JAVA und MySQL

Alfred Nussbaumer

Das für Unterricht und nicht kommerzielle Anwendungen frei verfügbare Datenbanksystem MySQL hat sich seit einigen Jahren im Bereich von Webanwendungen etabliert. JAVA stellt seit der Version 1.2 eine leistungsfähige Schnittstelle zu verschiedenen Datenbanksystemen zur Verfügung; die Verbindung zu MySQL wird in diesem Artikel in einigen grundlegenden Beispielen besprochen.

1. Voraussetzungen

Ein MySQL-Server muss entweder am lokalen Rechner oder im Netzwerk erreichbar sein; im Rahmen der folgenden Beispiele wurde MySQL der Version 4.0.15 verwendet. Entsprechende Berechtigungen müssen für die Verbindung mit dem Server und für den Zugriff auf die Datenbank vorhanden sein. Im Beispiel hat der Benutzer **nus** lediglich Zugriff auf die Datenbank **mdat_nus** und keinerlei Rechte am Datenbankserver.

Für die Beispiele verwenden wir die einfach aufgebaute MySQL-Tabelle "weblinks":

```
+-----------+--------------+
| Field     | Type         |
+-----------+--------------+
| id        | int(11)      |
| kategorie | char(3)      |
| url       | varchar(120) |
| notiz     | text         |
| datum     | date         |
+-----------+--------------+
```

Es ist günstig, mit dieser Tabelle zunächst auf MySQL-Ebene zu arbeiten und einige Datensätze einzugeben, anzuzeigen etc.

Für JAVA ist zusätzlich zum java.sql-Package der MySQL-Treiber MySQL Connector/J erforderlich. Aktuelle Versionen liegen auf der MySQL-Webseite als **.zip-** oder **.tar.gz**-Dateien bereit (*[1]*). Für die folgenden Beispiele wurde die Version **mysql-connector-java-3.0.9-stable.tar.gz** verwendet. Nach dem Entpacken dieser Datei liest man am besten die mitgelieferte README-Datei: Hier wird u.A. angegeben, dass das notwendige **.jar**-Archiv **mysql-connector-java-3.0.9-stable-bin.jar** in das entsprechende **$JAVA_HOME**-Verzeichnis kopiert werden muss:

```
nus@ice:~> ls $JAVA_HOME/jre/lib/ext
dnsns.jar
localedata.jar
sunjce_provider.jar
ldapsec.jar
mysql-connector-java-3.0.9-stable-bin.jar
```

Alternativ dazu kann der Ort des **.jar**-Archives über die **CLASSPATH**-Variable angegeben werden.

2. Grundlagen

Im ersten Beispiel sollen alle Datensätze einer MySQL-Tabelle ausgegeben werden. Dazu ist es zunächst notwendig, eine Verbindung zum MySQL-Server herzustellen. Dies ist mittels JDBC (*Java Database Connectivity*) möglich, welches von der Firma SUN entwickelt wurde, um den Zugriff auf externe Datenbankschnittstellen zu ermöglichen. Das Package **java.sql** stellt die entsprechenden Interfaces und Klassen zur Verfügung, eine detallierte Beschreibung dazu findet sich in der Dokumentation zu Java (*[2]*).

Grundsätzlich erfolgt das Abarbeiten von SQL-Statements in folgenden Schritten:

