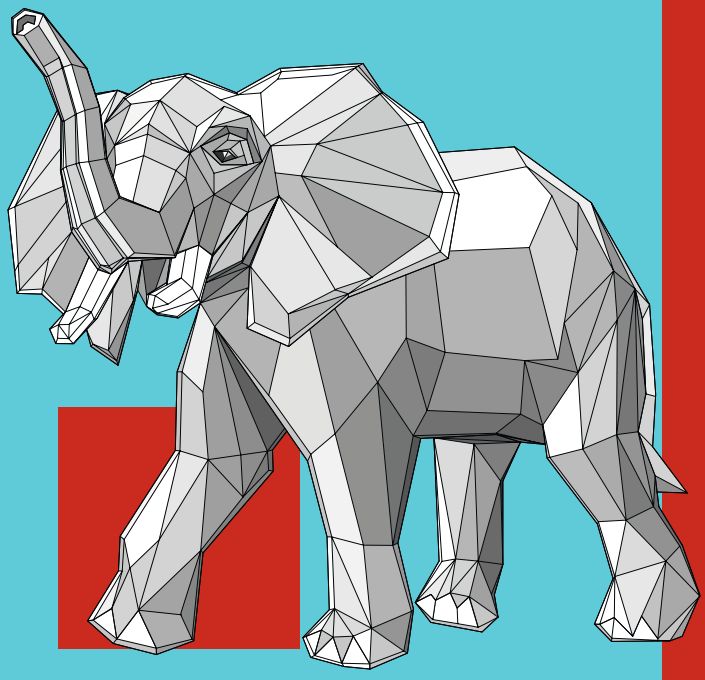

lutz FRÖHLICH

PostgreSQL 10

Praxisbuch für Administratoren und Entwickler

HANSER

Fröhlich

PostgreSQL 10

Lutz Fröhlich

PostgreSQL 10

Praxisbuch für Administratoren und Entwickler

HANSER

Der Autor:

Lutz Fröhlich, Darmstadt
lutz@lutzfrohelich.de

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Sylvia Hasselbach
Copy editing: Walter Saumweber, Ratingen
Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de
Umschlagrealisation: Stephan Rönigk
Gesamtherstellung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-45395-1
E-Book-ISBN: 978-3-446-45641-9

Inhalt

1 Einführung und Geschichte

Es sind fünf Jahre seit dem Erscheinen des Buches „PostgreSQL 9 – Praxisbuch für Administratoren und Entwickler“ vergangen. Aufgrund der positiven Kritiken und der großen Nachfrage haben sich Verlag und Autor entschlossen, für die Version 10 wiederum ein Buch zu veröffentlichen. Das Buch präsentiert sich im bekannten Stil des Autors. Es ist jedoch keine einfache Überarbeitung des Vorgängers, sondern wurde auf Grundlage der neuen Features der Version 10 neu strukturiert und wesentlich erweitert.

Die neuen Features seit der Version 9.1 und insbesondere die der Version 10 sind in Kapitel 6 herausgestellt. Kapitel 4 beschäftigt sich eingehend mit der Architektur, um das Verständnis für interne Prozesse und Abläufe zu fördern. Wesentlich ausgebaut wurde der Teil für Entwickler. Im Buch sind die populärsten Programmiersprachen mit zahlreichen Beispielen abgedeckt. Auch die Möglichkeiten von Erweiterungen sowie der Programmierung und Veröffentlichung eigener Module werden dargestellt.

Der Sprung zur Version 10 ist ein Major Release-Wechsel. Kapitel 3 gibt wertvolle Hinweise für das Upgrade. Wenn Sie von einem anderen Datenbanksystem kommen, dann unterstützen Sie die Kapitel 25 und 26 bei der Migration. Darüber hinaus finden Sie in diesem Buch praktische Erfahrungen zur Einbindung von PostgreSQL in eine bestehende IT-Landschaft.

Das Buch ist als Einstieg und Nachschlagewerk für IT-Profis geschrieben und setzt Basiskenntnisse von relationalen Datenbanken voraus. Auf eine Erläuterung von gängigen Begriffen wird deshalb bewusst verzichtet, auch um den Umfang des Buches überschaubar zu halten. Dennoch finden Sie viele Beispiele und Praxistipps, die auch Einsteigern die Möglichkeit bieten, sich in das Produkt einzuarbeiten.

PostgreSQL hat in den vergangenen Jahren an Verbreitung und Popularität erheblich zugenommen. Dazu hat in erheblichem Maß die permanente Erweiterung mit neuen Features und die Anpassung an die Belange der Anwender beigetragen. PostgreSQL ist der lebende Beweis, dass Open Source-Software nicht nur mit kommerziellen Produkten mithalten kann, sondern in vielen Bereichen sogar überlegen ist. Der kommerzielle Druck steht nicht im Vordergrund und lässt die Entwickler-Community frei arbeiten und Innovationen umsetzen.

Neben einem robusten Transaktionskern sowie einer hohen Zuverlässigkeit bietet PostgreSQL viele Features eines modernen Datenbank-Betriebssystems und kann problemlos in eine vorhandene IT-Infrastruktur integriert werden. Durch den hohen Kompatibilitätsgrad zu Oracle ist der Migrationsaufwand überschaubar und ein Mischbetrieb gut umzusetzen.

