PostgreSQL

Inhaltsverzeichnis

1 Postgresql

1.1 Installation, locales und postgres

Das Paket **postgresql-client-NR** (NR=Version, z.B. 12) ist in Linux meist enthalten. Das Paket **postgresql-NR** ist das Server-Paket und muss über den Paketmanager Synaptic installiert werden. Vor der Installation muss das System mit locale - a kontrolliert werden.

```
user@devuan:~$ locale -a
C
C.UTF-8
de_DE.utf8
POSIX
```

Wenn nur C erscheint, mit **dpkg-reconfigure locales** im Terminal die richtigen Einstellungen setzen. Danach im Terminal erneut kontrollieren und erst dann **postgresql-NR** installieren. Ansonsten gibt es Probleme beim Datum und den Double-Zahlenwerten. Der Datenbankserver startet nach Neustart automatisch.

1.2 Erstanmeldung über den root-Account

```
gottfried@CLARA:~$ su
Passwort:
root@CLARA:~# su postgres
postgres@CLARA:~$
postgres@CLARA:~$ psql postgres
Welcome to psql 8.3.7, the PostgreSQL interactive terminal.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit
postgres=#
```

1.3 Anlegen eines Nutzers mit Passwort und Superuser-Rechten

```
CREATE USER name PASSWORD 'passwort' CREATEDB;
```

(CREATEDB ist eigentlich das Recht Datenbanken zu erzeugen, wenn Daten per **COPY** aus Dateien eingefügt werden sollten, musste der User auch dieses Recht besitzen)

1.4 Erzeugen einer Datenbank für diesen User

```
CREATE DATABASE dbname WITH OWNER name ENCODING 'UTF8'
CREATE DATABASE handwerk WITH OWNER gottfried ENCODING 'UTF8' ;
```

2 Zugriff auf den Datenbankserver

Auf dem Server muss in der Datei **pg_hba.conf** folgender Abschnitt angepasst werden:

```
# IPv4 local connections:
host    all         all       127.0.0.1/32          md5
host    handwerk    all      192.168.178.0/24      md5
```

(! Die Zeilen dürfen offensichtlich nur Leerzeichen, keine Tabs enthalten)

Die Datei **postgresql.conf** muss ab dieser Zeile folgendermaßen aussehen:

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

Neben der Änderung '*' unbedingt auch das Kommentarzeichen # vor listen_adresses entfernen! Die Portnummer war nach Installation 5433 und musste geändert werden.

3 Datensicherung (auch zum Datenbank übertragen)

3.1 Sicherung

Datenbankinhalt als User = OWNER mit pg_dump in eine Textdatei sichern, und zwar per Terminal als user, (nicht mit psql reingehen)

```
# Backup full database
pg_dump -h host -U user -W database > dbname.sql    (erfordert Passwort)
pg_dump database > dbname.sql                        (ohne Passwort)
```

Aus der erstellten Backupdatei kann die Datenbank wieder hergestellt werden, in eine leere Datenbank, gleicher Name, gleicher OWNER, gleiche Kodierung.

3.2 Sicherungsdatei übertragen

Vorhandene Datenbank entfernen

```
DROP DATABASE name;
```

Dann die Datenbank neu erstellen

```
CREATE DATABASE dbname WITH OWNER name ENCODING 'UTF8' ;
```

Mit Einfügen der Sicherungsdatei den Inhalt komplett wiederherstellen

```
psql dbname < /pfad/zu/dbname.sql
```

Folgender Befehl soll gleiches bewirken, noch nicht getestet:

```
psql -d database -f dbname.sql
```

Achtung: Der folgende Befehl sichert nur die Struktur der DB, ohne Daten:

```
pg_dump -s -f dbname.sql dbname
```

Anderer PC, anderer USER: Die Sicherungsdatei enthält zu jedem einzelnen Element den OWNER. Diesen 'namen' kann man mittels Suchen/Ersetzen durch 'neuname' - den Namen des Datenbank-Eigentümers auf dem neuen System ersetzen, um die Sicherung dann einzufügen. (Kopie der unveränderten Sicherungsdatei empfohlen)

4 SQL - Befehle (Beispiele)

4.1 psql

```
\l                (nach psql postgres)
```

listet die vorhandenen Datenbanken auf

```
\d basiszins      (nach psql dbname)
```

Zeigt Tabellenspalten von Tabelle basiszins an

```
\d                (nach psql dbname)
```