1. Laden des passenden Datenbanktreibers. Der MySQL-Treiber wird mit der Methode `Class.forname("com.mysql.jdbc.Driver")` geladen.
2. Eine Verbindung zum Datenbankserver herstellen. Als Ergebnis liefert die Methode `DriverManager.getConnection()` das Connection-Objekt `con`, das die Verbindung zum Datenbankserver offen hält. Als Parameter müssen die Datenquelle mit URL und Name der Datenbank, der Name des Benutzers, der auf die Datenbank zugreifen darf, und das Passwort des Benutzers angegeben werden.
3. Ein JDBC-Statement erzeugen. Die Methode `con.createStatement()` liefert dabei das Statement-Objekt `stmt`, mit dem die gewünschten SQL-Anweisungen an den Dabenbankserver gesendet werden können.
4. SQL-Anweisungen ausführen. Je nach Aufgabe werden hier verschiedene Methoden verwendet, die in den folgenden Beispielen besprochen werden.
5. Die Ergebnismenge ausgeben. Liefert die SQL-Anweisung `SELECT` eine Ergebnismenge zurück, so sind ihre Elemente in einem ResultSet-Objekt enthalten. Sie werden mit geeigneten Methoden ausgelesen.

3. Datensätze anzeigen

`SELECT`-Befehle werden mit der Methode `stmt.executeQuery()` gesendet. Das Ergebnis der Abfrage wird im ResultSet-Objekt `rslt` zurückgegeben. Ein Zeiger auf die Zeilen dieser Ergebnismenge wird verwendet, um alle Daten ausgeben zu können: Zu Beginn weist dieser Zeiger auf die erste Zeile, er wird mit der Methode `rslt.next()` auf den nächsten Datensatz verschoben. So lange der Zeiger auf einen gültigen Datensatz weist, liefert er den boole'schen Wert `true`; damit eignet sich der Aufruf der Methode `rslt.next()` als Schleifenbedingung.

```
import java.sql.*;

public class mysql1 {
  public static void main(String args[]) {
    treiber_laden();
    daten_ausgeben();
  }

  public static void treiber_laden() {
    try {
      Class.forName("com.mysql.jdbc.Driver");
    }
    catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }

  public static void daten_ausgeben() {
    try {
      Connection con =
        DriverManager.getConnection
          ("jdbc:mysql://localhost/mdat_nus","nus","pwd");
      Statement stmt = con.createStatement();
      ResultSet rslt = stmt.executeQuery("select * from weblinks");
      while (rslt.next()) {
        System.out.println(rslt.getInt(1) + "\t" +
                           rslt.getString(2) + "\t" +
                           rslt.getString(3) + "\t" +
                           rslt.getDate(5));
      }
      stmt.close();
      con.close();
    }
    catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

Nach dem Kompilieren und Ausführen erhalten wir eine einfach formatierte Ausgabe der Datensätze; dabei wurde die vierte Spalte nicht ausgegeben:

```
1    edv       http://www.mysql.com      2003-12-30
2    phy       http://www.cern.ch        2003-12-31
```

Man beachte die ResultSet-Methoden `rslt.getInt()`, `rslt.getString()` und `rslt.getDate()`, mit denen die einzelnen Datenfelder des jeweiligen Datensatzes ausgegeben werden. Als Parameter wird jeweils die Position des Datenfeldes als ganze Zahl (beginnend ab 1) übergeben. Weitere `rslt.get`-Methoden werden in der JAVA-Dokumentation (*[2]*) bzw. in den zugehörigen Tutorials (*[11]*, *[12]*) beschrieben.

alfred.nussbaumer@schule.at

Die `close()`-Methode schließt ein Statement- bzw. Connection-Objekt.

4. Datensätze hinzufügen

Die Statement-Methode `stmt.executeUpdate()` wird für SQL-Anweisungen wie `INSERT`, `DELETE` und `UPDATE` verwendet. Die Methode `executeUpdate()` übermittelt das SQL-Statement, sie liefert jedoch kein ResultSet-Objekt zurück. Allfällige SQL-Fehler werden als Exception erkannt. Im folgenden Beispiel wird `stmt.executeUpdate()` verwendet, um neue Datensätze an das Ende der Tabelle hinzuzufügen.