Die Version 10 beeindruckt durch neue Features wie „Native Table Partitioning“ und „Logische Replikation“ sowie Erweiterungen im Bereich „Parallel Query“. Trotz einer zunehmenden Anzahl von Features ist PostgreSQL eine schlanke und sehr gut zu verwaltende Datenbank geblieben. Sie konzentriert sich auf das Kerngeschäft, der Verwaltung von Datenbeständen.

PostgreSQL kann auf allen populären Plattformen wie Linux, MacOS, Solaris oder Windows eingesetzt werden. Obwohl es sich um ein Open Source-Produkt handelt, kann kommerzieller Support zu einem vernünftigen Preis hinzugekauft werden. Einem professionellen Einsatz steht damit nichts im Wege.

Freuen Sie sich auf einen PostgreSQL-Server 10 mit spannenden neuen Features!

1.1 Die Geschichte von PostgreSQL

PostgreSQL geht zurück auf das POSTGRES-Projekt, das an der University of California at Berkeley in den 80er-Jahren angesiedelt war. Die erste vorzeigbare Version erschien im Jahre 1987 als Postgres-Version 1. Als Reaktion auf die ersten Kritiken wurde das noch heute in PostgreSQL vorhandene Rule-System entwickelt. Version 3 erschien im Jahre 1991 mit einer Weiterentwicklung der Abfrageeinheit. 1993 beendete die University of California das Projekt mit der Version 4.2, um die rasant wachsenden Supportanforderungen nicht mehr tragen zu müssen.

Nach Hinzufügen eines SQL-Abfrageinterpreters im Jahre 1995 wurde die Software unter dem Begriff Postgres95 ins Web gestellt, mit dem Quellcode des originalen Berkeley-Postgres. Das Produkt war zu dieser Zeit komplett in ANSI C geschrieben. Durch Verbesserung in den Bereichen Wartbarkeit und Performance lief es schließlich bis zu 50 % schneller als das originale Berkeley-Postgres.

Die Entscheidung, die Jahreszahl aus dem Produktnamen zu entfernen, fiel im Jahre 1996. Damit wurde Postgres95 zu PostgreSQL, und es begann die ständige Weiterentwicklung von PostgreSQL als Open-Source-Produkt. Obwohl Letzteres über viele Jahre ein Schattendasein im Licht der großen kommerziellen Datenbanken, aber auch der durch den Internet-Boom schnell verbreiteten Open-Source-Datenbank MySQL führte, erfolgte seine konsequente Weiterentwicklung durch die Community.

Heute präsentiert sich PostgreSQL als ausgereift und stabil und erfüllt (fast) alle Anforderungen an ein modernes relationales Datenbanksystem. Für viele überraschend: Die Performance ist vergleichbar mit so manchem kommerziellen Produkt.

1.2 Verwendete Version

Das Buch bezieht sich auf die während der Manuskripterstellung vorliegende aktuelle Version 10.3. Schauen Sie regelmäßig nach weiteren Veröffentlichungen, insbesondere für neuere Features und Versionen, auf der Webseite des Verlags und der Autoren-Webseite vorbei. Alles rund um die PostgreSQL-Community finden Sie auf der Webseite *http://www.postgresql.org*.

1.3 Konventionen

Begriffe in spitzen Klammern bezeichnen eine zu ersetzende Variable (so ist zum Beispiel der Ausdruck <VERSION> in der Regel durch die aktuelle Version 10.3 zu ersetzen). Die meisten Darstellungen beziehen sich gleichermaßen auf UNIX- und Windows-Betriebssysteme. Die Darstellung der Umgebungsvariablen erfolgt im Wesentlichen im UNIX-Format, das heißt z. B. *$BIN* statt *%BIN%* für Windows. Sie können das Format einfach nach Windows übertragen. Das Gleiche gilt für das Trennzeichen der Pfade: „/“ unter Unix sowie „\“ unter Windows.

1.4 Software und Skripte

Sie können die aktuelle Version von PostgreSQL aus dem Internet herunterladen und installieren. Es wird die Installation aus dem Quellcode empfohlen, um alle Beispiele nachvollziehen zu können. Ideal ist, wenn Sie auf einem Linux- oder Windows-Betriebssystem arbeiten.

Alle nummerierten Listings im Buch können als Datei von der Webseite des Verlags sowie von der Autoren-Webseite heruntergeladen werden:

http://www.hanser-fachbuch.de

http://www.lutzfroehlich.de

Darmstadt und Oasis del Sol, im April 2018

Lutz Fröhlich

lutz@lutzfroehlich.de

2 Installation aus Paketen und Quellcode

Die Installation kann sowohl von vorgefertigten Paketen als auch aus dem Quellcode erfolgen. Wenn Sie eine schnelle Installation bevorzugen, ist die einfachere Paketinstallation zu empfehlen. Allerdings müssen Sie dann mit dem Setup leben, die das Paket zur Verfügung stellt. Mit der Installation aus dem Quellcode sind Anpassungen und Erweiterungen möglich. Für den produktiven Einsatz ist zu beachten, dass Supportleistungen von Anbietern sich ausschließlich auf die entsprechenden Pakete beziehen. Diese Pakete sind bis zu einem gewissen Grad getestet. Beachten Sie, dass damit eine Abhängigkeit vom Release-Zyklus des Drittanbieters besteht. Ein wichtiger Vorteil der Paketinstallation ist, dass einheitliche Versionsstände ausgerollt werden können.

