Ausarbeitung zum Versuch IIS 4
Kopplung einer relationalen Datenbank über ein Netzwerk

Gruppe 6:

Thomas Espeter

Kai Schmitz-Hofbauer

Protokollführer: Kai Schmitz-Hofbauer

1 Einleitung

1.1 Ziel des Versuches

Ziel dieses Versuches war es, den Aufbau und die Handhabung relationaler Datenbanken kennenzulernen. Als Stellvertreter der relationalen Datenbanken wurde hier das DBMS *(Database Management System) PostgreSQL* behandelt.

1.2 Relationale Datenbanken

Eine Datenbank dient dem Beschreiben, Speichern und Wiedergewinnen von (meist größeren) Datenmengen. Datenbanken werden oft von mehreren Anwendungen in Anspruch genommen und sind meist Teil eines umfangreichen Informationssystems, welches von der Datenbank Informationen anfordert, auswertet und der Datenbank Daten zur Speicherung übergibt. Durch die Trennung von Daten und Programmen können unterschiedlichste Anwendungen mit dem Datenbestand arbeiten und es kann der Programmcode gepflegt werden, ohne dass die Organisation des Datenbestandes verändert werden muss. Daher erleichtern Datenbanken und Datenbankinformationssysteme die Datenverarbeitung in erheblichem Maße. Bei relationalen Datenbanken werden die Daten nicht in einer Datei, sondern geordnet nach Themenkreisen (Entities) in Form von Tabellen verwaltet. Beziehungen zwischen Entities werden durch sogenannte Relations beschrieben. Eine Relation muss wenigstens zwei Entities zueinander in Beziehung setzen. Die Daten der einzelnen Tabellen werden erst bei Abfragen mit SQL (vgl. nächsten Abschnitt) miteinander verbunden. Eine Zeile einer Tabelle wird als Datensatz bezeichnet. Eine Spalte hingegen wird als Attribut bezeichnet.

1.3 Die Abfragesprache SQL

Die Abkürzung SQL steht für *Structured Query Language (strukturierte Abfragesprache)*. SQL ist die Standardsprache, mit der auf Daten einer relationalen Datenbanken zugegriffen werden kann . Im Wesentlichen lassen sich mit SQL folgende Aufgaben durchführen:

- Datendefinition (z. B. das Erstellen neuer Tabellen)
- Datenmanipulation (z. B. Ändern oder Löschen von Daten in Tabellen)
- Datenabruf

Im folgenden werden die wichtigsten SQL-Befehle kurz vorgestellt:
Zum Erstellen neuer Tabellen steht der Befehl `CREATE TABLE` *Tabellenname* zur Verfügung. Sollen Tabellen aus der Datenbank entfernt werden, so kann dies mit dem Befehl `DROP TABLE` *Tabellenname* vollzogen werden. Um in einer Tabelle Datensätze / Zeilen hinzuzufügen, gibt es den Befehl `INSERT INTO` *Tabellenname...* . Zum Löschen von Datensätzen aus einer Tabelle dient der Befehl `DELETE FROM` *Tabellenname...* . Möchte man eine oder mehrere Tabellen durchsuchen und filtern, verwendet man den Befehl `SELECT...` . Zum Ändern von bereits vorhandenen Daten steht der Befehl `UPDATE...` zur Verfügung.

1.4 JDBC

JDBC (*Java Database Connectivity*) ist eine Programmierschnittstelle (engl. *API Application Programmer Interface*), die es Programmierern ermöglicht, von einem Java-Programm auf Datenbanken zuzugreifen. JDBC eignet sich nicht nur, um auf eine lokale Datenbank zuzugreifen, sondern es ermöglicht auch den entfernten Zugriff auf diese über ein Netzwerk. Die zugehörigen

Klassen und Schnittstellen befinden sich im Pakte `java.sql`. Die wichtigsten Elemente von JDBC werden im nächsten Abschnitt bei der Schilderung der Versuchsdurchführung vorgestellt.

2 Versuchsdurchführung

Der Versuch bestand aus zwei Aufgaben. Zum einen sollte eine Klasse `Verbindung.java`, die den Datenbankzugriff kapselt, implementiert werden. Zum anderen sollte ein grafisches Frontend `SQLEdit.java` erstellt werden, das die Benutzung von `Verbindung.java` ermöglicht.

2.1 Verbindung.java

In diesem Versuchsteil galt es eine Klasse `Verbindung.java` zu erstellen. Folgende Anforderung waren vorgeben :
Mit Hilfe der Klasse soll

- eine Verbindung zur Datenbank aufgebaut werden können.
- ein SQL-Befehl gesendet und ausgeführt werden können.
- die Verbindung geschlossen werden können.

