDatenbank.Size = New System.Drawing.Size
(144, 20)
                Me.textBoxDatenbank.TabIndex = 5
                Me.textBoxDatenbank.Text = ""
                '
                'buttonSQLBefehl
                '
                Me.buttonSQLBefehl.Enabled = false
                Me.buttonSQLBefehl.Location = New System.Drawing.Point
(328, 136)
                Me.buttonSQLBefehl.Name = "buttonSQLBefehl"
                Me.buttonSQLBefehl.Size = New System.Drawing.Size(120,
23)
                Me.buttonSQLBefehl.TabIndex = 3
                Me.buttonSQLBefehl.Text = "SQLBefehl ausführen"
                AddHandler Me.buttonSQLBefehl.Click, AddressOf
Me.ButtonSQLBefehlClick
                '
                'MainForm
                '
                Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
                Me.ClientSize = New System.Drawing.Size(456, 326)
                Me.Controls.Add(Me.labelSQLErgebnis)
                Me.Controls.Add(Me.dataGridSQL)
                Me.Controls.Add(Me.buttonSQLBefehl)
                Me.Controls.Add(Me.labelSQLBefehl)
                Me.Controls.Add(Me.textBoxSQLBefehl)
                Me.Controls.Add(Me.groupBox1)
                Me.Name = "MainForm"
                Me.Text = "pgDaten"
                Me.groupBox1.ResumeLayout(false)
                CType
(Me.dataGridSQL,System.ComponentModel.ISupportInitialize).EndInit
                Me.ResumeLayout(false)
        End Sub
        #End Region



        Private Sub ButtonAnmeldenClick(sender As System.Object, e As
System.EventArgs)
                Dim strConn as String
```

```
                strConn ="Server=localhost;Post=5432;User ID="
                strConn = strConn & textBoxName.Text & ";Passwort="
                strConn = strConn & textBoxPasswort.Text
                strConn = strConn & ";Database=" &
textBoxDatenbank.Text
                ' zur Laufzeit zuweisen
                conn = new NpgsqlConnection(strConn)
                Dim comBefehl as new NpgsqlCommand("select count(*)
from kunde",conn)
                Dim oRueckgabe As Object

                Try
                        conn.open()
                        labelInfo.Text="angemeldet"
                        textBoxSQLBefehl.Enabled=True
                        buttonSQLBefehl.Enabled=True
                Catch eX As Exception
                        labelInfo.Text=eX.Message
                End Try
        End Sub

        Private Sub ButtonSQLBefehlClick(sender As System.Object, e As
System.EventArgs)
                If textBoxSQLBefehl.Text<>"" Then
                        SQLBefehl
                Else
                        labelSQLErgebnis.Visible=true
                        labelSQLErgebnis.Text = "Bitte SQL-Befehl
eingeben"
                End If
        End Sub

        Private Sub SQLBefehl()
                ' kurze Prüfung des SQL-Befehls
                Dim strH As String
                Dim i as Integer

                strH=textBoxSQLBefehl.Text
                strH=strH.ToUpper
                i=strH.IndexOf("SELECT")
                if i>=0 then
                        SQLselect(strH)
                Else
                        SQLausführen(strH)
                end if
        End Sub
        Private Sub SQLselect(strEin As String)
                ' Wertet die Rückgabe aus
                ' füllt das DataGrid, select-Abfrage
                Dim dsDat as new DataSet("Abfrage")
                Dim npgDat As New NpgsqlDataAdapter(strEin,conn)

                labelSQLErgebnis.Visible=false
                dataGridSQL.Visible=true
```

```
                Try
                        npgDat.Fill(dsDat)
                        dataGridSQL.DataSource=dsDat
                Catch eX As Exception
                        labelInfo.Text=eX.Message
                End Try

        End Sub

        Private Sub SQLausführen(strEin As String)
                ' die Rückgabe wird nicht ausgewertet
                ' führt z.B. einen create table aus
                Dim comBefehl as new NpgsqlCommand(strEin,conn)

                labelSQLErgebnis.Visible=true
                dataGridSQL.Visible=false
                Try
                        comBefehl.ExecuteScalar()
                        labelSQLErgebnis.Text="ausgeführt"
                Catch eX As Exception
                        labelSQLErgebnis.Text=eX.Message
                End Try
        End Sub

End Class
End Namespace
```

Abbildung 79 VB.NET - pgData 1

Abbildung 80 VB.NET - pgData 2

Mit Java auf die Datenbank zugreifen

Sie können entweder über ODBC (egal ob über unixODBC auf LINUX oder über ODBC auf Windows-Betriebssystemen) auf die Datenbank zugreifen (Zugriff über die JDBC-ODBC Brücke mit den Sun-Klassen), oder auch das JAVA-Package von Postgres einsetzen (Es ist ein JDBC-Treiber Typ 4, er greift direkt auf die Datenbank zu, ohne Windows-ODBC, bzw. unixODBC zu benutzen). Aus der unten gezeigten Tabelle entnehmen Sie die wichtigsten Punkte:

Punkt	***Inhalt***
Treiber vorbereiten	Bei der Installation finden Sie 2 JAR-Archive im Verzeichnis /usr/local/pgsql/share/java. Benennen Sie die Datei, die Sie verwenden möchten in ***postgresql.jar*** um (Verwenden Sie die jdbc2-Version ab der Java-Version 1.2). Wenn es Probleme mit den Umlauten gibt, sollten Sie die Datenbank mit der Option -E LATIN1 anlegen. Sichern Sie dann die Datenbank mit pg_dump, löschen sie danach und legen Sie sie mit der Option -E LATIN1 wieder an. Am Ende importieren Sie die Daten wieder mit dem Restore. ***TIPP:*** Vermerken Sie am besten die Klasse in der CLASSPATH-Variable (Stichwort: Datei /etc/profile).
Import-Anweisung (im Quellcode)	Das Paket java.sql importieren.
Treiber laden (im Quellcode oder zur Laufzeit)	Sie können den Treiber für die Datenbank mit der Methode ***Class.forName(„org.postgresql.Driver“)***; laden (nicht per Import, es gibt hier dann Probleme), oder auch per Befehlsparameter: ***java -D jdbc.drivers=org.postgresql.Driver Programm***
Verbindung zur Datenbank	Mit JDBC wird die Verbindung durch eine URL dargestellt. Sie haben dazu mehrere Formen: **- jdbc:postgresql:*datenbank*** **- jdbc:postgresql://*host/datenbank*** **- jdbc:postgresql://*host:port/datenbank***
Daten löschen, einfügen, ändern	Mit der Methode ***executeUpdate()***. Im Gegensatz zum Lese-Zugriff erhalten Sie hier kein ResultSet nach der Ausführung zurück, sondern nur die Anzahl der Zeilen, die von der Aktion betroffen waren.

Punkt	Inhalt
Daten lesen	Mit der Methode ***executeQuery()*** des Interfaces ResultSet. Um nach der Ausführung des SQL-Statements auf die Daten zugreifen zu können, müssen Sie erstmals die Methode ***next()*** aufrufen. Wenn Sie das Interface nicht mehr benötigen, schließen Sie es mit ***close()***.
Verbindung zur Datenbank schliessen	Mit der Methode ***close()*** des Connection Objektes.

Das nun folgende Programm dient zur Demonstration.

```
import java.lang.*;
import java.sql.*;
import java.util.*;

/*
Zugriff auf die Tabelle Kunden
*/
public class ifield
{

        // für ODBC
        Connection con;
        Statement  stmt;
        String strODBC = new String();
        ResultSet rs;
        String user = new String("postgres");
        String password = new String("");

        //
        // verbinden
        //
        public ifield()
        {
                try
                {
                        // Zugriff auf die JAVA-Klassen für Postgres
                        Class.forName("org.postgresql.Driver");
                        String url = "jdbc:postgresql:test";
// weitere Optionen: jdbc:postgresql://host/database oder
// jdbc:postgresql://host:port/database
                        con   =   DriverManager.getConnection(url,   user,
password); // ACHTUNG: das ist eine Zeile
                        stmt = con.createStatement();
                        rs = stmt.executeQuery("select * from kunden");
                        // Zugriff auf den ersten Datensatz
                        rs.next();
                        System.out.println("Column      1:"+rs.getString
```

```
(1));
                        System.out.println("Column      2:"+rs.getString
(2));
                        System.out.println("Column      1      [Object]:
"+rs.getObject(1));
                        rs.close();
                        stmt.close();
                }
                catch(Exception ex)
                {
                        System.out.println("Error   at   initialize   the
database access:"+ex);
                }
        }

        public static void main(String[] args)
        {
                ifield i = new ifield();
        }
}
```

Und so sieht dann die Ausgabe des Programms aus:

```
stefan@PC954SUSE:"> cd JAVA-Programme
stefan@PC954SUSE:"/JAVA-Programme> sh jr
Column 1:4711
Column 2:Hinz und Kunz
Column 1 [Object]:4711
stefan@PC954SUSE:"/JAVA-Programme>
```

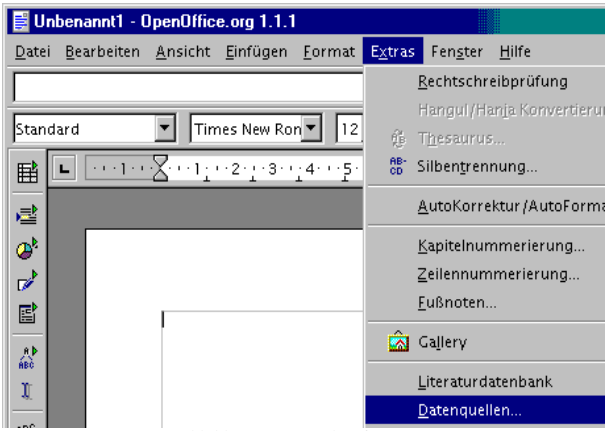
Abbildung 81 JAVA-Programm-Ausgabe

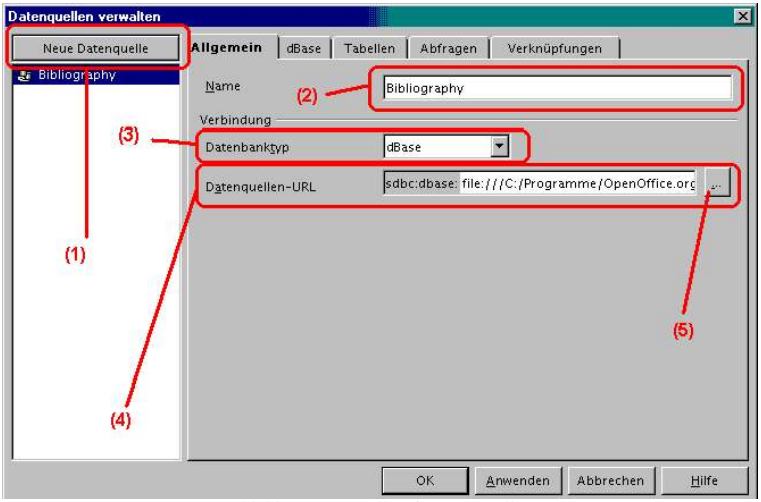
Serienbriefe mit OpenOffice

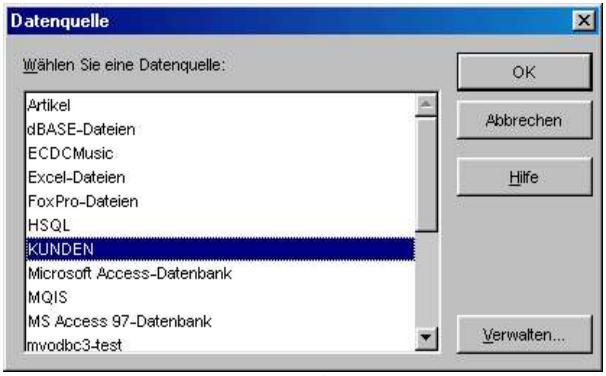
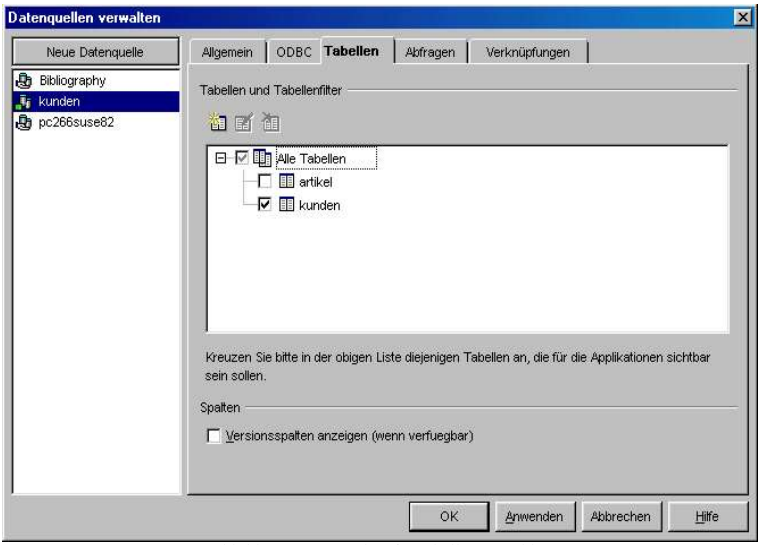
Von OpenOffice gibt es auch eine Möglichkeit auf Ihre Daten zuzugreifen. Dort hinterlegen Sie eine Abfrage. Sie ist dann die Quelle für einen Serienbrief.

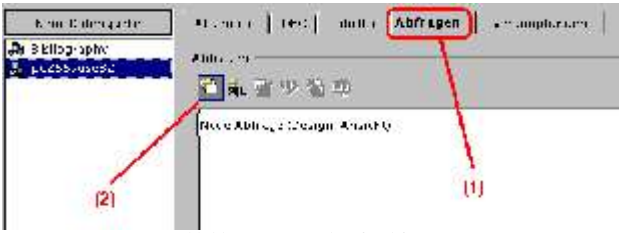
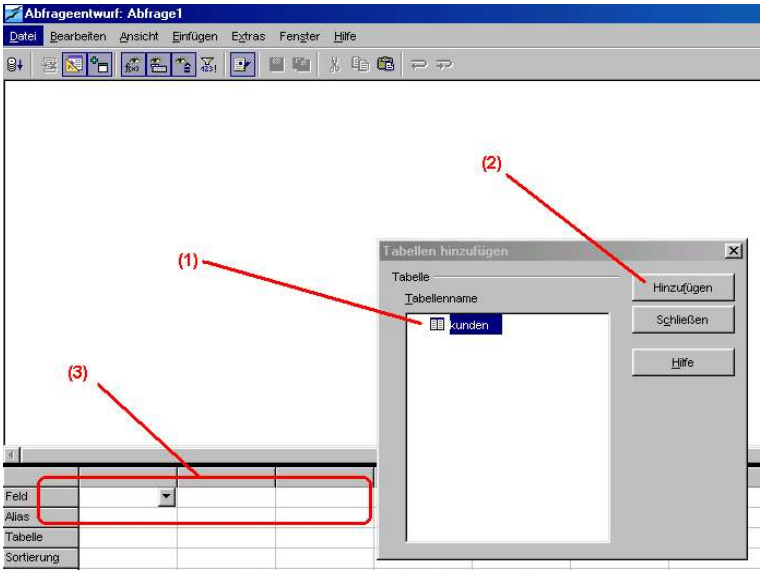
Zugriff einrichten

Als Mittler zwischen der Datenbank und dem Office-Programm verwenden Sie ODBC oder JDBC (Stichwort JAVA installieren).

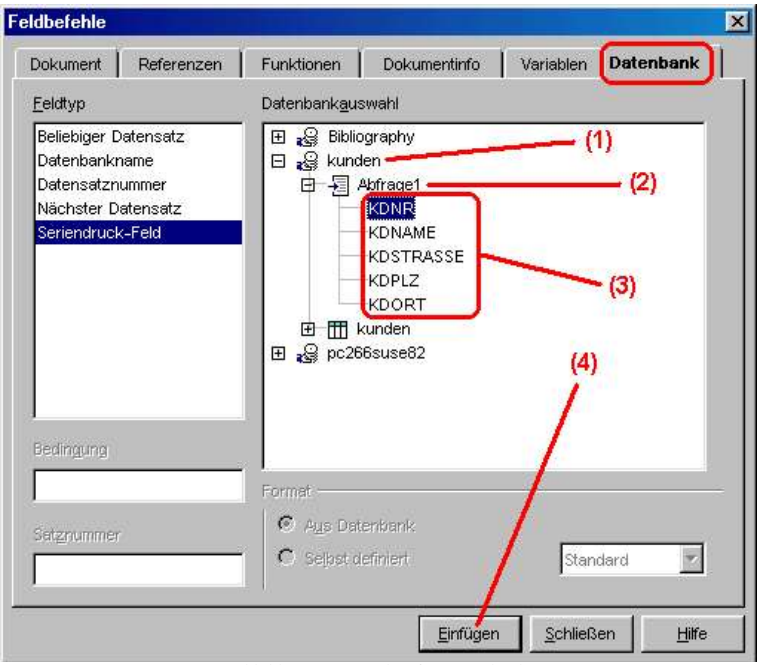
Schritt	***Inhalt***
1	Starten Sie OpenOffice
2	 *Abbildung 82 Serienbriefe-Menue* Rufen Sie über den Eintrag ***Extras*** den Eintrag ***Datenquellen*** auf.

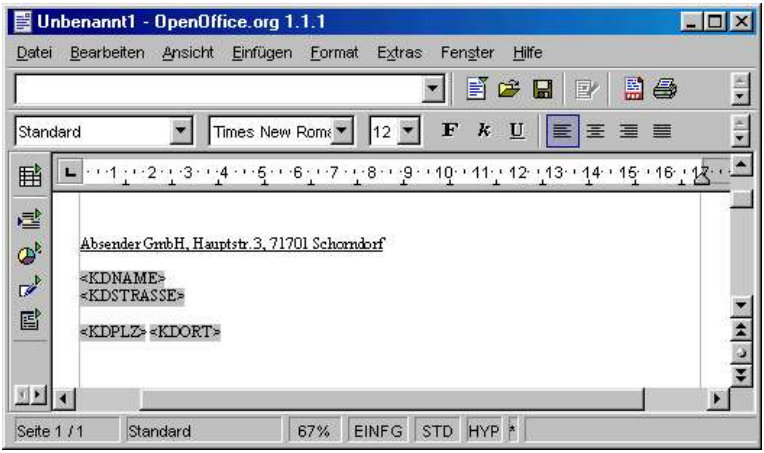
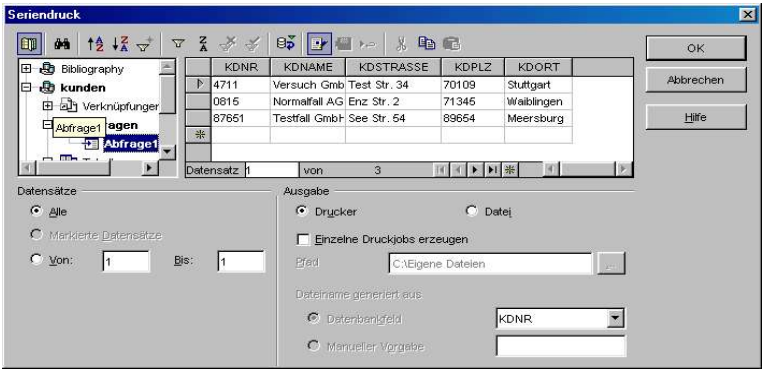
Schritt	*Inhalt*
3	 *Abbildung 83 Serienbriefe-Datenquellen* Drücken Sie als erstes den Knopf neue Datenquelle (siehe **(1)**). Vergeben Sie danach einen neuen Namen (siehe **(2)**). Im Punkt Datenbanktyp wählen Sie entweder den Eintrag ODBC oder JDBC aus (siehe **(3)**). Wenn Sie den JDBC-Treiber einsetzen, dann geben Sie den Dateinamen und das Verzeichnis der Klasse Driver.class ein (z.B. c:\org\postgres\driver) (siehe **(4)**). JDBC setzt allerdings JAVA voraus. Im andern Fall sollten Sie vorher den ODBC-Treiber installiert haben. Zur Installation der ODBC-Datenquelle drücken Sie einfach den Knopf (siehe **(5)**) in der Zeile Datenquelle-URL (**ACHTUNG:** System- und Benutzer-DSN werden zusammen angezeigt, siehe Schritt **4**).

Schritt	*Inhalt*
4	 *Abbildung 84 Serienbriefe-Quelle auswählen* Wählen Sie hier die Datenquelle aus (hier im Beispiel den Eintrag **KUNDEN**). Danach klicken Sie den Karteireiter ODBC an. In dem nun gezeigten Fenster wählen Sie die gewünschten Tabellen aus.
5	 *Abbildung 85 Serienbriefe-Tabelle auswählen* Zugriff über ODBC-Treiber einrichten: In dem nun gezeigten Fenster wählen Sie die gewünschten Tabellen aus.

Schritt	Inhalt
6	 *Abbildung 86 Serienbriefe-Abfragen* Wechseln Sie auf den Karteireiter Abfragen (siehe **(1)**) und drücken Sie den Knopf SQL (siehe **(2)**).
7	 *Abbildung 87 Serienbriefe-Abfrage erstellen* Wählen Sie hier als Erstes die gewünschten Tabellen (siehe **(1)** und **(2)**) aus. Danach geht es weiter mit den einzelnen Feldern (siehe **(3)**).

Serienbrief verfassen

Schritt	*Inhalt*
1	Im Hauptdokument können Sie nun im Text die einzelnen Felder auswählen. Gehen Sie über die Menüleiste vom Eintrag **Einfügen**, **Feldbefehl** und **Andere ...**. .  *Abbildung 88 Serienbriefe - Datenbank* Nach der Auswahl des Karteireiters Datenbank arbeiten Sie hier die Schritte 1 bis 4 ab.

Schritt	*Inhalt*
2	Das Ergebnis sieht dann so aus:  *Abbildung 89 Serienbriefe - Felder*
3	Gehen Sie dann über den Menüpunkt **Extras** zum Eintrag **Seriendruck**. In dem Window Seriendruck selektieren Sie dann Ihre Abfrage. Zur Bestätigung sehen Sie dann die Datensätze. Drücken Sie dann die **OK**-Taste.  *Abbildung 90 Serienbriefe - Datensätze*

Daten in Postgres übernehmen

Für die Übernahme in PostgreSQL steht Ihnen der SQL-Befehl COPY zur Verfügung. Mit diesem Befehl importieren Sie CSV-Dateien. Für alle anderen Dateiformate müssen Sie leider ein extra Import-Programm schreiben (XML-Dateien sind in der Version 8.1 vorgesehen).

Aufbau:
COPY Tabellenname from 'Verzeichnis und Dateiname' DELIMITER 'Tennzeichen';

TIPP: Wenn Sie die Daten importieren, dann achten Sie auch unbedingt auf den Tabellenaufbau. Bleibt z.B. ein Feld unberücksichtigt, dann bricht der COPY-Befehl ab (siehe Bild unten, in der Import-Datei wird das Feld vknr leider nicht bedient).

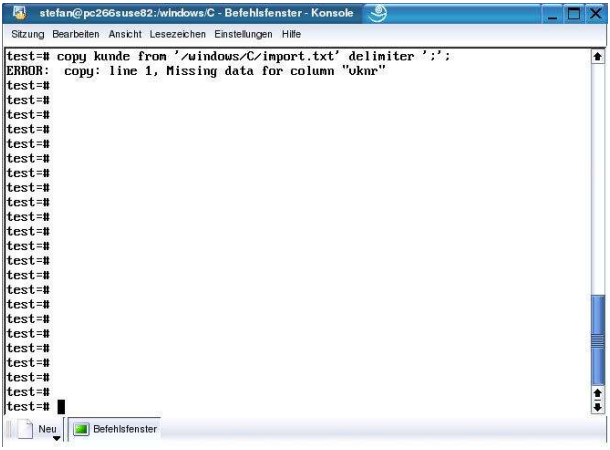


Abbildung 91 Daten in Postgres übernehmen , Fehler

Im folgenden Beispiel sehen Sie wie eine Textdatei aus Windows in eine Postgres -Datenbank übernommen wird. Die Tabelle ist so aufgebaut:
Vorname
Nachname
Strasse
PLZ
Ort
Kundennr

Die Importdatei sieht so aus:

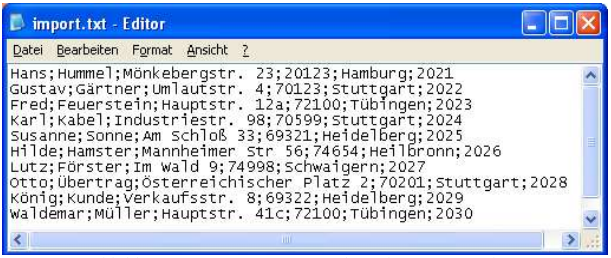


Abbildung 92 Daten in Postgres übernehmen, Importdatei

Später habe ich dann LINUX gestartet und die Windowspartition gemountet. Sicherheitshalber sollten Sie sich kurz noch einmal die Datei ansehen (siehe Beispiel unten).

Abbildung 93 Daten in Postgres übernehme, Daten prüfen

Rufen Sie anschließend das Programm psql auf importieren Sie die Daten mit dem COPY-Befehl. Wenn alles eingelesen wurde, erhalten Sie als Bestätigung die Meldung ***COPY***. Im Bild unten sehen Sie das Ergebnis der Importoperation.

Abbildung 94 Daten in Postgres übernehmen, das Ergebnis

TIPP: Wenn Sie Daten in die Windows-Version importieren möchten, dann lassen Sie den Laufwerksbuchstaben weg.

Anhang A - SQL-Befehle

Ändern	
Datensätze abändern	Update [Tabellenname] set [Feld] = [Wert],... Pflicht where [Feld] [Bedingung] [Vorgabe] kann Wenn es mehrere Felder gibt, dann ist das Komma das Trennzeichen. Rechen-Operationen sind auch möglich z.B. set Feld1 = Feld1 * 1,16. Gültige Operationen sind + - * / .
Tabellenname ändern	Alter table [Tabellenname] to [Tabellenname neu]
Feld umbenennen	Alter table [Tabellennamen] rename [Feld vorher] to [Feld nachher]

Erstellen	
Tabelle anlegen	Create table [Tabellenname] ([Feld] [Art]) Wenn es mehrere Felder gibt, dann kommt direkt nach der Art des Feldes ein Komma. Folgende Feldtypen gibt es: char(n) => Zeichenfeld, n ist die Anzahl der Bytes smallint => + und - Ganzzahl integer => + und - Ganzzahl decimal(n,m) => Dezimalzahl (n = Zahl vor dem Komma, m = Zahl nach dem Komma) money => Betragsfeld
Anlegen eines Indexfeldes	Create unique|distinct index [Indexname] on [Tabelle] (Feld,..) Der Index kann sich auch über mehrere Felder erstrecken unique => der Schlüssel wird nur einmal vergeben distinct => der Schlüssel darf mehrmals vorkommen
Erstellen einer Daten-bank	Create database [Datenbank] [ENCODING='LATIN1'] **ANMERKUNG:** Mit dem Shell-Befehl createdb dbname legen Sie auch eine Datenbank an. Ohne zusätzliche Option ist der aktuell angemeldete Benutzer der Owner. Übrigens auch das Encoding nicht vergessen.

Ein- und hinzufügen	
Eingeben von Daten-sätzen	Insert into [Tabellenname] ([Feld],...) values ([Wert],...) Anstelle der Felder und Werte kann man auch über eine select Anweisung verwenden.
Einfügen von Feldern in eine Tabelle	Alter table [Tabellenname] add column [Feld] [Feldtyp]

Auswählen	