In diesem Kapitel werden beide Varianten vorgestellt. Um alle in diesem Buch vorgestellten Features und Optionen nachvollziehen zu können, wird eine Installation aus dem Quellcode empfohlen.

2.1 Paketinstallation

Alle erforderlichen Informationen für die Installation befinden sich auf der Seite *http://postgresql.org/download* unter dem Abschnitt *Binary Packages*. Wir führen zuerst eine Paketinstallation für Linux und danach eine für Windows durch.

2.1.1 Paketinstallation unter Linux

Pakete für Linux stehen für die gebräuchlichsten Derivate zur Verfügung. Wir führen eine Installation unter Oracle Linux 7 durch, das weitgehend kompatibel mit Red Hat Linux ist. Wenn Sie bei der Installation des Betriebssystems ein yum-Repository erzeugt haben, dann können Sie einfach eine yum-Installation durchführen. Alternativ können Sie die RPM-Pakete herunterladen und manuell installieren.

Im folgenden Beispiel wird die Installation unter Oracle Linux Version 7 durchgeführt. Oracle Linux ist ein Ableger von Red Hat. Bereits bei der Installation des Betriebssystems

kann das yum-Repository angelegt werden. Die folgenden Schritte beschreiben die Installation der Version 10:

1. Installieren Sie das RPM im Repository.

```
# yum install https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/
pgdg-oraclelinux96-9.6-3.noarch.rpm
```

2. Zuerst wird das Paket für den Client installiert.

```
# yum install postgresql10
```

3. Danach erfolgt die Installation des Server-Pakets.

```
# yum install postgresql10-server
```

4. Zum Schluss können Sie den automatischen Start konfigurieren.

```
# /usr/pgsql-10.3/bin/postgresql10-setup initdb
Initializing database ... OK
# systemctl enable postgresql-10
# systemctl start postgresql-10
```

Falls keine Verbindung vom Datenbankserver zum Internet existiert, muss das Paket manuell heruntergeladen und übertragen werden. Gehen Sie dazu auf die Seite *https://yum.postgresql.org/rpmchart.php* und wählen Sie das passende Paket aus. Für die aktuelle Version lautet das Paket:

postgresql10-server-10.3-PGDG.rhel7.x86_64.rpm.

Die Installation kann wie gewohnt mit dem rpm-Utility erfolgen.

2.1.2 Paketinstallation unter Windows

Für die Paketinstallation unter Windows erfolgt eine Weiterleitung von der Downloadseite *postgresql.org/download* auf die Webseite von EnterpriseDB. Dort können Sie Version und Betriebssystem auswählen und im Fall von Windows den Windows-Installer herunterladen. EnterpriseDB ist ein Anbieter von vorgefertigten Paketen und Supportleistungen für diese Distributionen.

1. Starten Sie den Windows Installer.
2. Zuerst wird das Visual C++ 2013 Redistributable-Paket installiert. Das ist erforderlich, da während der PostgreSQL-Installation die Bibliotheken neu verbunden werden müssen.
3. Es erscheint das graphische Installationsprogramm.

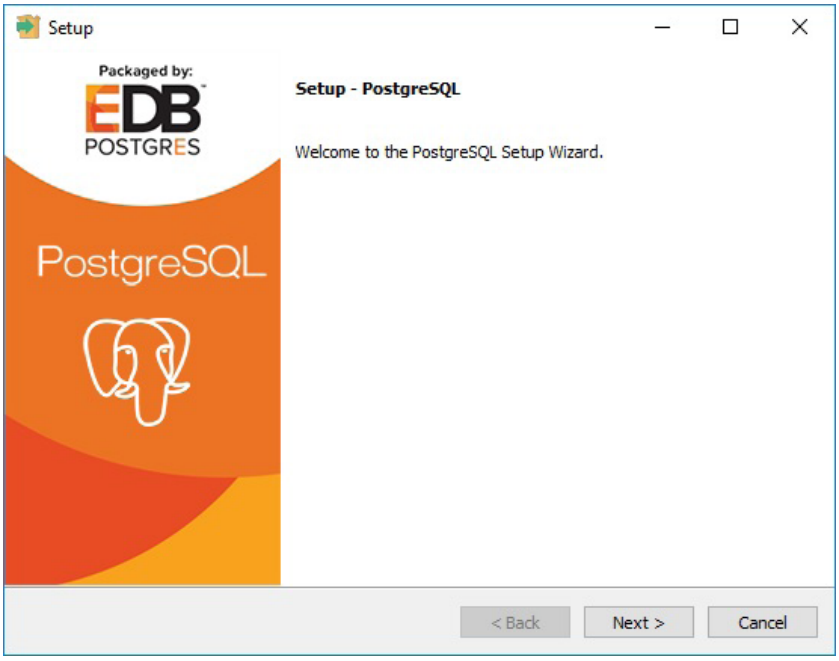

Bild 2.1 Installation des Pakets unter Windows mit dem Setup-Programm

4. Wählen Sie die Verzeichnisse für die Installation der Software und für die Datenbanken.
5. Danach werden das Passwort für den Superuser (postgres), die Portnummer des Listeners und die Locale abgefragt. Der Installer legt einen Windows-Dienst zur Verwaltung des PostgreSQL-Servers an.