Die Klasse wurde wie folgt implementiert :

```
01: import java.sql.*;
02:
03: public class Verbindung
04: {
05:    private Connection con;
06:    private Statement stmt;
07:
08:    public void verbindungAufbauen(String url, String user,
09:                            String passwd, String driver)
10:    {
11:      try
12:      {
13:        Class.forName(driver);
```

```
14:         con = DriverManager.getConnection(url, user, passwd);
15:         stmt = con.createStatement();
16:       }
17:       catch(Exception e)
18:       {
19:         e.printStackTrace();
20:       }
21:     }
22:
23:     public ResultSet sqlBefehlAusfuehren(String sql)
24:     {
25:       ResultSet rs = null;
26:       try
27:       {
28:         rs = stmt.executeQuery(sql);
29:       }
30:       catch(Exception e)
31:       {
32:         e.printStackTrace();
33:       }
34:       return rs;
35:     }
36:
37:     public void verbindungSchliessen()
38:     {
39:       try
40:       {
41:         stmt.close();
42:         con.close();
43:       }
44:       catch(Exception e)
45:       {
46:         e.printStackTrace();
47:       }
48:     }
49: }
```

Der Quelltext soll nun im folgenden erläutert werden :

In Zeile 05 wird ein Attribut `con` vom Typ `Connection` deklariert. Das Interface `Connection` repräsentiert eine Datenbankverbindung. In der folgenden Zeile wird eine Attribut `stmt` vom Typ `Statement` deklariert. Dieses Interface repräsentiert eine Sql-Anweisung (SQL-Statement). In der nachfolgenden Methode `verbindungAufbauen(....)` muss zunächst der Daten-

banktreiber geladen werden(Zeile 13). Nachdem der Treiber geladen wurde, kann mit Hilfe der Klasse `DriverManaegr` eine Verbindung hergestellt werden. Die Klassenmethode `getConnection(...)` des `DriverManager`s erzeugt unter Angabe des Datenbankpfades(url), des Benutzers und des zugehörigen Passwortes eine neue Connection und übergibt diese an `con`. Soll auf eine entfernte Datenbank zugegriffen werden, so muss dies bei in URL angegeben werden. Mit Hilfe der Connection kann nun ein `Statement` erzeugt werden (Zeile 17). Um einen SQL-Befehl auszuführen stellt das Statement-Objekt die Methode `executeQuery(...)` zur Verfügung (Zeile 28). Als Ergebnis gibt diese Methode ein `ResultSet` zurück. Über das `ResultSet` kann auf das Ergebnis des ausgeführten SQL-Befehls zugegriffen werden. Mit der Anweisung `con.close()` (Zeile 42) kann eine Datenbankverbindung geschlossen werden. Es ist ratsam vor dem Schließen der Datenbank auch sämtliche Statements zu schließen (Zeile 41). Da einige Aufrufe eine `SQLException` auslösen können, müssen entsprechende Aufrufe in `try - catch` - Blöcke eingebettet werden. Die Klasse `Verbindung.java` wurde in einen Testrahmen eingebettet und anschließend überprüft. Es konnte kein Fehlverhalten nachgewiesen werden.