```
public static void daten_hinzufuegen() {
  try {
    Connection con = DriverManager.getConnection
      ("jdbc:mysql://localhost/mdat_nus","nus","pwd");
    Statement stmt = con.createStatement();
    stmt.executeUpdate ("insert into weblinks values
      ('','edv','http://www.novell.com','',now())");
    stmt.close();
    con.close();
  }
  catch (Exception e) {
    System.out.println(e.getMessage());
  }
}
```

Das erste Datenfeld id wird automatisch inkrementiert, sodass kein Wert übergeben werden muss. Die MySQL-Funktion `NOW()` fügt in das fünfte Datenfeld das aktuelle Datum ein.

Um verschiedene Datensätze in die Tabelle eintragen zu können, kann man beispielsweise Argumente beim Aufruf der Klasse verwenden:

```
import java.sql.*;

public class mysql0 {

  public static void main(String args[]) {
    verbindung_herstellen();
    daten_eingeben(args);
  }

  public static void verbindung_herstellen() {
    try {
      Class.forName("com.mysql.jdbc.Driver");
    }
    catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }

  public static void daten_eingeben(String args[])
  {
    try {
      Connection con = DriverManager.getConnection
        ("jdbc:mysql://localhost/mdat_nus","nus","pwd");
      Statement stmt = con.createStatement();
      String sql_anweisung = "insert into weblinks values
        ('', '" + args[0] + "','" + args[1] +
         "','" + args[2] + "',now())";
      stmt.executeUpdate(sql_anweisung);
      stmt.close();
      con.close();
    }
    catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

Soll eine SQL-Anweisung wiederholt ausgeführt werden, ist es günstiger, so genannte PreparedStatement-Objekte zu verwenden: Mit der Connection-Methode `prepareStatement()` wird eine SQL-Anweisung zunächst mit Hilfe von Platzhaltern an den MySQL-Server gesendet und übersetzt; anschließend erfolgt mit den PreparedStatement-Methoden `setInt()`, `setString()`, usf. die Zuweisung von Inhalten. Die PreparedStatement-Methode `executeUpdate()` aktualisiert schließlich die MySQL-Tabelle.

```
public static void daten_hinzufuegen() {
  try {
    Connection con = DriverManager.getConnection
      ("jdbc:mysql://localhost/mdat_nus","nus","pwd");
    PreparedStatement ppstmt = con.prepareStatement
      ("insert into weblinks values('',?,?,?,now())");
    ppstmt.setString(1,"edv");
    ppstmt.setString(2,"http://www.heise.de");
    ppstmt.setString(3,"IT-Nachrichten online");
    ppstmt.executeUpdate();
    ppstmt.close();
    con.close();
  }
  catch (Exception e) {
    System.out.println(e.getMessage());
  }
}
```

Weitere `ppstmt.set`-Methoden werden in der JAVA-Dokumentation (*[2]*) beschrieben.

5. Datensätze löschen

Einzelne Datensätze sollen anhand der eindeutigen `id` ausgewählt und gelöscht werden. In der `prepareStatement`-Methode wird für die `WHERE`-Klausel zunächst ein Platzhalter für diese `id` verwendet:

```
public static void daten_loeschen() {
  try {
    Connection con = DriverManager.getConnection
      ("jdbc:mysql://localhost/mdat_nus","nus","pwd");
    PreparedStatement ppstmt = con.prepareStatement
      ("delete from weblinks where id = ?");
    ppstmt.setInt(1,4);
    ppstmt.executeUpdate();
    ppstmt.setInt(1,3);
    ppstmt.executeUpdate();
    ppstmt.close();
    con.close();
  }
  catch (Exception e) {
    System.out.println("e.getMessage());
  }
}
```

Zu beachten ist, dass die PreparedStatement-Methode `executeUpdate()` nach jeder Belegung des Parameters aufgerufen werden muss.