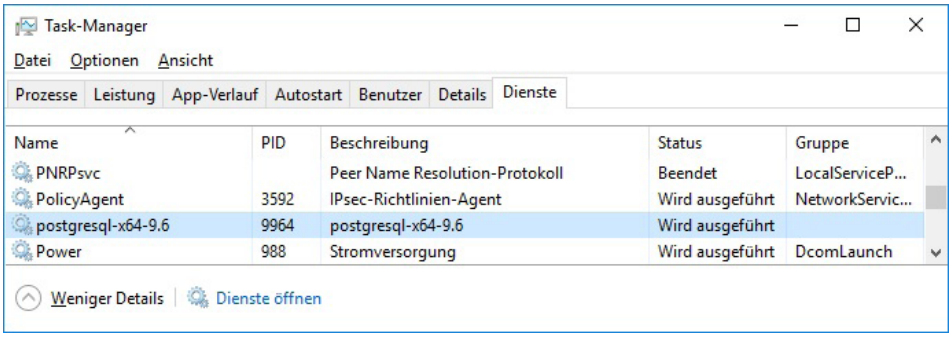

Bild 2.2 Der PostgreSQL-Dienst in Windows

Die Paketinstallation ist damit abgeschlossen und PostgreSQL kann genutzt werden.

2.2 Installation aus dem Quellcode

Das Vorgehen bei der Installation aus dem Quellcode ist für alle Betriebssysteme ähnlich. Die Quellprogramme werden kompiliert, gelinkt und anschließend im Installationsverzeichnis bereitgestellt. In diesem Abschnitt werden die Schritte für Linux und Windows vorgestellt. Eine Installation aus dem Quellcode kann besser individualisiert und erweitert werden. Sie können sogar eigene Programmkomponenten einbinden.

2.2.1 Installation aus dem Quellcode unter Linux

Wir installieren an dieser Stelle zunächst den Standardumfang und werden im weiteren Verlauf des Buches darauf hinweisen, wenn zusätzliche Optionen eingebunden werden.

Das folgende Vorgehen beschreibt die Installation unter Oracle Enterprise Linux Version 7:

1. Laden Sie den Quellcode von der Webseite *http://www.postgresql.org/download* herunter. Die Datei hat das Format *postgresql-<version>.tar.gz*.
2. Entpacken Sie den Quellcode und wechseln Sie in das Verzeichnis *postgresql-<version>*.

```
# tar -xvzf postgresql-10.3.tar.gz
```

3. Im ersten Schritt der Installation werden die Quellen für das Betriebssystem konfiguriert. Das Skript *configure* führt einige Tests aus, um die Werte einiger systemabhängiger Variablen zu bestimmen.

```
# ./configure
```

4. Starten Sie anschließend den Build-Prozess. Es werden die Libraries und Binaries erstellt. Verwenden Sie das GNU Make Utility und achten Sie auf die Erfolgsmeldung am Ende des Durchlaufs.

```
# make
. . .
All of PostgreSQL successfully made. Ready to install.
```

5. Im nächsten Schritt erfolgt die Installation der Binaries in die Verzeichnisse. Dieser Schritt muss ebenfalls unter dem Benutzer *root* ausgeführt werden, um die erforderlichen Berechtigungen zu erstellen. Standardmäßig erfolgt die Installation in das Verzeichnis */usr/local/pgsql*.

```
# make install
. . .
PostgreSQL installation complete.
```

6. Damit ist die Installation abgeschlossen. In der Nachbereitung sind folgende Schritte erforderlich:
   - Einen Benutzer *postgres* als Eigentümer der Software und des Servers erstellen:

```
# adduser postgres
```

- Ein Verzeichnis zum Speichern der Datenbankdateien erstellen:

```
# mkdir /usr/local/pgsql/data
# chown postgres /usr/local/pgsql/data
```

- Den Datenbankserver erstellen:

```
# su - postgres
$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
. . .
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
Success. You can now start the database server using:
    /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start
```

- Den Datenbankserver starten. Verifizieren Sie, dass die Prozesse laufen:

```
$ /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start
waiting for server to start.... done
server started
$ ps -ef|grep postg
postgres 20353     1  0 18:10 pts/1   00:00:00 /usr/local/pgsql/bin/postgres -D /usr/
local/pgsql/data
postgres 20355 20353  0 18:10 ?       00:00:00 postgres: checkpointer process
postgres 20356 20353  0 18:10 ?       00:00:00 postgres: writer process
postgres 20357 20353  0 18:10 ?       00:00:00 postgres: wal writer process
postgres 20358 20353  0 18:10 ?       00:00:00 postgres: autovacuum launcher process
postgres 20359 20353  0 18:10 ?       00:00:00 postgres: stats collector process
postgres 20360 20353  0 18:10 ?       00:00:00 postgres: bgworker: logical
replication launcher
```

- Eine Testdatenbank mit dem Namen *hanser* anlegen:

```
[postgres@localhost ~]$ /usr/local/pgsql/bin/createdb hanser
[postgres@localhost ~]$ /usr/local/pgsql/bin/psql hanser
psql (10.3)
Type "help" for help.
hanser=# select version();
                                                 version
-----------------------------------------------------------------------------
 PostgreSQL 10.3 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.4.7 20120313 (Red
Hat 4.4.7-17), 64-bit
(1 row)
```

Damit ist die Installation aus dem Quellcode abgeschlossen und der PostgreSQL-Server ist einsatzbereit.

2.2.2 Installation aus dem Quellcode unter Windows

Die Installation aus dem Quellcode auf einem Windows-Betriebssystem erscheint auf den ersten Blick etwas aufwendiger, was jedoch eher am Handling der Compiler sowie des Windows-Plattform-Supports liegt.