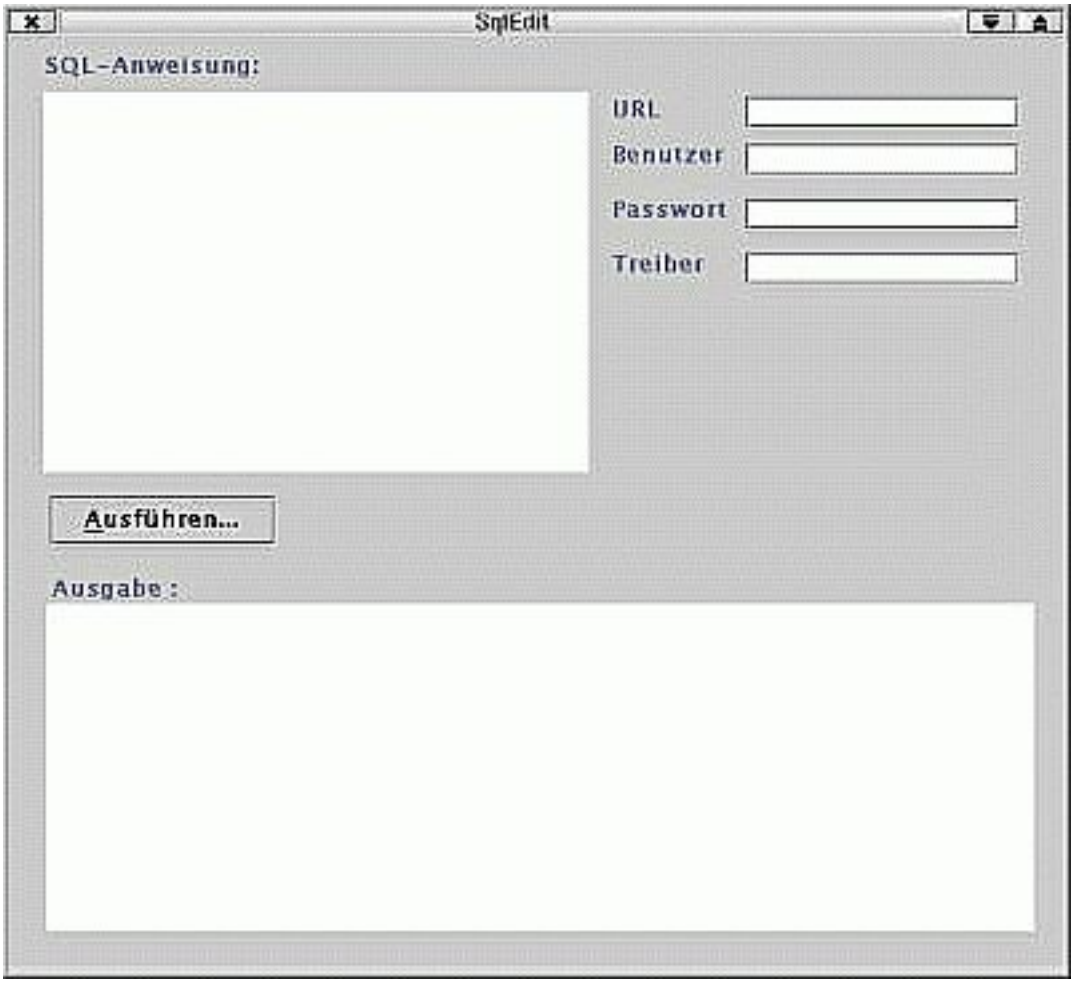
2.2 SQLEdit.java

In diesem Versuchsteil war es Aufgabe, ein GUI `SQLEdit` zu entwerfen, in dem

- eine SQL-Anweisung eingeben werden
- URL, Benutzer, Passwort und Datenbanktreiber angegeben werden
- das Ergebnis der SQL-Anweisung ausgegeben werden

kann.

Das GUI konnte während des Praktikums aus Zeitgründen nicht fertiggestellt werde. Das GUI könnte wie folgt aussehen:



Ein `ActionListener` könnte wie folgt implementiert werden :

```
00:  public class AktionsAbhoerer implements ActionListener
01:  {
02:
03:      public void actionPerformed(ActionEvent e)
04:      {
05:        String url = urlTextField.getText();
06:        String user = userTextField.getText();
07:        String passwd = passwordTextField.getText();
08:        String driver = driverTextField.getText();
09:        String sql = sqlTextArea.getText();
```

```
10:         String output = "";
11:
12:         Verbindung verbindung = new Verbindung();
13:         verbindung.verbindungAufbauen(url, user, passwd, driver);
14:
15:         ResultSet rs = verbindung.sqlBefehlAusfuehren(sql);
16:
17:         try
18:         {
19:           while (rs.next())
20:             output = output + rs.getString(1) + " \n ";
21:         }
22:         catch(Exception e)
23:         {
24:           e.printStackTrace();
25:         }
26:
27:         verbindung.verbindungSchliessen();
28:
29:         outputTextArea.setText(output);
30:       }
31:   }
```

In diesem Programmausschnitt bedürfen nur die Zeilen 19 und 20 einer näheren Erläuterung. `ResultSet` repräsentiert das Ergebnis einer SQL-Abfrage. Für jeden Datensatz der Ergebnismenge der Abfrage (`while (rs.next())`) wird der Wert der ersten Spalte (`rs.getString(1)`) ausgelesen. Weitere Spalten werden in diesem Beispiel aus Gründen der Einfachheit nicht berücksichtigt. Da die Methode `getString(...)` eine `SQLException` auslösen kann, müssen die Zeilen 19 und 20 in einen `try - catch`-Block eingebettet werden.

3 Diskussion der Ergebnisse

Der Versuch hat gezeigt, dass Datenbankzugriff mittels JDBC relativ einfach zu realisieren ist. Bei der Bedienung von JDBC macht es keinen Unterschied, ob man auf eine lokale Datenbank oder auf eine entfernte Datenbank in einem Netzwerk zugreift. Lediglich bei dem Aufbau der Datenbankverbindung muss

in der URL angegeben werden, ob es sich um eine entfernte oder um eine lokale Datenbank handelt. Auf diese Art und Weise stellt ein Zugriff auf eine entfernte Datenbank keine Schwierigkeit mehr dar, da sich ein Programmierer um netzwerkspezifische Details nicht kümmern muss. Auf eine Sache sei an dieser Stelle allerdings noch hingewiesen :
Auch wenn der Datenbankzugriff mittels JDBC unkompliziert ist, folgt daraus nicht, dass der Umgang mit Datenbanken im Allgemeinen eine einfache Angelegenheit ist. Abhängig von Anforderungen an eine Datenbank oder der Komplexität der Datenstruktur einer Datenbank kann sich die Handhabung einer Datenbank als sehr schwierig erweisen.