Lehrstuhl Informatik 5 (Informationssysteme)
Prof. Dr. S. Decker, Prof. Dr. M. Jarke
Dr. B. Heitmann, PD Dr. R. Klamma, C. Samsel

23. Mai 2017

Datenbanken und Informationssysteme (Sommersemester 2017)

Übung 5

Abgabe bis 30. Mai 10:00 Uhr. Zu spät eingereichte Übungen werden nicht berücksichtigt.

Bitte reichen Sie Ihre Lösung in Dreiergruppen ein. Die Lösung zu diesem Übungsblatt wird in den Übungen am 30. und 31. Mai vorgestellt. Bitte beachten Sie auch die aktuellen Ankündigungen im L^{2}P-Lernraum zur Vorlesung.

Stellen Sie sicher, das Ihre Lösung in **PostgreSQL** funktioniert.

Bitte den erstellten SQL Code zu den Aufgaben 5.1 und 5.3 *zusätzlich* in einer `.sql`-Datei hochladen. Dies vereinfacht die Korrektur.

Aufgabe 5.1 (SQL Anfragen) (10 Punkte)

Gegeben ist eine Datenbank mit dem Inhalt von `medizinDB.sql`. Formulieren Sie auf Grundlage der in der Vorlesung beschriebenen Syntax die folgenden Ausdrücke in SQL (ohne Ergebnisse).

a) Geben Sie die Namen aller Medikamente aus, welche teurer sind als 20.

b) Geben Sie den Namen und die IDs aller Ärzte aus, die bei keiner Behandlung beteiligt sind.

c) Geben Sie die Namen und den Preis aller Medikamente aus, aufsteigend sortiert nach dem Preis.

d) Erstellen Sie einen VIEW, der die Namen aller Patienten ausgibt, welche bei mindestens einer Behandlung beteiligt sind. Führen sie diesen VIEW aus.

e) Geben Sie die IDs aller Patienten aus, welche an der höchsten/maximalen Anzahl von Behandlungen beteiligt sind.

f) Geben Sie die Namen der Medikamente aus, welche bei mindestens vier Behandlungen verwendet werden.

g) Geben Sie alphabetisch sortiert die Namen aller Ärzte aus, inkl. der Anzahl der verschiedenen Medikamente, welche jeder Arzt bei seinen/ihren Behandlungen verwendet hat.

Hier das Schema der Datenbank:

```
CREATE TABLE arzt (
    arzt_id INTEGER PRIMARY KEY,
    name VARCHAR(80)
);

CREATE TABLE medikament (
    medikament_id INTEGER PRIMARY KEY,
    preis FLOAT,
    name VARCHAR(80)
);

CREATE TABLE patient (
    patient_id INTEGER PRIMARY KEY,
    alter INTEGER,
    name VARCHAR(80)
);

CREATE TABLE behandlung (
    behandlung_id INTEGER PRIMARY KEY,
    arzt INTEGER REFERENCES arzt(arzt_id),
    patient INTEGER REFERENCES patient(patient_id),
    medikament INTEGER REFERENCES medikament(medikament_id)
);
```

Aufgabe 5.2 (SQL Analyse) (5 Punkte)

Gegeben sei folgendes SQL-Schema:

```
CREATE TABLE filmkritiker(
        filmkritiker_id INTEGER PRIMARY KEY,
        name VARCHAR(50)
);

CREATE TABLE film(
        film_id INTEGER PRIMARY KEY,
        name VARCHAR(100)
);

CREATE TABLE rezension(
        kritiker INTEGER REFERENCES filmkritiker(filmkritiker_id),
        film INTEGER REFERENCES film(film_id),
        sterne INTEGER,
        PRIMARY KEY(kritiker,film),
        CHECK (sterne >= 1),
        CHECK (sterne <= 5)
);
```

Betrachten Sie die folgende SQL-Anfrage über das Schema:

```
SELECT name, AVG(sterne)
FROM rezension
JOIN film
ON film_id = film
JOIN(SELECT m.x AS z
     FROM(SELECT kritiker AS x, COUNT(film) AS y
          FROM rezension
          GROUP BY kritiker
          HAVING COUNT(film) >= 200) m
     JOIN(SELECT kritiker AS a, COUNT(film) AS b
          FROM rezension
          WHERE sterne = 5
          GROUP BY kritiker) n
     ON m.x = n.a
     WHERE n.b/m.y <= 0.05) l
ON kritiker = l.z
GROUP BY film_id
```

Welche Informationen liefert diese Anfrage bzw. wie würde die Aufgabenstellung zu dieser Anfrage lauten? Geben Sie jeweils auch eine Beschreibung der Ergebnisse für die Subqueries m, n und l an.

Aufgabe 5.3 (SQL) (5 Punkte)

Betrachten Sie die SQL Datenbank zur Datenhaltung der Foto-Community "SofortBild", die durch folgende SQL Ausdrücke erstellt wurde (`SofortBild.sql`):

```
CREATE TABLE users
(
  userID integer NOT NULL,
  email varchar(255) NOT NULL,
  name varchar(255) NOT NULL,
  age integer NOT NULL,
  CONSTRAINT pkey_users PRIMARY KEY (userID)
);

CREATE TABLE pictures
(
  pictureID integer NOT NULL,
  title varchar(500),
  filename varchar(50),
  timestamp timestamp NOT NULL,
  userID integer NOT NULL,
  location varchar(255) NOT NULL,
  CONSTRAINT pkey_pictures PRIMARY KEY (pictureID),
  CONSTRAINT fkey_users_userID FOREIGN KEY (userID)
    REFERENCES users (userID)
);

CREATE TABLE comments
(
  commentID integer NOT NULL,
  pictureID integer NOT NULL,
  userID integer NOT NULL,
  timestamp timestamp NOT NULL,
  text varchar(5000) NOT NULL,
  CONSTRAINT pkey_comments PRIMARY KEY (commentID),
  CONSTRAINT fkey_pictures_pictureID FOREIGN KEY (pictureID)
    REFERENCES pictures(pictureID),
  CONSTRAINT fkey_users_userID FOREIGN KEY (userID)
    REFERENCES users(userID)
);
```

Erstellen Sie einen Query, der die Timeline der Foto-Community darstellt. Dieser soll die Bilder der letzten 24 Stunden darstellen (neuste Bilder zuerst). Zu jedem Bild wird der Benutzername, Titel, Dateinamen, Aufnahmeort, Zeitstempel und die Anzahl der Kommentare angezeigt.
Hinweis: Informieren Sie sich in der PostgreSQL Dokumentation über die Verwendung von `timestamp` und entsprechenden Operatoren. Stellen Sie außerdem sicher, dass auch Bilder angezeigt werden zu denen es keine Kommentare gibt.