PostgreSQL kann mithilfe des Visual C++-Compilers von Microsoft übersetzt und verbunden werden.

HINWEIS: Verwenden Sie das Microsoft Windows SDK für das Erstellen der Software aus dem Quellcode. Alternativ kann das Visual Studio installiert werden, das ebenfalls alle erforderlichen Programme für das Kompilieren und Linken der Quellen enthält.

In den folgenden Schritten erfolgt die Installation unter Windows 10 auf einer 64-Bit-Plattform.

1. Installieren Sie Microsoft Windows SDK oder Visual Studio.
2. Für die Installation ist ein Perl-Interpreter erforderlich. Installieren Sie im Bedarfsfall ActiveState Perl. Das Paket sowie die Installationsanleitung finden Sie auf der Webseite *http://www.activestate.com*.
3. Entpacken Sie den heruntergeladenen PostgreSQL-Quellcode. Auch wenn die Datei die Endung *.tar.gz* besitzt, kann sie mit den meisten Kompressionswerkzeugen direkt unter Windows ausgepackt werden. Im vorliegenden Beispiel werden die Quellen in das Laufwerk *D:\PostgreSQL* gelegt. Beim Entpacken wird ein Unterverzeichnis mit der verwendeten Version angelegt.
4. Öffnen Sie eine Windows-Eingabeaufforderung (DOS-Fenster) mit Administratorrechten. Wechseln Sie in das Verzeichnis *D:\PostgreSQL\postgresql-<version>*.
5. Für eine Installation von 64-Bit-Programmen muss der Visual C++-Compiler auf 64 Bit gestellt werden. Das erfolgt durch Aufrufen der Batchdatei *vcvarsall.bat*.

```
D:\PostgreSQL\postgresql-10.3>"C:\Program Files (x86)\Microsoft Visual Studio 14.0\
VC\vcvarsall.bat" amd64
```

6. Wechseln sie in das Unterverzeichnis *..\src\tools\msvc* und starten Sie Kompilierung und Verbinden des Quellcodes. Achten Sie *auf die Erfolgsmeldung am Ende*.

```
D:\PostgreSQL\postgresql-10.3>cd src\tools\msvc
D:\PostgreSQL\postgresql-10.3\src\tools\msvc>build
. . .
Der Buildvorgang wurde erfolgreich ausgeführt.
0 Fehler
Verstrichene Zeit 00:04:37.13
```

7. Im letzten Schritt erfolgt die Installation. Geben Sie das gewünschte Verzeichnis an, unter dem die Software installiert werden soll.

```
D:\PostgreSQL\postgresql-10.3\src\tools\msvc>install D:\PostgreSQL
Installing version 10 for release in D:\PostgreSQL
. . .
Installation complete.
```

8. Analog zu den Installationsschritten unter Linux kann der Datenbankserver jetzt initialisiert werden.

```
D:\PostgreSQL\bin>initdb -D D:\PostgreSQL\data
. . .
Success. You can now start the database server using:
    pg_ctl -D ^"D^:^\PostgreSQL^\data^" -l logfile start
```

9. Zum Schluss wird der Server gestartet und eine Datenbank mit dem Namen *hanser* angelegt.

```
D:\PostgreSQL\bin>pg_ctl -D D:\PostgreSQL\data -l logfile start
waiting for server to start.... done
server started
D:\PostgreSQL\bin>createdb hanser
D:\PostgreSQL\bin>psql hanser
psql (10.3)
Type "help" for help.
hanser=# select version();
                           version
-------------------------------------------------------------
 PostgreSQL 10.3, compiled by Visual C++ build 1900, 64-bit
(1 row)
```

Damit ist die Installation abgeschlossen und der PostgreSQL-Server kann verwendet werden.

HINWEIS: Die Syntax sowie die meisten Beispiele im Buch beziehen sich auf ein Linux-System. Wenn Sie vorzugsweise unter Windows arbeiten, dann können Sie dennoch alles nachvollziehen, wenn Sie die betriebssystemspezifischen Abweichungen beachten. Das Verhalten von PostgreSQL ist plattformübergreifend analog.

Alternativ kann ein Windows-Dienst eingerichtet werden, der das Starten und Stoppen bei Start und Shutdown des Betriebssystems übernimmt.

```
D:\PostgreSQL\bin>pg_ctl register -D D:\PostgreSQL\data -N Postgres10
```

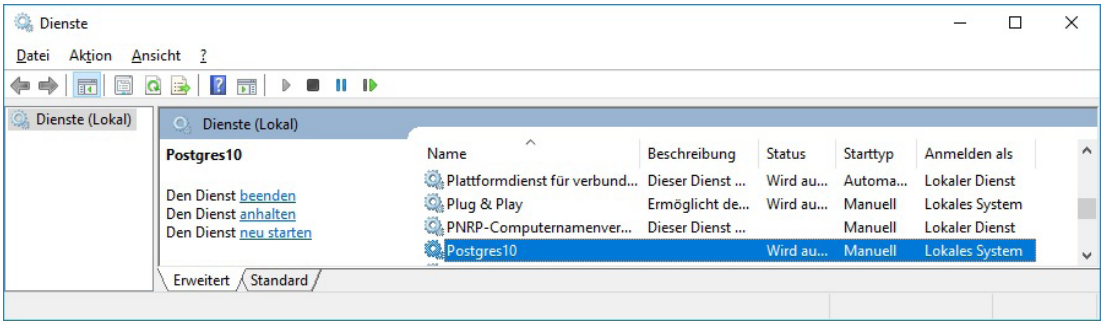

Bild 2.3 Einen Windows-Dienst für den PostgreSQL-Server anlegen

2.3 Erste Schritte

Für ein bequemes Handling sollten mindestens die Umgebungsvariablen *PATH* und *PGDATA* gesetzt werden.

