Prof. Dr. Stefan Brass
Institut für Informatik
MLU Halle-Wittenberg

22.07.2019

Vorlesung "Datenbank-Programmierung" — Klausur Sommer 2019, Erster Termin —

Name: _____

Studiengang: _____

Matrikelnummer: _____

Fühlen Sie sich gesundheitlich in der Lage die Prüfung abzulegen? ☐ Ja, ☐ Nein

Unterschrift: _____

(Melden Sie sich unbedingt bei der Aufsicht, wenn Sie sich krank fühlen sollten – eine Krankmeldung muss vor Ende der Prüfung erfolgen.)

Nr	Thema	Punkte	erreicht	Zeit
1	CREATE TABLE, INSERT	12		15 min
2	Indexe	3		5 min
3	Transaktionen, PostgreSQL	6		15 min
4	Konflikt-Serialisierbarkeit	5		10 min
5	SQL: Neuere Konstrukte	10		20 min
6	JDBC	10		15 min
7	Zugriffsrechte	4		10 min
8	PostgreSQL (2 Bonus-Punkte)	0		5 min
	Summe	50		95 min

Hinweise:

- Bearbeitungsdauer: 120 Minuten
- Drei A4-Blätter mit Notizen (Vorder- und Rückseite) sind erlaubt.
  Bücher, Aktenordner, Skript und andere Unterlagen, die über die drei Blätter hinausgehen, dürfen nicht verwendet werden.
- Auch Notebooks, PDAs, Handys etc. sind nicht erlaubt. Schalten Sie Mobiltelefone bitte aus (oder stumm und tun Sie die Telefone in eine geschlossene Tasche, die sie während der Klausur nicht anrühren). Wenn Sie unbedingt erreichbar sein müssen, setzen Sie sich in die erste Reihe und sprechen Sie mit der Aufsicht.
- Die Klausur hat 15 Seiten. Bitte prüfen Sie die Vollständigkeit.
- Verwenden Sie weder Rot- noch Bleistift zum Bearbeiten der Aufgaben.
- Bitte benutzen Sie den vorgegebenen Platz. Wenn Sie auf die Rückseite ausweichen müssen, markieren Sie klar, dass es eine Fortsetzung gibt.
- Tauschen Sie keinesfalls irgendwelche Dinge mit den Nachbarn aus. Notfalls rufen Sie eine Aufsichtsperson zur Kontrolle.
- Bei Aufgaben zum Ankreuzen sollten Sie wenigstens raten, wenn Sie die richtige Lösung nicht wissen. (Wenn Sie nichts ankreuzen, haben Sie den Punkt auf jeden Fall verloren). Es ist jeweils genau eine Antwort pro Teilaufgabe richtig.
- Fragen Sie, wenn Ihnen eine Aufgabe nicht klar ist!

Beispiel-Datenbank

Gegeben sei eine Datenbank von Sehenswürdigkeiten (Museen, Parks, Schlösser, etc.).

- Die wichtigste Tabelle ist

  `SEHENSWERT(ID, NAME, STADT, ART)`

  `NAME` und `STADT` zusammen sind ein Alternativschlüssel. Einige Beispiel-Daten sind:

SEHENSWERT			
ID	NAME	STADT	ART
1	Händel-Haus	Halle (Saale)	Museum
2	Kunstmuseum Moritzburg	Halle (Saale)	Museum
3	Burg Giebichenstein	Halle (Saale)	Burgruine
4	Landesmuseum für Vorgeschichte	Halle (Saale)	Museum
5	Stadtgottesacker	Halle (Saale)	Friedhof
6	Bergzoo Halle	Halle (Saale)	Zoo
7	Neue Residenz	Halle (Saale)	Sonstiges
8	Herrenhäuser Gärten	Hannover	Park
9	Sprengel Museum Hannover	Hannover	Museum
10	Neues Rathaus	Hannover	Sonstiges

