Technische Universität Braunschweig
Institut für Informationssysteme
http://www.ifis.cs.tu-bs.de

Wolf-Tilo Balke, Philipp Wille

SQL-Lab – Aufgabenblatt 5 – Application Programming

Szenario

Damit *NET.FIRST* (siehe vorletztes Aufgabenblatt) ein großer kommerzieller Erfolg wird und um die Benutzerzahlen zu steigern, möchte euer Boss die Attraktivität des Angebots erhöhen, indem angemeldeten Benutzern verschiedene „*APPS*“ zur Verfügung gestellt werden. Zum Glück hat er genug Vertrauen, um die Aufgabe nicht an eine Drittfirma auszulagern, sondern *EUCH* damit zu betrauen. In der letzten Aufgabe habt ihr bereits Vorbereitungen dafür getroffen. Eine mögliche Lösung für das letzte Blatt könnte wie folgt aussehen:

Aufsetzen der Datenbank

Dieses Aufgabenblatt baut auf den Zwischenergebnissen des letzten Aufgabenblattes auf. Um Probleme mit verschiedenen Lösungen für das letzte Aufgabenblatt zu vermeiden, **verwendet bitte die bereitgestellten Zwischenergebnisse**. Ihr findet sie auf dem ergänzenden Aufgabenblatt (http://www.ifis.cs.tu-bs.de/webfm_send/1206).

Benutzt die bereitgestellten Zwischenergebnisse zum Aufsetzen der Datenbank. Dazu entfernt zunächst alle alten Tabellen aus eurem Schema. Führt dann die Statements 1 bis 4 aus. Bevor ihr die nächsten beiden Statements eingebt, könnt ihr zuerst die Auswirkungen von Indexen auf die Performanz eurer Anfragen ausprobieren:

Bei Statement 7 handelt es sich um eine Anfrage zum Finden aller möglichen *Kevin Bacon Zahlen* zweier Schauspieler und den Filmen, die sie verbinden. Korrekte Instanzen von *Kevin Bacon Zahlen* sind alle Pfade kleinster Länge (*num* ist Minimal). Führt Statement 7 in eurem Schema aus (dauert ungefähr 30 Sekunden). Jetzt führt Statement 5 aus und erstellt so einen Index auf *actor_from*. Führt Statement 7 ein zweites Mal aus (dauert ungefähr 15 Sekunden). Wenn ihr jetzt mit Statement 6 einen zweiten Index baut und wieder Statement 7 ausführt, hat sich die Ausführungszeit der Anfrage um zwei Drittel reduziert (dauert ungefähr 10 Sekunden).

Java Programm schreiben

In der kommenden RDB1 Vorlesung *Application Programming 2* wird die JDBC (Java Database Connectivity) API vorgestellt. Mit Hilfe dieser Bibliothek kann sich ein Java Programm zu einer Datenbank verbinden und SQL Statements ausführen. Das von euch programmierte Programm soll auf JDBC aufbauen.

Institut für Informationssysteme
Technische Universität Braunschwe

Technische Universität Braunschweig
Institut für Informationssysteme
http://www.ifis.cs.tu-bs.de

Wolf-Tilo Balke, Philipp Wille

Aufgabe A

a. Schreibt auf Basis der Vorlesung die Methode getConnection(), die sich zu der Datenbank verbindet und eine Connection zurückgibt. Mit der zurückgegebenen Connection kann dann weitergearbeitet werden. Den benötigten JDBC Driver findet ihr unter http://www.ifis.cs.tu-bs.de/webfm_send/297. Die Signatur der Methode lautet: *public* Connection getConnection() *throws* SQLException
b. Benutzt die zurückgegebene Connection aus a., um eure Datenbank aufzusetzen. Eine Beschreibung dazu findet ihr auf der ersten Seite. Wichtig sind die Statements 1 bis 6 aus der Beschreibung. Benutzt zum Stellen der SQL Statements normale Statements (*keine Prepared Statements*).
c. Schreibt ein Programm, das die Aufgaben a. und b. automatisch beim Programmstart ausführt. Nennt die Hauptklasse des Programms AufgabeA. Achtet besonders darauf, dass beim Programmstart alle alten Tabellen aus eurem Schema entfernt werden.

Aufgabe B

a. Erstellt das auf der vorherigen Seite beschriebene SQL Schema entweder manuell oder indem ihr das in Aufgabe A entwickelte Programm benutzt.
b. Fordert den Benutzer über ein Menu auf (*Konsole reicht völlig aus!*), zwei Namen von Schauspielern einzugeben. Nennt eure Hauptklasse AufgabeB.
c. Stellt dann eine Connection zur Datenbank her (evtl. indem ihr die getConnection() Methode wiederverwendet). Mit dieser Connection soll das Statement 7 aus der Beschreibung zum Aufsetzen der Datenbank gestellt werden. Die beiden Schauspieler aus Aufgabe b. sollen dabei in das Statement 7 eingesetzt werden. Benutzt dazu ein Prepared Statement, um SQL Injections vorzubeugen! Zeigt dem Benutzer jeweils eine der möglichen kürzesten Verbindungen an, die Statement 7 zurückliefert.
d. Ermöglicht es dem Benutzer, eine andere kürzeste Verbindung anzuzeigen. Wenn es keine neue kürzeste Verbindung mehr gibt, wird wieder die erste Verbindung gezeigt!
e. Ein Benutzer soll die jeweils aktuell angezeigte Verbindung als *beste Verbindung* in der Datenbank speichern können. Legt dazu zunächchst die neue Tabelle *best_connections* an und fügt dann die *besten Verbindungen* der Benutzer hinzu. Wenn ein Benutzer in der Folge nach zwei Schauspielern sucht und sich schon eine Verbindung zwischen den beiden in den *best_connections* befindet, soll zuerst diese Verbindung ausgegeben werden (das spart Berechnungszeit). Allerdings soll ein Benutzer immer die Möglichkeit haben, eine andere *best_connection* auszuwählen (worauf wieder eine *richtige* Suche über Statement 7 erfolgen muss).

Liste der Abgaben für Aufgabenblatt 5

Gebt **alle** von euch **geschriebenen Programme** sowohl als **.java** als auch als **.class** Files ab und zusätzlich **alle Dateien**, die zum **Ausführen** benötigt werden. Schreibt außerdem ein **README.txt** file, das genau erklärt, wie die Programme kompiliert, bzw. ausgeführt werden können (u.a. **Classpath**!). Die Abgabe erfolgt per **Mail**.