Listing 2.1 Den Ausführungspfad sowie die Variable PGDATA setzen

```
$ export PATH=/usr/local/pgsql/bin:$PATH
$ export PGDATA=/usr/local/pgsql/data
```

Das Starten und Stoppen des Servers erfolgt mit dem Utility *pg_ctl*.

Listing 2.2 Den PostgreSQL-Server starten und stoppen

```
$ pg_ctl start -l logfile
waiting for server to start.... done
server started
$ pg_ctl stop
waiting for server to shut down.... done
server stopped
```

Sobald der Server gestartet ist, können Verbindungen zur Datenbank hergestellt werden. Das Kommandozeilen-Utility *psql* ist Bestandteil der Installation und kann direkt auf dem Server aufgerufen werden.

Listing 2.3 Eine Verbindung zur Datenbank mit psql herstellen

```
$ psql hanser
psql (10.3)
Type "help" for help.
hanser=# SELECT current_database();
 current_database
------------------
 hanser
(1 row)
hanser=# SELECT version();
                                                  version
-------------------------------------------------------------------------------------
 PostgreSQL 10.3 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.4.7 20120313 (Red
Hat 4.4.7-17), 64-bit
(1 row)
hanser=# \q
```

Aus Sicherheitsgründen ist der Zugriff von einem entfernten Server oder Client standardmäßig abgeschaltet. Der erste Versuch, von einem Client mit der SQL-Shell auf den PostgreSQL-Server zuzugreifen, wird deshalb mit der folgenden Fehlermeldung scheitern:

```
D:\PostgreSQL\bin>psql -h 192.168.56.101 -p 5432 -U postgres -W
Password for user postgres:
psql: FATAL:  no pg_hba.conf entry for host "192.168.56.1", user "postgres", database
"postgres"
```

Um einen Zugriff von entfernten Computern zu gestatten, müssen noch Anpassungen an den Konfigurationsdateien vorgenommen werden:

1. Stoppen Sie den PostgreSQL-Server.
2. Wechseln Sie in das Datenverzeichnis (*$PGDATA*) und editieren Sie die zentrale Konfigurationsdatei *postgresql.conf*. Ändern Sie den Eintrag für den Parameter *listen_addresses* von `"localhost"` auf den Servernamen oder dessen IP-Adressen, unter der dieser über das Netzwerk-Interface erreichbar ist. Mit dem Eintrag `"localhost"` hört der Server nur auf ankommende Anfragen, die vom Server kommen. Eine Verbindung von außen ist damit nicht möglich.

```
listen_addresses = '192.168.56.101'     # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for all
                                        # (change requires restart)
port = 5432                             # (change requires restart)
```