- Bewertungen der Sehenswürdigkeiten können von registrierten Nutzern eingetragen werden. Diese stehen in folgender Tabelle:

  `NUTZER(UID, EMAIL, PASSWORD)`

  `EMAIL` ist Alternativschlüssel. Beispieldaten sind:

NUTZER		
UID	EMAIL	PASSWORT
1001	andreas@acm.org	123456
1002	bettina@gmx.de	geheim
1003	christoph@gmail.com	passwort

- Nutzer können Sehenswürdigkeiten 1 bis 5 Sterne geben:

  `BEWERTUNG(ID→SEHENSWERT, UID→NUTZER, STERNE)`

BEWERTUNG		
ID	UID	STERNE
1	1001	5
1	1002	5
2	1001	3
4	1001	3
6	1002	3
6	1003	5
7	1001	5
7	1002	4
8	1003	4

Aufgabe 1 (CREATE TABLE, INSERT) 12 Punkte

a) Schreiben Sie eine `CREATE TABLE` Anweisung für die Tabelle `SEHENSWERT`:

`SEHENSWERT(ID, NAME, STADT, ART)`

- Dabei sei `ID` eine maximal dreistellige Zahl mit möglichen Werten von `1` bis `999`.
- `ID` ist Primärschlüssel der Tabelle.
- `NAME` sei Zeichenkette bis zur Maximallänge 40,
- `STADT` entsprechend bis zur Länge 30,
- `ART` bis zur Länge 20. Für diese Aufgabe sei `ART` eine beliebige Zeichenkette, Sie sollen nicht prüfen, dass nur bestimmte Werte möglich sind.
- Keine der vier Spalten soll Nullwerte erlauben.
- `NAME` und `STADT` zusammen sind Alternativschlüssel der Tabelle.

b) Schreiben Sie eine `INSERT`-Anweisung für die erste Zeile der Tabelle:

ID	NAME	STADT	ART
1	Händel-Haus	Halle (Saale)	Museum

c) Schreiben Sie eine `CREATE TABLE` Anweisung für die Tabelle `BEWERTUNG`:

`BEWERTUNG(`<u>`ID`</u>`→SEHENSWERT, `<u>`UID`</u>`→NUTZER, STERNE)`

Sie können davon ausgehen, dass die Tabelle `NUTZER` schon deklariert ist. Ihr Primärschlüssel `UID` ist eine vierstellige Zahl. Deklarieren Sie auch Schlüssel und Fremdschlüssel der Tabelle `BEWERTUNG`, und stellen Sie sicher, dass `STERNE` wirklich nur die ganzen Zahlen von `1` bis `5` annehmen kann. Kein Attribut erlaubt Nullwerte.

Aufgabe 2 (Indexe) 3 Punkte

a) Wie heisst die typische Datenstruktur für Indexe in relationalen Datenbank-Systemen?

b) Was ist die Komplexität, um zu gegebenem Schlüsselwert die zugehörige Zeile in einer Tabelle mit n Einträgen zu finden?

- ☐ Linear: $O(n)$
- ☐ Logarithmisch: $O(\log(n))$
- ☐ $O(n * \log(n))$
- ☐ Quadratisch: $O(n^2)$

c) Indexe beschleunigen (manchmal) Anfragen, aber haben auch Nachteile. Nennen Sie einen Nachteil von Indexen. (Es gibt keine Zusatzpunkte für die Nennung mehrerer Nachteile.)