6. Datensätze ändern

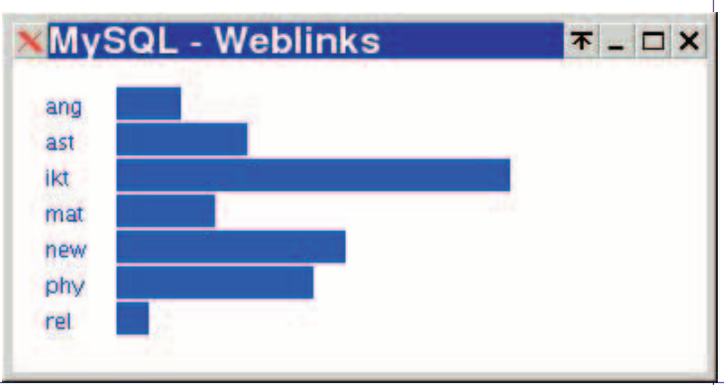
In diesem Beispiel soll die Kurzbezeichnung für die Kategorie geändert werden. Dabei wird die neue Bezeichnung `ikt` für alle Datensätze eingetragen, für die die Klausel `where kategorie = edv` erfüllt ist.

```
public static void daten_aendern() {
  try {
    Connection con = DriverManager.getConnection
      ("jdbc:mysql://localhost/mdat_nus","nus","pwd");
    PreparedStatement ppstmt = con.prepareStatement
      ("update weblinks set kategorie=? where kategorie=?");
    ppstmt.setString(1,"ikt");
    ppstmt.setString(2,"edv");
    ppstmt.executeUpdate();
    ppstmt.close();
    con.close();
  }
  catch (Exception e) {
    System.out.println(e.getMessage());
  }
}
```

7. Aufgaben, Ausblick

Die vorgestellten, einfachen Beispiele können einerseits durch die enormen Möglichkeiten erweitert werden, die SQL bietet; andererseits sollen Java-Applikationen mit einer entsprechenden GUI-Programmierung erstellt werden.

1. Eine Applikation soll eine Statistik über die Anzahl der Linkeinträge zu den bestimmten Kategorien grafisch ausgeben. Wie oft ein



Eintrag zu einer bestimmten Kategorie besteht kann am besten mit Hilfe der MySQL-Aggregatfunktion `COUNT()` bestimmt werden:

```
select kategorie, count(kategorie)
from weblinks
group by kategorie;
```

(Der JAVA-Code zu dieser kurzen Applikation kann unter ***[13]*** nachgelesen werden).

2. Eine Applikation ist zu erstellen, mit der die Eingabe und Ausgabe der Datenbankinhalte über entsprechende GUI-Objekte erfolgen (Textfelder, Labels, Buttons, etc.).
3. Mit Hilfe von GUI-Objekten sollen Datensätze zum Editieren ausgewählt werden. Die veränderten Spalten sollen schließlich am Datenbankserver aktualisiert werden.
4. Mit Hilfe der JDBC-ODBC-Bridge für MS-ACCESS sind Datensätze einer ACCESS-Tabelle auszugeben.

   Für den Unterricht ergeben sich darüber hinaus wichtige Querverbindungen zu bereits bekannten Datenbanken wie MS-ACCESS und bekannten Unterrichtsfeldern wie das Erstellen dynamischer Webseiten mittels PHP und MySQL.

8. Literatur, Weblinks

[1] *http://www.mysql.com/downloads* (Download aktueller JDBC-Treiber)

[2] *http://java.sun.com/j2se/1.4.2/docs/index.html* (Documentation aller verfügbaren Packages)

[3] Paul Dubois, "MySQL - Entwicklung, Implementierung und Referenz", Markt + Technik

[4] *http://www.mysql.de/doc/de/* (MySQL-Handbuch, Referenz)

[5] Michael Kofler, "MySQL - Einführung, Programmierung, Referenz", Addison-Wesley

[6] *http://www.mysql.de/doc/de/Java.html* (Kurzbeschreibung der MySQL-API für Java)

[7] Herbert Schildt, "Java 2 Ent-Packt", mitp-Verlag

[8] Andreas Eberhart, Stefan Fischer, "Java-Bausteine für E-Commerce-Anwendungen", Hanser