Listet Tabellen mit Eigentümer auf

```
\z                (nach psql dbname)
```

Listet Tabellen mit Zugriffsrechten auf

```
\set                    (2)   \set HISTSIZE '3000'
```

zeigt alle Konstanten an (2) setzt eine Konstante neu

4.2 Datenbank admin vom Client aus mit psql

folgenden Befehl im Terminal eingeben, daraufhin wird das Passwort abgefragt und dann wird verbunden:

```
psql --host=simone --port=5432 --username=simone --password --dbname=handwerk
```

4.3 Tabellen mit Beziehungen

1. Tabelle erstellen

```
CREATE TABLE kunden (
knr SERIAL PRIMARY KEY,
anr varchar(30), vname varchar(40), nname varchar(60),
...
);
```

2. Tabelle erstellen

```
CREATE TABLE vorgang (
vid SERIAL PRIMARY KEY,  knr integer, vdat date,
...
knr INTEGER NOT NULL,
FOREIGN KEY (knr) REFERENCES kunden (knr)
);
```

PostgreSQL erstellt für den Datentyp SERIAL (Autowert) automatisch eine Sequenz, in den mit pg_dump erstellten Sicherungsdateien sieht das für die Tabelle kunden z.B. so aus:

```
CREATE SEQUENCE public.kunden_knr_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

4.4 Tabellen ändern ALTER TABLE

ALTER TABLE vorlage ADD COLUMN titel varchar(100);

Fügt der Tabelle vorlage die Spalte titel für Text, Länge 100 hinzu

ALTER TABLE vorlage DROP COLUMN titel;

Löscht die Spalte titel aus der Tabelle vorlage

ALTER TABLE vorgang RENAME COLUMN alt TO neu;

ändern einer Spaltenbezeichnung (verändert nicht den Inhalt)

ALTER TABLE vorlage RENAME TO vorlagen;

Umbenennen einer Tabelle, Inhalt bleibt erhalten

ALTER TABLE vorlage ALTER COLUMN bez DROP NOT NULL;

Entfernt die Eigenschaft Eingabe erforderlich=ja

ALTER TABLE artikel DROP UNIQUE(sortnr);

Hinzufügen der Eigenschaft Keine Duplikate zu einer Spalte

ALTER TABLE leistungen ALTER COLUMN anz TYPE NUMERIC(6,2);
ALTER TABLE vorgang ALTER COLUMN skont TYPE INT4;

Typ einer Spalte neu festlegen

4.5 Weitere Befehle

DROP TABLE vorgang;

Löscht die Tabelle vorgang

DELETE FROM vorgang;

Löscht den gesamten Inhalt der Tabelle vorgang

DELETE FROM vorgang WHERE datum < '01.01.2005';

Löscht Datensätze mit Datum vor dem 01.01.2005 aus der Tabelle vorgang

CREATE TABLE tablename(feld(attr), feld2,

siehe Basic: Anlegen von Tabellen

INSERT INTO tablename VALUES (.....)

siehe Basic: fügt Inhalte in der angegebenen Feldreihenfolge ein

COPY texte FROM '/home/user/database/hwdb/artikel.ods';

Kopiert den Inhalt einer Datei in die angegebene Tabelle texte

COPY texte (tnr, tkenn, tbez, txta, txtm, txts, res) TO '/tmp/hw_texte.csv' WITH DELIMITER AS '\t';

Kopiert die aufgeführten Spalten der Tabelle texte mit Trennzeichen in eine Datei (Pfad) und gibt als Trennzeichen Tabulator \t an

```
UPDATE vorlagen SET text = 'Forderungen' WHERE text = 'gefordert';
```

Ersetzt alle gefundenen Worte 'gefordert' in der Spalte text der Tabelle vorlagen durch das Wort 'Forderungen'

```
UPDATE import SET such = nname|| vname WHERE such = '';
```

kopiert Nachname und Vorname zusammengesetzt in die Spalte **such**, wenn diese leer ist

```
UPDATE kunden SET bem = 'x';
UPDATE kunden SET bem = nname||', '||vname WHERE bem = 'x';
```

Befehl zum Einarbeiten der Rechtschreibreform, Beispiel (ersetzt auch Anschluß)

```
UPDATE artikel SET bez = regexp_replace(bez, 'anschluß', 'anschluss')
```

Entfernen von Leerzeichen in einer Tabellenspalte

```
gcode=> UPDATE gcodes SET gckenn = regexp_replace(gckenn, '\s', '', 'g');
UPDATE 56
```

4.6 Rechte