Von Daten	Select [Feld],... Pflicht from [Tabellenname] Pflicht [join] kann where [Auswahlliste schränkt die Auswahl ein] kann group by [Gruppierungsliste fasst Feldinhalte zusammen] kann order by [Sortierliste sortiert die Ausgabe] kann into [Tabellenname die Ausgabe erfolgt in eine extra Tabelle] kann Anstelle der Felder kann auch ein Stern als Wildcard eingesetzt werden. Die Gruppierungs- und Sortierliste enthält die Feldnamen. **ANMERKUNG:** Möchten Sie nur eine bestimmte Anzahl von Datensätzen angezeigt haben, dann geben Sie nach der Tabelle LIMIT und die Anzahl der Datensätze ein. Z.B. bei SELECT * FROM KUNDEN LIMIT 10 erhalten Sie die ersten 10 Datensätze angezeigt. Mit der Option OFFSET <Anzahl Zeilen> überspringen Sie eine Anzahl von Zeilen.. Sie können auch mit dem Select -Befehl plsql-Funktionen aufrufen, z. B. mit select now() das aktuelle Datum und die aktuelle Zeit.

Löschen	
Löschen eines Indexfeldes	Drop index [Indexname]
Löschen einer Datenbank	Drop database [Datenbank] **ANMERKUNG:** Mit dem Befehl dropdb dbname als Shell-Befehl löschen Sie auch eine Datenbank
Löschen einer Tabelle	Drop table [Tabellenname]
Löschen eines oder mehrerer Datensätze	Delete from [Tabellenname] Pflicht where [Bedingung] kann Mit der Where-Bedingung schränkt man die zu löschenden Datensätze ein. ACHTUNG: Fehlt die Where-Bedingung, dann löscht man alle Datensätze.
Löschen eines Feldes in einer Tabelle	Alter table [Tabellename] drop column [Feld]

Daten importieren und exportieren	
Import	copy [binary] Tabellenname from 'Dateiname' delimiters '[das Zeichen]' [with null as 'Ersatzzeichen'] **ANMERKUNG:** Als Dateiname können Sie auch stdin angeben.
Export	copy [binary] Tabellenname to 'Dateiname' delimiters '[das Zeichen]' [with null as 'Ersatzzeichen'] **ANMERKUNG:**Als Dateiname können Sie auch stdout angeben. Lassen Sie übrigens den Zusatz WITH NULL AS weg, dann gibt das Programm \n anstelle von null aus.

Autocommit	
Starten	commit
Beenden	begin

ANMERKUNG: Das Autocommit bewirkt, dass die Änderung sofort in die Datenbank geschrieben wird. Ohne diese Möglichkeit müssen Sie vor der Transaktion den Befehl BEGIN absetzen, dann Ihre Änderungen vornehmen und mit COMMIT dann festschreiben.

Agenda	
[Felder],...	Eingabe eines oder mehrerer Felder, bei mehreren Feldern muss man die Felder mit einem Komma trennen.
[Bedingung]	Eine Bedingung setzt sich aus einem Feld, einer Bedingung und einem Vergleichswert zusammen. Gültige Bedingungen sind: = (gleich) like (Suche nach Textmustern in einem Feld, % (bei ACCESS steht es für *) und ? sind als Wildcards zulässig) <> (ungleich) < (kleiner) <= (kleiner gleich) > (größer) >= (größer gleich) Eine Bedingung kann auch mehrere Felder enthalten. Siehe Beispiele unten: PLZ="70565" => *ein Feld* KUNDENGRUPPE ="A" AND ORT ="ESSEN" => *zwei Felder*

Join	Über join verknüpfen Sie Tabellen mit einander. Es gibt insgesamt 3 Arten von join: – CROSS JOIN: (z.B. Tabelle1 CROSS JOIN Tabelle2) Hier verknüpft die Abfrage jede Zeile aus der Tabelle1 mit der Tabelle2. – INNER JOIN: (z.B. Tabelle1 INNER JOIN Tabelle2) INNER ist die Grundeinstellung. Hier erhalten Sie alle Datensätze aus den Tabellen1 und Tabellen2, welche die Abfragebedingung erfüllen. – LEFT OUTER JOIN: z.B. Tabelle1 LEFT OUTER JOIN Tabelle2) Alle Datensätze aus der Tabelle1 die nicht die Abfragebedingung mit der Tabelle 2 erfüllen werden ausgegeben. – RIGHT OUTER JOIN: z.B. Tabelle1 RIGHT OUTER JOIN Tabelle 2) Hier erhalten Sie alle Datensätze aus der Tabelle2, die nicht die Verknüpfungsbedingung mit der Tabelle1 erfüllen. – FULL OUTER JOIN: z.B. Tabelle1 FULL OUTER JOIN Tabelle2) Sie sehen hier alle Datensätze, die überhaupt nicht die Abfragebedingung erfüllen.

Anhang B - Verzeichnisbaum unter LINUX

Verzeichnis	Inhalt
/bin	Alle elementaren Linux-Kommandos
/boot	Alle Dateien des LILO-Bottmanagers
/dev	Die Device-Dateien für die Hardware (Festplatten, Floppy, CD-Rom usw.)
/etc	Die Konfigurationsdateien zum Systemstart
/home	Unter ihm befinden sich die Homeverzeichnisse der Anwender
/lib	Die gemeinsamen Bibliotheken
/lost+found	Hier befinden sich die defekten Dateien, die beim nicht ordnungsgemäßen Verlassen von Linux entstehen.
/mnt	Hier wird z.B. das CD-Rom-Laufwerk eingebunden
/opt	Enthält zusätzliche Programme (z.B. StarOffice)
/proc	Enthält Unterverzeichnisse für alle laufenden Prozesse
/sbin	Kommandos zur Systemverwaltung. Sie dürfen nur vom Superuser ausgeführt werden
/tmp	Die temporären Dateien
/usr	Es beinhaltet die wichtigsten Anwendungsprogramme
/var	Es enthält die veränderbaren Dateien

Anhang C - weitere Möglichkeiten zur Administration der Datenbank

Auf der Homepage www.postgres.de finden Sie unter der Rubrik Software eine Übersicht (incl. Link auf die Download-Bereiche) über verschiedene grafische Schnittstellen zur Datenbank. Von den Programmen habe ich nicht alle untersucht, sondern nur 2 ausgewählt. Sie finden dazu eine Screenshot und einen Verweis auf den Download-Bereich.

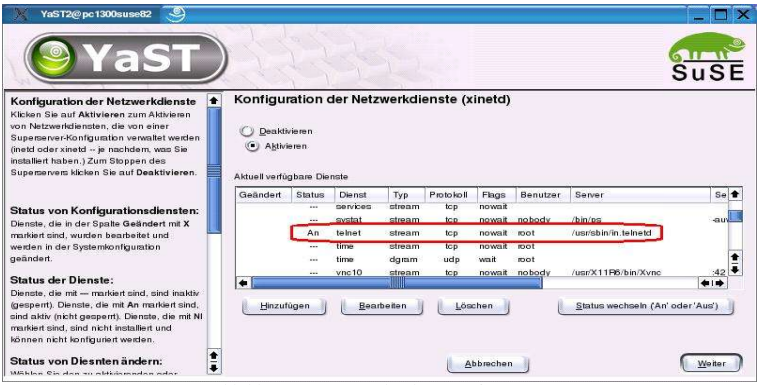
Administration über das Programm telnet

Das Programm finden Sie sowohl auf LINUX, als auch auf den MS-Windows Betriebssystemen. Dazu müssen Sie auf Ihrem LINUX-Host den Zugriff über ***telnet*** zulassen. Bei einer telnet-Sitzung können Sie nicht mit X-Windows arbeiten. Für die Einrichtung von ***telnet*** benötigen Sie wieder das Programm ***YaST*** (das Programm gibt es nur unter der SuSE Distribution). Die einzelnen Schritte finden Sie auf der nächsten Seite (***ANMERKUNG***: Fedora3 weist Sie daraufhin, dass Passwörter und Benutzernamen unverschlüsselt über das Netz gehen).

SuSE-LINUX und telnet

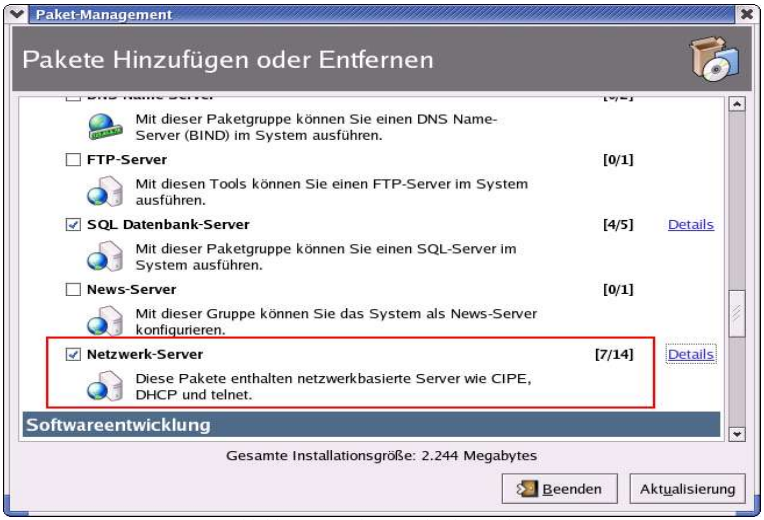
Unter SuSE-LINUX richten Sie es wie folgt ein:

Schritt	Inhalt
1	 *Abbildung 95 SuSE und Telnet-Netzwerkdienste* Wählen Sie hier ***Netzwerkdienste*** und ***Netzwerkdienste (inetd)*** aus (bei SuSE 8.2). Bei der Version 8.0 wählen Sie ***Netzwerk/Basis*** und ***Start oder Stop von Systemdiensten*** aus.

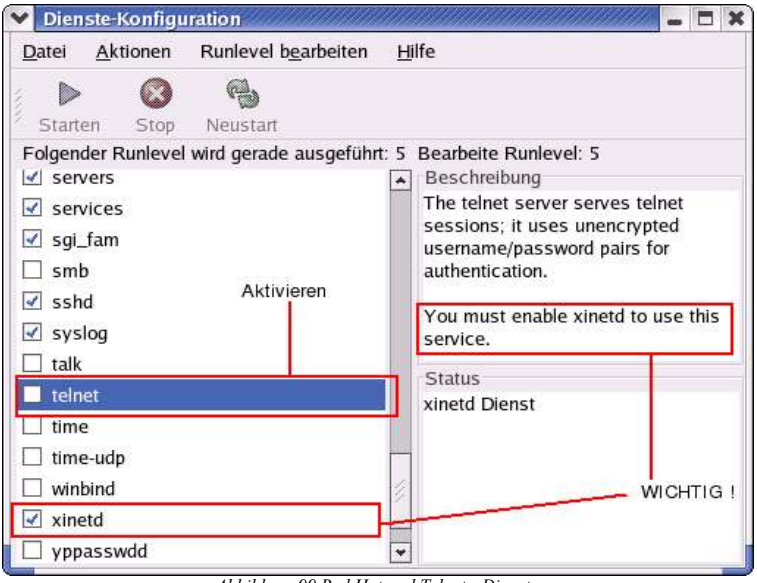
Schritt	Inhalt
2	 *Abbildung 96 SuSE und Telnet-Konfiguration* Markieren Sie den Dienst ***telnet***. Drücken Sie anschließend den Knopf ***Status wechseln ('An' oder 'Aus')*** **WICHTIG:** Achten Sie übrigens auch darauf, dass die Netzwerkdienste aktiviert sind!!
3	Prüfen Sie nun die Verbindungsaufnahme mit telnet. Wenn es nicht funktioniert, dann müssen Sie den inetd Dämon beenden und erneut starten (mit den Befehlen ***killall inetd*** und ***inetd*** als Superuser).
4	Weiter geht es nun mit dem Kapitel Mit telnet arbeiten.

Red Hat/Fedora LINUX und telnet

Die Einrichtung des telnet-Servers gestaltet sich wie folgt:

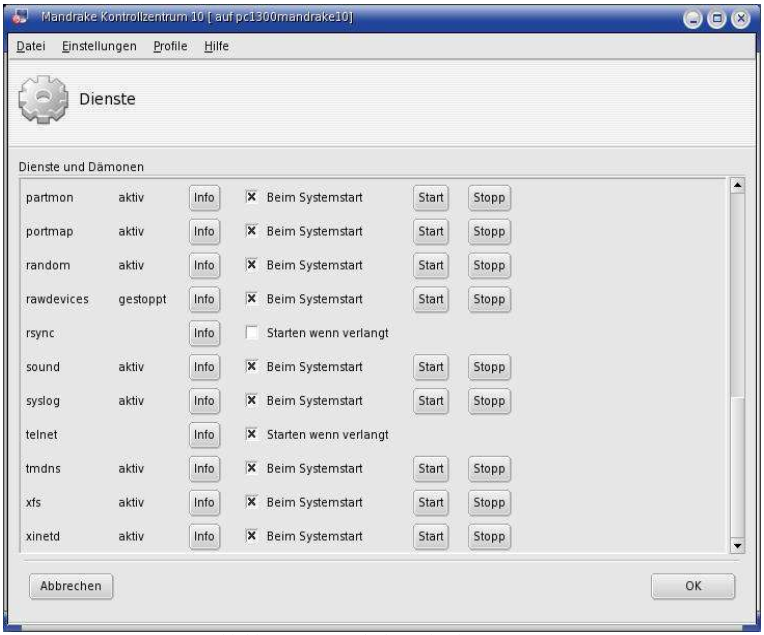
Schritt	Inhalt
1	Drücken Sie die Start-Taste (mit dem roten Hut). Danach geht es weiter mit den Menüpunkten ***Systemeinstellungen*** und ***Hinzufügen / Entfernen von Applikationen*** .
