Datenzugriff via Cursor am Beispiel JDBC

Datenbanksysteme 4. Jahrgang

Abteilung Informationstechnik

WAS BISHER GESCHAH ...

Bisher:

Abfragen liefern bisher stets eine Menge an Datensätzen zurück.

```
SELECT *
FROM tabelle;
```

Spalte1	Spalte2	Spalte3
Wert1	Wert2	Wert3
Wert4	Wert5	Wert6
Wert7	Wert8	Wert9

Nachteil: u.U. große Datenmenge
--> Transport über Netzwerk?

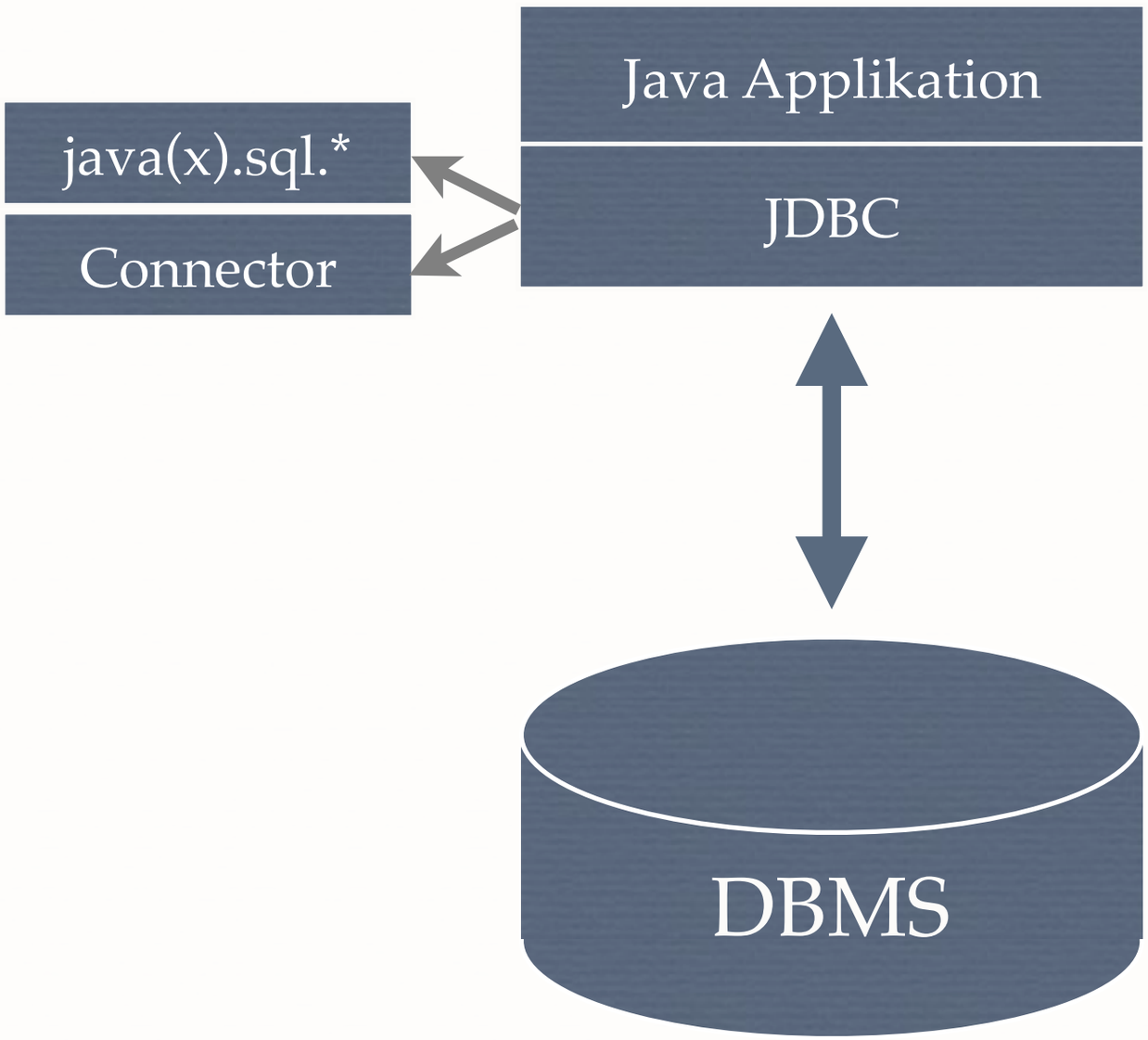
CURSOR

Abhilfe: der Cursor

- ★ ist ein Zeiger auf **einen** Datensatz eines Ergebnisses
- ★ 1. Position **vor dem ersten** Ergebnisdatensatz
  --> dadurch Zeiger auf leeres Ergebnis möglich
- ★ Letzte Position **hinter dem letzten** Ergebnisdatensatz
- ★ wird auf Kommando weiterbewegt
- ★ üblicherweise nur eine Fetch-Richtung (vorwärts)*
  --> Ergebnis kann nur 1 Mal durchwandert werden

  * in moderneren Implementierungen nicht mehr so

STRUKTUR EINER JDBC-ANWENDUNG



Client

proprietäres Protokoll des DBMS

Datenbank-Server

JDBC - DIE JAVA-SEITE

java.sql.*

Classes
Date
DriverManager
DriverPropertyInfo
SQLPermission
Time
Timestamp
Types

Enums
ClientInfoStatus
RowIdLifetime

Interfaces
Array
Blob
CallableStatement
Clob
Connection
DatabaseMetaData
Driver
NClob
ParameterMetaData
PreparedStatement
Ref
ResultSet
ResultSetMetaData
RowId
Savepoint
SQLData
SQLInput
SQLOutput
SQLXML
Statement
Struct
Wrapper