[9] Christian Ullenboom, "Java ist auch eine Insel", Galileo Computing

[10] *http://java.sun.com/products/jdbc/* (JDBC-Technologie, Links zu Dokumentation, Tutorials)

[11] *http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html* (Tutorial zu JDBC)

[12] *http://java.sun.com/developer/Books/JDBCTutorial/index.html* (Tutorial zu JDBC)

[13] *http://nus.lugsp.at/wpf/informatik/JAVA* (Unterrichtsbeispiele zum Progarmmieren mit JAVA)

Online

http://listen.pcnews.at/

DotNet-Entwicklung

Franz Fiala, Werner Illsinger

Wie schon einmal berichtet, sind der CCC und PCC der INETA beigetreten (das ist eine Vereinigung von Usergroups, die sich mit DotNet-Programmierung beschäftigen). (*http://www.ineta.org/*).

Preisnachlass für Mitglieder des CCC und PCC

Mitglieder dieser Usergroups erhalten bei vielen Produkten einen Preisnachlass, bitte informieren Sie sich bei dieser Seite: *http://www.ineta.org/DesktopDefault.aspx?tabindex=4&tabid=15*

Materialien für Entwickler

Wir erhalten von der INETA in jedem Quartal eine Lieferung von Materialien, für die Entwicklung von DotNet-Programmen und wollen diese CDs und Bücher (natürlich kostenlos) an unsere Mitglieder weitergeben.

Clubmitglieder wählen aus der folgenden Liste ein Produkt (CD, Poster oder Buch) aus und erhalten es mit der Post zugeschickt. Wir haben zwei CD-Sätze bekommen. Verstehen Sie bitte, dass wir entsprechend dem Aufdruck auf den CDs nur die Originale und nur an Clubmitglieder weitergeben. Weiters ist zu beachten, dass die CDs nur zum Zwecke der Programmentwicklung eingesetzt werden dürfen.

CDs

- Microsoft Windows Security Update CD (1 CD) (mit Key) enthält *Security Update MS 03-039* und Hinweise für die Benutzung einer Firewall und Antivirus Software
- *Visual Studio Tools für Office 2003* (1 CD)
- *Windows 2000 SP4* (nur SP) (1 CD)
- *Microsoft SQLserver 2000 Developer Edition* (1 CD) (mit Key)
- *Microsoft SQLserver 2000 SP3a* (nur SP) (1 CD)
- *MSDN Library* (3 CDs)
- *Access 2003 Developer Extensions* (1 CD)
- *Visual Studio .net Prerequisites* (1 CD)
- *Visual Basic .net Standard Englisch* (1 CD) (mit Key)

Poster

Wir haben 5 Postersätze bekommen (Sprachen Deutsch, Englisch, Französisch, Italienisch, Spanisch). Ein Postersatz enthält

- *Visual Basic .net*
- *Microsoft Foundation Class Library*
- *Namespaces an Selected Classes in the .net-Framework*
- *Business Object*
- *Visual Studio .net Data Access*
- *XML Web Services*

Auch die fremdsprachigen Folder sind nützlich, weil alle Bezeichner ohnehin in englischer Sprache abgebildet sind.

Bücher

Wir haben auch zwei Bücher bekommen:

- *Developing Windows-Based Applications with Visual Basic .Net and Visual Basic C# .Net*. Aktuell für .Net 2003 mit Trainigs CD und Gutschein für 15 % Preisnachlass für Vue-Zertifikat
- *Writing Secure Code*

Wenn Sie eines der Bücher bestellen, ersucht die **PCNEWS**-Redaktion um eine kurze Besprechung für alle anderen Leser für die kommende Ausgabe der **PCNEWS**.

Für regelmäßige Informationen bestellen Sie die Mailing-Liste **PCNINFO** (***siehe Seite 47***).

franz@fiala.cc