2	Wählen Sie als nächsten Punkt den Eintrag ***Netzwerk Server*** aus (siehe unten). Überprüfen Sie auf jeden Fall, die Details zu diesen Paketen. Klicken Sie auf die Information ***Details*** von ***Netzwerk Server***. Anschließend geht es beim Schritt 3 weiter.  *Abbildung 97 Red Hat und Telnet-Pakete* Bei Fedora3 finden Sie das Paket unter dem Eintrag ***Alte Netzwerk-Server***.

Schritt	Inhalt
3	Der telnet-Server muss in diesem Fenster dann aktiviert sein.  *Abbildung 98 Red Hat und Telnet-Serverprogramm* Bei Fedora3 klicken Sie ebenfalls diesen Eintrag an.
4	Nun drücken Sie den Knopf ***Schließen*** und die Software wird auf Ihrem Host installiert. In der Fedora-Distribution sehen Sie noch einen Hinweis und drücken den Knopf ***Weiter***.

Schritt	Inhalt
5	 *Abbildung 99 Red Hat und Telnet - Dienste* Im letzten Schritt aktivieren Sie noch beide Dienste (***Start-Taste, Systemeinstellungen, Servereinstellungen***). Bitte achten Sie auch auf den Dienst ***xinetd***, da telnet ihn benötigt (siehe im Bild den Punkt WICHTIG).
6	Weiter geht es nun mit dem Kapitel Mit Telnet arbeiten.

Mandrake 10 und telnet

Schritt	Inhalt
1	Drücken Sie die Start-Taste. Danach geht es weiter mit den Menüpunkten ***System, Einstellungen, Paketierung*** und ***Software installieren***.
2	Um die Suche nach dem Softwarepaket telnet einfacher zu gestalten, geben Sie den Namen als Suchbegriff ein.  *Abbildung 100 Mandrake und Telnet - Software auswählen* Wählen Sie dann den Eintrag ***telnet-server-krb5*** aus (**ACHTUNG:** in der momentanen Einstellung zeigt das Programm die schon installierten Softwarepakete nicht an). Drücken Sie danach den Knopf ***installieren***.

Schritt	Inhalt
3	 *Abbildung 101 Mandrake und Telnet - Dienste* Nach der Installation von telnet können Sie die Eintragungen zu diesem Dienst im Mandrake Kontrollzentrum überprüfen.

Mit Telnet arbeiten

Die Administration mit ***telnet*** gestaltet sich nun sehr einfach. Sie können hier mit allen textorientierten Programmen arbeiten.

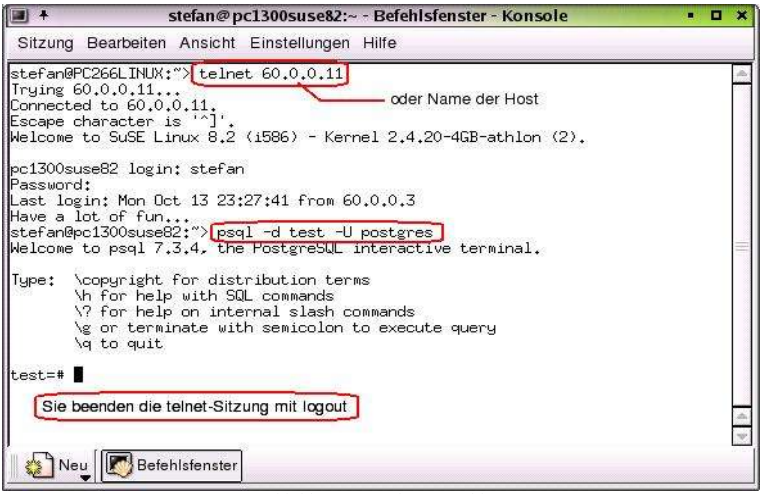


Abbildung 102 Telnet anmelden

Nachdem Sie die Datenbank-Sitzung beendet haben, melden Sie sich mit dem Befehl ***logout*** von der LINUX-Host ab.

Administration über das Programm EMS PostgreSQLManager

Das Programm finden Sie im Internet unter www.ems-hitech.com/pgmanager/download.phtml. Neben ***pgAdmin*** gibt es auch noch weitere Programme zur Administration der Datenbank. Es ist allerdings nicht kostenlos. Für 30 Tage können Sie es ohne Lizenzgebühren einsetzen. Im folgenden Bild sehen Sie einige Möglichkeiten, die das Programm bietet.

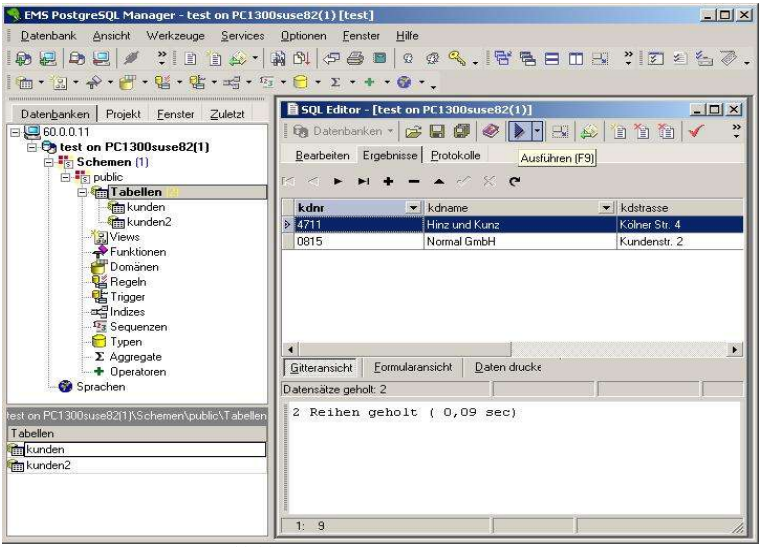


Abbildung 103 EMS PostgreSQL Manager

Administration über das Programm pgAdminIII

Das Programm finden Sie im Internet unter pgadmin.postgresql.org. Der pgAdminIII administriert nur Datenbanken ab der Version 7.3 (***ANMERKUNG:*** Bei der Windows-Version 8.0 ist es schon dabei). Mit dem Programm pgAdminIII können Sie auch mehrere Datenbank-Server administrieren. Einen neuen Datenbank-Server fügen Sie sehr einfach hinzu. Markieren Sie einen Datenbank-Server oder den Eintrag Server. Drücken Sie die rechte Maustaste. Danach sehen Sie anschließend im Menue die Zeile ***Server hinzufügen***.

Abbildung 104pgAdmin III - Server hinzufügen

An einem Datenbank-Server melden Sie sich entweder durch einen Doppelklick auf den Eintrag an, oder auch über die rechte Maustaste (siehe unten).

Abbildung 105 pgAdmin III - Anmelden

Abbildung 106 pgAdmin III-Eingabe Passwort

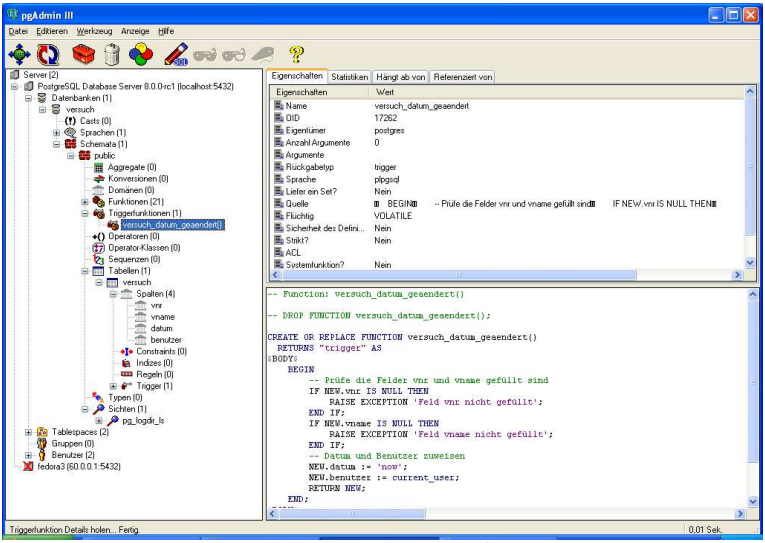


Abbildung 107 pgAdmin III Aufbau

Ähnlich wie beim Enterprise Manager vom MS SQL-Server können Sie sich nun durch den Baum hangeln. Generell gilt: Neue Einträge fügen Sie über die rechte Maustaste und den dann folgenden Menueeintrag hinzu.

Anlage D – die Funktionen von PostgreSQL

Hier erhalten Sie eine Übersicht der verschiedenen Funktionen (ein Auszug aus dem Buch PostgreSQL: Das offiziellen Handbuch).

Mathematische Funktionen

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*	*Ergebnis*
abs(*x*)	(gleicher wie *x*)	Betrag	abs(-17.4)	17.4
cbrt(dp)	dp	Kubikwurzel	cbrt(27.0)	3
ceil(dp oder numeric)	numeric	kleinste ganze Zahl, die nicht kleiner als das Argument ist	ceil(-42.8)	-42
degrees(dp)	dp	Radiant in Grad umwandeln	degrees(0.5)	28.6478897565412
exp(dp oder numeric)	dp	Potenzierung	exp(1.0)	2.71828182845905
floor(dp oder numeric)	numeric	größte ganze Zahl, die nicht größer als das Argument ist	floor(-42.8)	-43
ln(dp oder numeric)	dp	natürlicher Logarithmus	ln(2.0)	0.693147180559945
log(dp oder numeric)	dp	Logarithmus zur Basis 10	log(100.0)	2
log(*b* numeric, *x* numeric)	numeric	Logarithmus zur Basis *b*	log(2.0, 64.0)	6.00000000000
mod(*y*, *x*)	(gleicher wie Argumenttypen)	Rest von *y*/*x*	mod(9,4)	1
pi()	dp	Konstante „π"	pi()	3.14159265358979
pow(*a* dp, *b* dp)	dp	*a* hoch *b*	pow(9.0, 3.0)	729
pow(*a* numeric, *b* numeric)	numeric	*a* hoch *b*	pow(9.0, 3.0)	729
radians(dp)	dp	Grad in Radiant umwandeln	radians(45.0)	0.785398163397448
random()	dp	zufälliger Wert zwischen 0.0 und 1.0	random()	

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*	*Ergebnis*
round(dp or numeric)	(wie Argument)	zur nächsten ganzen Zahl runden	round(42.4)	42
round(*v* numeric, *s* integer)	numeric	auf *s* Dezimalstellen runden	round(42.4382, 2)	42.44
sign(dp oder numeric)	(wie Argument)	Vorzeichen des Arguments (-1, 0, +1)	sign(-8.4)	-1
sqrt(dp)	dp	Quadratwurzel	sqrt(2.0)	1.4142135623731
trunc(dp)	dp	Richtung null runden (abschneiden)	trunc(42.8)	42
trunc(numeric, *r* integer)	numeric	auf *r* Dezimalstellen abschneiden	trunc(42.4382, 2)	42.43

Trigonometrische Funktionen

Funktion	***Beschreibung***
acos(*x*)	Arkuskosinus
asin(*x*)	Arkussinus
atan(*x*)	Arkustangens
atan2(*x*, *y*)	Arkustangens von *x*/*y*
cos(*x*)	Kosinus
cot(*x*)	Kotangens
sin(*x*)	Sinus
tan(*x*)	Tangens

SQL-Zeichenkettenfunktionen und -operatoren

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*	*Ergebnis*
string || *string*	text	Zeichenketten aneinanderhängen	'Post' || 'greSQL'	PostgreSQL
bit_length(*string*)	integer	Anzahl Bits in Zeichenkette	bit_length('jose')	32
char_length(*string*) or character_length (*string*)	integer	Anzahl Zeichen in Zeichenkette	char_length('jose')	4
convert(*string* using *konversionsname*)	text	Ändert die Kodierung mit der angegebenen Konversion. Konversionen können mit CREATE CONVERSION erzeugt werden. Es gibt auch vordefinierte Konversionen;	convert('PostgreSQL' using iso_8859_1_to_utf_8)	'PostgreSQL' in Unicode-Kodierung (UTF-8)
lower(*string*)	text	in Kleinbuchstaben umwandeln	lower('TOM')	tom
octet_length(*string*)	integer	Anzahl Bytes in Zeichenkette	octet_length('jose')	4
overlay(*string* placing *string* from integer [for integer])	text	Teilzeichenkette ersetzen	overlay('Txxxxas' placing 'hom' from 2 for 4)	Thomas
position(*substring* in *string*)	integer	Position der angegebenen Teilzeichenkette	position('om' in 'Thomas')	3