```
GRANT ALL ON ALL TABLES IN SCHEMA public TO username;
```

```
Beispiel: handwerk=# GRANT ALL ON ALL TABLES IN SCHEMA public TO rolf;
GRANT
```

Wird nach Einloggen in die DB mit 'psql dbname' ausgeführt. Erlaubt einem neuen User vollen Zugriff auf alle Tabellen der betreffenden Datenbank. Muss im Mehrbenutzerbetrieb für jeden User der über einen Client Zugriff haben soll ausgeführt werden.

4.7 Type Casting in PostgreSQL

Examples of common types in PostgreSQL:

```
-- Cast text to boolean
select 'true'::boolean;
-- Cast float to integer
select 1.0::integer;

-- Cast integer to float
select '3.33'::float;
select 10/3.0; -- This will return a float too

-- Cast text to integer
select '1'::integer;

-- Cast text to timestamp
select '2018-01-01 09:00:00'::timestamp;

-- Cast text to date
select '2018-01-01'::date;

-- Cast text to interval
select '1 minute'::interval;
select '1 hour'::interval;
select '1 day'::interval;
```

4.8 Thema: Kopieren von Daten mit INSERT / UPDATE

Daten aus einer Spalte einer Tabelle in eine Spalte einer anderen Tabelle kopieren

```
INSERT INTO firma (name, zusatz, adr, plz_ort, bank, iban, bic, nrst, mwst, lohn, tel, fax, mail ) SELECT fa_name, fa_bem, fa_adr, fa_plzort, bank, kto, blz, stnr, mwst, lph, fa_tel, fa_fax, fa_mail FROM intern WHERE pid = 1;
UPDATE vorgang SET nname = kunden.nname FROM kunden WHERE vorgang.knr=kunden.knr;
```

4.9 Kopieren mit COPY TO / COPY FROM

```
handwerk=> COPY texte TO '/var/lib/postgresql/9.6/main/texte.csv';
FEHLER:  nur Superuser können COPY mit Dateien verwenden
TIP:  Jeder kann COPY mit STDOUT oder STDIN verwenden. Der Befehl \copy in psql funktioniert auch für jeden:
handwerk=> \copy texte TO 'texte.csv';
COPY 28
(Die Datei befindet sich dann im $HOME des Aufrufenden)
```

COPY ziel FROM '/pfad/zur/postgres/ziel.csv

Kopiert den Inhalt der angegebenen Datei in die Tabelle ziel. Benötigt erweiterte Rechte, die Dateien müssen dem user 'postgres' gehören, siehe Tipp oben. Oder (TO):

```
\copy (SELECT * FROM tabelle WHERE option) TO ‘Pfad/dateiname.csv’ WITH DELIMITER ‘,’ CSV HEADER;
```

Syntax für die \ copy-Befehle:

```
\copy { table | ( query ) } { from | to } { ‘filename’ } [ [ with ] ( option [, …] ) ]
```

Beispiel (FROM):

```
\copy tablename FROM ‘Dateipfadf/dateiname.csv’ WITH DELIMITER ‘,’ CSV HEADER;
```

```
WITH DELIMITER ',' CSV HEADER
```

bedeuten, dass das Trennzeichen der Datei mit den Daten ein Komma ist und dass diese eine Kopfzeile enthält.

Kopieren ohne Auto-Integer Wert, dann mit Spaltenliste

```
\copy artikel (me, sort, bez, ekp, anz, mk, fmat, lmin, min, lk, kalk_ma, kalk_la, kalk_gp, flk) FROM '/home/gottfried/leistpos.csv' WITH DELIMITER '|';
COPY 51
```

4.10 Sequenz neu setzen, z.B. nach dem Einfügen von Datensätzen aus einer CSV-Datei in eine Tabelle

ERROR: duplicate key violates unique constraint

Wenn Sie diese Meldung erhalten, wenn Sie versuchen, Daten in eine PostgreSQL-Datenbank einzufügen, ist wahrscheinlich die Primärschlüsselsequenz in der Tabelle nicht mehr synchron. Um das zu prüfen, führen Sie diese beiden Befehle aus:

```
aldb=# SELECT MAX(artnr) FROM artikel;
aldb=# SELECT nextval('artikel_artnr_seq');
```

Wenn der erste Wert höher als der zweite Wert ist, ist die Sequenz nicht synchron.

Das Neu Setzen der Sequenz:

```
aldb=# SELECT setval('artikel_artnr_seq', 37);
```

Z.B. wäre 37 richtig, wenn SELECT MAX(artnr) FROM artikel; den Wert 36 ergeben hat.

4.11 Suchen und Ersetzen

Sucht in der Spalte lbez in Artikel nach 'Anschluß' und ersetzt dieses Wort durch 'Anschluss'

```
UPDATE artikel SET lbez = REPLACE(lbez,'Anschluß', 'Anschluss');
```

Sucht in der Spalte lbez in Artikel nach '.' Punkt und ersetzt ihn mit ',' Komma

