Freie Universität Berlin

Prof. Dr. Agn‘es Voisard, Nicolas Lehmann

Datenbanksysteme, SoSe 2017

Übungsblatt 4

Tutor: Nicolas Lehmann
Tutorium 10

Boyan Hristov, Julian Habib

26. Mai 2017

Link zum Git Repository: https://github.com/BoyanH/Freie-Universitaet-Berlin/tree/master/Datenbanksysteme/Solutions/homework4

1 Aufgabe

a)

```
[hristov@Edgy ~]$ sudo su // als Root Nutzer sich einmelden (Root
  rechte für die folgende Operationen notwendig, man kann auch
  jedes mal sudo schreiben)
[root@Edgy hristov]# systemctl start postgresql.service // starte
  den service für postgresql, damit wir postgresql benutzen können
[root@Edgy hristov]# su -l postgres // zu postgres nutzer wächseln
[postgres@Edgy ~]$ exit // wieder zu root Nutzer wechseln, da
  postgres keine Root Rechte hat und wir den Password dafür nicht
  mehr kennen (oops)
[root@Edgy hristov]# passwd postgres -d // lösche den Password für
  Nutzer postgres
[root@Edgy hristov]# passwd postgres // verändere den password.
  Zwei Console Prompts folgen
New password: postgres
Retype new password: postgres
[root@Edgy hristov]# su -l postgres
[postgres@Edgy ~]$ createdb testdb // erstelle Datenbank 'testdb'
[postgres@Edgy ~]$ dropdb testdb // lösche Datenbank 'testdb'
[postgres@Edgy ~]$ createdb dbs // erstelle Datenbank 'testdb'

```

b)

```
 [hristov@Edgy homework4]$ sudo su -l postgres // als postgres
  Nutzer sich anmelden
 [postgres@Edgy ~]$ createuser testuser --password --interactive
 Shall the new role be a superuser? (y/n) n
 Shall the new role be allowed to create databases? (y/n) y
 Shall the new role be allowed to create more new roles? (y/n) y
```

```
Password: testpassword // Erstelle Nutzer 'testuser' mit Password
 'testpassword', der Datenbanken und Rollen erstellen kann, aber
 keine Superrechte hat
[postgres@Edgy ~]$ psql // in PostgreSQL Shell wächseln
postgres=# ALTER USER "testuser" WITH PASSWORD 'testpass'; //
 Password von 'testuser' ändern
postgres=# \q
[postgres@Edgy ~]$ exit
logout
[hristov@Edgy homework4]$ // wieder zu eigenen Nutzer wächseln


```

c)

```
[hristov@Edgy homework4]$ sudo su -l postgres
[postgres@Edgy ~]$ psql -d dbs -U testuser
dbs=> CREATE TABLE Student ( // erstelle Tabelle wie in der
 Übungsaufgabe beschrieben
matrikelnummer integer NOT NULL,
vorname character varying(20) NOT NULL,
nachname character varying(20) NOT NULL
);
CREATE TABLE // output
dbs=> \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | student | table | testuser
(1 row)


```

d)

```
dbs=> ALTER TABLE Student
ADD PRIMARY KEY (matrikelnummer)
;
ALTER TABLE

```

2 Aufgabe

a)
```
SELECT Vorname, Nachname
FROM Passagier
WHERE Alter > 42
```

b)
```
SELECT P.Kreditkarennummer
FROM Passagier P, Fluggesellschaft G, Flug F
WHERE P.ID = F.Passagier-ID AND
        F.Fluggesellschaft-ID = G.ID AND
        Datum > 18.01.2014 AND Datum < 27.09.2014
```

c)
```
SELECT G.Name
FROM Flug F, Wetter W, Fluggesellschaft G
```

```
WHERE F.Datum = W.Datum AND
        F.Fluggesellschaft-ID = G.ID AND
        W.Sonnenscheindauer > 8
```

d)
```
SELECT P.Vorname, P.Nachname
FROM Passagier P, Flug F, Wetter W
WHERE F.Passagier-ID = P.ID AND
        F.Datum = W.Datum AND
        NOT (Temperatur > 20 AND Regenmenge < 10 AND Sonnenscheindauer > 6)
```

3 Aufgabe

)
```
SELECT DISTINCT TOP 3 Name
FROM (
        SELECT F.Fluggesellschaft-ID, F.Datum, Count(*), G.Name
        FROM FLuggesellschaft G, Flug F
        WHERE G.ID = F.Fluggesellschaft-ID
        GROUP BY F.Fluggesellschaft-ID, F.Datum
        ORDER BY Count(*)
        )
```

b)
```
SELECT DISTINCT TOP 10 Vorname, Nachname
FROM (
        SELECT P.Vorname, P.Nachname, F.Passagier-ID, Count(*)
        FROM Passagier P, Flug F
        WHERE P.ID = F.Passagier-ID
        GROUP BY F.Passagier-ID
        ORDER BY Count(*) ASC
        )
WHERE
```