substring(*string* [from integer] [for integer])	text	Teilzeichenkette ermitteln	substring('Thomas' from 2 for 3)	hom
substring(*string* from *patterm*)	text	Teilzeichenkette nach POSIX regulärem Ausdruck ermitteln	substring('Thomas' from '...$')	mas
substring(*string* from *patterm* for *escape*)	text	Teilzeichenkette nach SQL regulärem Ausdruck ermitteln	substring('Thomas' from '%#"o_a#"_' for '#')	oma
trim([leading | trailing | both] [*zeichen*] from *string*)	text	Entfernt die längste Zeichenkette, die nur aus *zeichen* (als Vorgabe ein Leerzeichen) besteht, von Anfang/Ende/beiden Enden von *string*	trim(both 'x' from 'xTomxx')	Tom
upper(*string*)	text	in Großbuchstaben umwandeln	upper('tom')	TOM

Andere Zeichenkettenfunktionen

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*	*Ergebnis*
ascii(text)	integer	ASCII-Code des ersten des Arguments	ascii('x')	120
btrim(*string* text, *zeichen* text)	text	Entfernt die längste Zeichenkette, die nur aus Zeichen in *zeichen* steht, vom Anfang und Ende von *string*	btrim('xyxtrimyyx','xy')	trim
chr(integer)	text	Zeichen mit dem angegebenen ASCII-Code	chr(65)	A
convert(*string* text, [*quellkodierung* name,] *zielkodierung* name)	text	Wandelt Zeichenkette von *quellkodierung* nach *zielkodierung* um. Wenn *quellkodierung* ausgelassen wird, dann wird die Datenbankkodierung angenommen.	convert('text_in_unicode', 'UNICODE', 'LATIN1')	text_in_unicode in der Kodierung ISO 8859-1
decode(*string* text, *typ* text)	bytea	Dekodiert binäre Daten von *string*, die zuvor mit encode kodiert wurden. Parametertyp ist genauso wie bei encode.	decode('MTIzAAE=', 'base64')	123\000\001
encode(*data* bytea, *typ* text)	text	Kodiert binäre Daten in eine Darstellung nur aus ASCII-Zeichen. Unterstützte Typen sind: base64, hex, escape.	encode('123\\000\\001', 'base64')	MTIzAAE=
initcap(text)	text	Wandelt ersten Buchstaben jedes Wortes (durch Whitespace getrennt) in Großbuchstaben um	initcap('hi thomas')	Hi Thomas
length(*string*)	integer	Anzahl Zeichen in Zeichenkette	length('jose')	4
lpad(*string* text, *länge* integer [, *füllung* text])	text	Füllt *string* bis zur Länge *länge*, indem Zeichen *füllung* (ein Leerzeichen als Vorgabe) vorangestellt werden. Wenn *string* schon länger als *länge* ist, dann wird es rechts abgeschnitten.	lpad('hi', 5, 'xy')	xyxhi
ltrim(*string* text, *zeichen* text)	text	Entfernt die längste Zeichenkette, die nur aus Zeichen in *zeichen* besteht, vom Anfang der Zeichenkette.	ltrim('zzzytrim','xyz')	trim
pg_client_encoding()	name	aktuelle Clientkodierung	pg_client_encoding()	SQL_ASCII
quote_ident(*string* text)	text	Gibt die Zeichenkette so zurück, dass sie als Name in einem SQL-Befehl verwendet werden kann. Anführungszeichen werden hinzugefügt wenn nötig (d.h. Zeichenkette enthält Sonderzeichen oder Großbuchstaben). Vorhandene Anführungszeichen werden korrekt verdoppelt.	quote_ident('Foo')	"Foo"

SQL-Funktionen und -Operatoren für binäre Datenbanken

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*	*Ergebnis*
daten || *daten*	bytea	Datenketten aneinanderhängen	'\\\\Post'::bytea || '\\047greSQL\\000'::bytea	\\Post'greSQL\000
octet_length(*daten*)	integer	Anzahl Bytes in Daten	octet_length ('jo\\000se'::bytea)	5
position(*subdaten* in *daten*)	integer	Position der angegebenen Teildatenkette	position('\\000om'::bytea in 'Th\\000omas'::bytea)	3
substring(*daten* [from integer] [for integer])	bytea	Teildatenkette ermitteln	substring('Th\\000omas'::bytea from 2 for 3)	h\000o
trim([both] *bytes* from *daten*)	bytea	Entfernt die längste Datenkette, die nur aus Bytes in *bytes* besteht, vom Anfang und Ende von *daten*.	trim('\\000'::bytea from '\\000Tom\\000'::bytea)	Tom

Andere Funktionen für binäre Daten

Funktion	*Ergebnistyp*	*Beschränkung*	*Beispiel*	*Ergebnis*
btrim(*daten* bytea *bytes* bytea)	bytea	Entfernt die längste Datenkette, die nur aus Bytes in *bytes* steht, vom Anfang und Ende von *daten*	btrim ('\\000trim\\000'::bytea,'\\000'::bytea)	trim
length(*daten*)	integer	Anzahl Bytes in Daten	length('jo\\000se'::bytea)	5
decode(*daten* text, *typ* text)	bytea	Dekodiert binäre Daten in *daten*, die zuvor mit encode kodiert wurden. Parametertyp ist genauso wie bei encode.	decode('123\\000456', 'escape')	123\000456
encode(*daten* bytea, *typ* text)	text	Kodiert binäre Daten in eine Darstellung nur aus ASCII-Zeichen. Unterstützte Typen sind: base64, hex, escape.	encode('123\\000456'::bytea, 'escape')	123\000456

Datentyp-Formatierungsfunktionen

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*
to_char(timestamp, text)	text	timestamp in Zeichenkette umwandeln	to_char(current_timestamp, 'HH12:MI:SS')
to_char(interval, text)	text	interval in Zeichenkette umwandeln	to_char(interval '15h 2m 12s', 'HH24:MI:SS')
to_char(int, text)	text	integer in Zeichenkette umwandeln	to_char(125, '999')
to_char(double precision, text)	text	real/double precision in Zeichenkette umwandeln	to_char(125.8::real, '999D9')
to_char(numeric, text)	text	numeric in Zeichenkette umwandeln	to_char(-125.8, '999D99S')
to_date(text, text)	date	Zeichenkette in date umwandeln	to_date('05 Dec 2000', 'DD Mon YYYY')
to_timestamp(text, text)	timestamp	Zeichenkette in timestamp umwandeln	to_timestamp('05 Dec 2000', 'DD Mon YYYY')
to_number(text, text)	numeric	Zeichenkette in numeric umwandeln	to_number('12,454.8-', '99G999D9S')

Mustervorlagen

Mustervorlagen für die Formatierung von Datum/Zeit

Muster	***Beschreibung***
HH	Stunde (01-12)
HH12	Stunde (01-12)
HH24	Stunde (00-23)
MI	Minute (00-59)
SS	Sekunde (00-59)
MS	Millisekunde (000-999)
US	Mikrosekunde (000000-999999)
SSSS	Sekunden nach Mitternacht (0-86399)
AM oder A.M. oder PM oder P.M.	Vormittags-/Nachmittagsangabe (Großbuchstaben)
am oder a.m. oder pm oder p.m.	Vormittags-/Nachmittagsangabe (Kleinbuchstaben)
Y,YYY	Jahr (mind. 4 Ziffern) mit Komma zur Zifferngruppierung
YYYY	Jahr (mind. 4 Ziffern)
YYY	die letzten 3 Ziffern des Jahres
YY	die letzten 2 Ziffern des Jahres
Y	die letzte Ziffer des Jahres
BC oder B.C. oder AD oder A.D.	englische Angabe der Zeitrechnung (Großbuchstaben)
bc oder b.c. oder ad oder a.d.	englische Angabe der Zeitrechnung (Kleinbuchstaben)
MONTH	voller englischer Monatsname in Großbuchstaben (mit Leerzeichen auf 9 Zeichen aufgefüllt)
Month	voller englischer Monatsname in gemischten Buchstaben (mit Leerzeichen auf 9 Zeichen aufgefüllt)
month	voller englischer Monatsname in Kleinbuchstaben (mit Leerzeichen auf 9 Zeichen aufgefüllt)
MON	abgekürzter englischer Monatsname in Großbuchstaben (3 Zeichen)

Muster	***Beschreibung***
Mon	abgekürzter englischer Monatsname in gemischten Buchstaben (3 Zeichen)
mon	abgekürzter englischer Monatsname in Kleinbuchstaben (3 Zeichen)
MM	Nummer des Monats (01-12)
DAY	voller englischer Wochentag in Großbuchstaben (mit Leerzeichen auf 9 Zeichen aufgefüllt)
Day	voller englischer Wochentag in gemischten Buchstaben (mit Leerzeichen auf 9 Zeichen aufgefüllt)