```
UPDATE artikel SET lbez = translate(lbez,'.',',');
```

TRANSLATE ist nur für Einzelwerte, nicht für Wörter geeignet

4.12 Doppelte Einträge finden

```
SELECT sortnr, COUNT(sortnr) AS count FROM artikel GROUP BY sortnr HAVING COUNT(sortnr) > 1;
```

4.13 TO_CHAR

Function	Return Type	Description	Example
to_char(timestamp, text)	text	convert time stamp to string	to_char(current_timestamp, 'HH12:MI:SS')
to_char(interval, text)	text	convert interval to string	to_char(interval '15h 2m 12s', 'HH24:MI:SS')
to_char(int, text)	text	convert integer to string	to_char(125, '999')
to_char(double precision, text)	text	convert real/double precision to string	to_char(125.8::real, '999D9')
to_char(numeric, text)	text	convert numeric to string	to_char(-125.8, '999D99S')
to_date(text, text)	date	convert string to date	to_date('05 Dec 2000', 'DD Mon YYYY')
to_number(text, text)	numeric	convert string to numeric	to_number('12,454.8-', '99G999D9S')
to_timestamp(text, text)	timestamp with time zone	convert string to time stamp	to_timestamp('05 Dec 2000', 'DD Mon YYYY')
to_timestamp(double precision)	timestamp with time zone	convert Unix epoch to time stamp	to_timestamp(1284352323)

5 Spezielle Abfragen

5.1 Unterabfrage mit IN bzw. NOT IN

Es wird nach Artikeln gesucht, die keine Aufträge haben

```
SELECT artikel_nr, bezeichnung
FROM artikel
WHERE artikel_nr not in
   (
   SELECT artikel_nr
   FROM auftrag_pos
   );
```

Man kann hier sehr gut sehen, dass der Sub-Query die SELECT Abfrage in den Klammern nach dem NOT IN ist. In diesem Beispiel wird eine neue Zwischenmenge erzeugt für jede Zeile in "ARTIKEL" , die alle Artikelnummern (artikel_nr) aus der Tabelle "AUFTRAG_POS" enthält. Dann wird für jede Zeile geprüft, ob die Artikelnummer, in der Zwischenmenge enthalten ist. Man hätte diesen Sub-Querie auch noch mit einem WHERE erweitern können.

Problematisch ist die NOT IN Abfrage deshalb, weil die erzeugte Zwischenmenge unnötig groß wird und deshalb aus performance-technischen Gründen nicht ausgeführt werden sollte.

5.2 Unterabfrage mit EXISTS bzw. NOT EXISTS

Das selbe Beispiel noch einmal, nur dieses mal mit NOT EXSITS:

Beispiel:

```
SELECT artikel_nr, bezeichnung
FROM artikel a
WHERE not exists
   (
   SELECT artikel_nr
   FROM auftrag_pos ap
   WHERE ap.artikel_nr = a.artikel_nr
   );
```

Diese Abfrage ist deutlich kleiner, da keine so große Zwischenmenge erzeugt wird. In diesem Beispeil werden die Artikelnummer (arikel_nr) der beiden Tabellen miteiander verglichen, um somit zu schauen ob es einen Auftrag zu finden, der mit keiner Artikelnummer verbunden ist.

5.3 Unterabfrage mit "="

Es wird nach Aufträge gesucht, welchen den höchsten Umsatz erzeugt haben:

Beispiel:

```
SELECT auftrag_nr
FROM auftrag_pos
WHERE anzahl * preis =
   (
   SELECT MAX(SUM(anzahl*preis))
   FROM auftrag_pos
   GROUP BY auftrag_nr
   );
```

Der "=" Operator kann auch für Unterabfragen verwendet werden. Im oberen Beispiel wird geschaut, welcher Auftrag den höchten Umsatz erzielt hat in dem durch MAX(SUM(anzahl*preis) die maximale höchste Summe eines Auftrags ermittelt wird

6 Import Microsoft Access MDB PostgreSQL (Linux)

Um eine Microsoft Access-MDB-Datei in eine PostgreSQL-Datenbank zu laden sind zwei Schritte notwendig, die in diesem Artikel vorgestellt werden.

Der erste Schritt ist, exportieren Sie die MDB-Tabellen mit mdbtools und Laden Sie die Daten in PostgreSQL mit psql von der Standardeingabe.

Um mdbtools zu verwenden, muss die Anwendung installiert werden:

6.1 Vorbereiten der PostgreSQL Zieldatenbank