c) Da am meisten und am wenigsten mit verschieden zwei separate Aussagen sind gibt es zwei Möglichkeiten die Aufgabe zu bearbeiten. Entweder zu sehen wie oft jedes Passagier mit je Fluggesellschaft geflogen ist und dann nach den Flüge die Resultate zu ordnen, oder als 2 separate stabile Sortierungen - erstmal nach Flüge absteigend, danach nach verschiedene Fluggesellschaften aufsteigend und am Ende die Top 5 selektieren. Wir haben uns für die 2. Variante geeinigt (klingt sinnvoller). Wir gehen hier davon aus, dass ORDER BY in SQL stabil ist.

```
SELECT DISTINCT TOP 5 Vorname, Nachname
FROM (
        SELECT Fluggesellschaft-ID, Count(*), Vorname, Nachname
        FROM (
                SELECT P.Vorname, P.Nachname, F.Passagier-ID, F.Fluggesellschaft-ID
                FROM Passagier P, Flug F
                WHERE P.ID = F.Passagier-ID
                GROUP BY F.Passagier-ID
                ORDER BY Count(*) DESC
                )
        WHERE
        GROUP BY Fluggesellschaft-ID
        ORDER BY Count(*) ASC
        )
WHERE
```

4 Aufgabe

a) Den cabal-install tool installieren

```
[hristov@Edgy exercise4]$ sudo pacman -S cabal-install
```

Letzten cabal Package List von hackage.haskell.org herunterladen

```
[hristov@Edgy exercise4]$ cabal update
```

Und postgresql-simple Haskell Module herunterladen

```
[hristov@Edgy exercise4]$ cabal install postgresql-simple
```

```haskell
{-# LANGUAGE OverloadedStrings #-}

import Control.Monad
import Control.Applicative
import Database.PostgreSQL.Simple
import qualified Data.Text as Text


main = do
 conn <- connectPostgreSQL "dbname='dbs' user='testuser' host='localhost' password='testpass'"
 putStrLn "Connected to DB! Querying all students"
 xs <- (query_ conn "SELECT vorname, nachname, matrikelnummer FROM Student")
 forM_ xs $ \(fname, lname, matrikelnummer) ->
    putStrLn $ Text.unpack "Matrikelnummer: " ++ show (matrikelnummer :: Int) ++ "; Vorname: " ++ fname ++ "; Nachname: " ++ lname
```

b) Erstmal die Bibliothek für Arbeit mit PostgreSQL installieren.

```
[hristov@Edgy exercise4]$ sudo pacman -S python-psycopg2
```

PostgreSQL Service starten

```
[hristov@Edgy exercise4]$ sudo systemctl start postgresql.service
```

```python
 import psycopg2
import uuid

class Student:

 def __init__(self, fname, lname):
  self.firstName = fname
  self.lastName = lname
  self.id = uuid.uuid4().int & (1<<16)-1 # turn into a 16 bit integer with mask

 def addToDB(self, connection):
  c = connection.cursor()

  c.execute("SELECT * FROM Student")
  students = c.fetchall()
  print("Students in Student table: ")
  for student in students:
   print(student)

  insertStudentSql = "INSERT INTO Student (vorname, nachname, matrikelnummer) VALUES ('{0}', '{1}', '{2}')"
  sqlCommand = insertStudentSql.format(self.firstName, self.lastName, self.id)
```

```
  c.execute(sqlCommand)
  connection.commit()

dbUser = "testuser"
dbName = "dbs"
host = "localhost"
dbUserPassword = "testpass"

connectionStringTemplate = "dbname='{0}' user='{1}' host='{2}' password='{3}'"
connectionString = connectionStringTemplate.format(dbName, dbUser, host,
    dbUserPassword)

try:
 connection = psycopg2.connect(connectionString)
 print("Connected to DB!")
 randomStudent = Student("Will", "Smith")
 randomStudent.addToDB(connection)
 print("Successfully addded Student to DB!")

 connection.close()
except Exception as er:
 print("Failed connecting to DB!")
 print(er)
```

c) Erstmal den Postgresql JDBC Driver für Java herunterladen

```
[hristov@Edgy ~]$ yaourt postgresql-jdbc
```

Füge das .jar zu den Classpath von dem DBTest.java

z.B javac -classpath postgresql-42.1.1.jre6.jar DBTetst.java

```
package fu.alp4;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;


public class DBTest {

    public static void main(String[] args) {

        try {

            Class.forName("org.postgresql.Driver");

        } catch (ClassNotFoundException e) {

            System.out.println("PostgreSQL JDBC Driver not included in class path!");
            return;

        }

        System.out.println("JDBC registered succesffully :)");

        Connection connection = null;
        Statement statement = null;
        ResultSet rs = null;

        try {
            connection = DriverManager.getConnection("jdbc:postgresql://localhost/dbs"
    , "testuser", "testpass");
```

```
35              statement = connection.createStatement();
36          } catch (SQLException e) {
38              System.out.println("Failed connecting to DB!");
39              return;
41          }
43          System.out.println("Successfully connected to DB!");
46          try {
47              System.out.println("Students in DB: ");
48              rs = statement.executeQuery("SELECT matrikelnummer, vorname, nachname FROM
     Student");
49              while (rs.next()) {
50                  int matrikelnummer = rs.getInt("matrikelnummer");
51                  String firstName = rs.getString("vorname");
52                  String lastName = rs.getString("nachname");
53                  System.out.println(String.format("Matrikelnummer: %s; Vorname: %s;
    Nachname: %s",
54                          matrikelnummer, firstName, lastName));
55              }
56              connection.close();
57          }
58          catch(SQLException e) {
59              System.out.println("Error while executing sql query!");
60          }
62      }
63  }
```