

Prof. Dr. Agn es Voisard, Nicolas Lehmann

Datenbanksysteme, SoSe 2017

 bungsblatt 4

Tutor: Nicolas Lehmann

Tutorium 10

Boyan Hristov, Julian Habib

25. Mai 2017

Link zum Git Repository: <https://github.com/BoyanH/Freie-Universitaet-Berlin/tree/master/Datenbanksysteme/Solutions/homework3>

1 Aufgabe

a)

```
1 [hristov@Edgy ~]$ sudo su // als Root Nutzer sich einmelden (Root
   rechte f r die folgende Operationen notwendig, man kann auch
   jedes mal sudo schreiben)
2 [root@Edgy hristov]# systemctl start postgresql.service // starte
   den service f r postgresql, damit wir postgresql benutzen k nnen
3 [root@Edgy hristov]# su -l postgres // zu postgres nutzer w chseln
4 [postgres@Edgy ~]$ exit // wieder zu root Nutzer wechseln, da
   postgres keine Root Rechte hat und wir den Password daf r nicht
   mehr kennen (oops)
5 [root@Edgy hristov]# passwd postgres -d // l sche den Password f r
   Nutzer postgres
6 [root@Edgy hristov]# passwd postgres // ver ndere den password.
   Zwei Console Prompts folgen
7 New password: postgres
8 Retype new password: postgres
9 [root@Edgy hristov]# su -l postgres
10 [postgres@Edgy ~]$ createdb testdb // erstelle Datenbank 'testdb'
11 [postgres@Edgy ~]$ dropdb testdb // l sche Datenbank 'testdb'
12 [postgres@Edgy ~]$ createdb dbs // erstelle Datenbank 'testdb'
13
```

b)

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

```

6 Password: testpassword // Erstelle Nutzer 'testuser' mit Passwort
  'testpassword', der Datenbanken und Rollen erstellen kann, aber
  keine Superrechte hat
7 [postgres@Edgy ~]$ psql // in PostgreSQL Shell wechseln
8 postgres=# ALTER USER "testuser" WITH PASSWORD 'testpass'; //
  Password von 'testuser' ändern
9 postgres=# \q
10 [postgres@Edgy ~]$ exit
11 logout
12 [hristov@Edgy homework4]$ // wieder zu eigenen Nutzer wechseln
14

```

c)

```

1 [hristov@Edgy homework4]$ sudo su -l postgres
2 [postgres@Edgy ~]$ psql -d dbs -U testuser
3 dbs=> CREATE TABLE Student ( // erstelle Tabelle wie in der
  Übungsaufgabe beschrieben
4 matrikelnummer integer NOT NULL,
5 vorname character varying(20) NOT NULL,
6 nachname character varying(20) NOT NULL
7 );
8 CREATE TABLE // output
9 dbs=> \dt
10          List of relations
11 Schema | Name      | Type  | Owner
12 -----+-----+-----+-----
13 public | student   | table | testuser
14 (1 row)
16

```

d)

```

1 dbs=> ALTER TABLE Student
2 ADD PRIMARY KEY (matrikelnummer)
3 ;
4 ALTER TABLE
5

```

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

```