```
sudo apt-get install mdbtools
```

Sobald mdbtools installiert ist, führen Sie den folgenden Befehl aus, um die Namen aller Tabellen in der MDB-Datei abrufen:

```
mdb-tables datenbankname.mdb
```

Führen Sie den mdb-Schema Befehl aus, um die Zieltabellen zu erstellen, entweder für alle Tabellen in der MDB-Datei:

```
mdb-schema datenbankname.mdb postgres | psql -d datenbank -U username -W -h localhost
```

-d ist die Datenbank, -U der Benutzer -W psql Passwort Option und -H ist der Host-Name

Um nur eine bestimmte Tabelle zu erstellen, verwenden Sie den folgenden Befehl:

```
mdb-schema -T tabelle datenbankname.mdb postgres | psql -d datenbank -U username -W -h localhost
```

6.2 Laden der Daten in PostgreSQL

Direktes Laden der Daten einer Tabelle (Tabelle1) in PostgreSQL:

```
mdb-export -I postgres -H -D '%Y-%m-%d %H:%M:%S' -q \' DDDB2002.mdb Tabelle1 | psql -d mdbdaten -U gottfried -W -h localhost
```

6.3 Extrahieren der Daten ins CSV-Format

Mit folgendem Befehl können Sie eine Tabelle in einer externen CSV-Datei exportieren

```
mdb-export datenbankname.mdb Tabelle1 > tabelle1.csv
```

6.4 Probleme bei *.mdb Import nach PostgreSQL

Die Tabelle1 aus den 4 mdb Datenbanken konnte nach o.g. Verfahren problemlos übernommen werden, da alle den gleichen Primärschlüssel hatten. Dadurch wurden Datensätze, die in zwei der *.mdb Datenbanken vorkamen automatisch ausgesondert. Abweichende Spaltenbezeichnungen (für die datenmäßig gleiche Spalte) konnten durch vorheriges und nachträgliches Umbenennen der Spalten in der Zieltabelle als Problem ausgeschaltet werden.

Anders bei Tabelle2, ohne Primärschlüssel konnten die Daten nach der Übernahme nicht bearbeitet, doppelte Datensätze nicht entfernt werden.

Lösung:

- Manuelles Erzeugen einer Tabelle mit Primärschlüssel

```
mdbdaten=# CREATE TABLE leistung (lid serial, id int4 REFERENCES vorgang(id) ON UPDATE CASCADE ON DELETE CASCADE, lart VARCHAR(10), pos VARCHAR(10), anz NUMERIC(6,2), me VARCHAR(10), bez1 VARCHAR(200), bez2 VARCHAR(200), ep NUMERIC(6,2), gp NUMERIC(6,2), gpd NUMERIC(6,2), gpk NUMERIC(6,2),gpz NUMERIC(6,2), PRIMARY KEY (lid));
CREATE TABLE
```

- Erzeugen der 4 CSV-Dateien und Zusammenführen in eine CSV-Datei
- Hinzufügen und Ausfüllen (Reihe) der Spalte lid (Primärschlüssel)
- Kopieren des Inhaltes ohne Header
- Einfügen über die Base-Datenbank (Tabelle markieren, Einfügen) Erfolg trotz einer Fehlermeldung bezüglich der Integer-Spalten EP,GP usw.
- Anschließend Löschen doppelter Leistungsdatensätze

```
DELETE FROM leistung USING leistung leistung2 WHERE leistung.bez1 = leistung2.bez1 AND leistung.id = leistung2.id ANDleistung.lid < leistung2.lid;
```

- und Löschen leerer Zeilen

```
DELETE FROM leistung WHERE leistung.bez1  IS NULL AND leistung.pos IS NULL AND leistung.ep IS NULL;
```

7 Crontab

Um Sicherungen automatisch ausführen zu lassen, bietet sich ein Eintrag in der Datei ***etc/*crontab** an. Beispiel eines Eintrags:

---

10 19 * * * gottfried /usr/bin/AllSic

#

Hier eine Übersicht, wie sich ein Cronjob aufbaut:

```
*     *     *     *     *  Befehl der ausgeführt werden soll
-     -     -     -     -
|     |     |     |     |
|     |     |     |     +----- Wochentag (0 – 7 | Sonntag ist 0 | * jeder Tag)
|     |     |     +------- Monat (1 – 12  | * jeder Monat)
|     |     +--------- Tag (1 – 31  | * alle )
|     +----------- Stunde (0 - 23)
+------------- Minute (0 - 59)
```

Wichtig ist, dass am Ende der Tabelle ein Kommentar oder eine Leerzeile steht. Ähnlich wie die fstab muss die **crontab** mit einer Leerzeile enden!