day	voller englischer Wochentag in Kleinbuchstaben (mit Leerzeichen auf 9 Zeichen aufgefüllt)
DY	abgekürzter englischer Wochentag in Großbuchstaben (3 Zeichen)
Dy	abgekürzter englischer Wochentag in gemischten Buchstaben(3 Zeichen)
dy	abgekürzter englischer Wochentag in Kleinbuchstaben (3 Zeichen)
DDD	Tag im Jahr (001-366)
DD	Tag im Monat (01-31)
D	Wochentag (1-7; Sonntag ist 1)
W	Woche im Monat (1-5) (Die erste Woche fängt am ersten Tag des Monats an)
WW	Woche im Jahr (1-53) (Die erste Woche fängt am ersten Tag des Jahres an)
IW	ISO-Wochennummer (Der erste Donnerstag des neuen Jahres ist in Woche Nummer 1)
CC	Jahrhundert (2 Ziffern)
J	Julianischer Tag (Tage seit 1. Januar 4712 v.u.Z.)
Q	Quartal
RM	Monat in römischen Zahlen (I-XII; I=Januar) (Großbuchstaben)
rm	Monat in römischen Zahlen (i-xii; i=Januar) (Kleinbuchstaben)
TZ	Zeitzonenname (Großbuchstaben)
tz	Zeitzonenname (Kleinbuchstaben)

Mustervorlagen für die Formatierung von numerischen Werten

Muster	***Beschreibung***
9	Wert mit der angegebenen Anzahl Ziffern
0	Wert mit führenden Nullen
. (Punkt)	Punkt zur Trennung von Nachkommastellen
, (Komma)	Komma als Tausendergruppierung
PR	negativer Wert in spitzen Klammern
S	Vorzeichen direkt neben der Zahl (verwendet Locale)
L	Währungssymbol (verwendet Locale)
D	Trennzeichen für Nachkommastellen nach Locale (also Komma auf Deutsch)
G	Zeichen zur Tausendergruppierung nach Locale (Punkt auf Deutsch)
MI	Minuszeichen auf angegebener Position (wenn Zahl < 0)
PL	Pluszeichen auf angegebener Position (wenn Zahl > 0)
SG	Plus-/Minuszeichen auf angegebener Position
RN	römische Zahl (Eingabe zwischen 1 und 3999)
TH oder th	englische Ordnungszahlendung

Funktionen für Datum/Zeit

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*	*Ergebnis*
age(timestamp)	interval	Subtraktion vom heutigen Datum	age(timestamp '1957-06-13')	43 years 8 mons 3 days
age(timestamp, timestamp)	interval	Subtraktion	age('2001-04-10', timestamp '1957-06-13')	43 years 9 mons 27 days
current_date	date	heutiges Datum		
current_time	time with time zone	aktuelle Zeit		
current_timestamp	timestamp with time zone	aktuelle Zeit mit Datum		
date_part(text, timestamp)	double precision	Teilfeld ermitteln (gleichbedeutend mit extract)	date_part('hour', timestamp '2001-02-16 20:38:40')	20
date_part(text, interval)	double precision	Teilfeld ermitteln (gleichbedeutend mit extract)	date_part('month', interval '2 years 3 months')	3
date_trunc(text, timestamp)	timestamp	auf angegebene Genauigkeit abschneiden	date_trunc('hour', timestamp '2001-02-16 20:38:40')	2001-02-16 20:00:00+00
extract(*field* from timestamp)	double precision	Teilfeld ermitteln	extract(hour from timestamp '2001-02-16 20:38:40')	20
extract(*field* from interval)	double precision	Teilfeld ermitteln	extract(month from interval '2 years 3 months')	3
isfinite(timestamp)	boolean	auf endlichen Wert prüfen (ungleich infinity)	isfinite(timestamp '2001-02-16 21:28:30')	true
isfinite(interval)	boolean	auf endlichen Wert prüfen (ungleich infinity)	isfinite(interval '4 hours')	true
localtime	time	aktuelle Zeit		
localtimestamp	timestamp	aktuelle Zeit mit Datum		
now()	timestamp with time zone	aktuelle Zeit mit Datum (gleichbedeutend mit current_timestamp)		
timeofday()	text	aktuelle Zeit mit Datum	timeofday()	Wed Feb 21 17:01:13.000126 2001 EST

Geometrische Funktionen

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiele*
area(*objekt*)	double precision	Flächeninhalt	area(box '((0,0),(1,1))')
box(box, box)	box	Schnittrechteck	box(box '((0,0),(1,1))',box '((0.5,0.5),(2,2))')
center(*objekt*)	point	Mittelpunkt	center(box '((0,0),(1,2))')
diameter(circle)	double precision	Durchmesser des Kreises	diameter(circle '((0,0),2.0)')
height(box)	double precision	Höhe des Rechtecks	height(box '((0,0),(1,1))')
isclosed(path)	boolean	geschlossener Pfad?	isclosed(path '((0,0),(1,1),(2,0))')
isopen(path)	boolean	offener Pfad?	isopen(path '[(0,0),(1,1),(2,0)]')
length(*objekt*)	double precision	Länge	length(path '((-1,0),(1,0))')
npoints(path)	integer	Anzahl der Punkte	npoints(path '[(0,0),(1,1),(2,0)]')
npoints(polygon)	integer	Anzahl der Punkte	npoints(polygon '((1,1),(0,0))')
pclose(path)	path	Pfad in geschlossenen umwandeln	popen(path '[(0,0),(1,1),(2,0)]')
popen(path)	path	Pfad in offenen umwandeln	popen(path '((0,0),(1,1),(2,0))')
radius(circle)	double precision	Radius des Kreises	radius(circle '((0,0),2.0)')
width(box)	double precision	Breite des Rechtecks	width(box '((0,0),(1,1))')

Geometrische Typumwandlungsfunktionen

Funktion	*Ergebnistyp*	*Beschreibung*	*Beispiel*
box(circle)	box	Kreis in Rechteck	box(circle '((0,0),2.0)')
box(point, point)	box	Punkte in Rechteck	box(point '(0,0)', point '(1,1)')
box(polygon)	box	Polygon in Rechteck	box(polygon '((0,0),(1,1),(2,0))')
circle(box)	circle	Rechteck in Kreis	circle(box '((0,0),(1,1))')
circle(point, double precision)	circle	Punkt und Radius in Kreis	circle(point '(0,0)', 2.0)
lseg(box)	lseg	Rechteckdiagonale in Strecke	lseg(box '((-1,0),(1,0))')
lseg(point, point)	lseg	Punkte in Strecke	lseg(point '(-1,0)', point '(1,0)')
path(polygon)	point	Polygon in Pfad	path(polygon '((0,0),(1,1),(2,0))')
point(circle)	point	Mittelpunkt des Kreises	point(circle '((0,0),2.0)')
point(lseg, lseg)	point	Schnittpunkt	point(lseg '((-1,0),(1,0))', lseg '((-2,-2),(2,2))')
point(polygon)	point	Mittelpunkt des Polygons	point(polygon '((0,0),(1,1),(2,0))')
polygon(box)	polygon	Rechteck in 4-Punkte-Polygon	polygon(box '((0,0),(1,1))')
polygon(circle)	polygon	Kreis in 12-Punkte Polygon	polygon(circle '((0,0),2.0)')
polygon(*npkte*, circle)	polygon	Kreis in *npkte*-Punkte-Polygon	polygon(12, circle '((0,0),2.0)')
polygon(path)	polygon	Pfad in Polygon	polygon(path '((0,0),(1,1),(2,0))')

Sitzungsinformationsfunktionen

Name	***Ergebnis-typ***	***Beschreibung***
current_database()	name	Name der aktuellen Datenbank
current_schema()	name	Name des aktuellen Schemas
current_schemas(boolean)	name[]	Name der Schemas in Suchpfad, wahlweise einschließlich der impliziten Schemas
current_user	name	Benutzername der aktuellen Ausführungsumgebung
session_user	name	Benutzername der Sitzung
user	name	äquivalent mit current_user
version()	text	PostgreSQL-Versionsinformationen

Aggregatfunktionen

Funktion	*Argumenttyp*	*Ergebnistyp*	*Beschreibung*
avg(*ausdruck*)	smallint, integer, bigint, real, double precision, numeric oder interval	numeric für ganzzahlige Argumente, double precision für Fließkommaargumente, ansonsten gleich dem Argumentdatentyp	Durchschnitt (arithmetisches Mittel) aller Eingabewerte
count(*)		bigint	Anzahl der Eingabewerte
count (*ausdruck*)	egal	bigint	Anzahl der Eingabewerte, für die *ausdruck* nicht gleich dem NULL-Wert ist
max(*ausdruck*)	numerisch, Zeichenkette oder Datum/Zeit	gleich dem Argumenttyp	Maximalwert von *ausdruck* aus allen Eingabewerten