Aufgabe 3 (Transaktionen, PostgreSQL) 6 Punkte

a) Sie nutzen PostgreSQL über die Terminal-Schnittstelle `psql` und geben Folgendes ein:

```
DELETE FROM BEWERTUNG;
ROLLBACK;
```

Was werden Sie beobachten? (1 Punkt)

- ☐ `DELETE` gibt einen Syntaxfehler, weil `WHERE` fehlt.
- ☐ `ROLLBACK` gibt eine Warnung, die Daten sind weg.
- ☐ Die Daten sind durch das `ROLLBACK` wieder hergestellt.

b) Nutzer A gibt über `psql` folgende Befehle ein:

```
START TRANSACTION;
UPDATE SEHENSWERT SET ART = 'Museum' WHERE ID = 10;
```

Nun fragt Nutzer B diesen Datensatz ab:

```
SELECT * FROM SEHENSWERT WHERE ID = 10;
```

Was geschieht? (1 Punkt)

- ☐ Nutzer B bekommt eine Fehlermeldung, weil der Datensatz gerade geändert wird.
- ☐ Nutzer B muss warten, bis Nutzer A `COMMIT` oder `ROLLBACK` eingibt.
- ☐ Nutzer B bekommt die alte Version des Datensatzes (mit der Art “Sonstiges”).
- ☐ Nutzer B bekommt die neue Version des Datensatzes (mit der Art “Museum”).

c) Sie wollen einen Deadlock ausprobieren. Was können Sie in zwei parallelen Sitzungen eingeben, um einen Deadlock zu provozieren? Verwenden Sie dazu eine oder mehrere der obigen Tabellen. Sie dürfen SQL-Befehle hier leicht abkürzen, solange klar ist, was Sie meinen. (3 Punkte)

Sitzung A	Sitzung B
`START TRANSACTION`	
	`START TRANSACTION`

Aufgabe 4 (Konflikt-Serialisierbarkeit) 5 Punkte

Gegeben sei folgender Schedule:

T1	T2	T3	T4
read(A)			
	read(B)		
	write(B)		
		write(B)	
		read(A)	
			read(A)
write(A)			
			read(B)

a) Geben Sie einen Konfliktgraphen für den Schedule an.

b) Ist der Schedule serialisierbar? Begründen Sie Ihre Aussage. Geben Sie ggf. die Reihenfolge der Transaktionen für einen äquivalenten seriellen Schedule an.

Aufgabe 5 (SQL: Neuere Konstrukte) 10 Punkte

a) Schreiben Sie eine SQL-Anfrage, die die drei Sehenswürdigkeiten in “Halle (Saale)” mit der besten durchschnittlichen Bewertung liefert.

- Es sollen der Name der Sehenswürdigkeit und die durchschnittliche Bewertung ausgegeben werden.
- Die Spalte mit der durchschnittlichen Anzahl Sterne soll “DURCHSCHNITT” heißen (Groß-/Kleinschreibung ist egal).
- Bei gleicher Bewertung ist egal, welche Sehenswürdigkeit geliefert wird. Es sollen aber nicht mehr als drei ausgegeben werden.
- Die Ausgabe soll nach der durchschnittlichen Bewertung sortiert sein.

Ihre Anfrage muss in PostgreSQL funktionieren, also insbesondere syntaktisch korrekt sein. Im Beispiel-Zustand soll sie folgende Ausgabe liefern:

NAME	DURCHSCHNITT
Händel-Haus	5.0
Neue Residenz	4.5
Bergzoo Halle	4.0

Hinweis: PostgreSQL versteht die Standard-Syntax für “Top-N Anfragen” und die MySQL-Syntax.

b) Produzieren Sie eine Tabelle, die Folgendes enthält. Dabei sollen nur Sehenswürdigkeiten in “Halle (Saale)” berücksichtigt werden.

- die Einzelbewertungen der Nutzer für jede Sehenswürdigkeit (wie in der Tabelle `BEWERTUNG`, aber mit dem Namen der Sehenswürdigkeit und der EMail-Adresse des Nutzers).
- den Durchschnitt aller Bewertungen für eine Sehenswürdigkeit (mit einem Nullwert in der Nutzer-Spalte),
- den Durchschnitt aller Bewertungen eines Nutzers (mit einem Nullwert in der Spalte für die Sehenswürdigkeit).
- den Durchschnitt über alle Bewertungen für alle Sehenswürdigkeiten.