3. PostgreSQL besitzt eine Konfigurationsdatei mit dem Namen *pg_hba.conf* für die Authentifizierung von Clients. Sie befindet sich im selben Verzeichnis. Fügen Sie den folgenden Eintrag mit der IP-Adresse des Clients hinzu, von dem aus Sie zugreifen möchten. Weitere Informationen zu diesem Thema finden Sie im Kapitel „Sicherheit und Überwachung“.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             192.168.56.1/32         trust
```

4. Starten Sie den PostgreSQL-Server.

Jetzt können Sie sich von dem frei geschalteten Computer aus verbinden.

Umgebungsvariablen

Wie Sie sicherlich bemerkt haben, müssen dem psql-Utility für die Verbindung zu einem entfernten Server die Verbindungsinformationen mitgegeben werden. Alternativ können diese in den folgenden Umgebungsvariablen hinterlegt werden:

PGHOST: Name oder IP-Adresse des Servers

PGPORT: TCP/IP-Port des PostgreSQL-Servers (Standard 5432)

PGDATABASE: Name der Datenbank

PGUSER: Benutzername für die Verbindung

Für die Verwaltung von Server und Datenbanken gibt es eine Anwendung mit grafischer Oberfläche mit dem Namen *pgAdmin*. Es ist ebenfalls ein Open Source-Programm und kann von der Webseite *https://www.pgadmin.org/download* heruntergeladen werden. Erstellen Sie eine neue Serververbindung durch Eingabe der üblichen Parameter. Da wir den Client bereits freigeschaltet haben, wird die Verbindung direkt funktionieren.

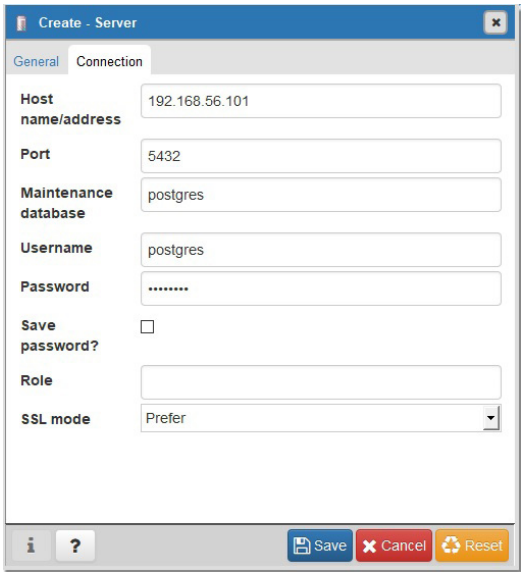

Bild 2.4 Einen Server im pgAdmin-Tool registrieren

Das Tool vereinfacht die Navigation durch Server und Datenbanken und verfügt über zusätzliche Funktionen im Vergleich zur Kommandozeile.

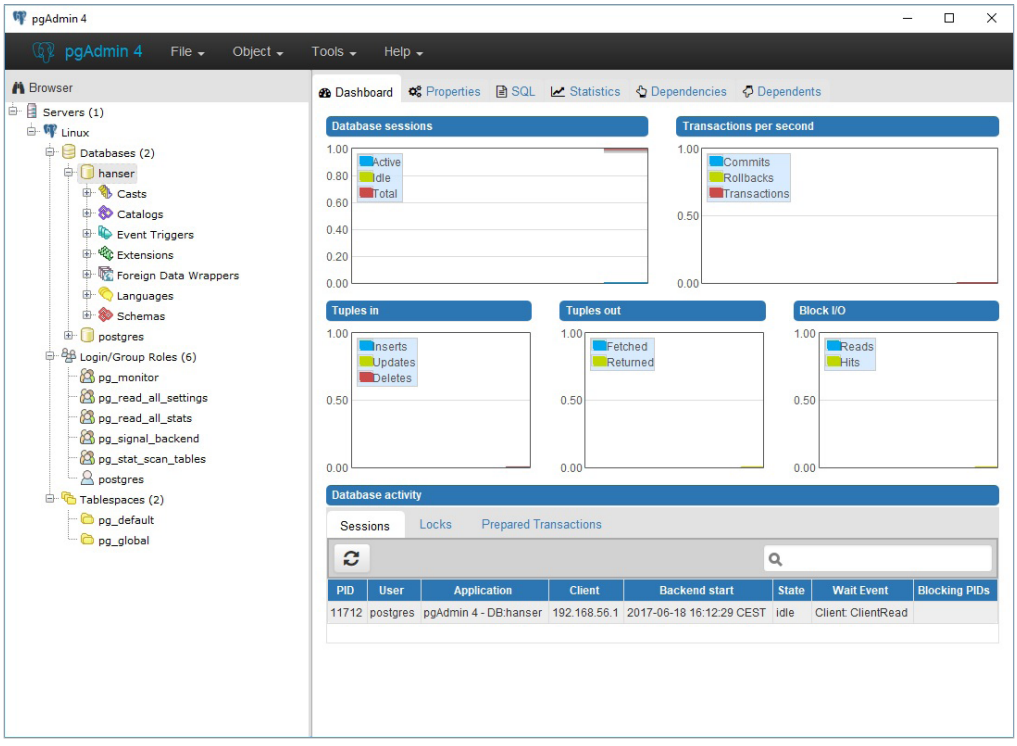

Bild 2.5 Das pgAdmin-Tool

Wie Sie vielleicht bemerkt haben, sendet der Server seine Nachrichten nach stdout, wenn er ohne die Logfile-Option gestartet wird. In der Standardkonfiguration ist das Schreiben von Nachrichten in Logdateien ausgeschaltet. Mit der folgenden Einstellung in der Konfigurationsdatei *postgresql.conf* wird ein minimales Schreiben von Fehlermeldungen und Servernachrichten eingeschaltet:

```
log_destination = 'stderr'              # Valid values are combinations of
                                        # stderr, csvlog, syslog, and eventlog,
                                        # depending on platform.  Csvlog
                                        # requires logging_collector to be on.
# This is used when logging to stderr:
logging_collector = on                  # Enable capturing of stderr and csvlog
                                        # into log files. Required to be on for
                                        # csvlogs.
                                        # (change requires restart)
# These are only used if logging_collector is on:
log_directory = 'pg_log'                # directory where log files are written,
```

Mit dem Neustart erscheint die Meldung:

```
2017-06-18 20:48:49.855 CEST [2896] HINT:  Future log output will appear in directory
"pg_log".
```

Unter *$PGDATA* wurde das Verzeichnis *pg_log* angelegt. Darin befindet sich die Logdatei des PostgreSQL-Servers:

```
$ cat postgresql-2017-06-18_204849.log
2017-06-18 20:48:49.910 CEST [2898] LOG:  database system was shut down at 2017-06-18
16:45:54 CEST
2017-06-18 20:48:49.977 CEST [2896] LOG:  database system is ready to accept
connections
```

Wenn Sie den Server mit „pg_ctl stop“ heruntergefahren haben, dann konnten Sie feststellen, dass der Server gestoppt wurde, obwohl noch Clients verbunden waren. Hier hat es eine Veränderung des Standards gegeben. Mit der Version 9.5 wurde der Standard der Shutdown-Option von „smart“ auf „fast“ geändert. Auch in der Version 10 ist der Standard „fast“, d. h. der Befehl ist identisch mit *pg_ctl stop -m fast*. Tabelle 2.1 beschreibt die Optionen.

Tabelle 2.1 Optionen zum Stoppen des PostgreS“QL-Servers

Option	Aktion
smart	Stoppen, wenn sich alle Clients abgemeldet haben
fast	Verbindungen trennen, Server sauber herunterfahren
immediate	Prozesse sofort beenden, ohne den Server sauber herunterzufahren Erfordert ein Recovery beim nächsten Start

Wenn Sie mit den Stopp-Optionen von Oracle vertraut sind, werden Sie feststellen, dass bei PostgreSQL gleiche Begriffe auftauchen, die allerdings anders interpretiert werden. Tabelle 2.2 vergleicht die Shutdown-Optionen zwischen Oracle und PostgreSQL.

Tabelle 2.2 Vergleich der Shutdown-Optionen zwischen Oracle und PostgreSQL

PostgreSQL	Oracle
smart	normal
nicht vorhanden	transactional
fast	immediate
immediate	abort

3 Upgrade auf Version 10

Mit der Version 10 wurde die Nummerierung der Versionen geändert. Ein Major Release-Wechsel bedeutet, dass sich die erste Nummer ändert, also zum Beispiel von 10 auf 11. Vorher wurde ein Major Release durch die zweite Stelle gekennzeichnet. Demnach ist ein Upgrade von Version 9.5 auf 9.6 ein Major Release-Wechsel.

Ein Minor Release-Wechsel liegt vor, wenn sich maximal der zweite Teil der Versionsnummer ändert, zum Beispiel bei einem Upgrade von Version 10.0 auf 10.1. Grundsätzlich werden in diesem Fall nur Bug Fixes implementiert.

TIPP: Obwohl Upgrades immer ein Risiko mit sich bringen, sollten Sie regelmäßig auf das letzte Minor Release gehen. Zusätzlich zu allgemeinen Bug Fixes werden auch Sicherheitslücken geschlossen. Das Risiko, das mit dem Überspringen von Upgrades verbunden ist, wird als höher eingeschätzt.

Ein Upgrade von Version 9 auf Version 10 ist ein Major Release-Wechsel. Mit einem neuen Major Release ändert sich unter anderem das interne Format von Systemtabellen und Datenfiles. Diese Änderungen sind in der Regel sehr komplex, sodass eine Rückwärtskompatibilität nicht gegeben ist.

Es gibt zwei grundlegende empfohlene Methoden für ein Major Release-Upgrade:

- Entladen der Datenbanken mit *pg_dumpall* und Laden in die neue Version.
- Upgrade mit *pg_upgrade*.

Wir führen ein Upgrade von der Version 9.6 auf die Version 10.3, also einen Major Release-Wechsel durch.

3.1 Upgrade mit pg_dumpall

Mit der Hilfe von *pg_dumpall* kann ein logisches Backup vom PostgreSQL-Cluster mit der niedrigeren Major Release-Nummer erstellt und in das Cluster mit der höheren Release-Nummer geladen werden. Es wird empfohlen, für das Entladen das *pg_dumpall*-Utility der

höheren Version zu verwenden. Diese Vorgehensweise funktioniert rückwärts betrachtet bis zur Version 7.0.

Da es sich um ein Out-of-place-Upgrade handelt, müssen beide Versionen parallel installiert sein. Dies kann sowohl auf derselben als auch auf getrennter Hardware der Fall sein. Befinden sich beide Cluster auf derselben Hardware, dann müssen Sie zwischen beiden Umgebungen hin und herschalten.

Es wird empfohlen, das Utility *pg_dumpall* der höheren Version zu verwenden. Es kann allerdings nicht einfach in die Binary-Installation der alten Version kopiert werden. Das heißt, auch wenn Sie auf eine neue Hardware migrieren, benötigen Sie eine zweite PostgreSQL-Installation mit der Version 10. Die folgenden Schritte beschreiben, wie eine zweite Version installiert werden kann:

1. Wir nehmen an, es existiert bereits eine Installation mit der Version 9.6. Die Distribution befindet sich im Standardverzeichnis */usr/local/pgsql*.
2. Führen Sie die Installation aus dem Quellcode, so wie in Kapitel 2 beschrieben, durch. Im Standardverzeichnis befindet sich bereits die Installation der Version 9.6. Verwenden Sie deshalb für den `configure`-Befehl eine zusätzliche Option, um die Distribution in ein anderes Verzeichnis zu lenken:

```
./configure -prefix=/usr/local/pgsql10
```

3. Damit befinden sich zwei Binary-Installationen auf dem Server und Sie können hin- und herschalten.

Die folgenden Schritte zeigen eine logische Migration von der Version 9.6 auf die Version 10.3:

1. Stellen Sie sicher, dass während des Exports keine Transaktionen auf der Datenbank stattfinden. Eine einfache Methode ist die Anpassung der Datei *pg_hba.conf* und ein Neustart des Clusters.
2. Setzen Sie die Umgebung für die Daten auf die alte Version und für die Binaries auf die neue Version.

```
$ export PGDATA=/usr/local/pgsql/data
$ export PATH=/usr/local/pgsql10/bin:$PATH
$ which pg_dumpall
/usr/local/pgsql10/bin/pg_dumpall
```

3. Führen Sie das Backup des Clusters durch.

```
$ pg_dumpall > backup_v96.sql
```

4. Das Backup ist eine Textdatei bestehend aus SQL-Befehlen zum Wiederherstellen der Komponenten des Clusters. Aus diesem Grund wird diese Methode auch logische Migration genannt. Da keine Abhängigkeiten zur Binary-Installation bestehen, kann diese Methode auch zur Migration in ein anderes Betriebssystem, zum Beispiel Windows, eingesetzt werden.
5. Stoppen Sie das alte PostgreSQL-Cluster.