min(*ausdruck*)	numerisch, Zeichenkette oder Datum/Zeit	gleich dem Argumenttyp	Minimalwert von *ausdruck* aus allen Eingabewerten
stddev (*ausdruck*)	smallint, integer, bigint, real, double precision oder numeric	double precision für Fließkommaargumente, ansonsten numeric	Standardabweichung der Eingabewerte
sum(*ausdruck*)	smallint, integer, bigint, real, double precision, numeric oder interval	bigint für smallint- oder integer-Argumente, numeric für bigint-Argumente, double precision für Fließkommaargumente, ansonsten gleich dem Argumenttyp	Summe von *ausdruck* für alle Eingabewerte
variance (*ausdruck*)	smallint, integer, bigint, real, double precision oder numeric	double precision für Fließkommaargumente, ansonsten numeric	Varianz der Eingabewerte (Quadrat der Standardabweichung)

Anhang E – die Zeichensätze

In dieser Anlage finden Sie die vom Server unterstützten Zeichensätze

Name	*Beschreibung*
SQL_ASCII	ASCII
EUC_JP	Japanischer EUC
EUC_CN	Chinesischer EUC
EUC_KR	Koreanischer EUC
JOHAB	Koreanischer EUC (Basis-Hangul)
EUC_TW	Taiwanischer EUC
UNICODE	Unicode (UTF-8)
MULE_INTERNAL	Mule Internal Code
LATIN1	ISO 8859-1/ECMA 94 (Lateinisches Alphabet Nr.1)
LATIN2	ISO 8859-2/ECMA 94 (Lateinisches Alphabet Nr.2)
LATIN3	ISO 8859-3/ECMA 94 (Lateinisches Alphabet Nr.3)
LATIN4	ISO 8859-4/ECMA 94 (Lateinisches Alphabet Nr.4)
LATIN5	ISO 8859-9/ECMA 128 (Lateinisches Alphabet Nr.5)
LATIN6	ISO 8859-10/ECMA 144 (Lateinisches Alphabet Nr.6)
LATIN7	ISO 8859-13 (Lateinisches Alphabet Nr.7)
LATIN8	ISO 8859-14 (Lateinisches Alphabet Nr.8)
LATIN9	ISO 8859-15 (Lateinisches Alphabet Nr.9)
LATIN10	ISO 8859-16/ASRO SR 14111 (Lateinisches Alphabet Nr.10)
ISO-8859-5	ISO 8859-5/ECMA 113 (Lateinisch/Kyrillisch)
ISO-8859-6	ISO 8859-6/ECMA 114 (Lateinisch/Arabisch)
ISO-8859-7	ISO 8859-7/ECMA 118 (Lateinisch/Griechisch)
ISO-8859-8	ISO 8859-8/ECMA 121 (Lateinisch/Hebräisch)
KOI8	KOI8-R(U)
WIN	Windows CP1251
ALT	Windows CP866
WIN1256	Windows CP1256 (Arabisch)
TCVN	TCVN-5712/Windows CP1258 (Vietnamesisch)
WIN874	Windows CP874 (Thai)

Anhang F - Liste der Dienstprogramme

In der Tabelle sind die Dienstprogramme aufgeführt. Einige von ihnen haben Sie ja schon kennengelernt.

Dienstprogramm	***Zweck***
clusterdb	Reorganisiert die Cluster einer Tabelle in einer Postgres Datenbank
createdb	Legt eine neue Datenbank an
createlang	Installiert eine prozedurale Sprache auf einer Datenbank
createuser	Legt einen Datenbank-Benutzer an
dropdb	Löscht eine Datenbank
droplang	Löscht eine prozedurale Sprache aus einer Datenbank
dropuser	Löscht einen Datenbank-Benutzer
initdb	Legt einen neuen Datenbank-Cluster an
pg_config	Gibt Informationen über die installierte Datenbankversion aus.
pg_controldata	Zeigt Informationen über den Server an
pg_ctl	Startet, beendet, oder führt einen Restart eines Datenbankservers durch
pg_dump	Für die Datensicherung einer einzelnen Datenbank. Die Ausgabe erfolgt in eine Datei
pg_dumpall	Für eine komplette Datensicherung. Das Programm schreibt alle Datenbanken in eine Datei
pg_resetxlog	Bereinigt den write-ahead log und den Inhalt der Datei pg_control
pg_restore	Stellt eine PostgreSQL Datenbank wieder her. Ausgangsbasis ist eine Datei, die von pg_dump erzeugt wurde
postgres	Betreibt die Datenbank im single-user mode
postmaster	Das Server-Programm erlaubt den Zugriff auf die Datenbank durch die verschiedenen Clients
psql	Ein Terminal-Programm für PostgreSQL
vacuumdb	Analysiert und bereinigt PostgreSQL Datenbanken

Anhang G – Fehlercode beim embedded SQL

-12: Out of memory in line %d.

Sollte normalerweise nicht auftreten. Gibt an, dass der virtuelle Speicher aufgebraucht ist.

-200 (ECPG_UNSUPPORTED): Unsupported type %s on line %d.

Sollte normalerweise nicht auftreten. Zeigt an, dass der Präprozessor etwas erzeugt hat, das die Bibliothek nicht kennt. Vielleicht haben Sie inkompatible Versionen von Präprozessor und Bibliothek verwendet.

-201 (ECPG_TOO_MANY_ARGUMENTS): Too many arguments line %d.

Dies bedeutet, dass der Server mehr Argumente zurückgegeben hat als wir passende Variablen haben. Vielleicht haben Sie ein paar Hostvariablen in der Liste INTO :var1, :var2 vergessen.

-202 (ECPG_TOO_FEW_ARGUMENTS): Too few arguments line %d.

Dies bedeutet, dass der Server weniger Argumente zurückgegeben hat als wir Hostvariablen haben. Vielleicht haben Sie zu viele Hostvariablen in der Liste INTO :var1, :var2.

-203 (ECPG_TOO_MANY_MATCHES): Too many matches line %d.

Dies bedeutet, dass die Anfrage mehrere Zeilen ergab, aber die angegebenen Variablen keine Arrays sind. Der SELECT-Befehl sollte nur eine Zeile ergeben.

-204 (ECPG_INT_FORMAT): Not correctly formatted int type: %s line %d.

Dies bedeutet, dass die Hostvariable vom Typ int ist und das Feld in der PostgreSQL-Datenbank einen anderen Typ hat und einen Wert enthält, der nicht als int interpretiert werden kann. Die Bibliothek verwendet strtol() für diese Umwandlung.

-205 (ECPG_UINT_FORMAT): Not correctly formatted unsigned type: %s line %d.

Dies bedeutet, dass die Hostvariable vom Typ unsigned int ist und das Feld in der PostgreSQL-Datenbank einen anderen Typ hat und einen Wert enthält, der nicht als unsigned int interpretiert werden kann. Die Bibliothek verwendet strtoul() für diese Umwandlung.

-206 (ECPG_FLOAT_FORMAT): Not correctly formatted floating-point type: %s line %d.

Dies bedeutet, dass die Hostvariable vom Typ float ist und das Feld in der PostgreSQL-Datenbank einen anderen Typ hat und einen Wert enthält, der nicht als float interpretiert werden kann. Die Bibliothek verwendet strtod() für diese Umwandlung.

-207 (ECPG_CONVERT_BOOL): Unable to convert %s to bool on line %d.

Dies bedeutet, dass die Hostvariable vom Typ bool ist und das Feld in der PostgreSQL-Datenbank weder 't' noch 'f' ist.

-208 (ECPG_EMPTY): Empty query line %d.

Die Anfrage war leer. (Das kann in einem eingebetteten SQL-Programm normalerweise nicht passieren und könnte daher einen internen Fehler aufzeigen).

-209 (ECPG_MISSING_INDICATOR): NULL value without indicator in line %d.

Ein NULL-Wert wurde zurückgegeben und keine NULL-Indikatorvariable wurde angegeben.

-210 (ECPG_NO_ARRAY): Variable is not an array in line %d.

Eine normale Variable wurde verwendet wo ein Array erforderlich ist.

-211 (ECPG_DATA_NOT_ARRAY): Data read from backend is not an array in line %d.

Die Datenbank gab eine normale Variable zurück wo ein Array erforderlich ist.

-220 (ECPG_NO_CONN): No such connection %s in line %d.

Das Programm versuchte auf eine Verbindung zuzugreifen, die nicht existiert.

-221 (ECPG_NOT_CONN): Not connected in line %d.

Das Programm versuchte auf eine Verbindung zuzugreifen, die existiert aber nicht geöffnet ist.

-230 (ECPG_INVALID_STMT): Invalid statement name %s in line %d.

Der Befehl, den Sie versuchen zu verwenden, wurde nicht vorbereitet.

