

12. DATENBANKSYSTEME: PROGRAMMIERSCHNITTSTELLE

403

JDBC

Java Database Connectivity

- Standardisierte Schnittstelle zur Anbindung von relationalen Datenbanken an Java
- Vorgehen
 - Verbindung (*Connection*) zur Datenbank herstellen
 - Statement* zur Ausführung erstellen
 - Parameter des Statements setzen
 - Statement ausführen, Resultat *ResultSet* (auch: *cursor*)
 - Lese Tupel vom *ResultSet*
- JDBC ist nicht auf SQL beschränkt.

404

Herstellen der Verbindung

Connection conn =
DriverManager.getConnection(urlDB, username, password)

- benötigt JDBC Treiber (vom DBMS zur Verfügung gestellt)

Für diese Vorlesung
download auf <http://dev.mysql.com/downloads/connector/j/>

- urlDB: Identifiziert die Datenbank eindeutig
 - ein Server kann mehrere Datenbanken verwalten
- Benutzername und Passwort
- Andere Einstellungen in der Konfiguration
 - Puffergrößen etc.

405

Wdh. Ausnahmen (Exceptions)

Ausnahmen sind Fehlerereignisse

- Werden oft vom System ausgelöst
- Können aber auch explizit im Programm (z.B. JDBC Bibliothek) ausgelöst werden
- Können abgefangen und behandelt werden

Strukturierung in Java durch "try" und "catch":

```
try {
    // Hier stehen Anweisungen, bei denen
    // eine Fehlerbedingung eintreten kann
} catch (Fehlertyp1) {
    // "Behandlung" dieses Fehlertyps;
} catch (Fehlertyp2) {
    // "Behandlung" dieses Fehlertyps;
} finally
{
    // Code, der immer ausgeführt wird, z.B. für Verbindungsabbau
}
```

406

Beispiel: Treiberinstallation, Verbindungsaufbau

```
import java.sql.*;
...
try{
    Class.forName("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}

String username = ...;
String password = ...;
String url = "jdbc:mysql://mysqlweb.ethz.ch/"+username;

Connection conn = null;
try{
    conn = DriverManager.getConnection(url, username, password);
} catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
    return;
}
...
```

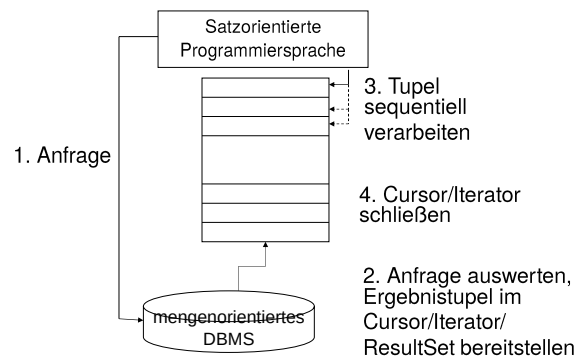
Installation des jdbc Treibers
für MySQL

Aufbau der Verbindung zur
Datenbank

407

Anfragen in Anwendungsprogrammen

- mehrere Tupel im Ergebnis



Beispiel: Namen aller Professoren

```
...
try{
    stmt = conn.createStatement();
    ResultSet result =
        stmt.executeQuery("select name from professoren");

    if (result != null)
    {
        while (result.next())
            System.out.println(result.getString(1));
    }
} catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
} finally {
    ... result.close();
    ... stmt.close();
}
```

Statement ausführen

Über Ergebnis
iterieren (Zeilen)

Fehlerbehandlung

Freigabe der
Ressourcen

409

Metadaten

- ResultSet stellt Methode bereit zum Abholen der Metadaten
ResultSetMetaData metaData = result.getMetaData();
- Anzahl Spalten, Namen, Typ (etc.)
- Bietet z.B. die Möglichkeit über die Spalten zu iterieren

410

Beispiel: Tabelle aller Studenten

```
try{
    stmt = conn.createStatement();

    String query = "select * from Studenten";
    if (stmt.execute(query))
        result = stmt.getResultSet();
    if (result != null)
    {
        ResultSetMetaData metaData = result.getMetaData();

        System.out.print("|");
        for (int col=1; col <= metaData.getColumnCount(); col++)
            System.out.print(metaData.getColumnName(col) + "|");
        System.out.println();
        System.out.println("-----");

        while (result.next()) { // Zeilen
            System.out.print("|");
            for (int col=1; col <= metaData.getColumnCount(); col++) // Spalten
                System.out.print(result.getString(col) + " |");
            System.out.println();
        }
    }
} catch (SQLException ex)
{ ... }
```

Legi	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoteles	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

411

PreparedStatements / Cursor Caching

- Verwende PreparedStatements!
- Beispiel:
insert into professor(name, level) values(?,?)
- Warum? Vermeide Overhead des Anfrageoptimierers für jeden Aufruf

412

Parametrisierte Anfragen

```
PreparedStatement s = conn.prepareStatement(
    "SELECT name FROM professoren WHERE rang = ?");
ResultSet r;
...

s.setString(1, "AP");
r = s.executeQuery();
while (r.next()) ...
```



413

Tipps und Tricks: Connection Pooling

- Mehrere Verbindungen zur Datenbank
- Hole eine der unbenutzten Verbindungen vor dem Datenbankzugriff
- Führe Statements mit dieser Verbindung aus
- Warum? Datenbank nicht mit schwierigen Anfragen blockieren
- Daumenregel: 5-10 Verbindungen

414

Zusammenfassung JDBC

- Einfaches Protokoll zum Senden von Nachrichten an die Datenbank
 - Datenbank typischerweise als Server implementiert
- SQL Syntax wird nicht zur Kompilierungszeit geprüft
 - Für Java sind das nur Strings
- Typsicherheit der Parameter zur Laufzeit geprüft
- Alle JDBC Statements können SQL Exceptions auslösen
 - sollten abgefangen werden
- Neuer Standard SQLJ

415

SQLJ

- SQL in Java eingebettet
- Benutzt Präprozessor zur Kompilierungszeit für Typsicherheit, SQL Syntax
- SQL Statements werden mit #sql annotiert
- Iterator (Cursor) ähnlich wie bei JDBC

```
#sql iterator ProfIterator(String name, String level);
ProfIterator myProfs;
#sql myProfs = { SELECT name, level FROM Professor };
while (myProfs.next()) {
    System.out.println(myProfs.name() + myProfs.level());
}
```

416