Die Sortierung soll nach dem Namen der Sehenswürdigkeit, und bei gleichem Namen nach der EMail-Adresse des Nutzers erfolgen. Nullwerte sollen jeweils am Ende einsortiert werden. Im Beispiel-Zustand soll folgendes Ergebnis geliefert werden:

NAME	EMAIL	STERNE
Bergzoo Halle	bettina@gmx.de	3.000
Bergzoo Halle	christoph@gmail.com	5.000
Bergzoo Halle		4.000
Händel-Haus	andreas@acm.org	5.000
Händel-Haus	bettina@gmx.de	5.000
Händel-Haus		5.000
Kunstmuseum Moritzburg	andreas@acm.org	3.000
Kunstmuseum Moritzburg		3.000
Landesmuseum für Vorgeschichte	andreas@acm.org	3.000
Landesmuseum für Vorgeschichte		3.000
Neue Residenz	andreas@acm.org	5.000
Neue Residenz	bettina@gmx.de	4.000
Neue Residenz		4.500
	andreas@acm.org	4.000
	bettina@gmx.de	4.000
	christoph@gmail.com	5.000
		4.125

Aufgabe 6 (JDBC) 10 Punkte

Gegeben sei das untenstehende Java-Programm, welches eine über die Kommandozeile übergebene Sehenswürdigkeit nach Name aus der Datenbank heraussuchen und ausgeben soll.

```
import java.sql.*;

public class SightFinder {
  public static void main(String[] args) {
    if (args.length != 1) {
      System.out.println("Bitte genau einen Suchparameter eingeben");
      System.exit(1);
    }
    try (Connection c = DriverManager
          .getConnection("jdbc:postgresql://srv/user",
                         "user", "password")) {
      Statement stm = c.createStatement();
      ResultSet result = stm
        .executeQuery("SELECT NAME, STADT, ART FROM SEHENSWERT" +
                      "WHERE NAME ILIKE '%" + args[0] + "%';");
      do {
        String name = result.getString(0);
        String stadt = result.getString(1);
        String art = result.getString(2);
        System.out.println(name + ", " + stadt + ", " + art);
      } while (result.next());
    } catch (Exception e) {
      System.exit(0);
    }
  }
}
```

Das Programm wird zum Beispiel mit `java SightFinder Haus` aufgerufen und es soll folgendes ausgegeben werden:

```
Händel-Haus, Halle (Saale), Museum
Neues Rathaus, Hannover, Sonstiges
```

Der SQL-Operator `ILIKE` ist wie `LIKE`, ignoriert aber Groß- und Kleinschreibung.

a) Dieses Programm hat einige grundsätzliche Probleme und Fehler. Nennen Sie 3 Fehler, markieren Sie diese Fehler im Quelltext, erläutern Sie das Problem und geben Sie an, wie dieses Problem zu lösen ist.

Gehen Sie davon aus, dass die Imports und die Verbindungsdaten (Verbindung, Nutzername, Passwort) stimmen.

- Problem 1:

- Problem 2:

- Problem 3:

b) Nennen Sie zwei Vorteile von Prepared Statements (gegenüber dem Einfügen der Daten in den SQL-String).

- Vorteil 1:

- Vorteil 2:

Aufgabe 7 (Zugriffsrechte) 4 Punkte

a) Welche vier Informationen speichert eine SQL-Datenbank typischerweise für jedes vergebene Zugriffsrecht im Systemkatalog?

1. _____
2. _____
3. _____
4. _____

b) Schreiben Sie einen SQL-Befehl, um dem Nutzer “`donald`” Leserechte für die Tabelle “`SEHENSWERT`” zu geben.

Aufgabe 8 (PostgreSQL (2 Bonus-Punkte)) 0 Punkte

a) Wie kann man in PostgreSQL/`psql` herausfinden, welche Tabellen es gibt?

b) Kann es in einer PostgreSQL-Datenbank verschiedene Tabellen mit gleichem Namen geben? Wenn ja, worin unterscheiden Sie sich?