-240 (ECPG_UNKNOWN_DESCRIPTOR): Descriptor %s not found in line %d.

Der angegebene Deskriptor wurde nicht gefunden. Der Befehl, den Sie versuchen zu verwenden, wurde nicht vorbereitet.

-241 (ECPG_INVALID_DESCRIPTOR_INDEX): Descriptor index out of range in line %d.

Der angegebene Deskriptorindex ist außerhalb des gültigen Bereichs.

-242 (ECPG_UNKNOWN_DESCRIPTOR_ITEM): Unknown descriptor item %s in line %d.

Der angegebene Deskriptor wurde nicht gefunden. Der Befehl, den Sie versuchen zu verwenden, wurde nicht vorbereitet.

-243 (ECPG_VAR_NOT_NUMERIC): Variable is not a numeric type in line %d.

Die Datenbank gab einen numerischen Wert zurück und die Variable war nicht numerisch.

-244 (ECPG_VAR_NOT_CHAR): Variable is not a character type in line %d.

Die Datenbank gab einen nichtnumerischen Wert zurück und die Variable war numerisch.

-400 (ECPG_PGSQL): '%s' in line %d.

Irgendein PostgreSQL-Fehler. Der Text enthält die Fehlermeldung vom PostgreSQL-Server.

-401 (ECPG_TRANS): Error in transaction processing line %d.

Der PostgreSQL-Server hat signalisiert, dass wir die Transaktion nicht starten, abschließen oder zurückrollen können.

-402 (ECPG_CONNECT): Could not connect to database %s in line %d.

Der Verbindungsversuch zur Datenbank ist fehlgeschlagen.

100 (ECPG_NOT_FOUND): Data not found line %d.

Dies ist ein „normaler“ Fehler, der aussagt, dass das, was sie abfragen wollen, nicht gefunden werden kann oder dass Sie am Ende des Cursors sind.

Quellen

1. Die Handbücher von Postgres, die im Lieferumfang enthalten sind
2. PostgreSQL Das offizielle Handbuch von Peter Eisentraut (von der PostgreSQL Global Development Group)
3. SuSE LINUX 8.0, die REFERENZ
4. Die Seite http://www.postgres.de
5. Linux von Michael Kofler
6. Die ersten Schritte in der Administration mit der PostgreSQL-Datenbank, von Stefan Kunick
7. Npgsql – User´s Manual
8. Zeitschrift iX – Dezember 2004, Artikel über PostgreSQL von Cornelia Boenigk und Ralf Burger

Abbildungsverzeichnis

Abbildung 1 SuSE YaST 12
Abbildung 2 SuSE Pakete auswählen 13
Abbildung 3 SuSE optimale Suche nach Paketen 13
Abbildung 4 SuSE Benutzerverwaltung 14
Abbildung 5 SuSE Benutzer anlegen 15
Abbildung 6 SuSE Benutzer und Gruppe 15
Abbildung 7 Red Hat Pakete Hinzufügen oder Entfernen Red Hat 16
Abbildung 8 Red Hat Paketdetails Red Hat 17
Abbildung 9 Red Hat Benutzerverwaltung 18
Abbildung 10 Red Hat Benutzereigenschaften Benutzerdaten 19
Abbildung 11 Red Hat Benutzereigenschaften Account-Info 19
Abbildung 12 Red Hat Benutzereigenschaft Passwort 20
Abbildung 13 Red Hat Benutzereigenschaft Gruppen 20
Abbildung 14 Mandrake - Pakete suchen 21
Abbildung 15 Mandrake - Pakete auswählen 22
Abbildung 16 Mandrake Benutzerverwaltung 23
Abbildung 17 Mandrake Benutzerverwaltung Gruppen 24
Abbildung 18 Mandrake Benutzerverwaltung Info über Konto 24
Abbildung 19 Mandrake Benutzerverwaltung Passwort-Info 25
Abbildung 20 Datenbank einrichten 26
Abbildung 21 Datenbank starten 27
Abbildung 22 PGDATA 28
Abbildung 23 Skript zum Start der Datenbank 29
Abbildung 24 Dienst-Konfiguration 31
Abbildung 25 Datenbank beenden 34
Abbildung 26 postgresql.conf – Inhalt 38
Abbildung 27 Postmaster 40
Abbildung 28 Datenbanken anlegen - createdb 43
Abbildung 29 createdb - ein Beispiel 44
Abbildung 30 Datenbank Aktivitäten überwachen 58
Abbildung 31 Version 8 Windows, Installation, Sprache auswählen 59
Abbildung 32 Version 8 Windows, Installation 59
Abbildung 33 Version 8 Windows, Installation, Hinweise 60
Abbildung 34 Version 8 Windows, Optionen 60
Abbildung 35 Version 8 Windows, Dienste-Konfiguration 61
Abbildung 36 Version 8 Windows, User postgres 61
Abbildung 37 Version 8 Windows, Password 62
Abbildung 38 Version 8 Windows, Service Rechte 62
Abbildung 39 Version 8 Windows, database cluster 63
Abbildung 40 Version 8 Windows, remote Zugriff 63
Abbildung 41 Version 8 Windows, prozedurale Sprachen 64

Abbildung 42 Version 8 Windows, weitere Module 65
Abbildung 43 Version 8 Windows, Ende der Installation 66
Abbildung 44Version 8 Windows, Dateien kopieren 66
Abbildung 45 Version 8 Windows, Installation vollständig 67
Abbildung 46 Version 8 Windows, psql Hinweis 68
Abbildung 47 Version 8 Windows, CMD 68
Abbildung 48Version 8 Windows, CMD Eigenschaften übernehmen 69
Abbildung 49 Version 8 Windows, Codepage 69
Abbildung 50 Version 8.0 Windows - Arbeiten mit psql 70
Abbildung 51 Version 8 Windows, Dienste 71
Abbildung 52 Version 8 Windows, Computerverwaltung 71
Abbildung 53 unixODBC-Softwarepakete 73
Abbildung 54 unixODBC-Data Source Administrator 74
Abbildung 55 unixODBC-Treiber 75
Abbildung 56 unixODBC-Datenquelle 76
Abbildung 57 unixODBC-Treibermanager 77
Abbildung 58 unixODBC-Red Hat Softwarepakete 78
Abbildung 59 unixODBC-Red Hat Softwarepakete 79
Abbildung 60 Mandrake - Installation psql 80
Abbildung 61 Windows ODBC-Treiber 81
Abbildung 62 Windows ODBC-Einstellungen 82
Abbildung 63 psql-Programmstart 83
Abbildung 64 psql-Liste verlassen 87
Abbildung 65 createuser ohne Optionen 90
Abbildung 66 createuser mit Optionen 90
Abbildung 67 createuser - Benutzer abfragen 90
Abbildung 68 dropuser – Beispiel 91
Abbildung 69 Tabellenaufbau anzeigen 110
Abbildung 70 einfache SQL-Funktion 115
Abbildung 71 Trigger – Insert 123
Abbildung 72 Trigger – Update 123
Abbildung 73 C-Programm-Kundenliste 130
Abbildung 74 C-Programm-Datensatz einfügen 133
Abbildung 75 Embedded SQL 135
Abbildung 76 ecpg – Fehlerbehandlung 138
Abbildung 77 ecpg - Programm Stammdaten 145
Abbildung 78 C#-Programm-Kundenliste 151
Abbildung 79 VB.NET - pgData 1 160
Abbildung 80 VB.NET - pgData 2 161
Abbildung 81 JAVA-Programm-Ausgabe 164
Abbildung 82 Serienbriefe-Menue 165
Abbildung 83 Serienbriefe-Datenquellen 166
Abbildung 84 Serienbriefe-Quelle auswählen 167
Abbildung 85 Serienbriefe-Tabelle auswählen 167
Abbildung 86 Serienbriefe-Abfragen 168
Abbildung 87 Serienbriefe-Abfrage erstellen 168

Abbildung 88 Serienbriefe – Datenbank 169
Abbildung 89 Serienbriefe - Felder 170
Abbildung 90 Serienbriefe - Datensätze 170
Abbildung 91 Daten in Postgres übernehmen , Fehler 171
Abbildung 92 Daten in Postgres übernehmen, Importdatei 172
Abbildung 93 Daten in Postgres übernehme, Daten prüfen 172
Abbildung 94 Daten in Postgres übernehmen, das Ergebnis 172
Abbildung 95 SuSE und Telnet-Netzwerkdienste 180
Abbildung 96 SuSE und Telnet-Konfiguration 181
Abbildung 97 Red Hat und Telnet-Pakete 182
Abbildung 98 Red Hat und Telnet-Serverprogramm 183
Abbildung 99 Red Hat und Telnet – Dienste 184
Abbildung 100 Mandrake und Telnet - Software auswählen 185
Abbildung 101 Mandrake und Telnet – Dienste 186
Abbildung 102 Telnet anmelden 187
Abbildung 103 EMS PostgreSQL Manager 188
Abbildung 104pgAdmin III - Server hinzufügen 189
Abbildung 105 pgAdmin III – Anmelden 189
Abbildung 106 pgAdmin III-Eingabe Passwort 189
Abbildung 107 pgAdmin III Aufbau 190