Informationssysteme SS 2016
Prof. Dr.-Ing. Sebastian Michel
MSc. Manuel Hoffmann
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Übungsblatt 8: Ausgabe 07.06.2016, Abgabe 13.06.2016

Aufgabe 0: Online Tutorial zu UDFs und Trigger (1 P.)

Arbeiten Sie das UDF- sowie das Trigger-Online-Tutorial durch. Den Link dazu finden Sie auf der Vorlesungswebseite. Diese Aufgabe wird nicht in den Übungen abgefragt und muss im OLAT nicht markiert bzw. abgegeben werden. Jeder bekommt also einen Punkt.

Aufgabe 1: User-Defined Functions (UDFs) (1 P.)

Sie können in Postgres die Funktion `random()` benutzen, um eine Zufallszahl aus $[0, 1]$ zufällig und gleichverteilt zu erhalten. Diese Funktion liefert bei jedem Aufruf einen neuen Wert.

a) Schreiben Sie eine UDF `vertiefung()`, die den Namen einer Vertiefung angibt. Dabei soll Informationssysteme mit 25% Wahrscheinlichkeit gewählt werden, Software Engineering mit 20%, sowie 10% Wahrscheinlichkeit für Intelligente Systems, Robotik, Kommunikationssysteme, Embedded Systems und Graphentheorie. In allen anderen Fällen soll "Keine Ahnung" zurückgegeben werden.

b) Schreiben Sie eine UDF `sample(chance real, student studenten)`, welche für einen Studenten mit Wahrscheinlichkeit `chance` die Matrikelnummer (int) zurückgibt und sonst NULL.

c) Schreiben Sie eine Funktion `sample_t(amount int)`, die eine zufällige Auswahl von Tupeln der Studententabelle ausgibt. Die Anzahl der Tupel ist durch den Parameter `amount` gegeben. Dabei sollen exakt `amount` Tupel ausgegeben werden, falls es mindestens so viele Einträge in der Studententabelle gibt.

Aufgabe 2: UDFs und Trigger (1 P.)

Wir erweitern das Universitätsschema durch Vorlesungstermine:

```
CREATE TABLE vorlesungstermine (
  terminid INTEGER PRIMARY KEY,
  vorlnr INTEGER REFERENCES vorlesungen(vorlnr),
  tag CHAR(2) NOT NULL, -- 'MO', 'DI', ...
  start_zeit TIME,
  end_zeit TIME
)
```

a) Schreiben Sie eine Funktion `minutes_between`, die zwei `TIME`-Objekte nimmt und die absolute Anzahl an Minuten zwischen diesen Zeiten zurückgibt. Zum Beispiel: `minutes_between('15:15:00', '13:45:00') = 90`. *Hinweis:* Auf folgende Weise kommen Sie an die Minuten eines `time`-Objekts: `select extract(minute from '10:03:06'::time)` (analog für Stunden und `hour`).

b) Erstellen Sie einen Trigger, der dafür sorgt, dass keine Vorlesungstermine eingefügt werden können, welche kürzer sind als 30 Minuten.

c) Verhindern Sie durch Trigger, dass bereits eingetragene Vorlesungstermine auf eine kürzere Dauer als 30 Minuten geändert werden. Das Ändern des Tages soll auch nicht erlaubt sein.

d) Erstellen Sie Trigger, die sicherstellen, dass Professoren nicht zur selben Zeit zwei unterschiedliche Termine haben.

Aufgabe 3: JDBC (1 P.)

Auf der Vorlesungsseite finden Sie eine Vorlage mit Javacode, die Sie benutzen können.

a) Geben Sie den Java-Klassen `Professoren` sowie `Vorlesungen` Attribute entsprechend des Uni-Schemas. Der Konstruktor der Klassen soll ein Resultset als Parameter haben und die Objektattribute entsprechend setzen. Ihr Programm soll Listen mit allen Professoren bzw. Vorlesungen aus der Datenbank lesen.

   Implementieren Sie (i) die Methode `nestedLoopJoin`, die einen brute-force Join beider Listen anhand des Prädikats Professoren.persnr=Vorlesungen.gelesenvon durchführt. Wie der Name sagt, soll mit zwei verschachtelten `for`-Schleifen jede Professoren-Vorlesungen-Kombination geprüft werden.

   Implementieren Sie (ii) die Methode `mergeJoin`, die davon ausgeht, dass die Professorenliste nach `persnr` und die Vorlesungsliste nach `gelesenvon` sortiert ist. Diese Methode soll die Sortierung ausnutzen, um unnötige Vergleiche zu minimieren.

   Zählen Sie bei beiden Join-Implementierungen die Anzahl der benötigten Prädikatsvergleiche sowie die Anzahl der tatsächlichen Ergebnisse.

b) Schreiben Sie ein Java-Programm, das als Parameter eine Vorlesungsnummer, einen Tag, sowie Start- und Endzeit entgegennimmt, und damit einen entsprechenden Vorlesungstermin mit neuer terminid in die Datenbank einfügt. Benutzen Sie für das INSERT ein Prepared Statement und nutzen Sie, dass Sie in SQL mit `CAST('10:00:00' AS TIME)` aus einem String ein Zeit-Objekt machen können. Das Wichtige ist hier die Interaktion mit der Datenbank, halten Sie sich nicht zu lange mit Inputvalidierung o. Ä. auf.