Exceptions
BatchUpdateException
DataTruncation
SQLClientInfoException
SQLDataException
SQLException
SQLFeatureNotSupportedException
SQLIntegrityConstraintViolationException
SQLInvalidAuthorizationSpecException
SQLNonTransientConnectionException
SQLNonTransientException
SQLRecoverableException
SQLSyntaxErrorException
SQLTimeoutException
SQLTransactionRollbackException
SQLTransientConnectionException
SQLTransientException
SQLWarning

aus javax.sql: Interface DataSource

JDBC - DIE DBMS-SEITE

MySQL: Connector/J

Download:

http://dev.mysql.com/downloads/connector/j/

Connection-String:

jdbc:mysql://*servername*:*port*/*database*

DataSource-Implementierung:

com.mysql.jdbc.jdbc2.optional.MysqlDataSource

JDBC - DIE DBMS-SEITE

PostgreSQL

Download:

http://jdbc.postgresql.org/

Connection-String:

jdbc:postgres://*servername*:*port*/*database*

DataSource-Implementierung:

org.postgresql.ds.PGSimpleDataSource

JDBC - DIE DBMS-SEITE

Oracle - mehrere Treiberarten: Thin/OCI/...

Download:

http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

Connection-String (Thin-Treiber):

jdbc:oracle:thin:@*servername*:*port*:*SID*

DataSource-Implementierung:

oracle.jdbc.pool.OracleDataSource

JDBC - DIE DBMS-SEITE

Spezialfall: Apache Derby

reine Java-Implementation eines RDBMS. Durch kleine Größe (ca. 3MB inkl. JDBC-Treiber) zum Einbetten in Java-Applikationen geeignet.

Download:

http://db.apache.org/derby/

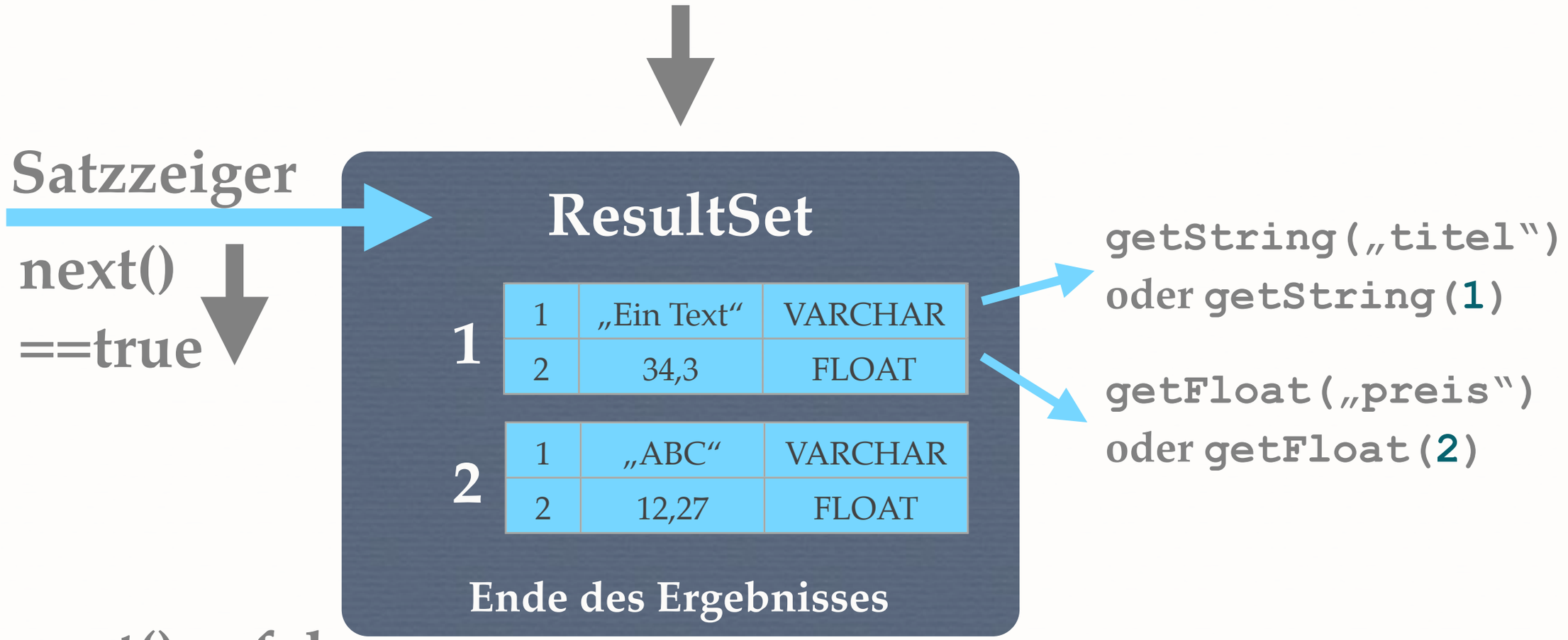
Connection-String:

jdbc:derby:*database*

CURSOR IN JDBC: RESULTSET

Aufbau eines ResultSet

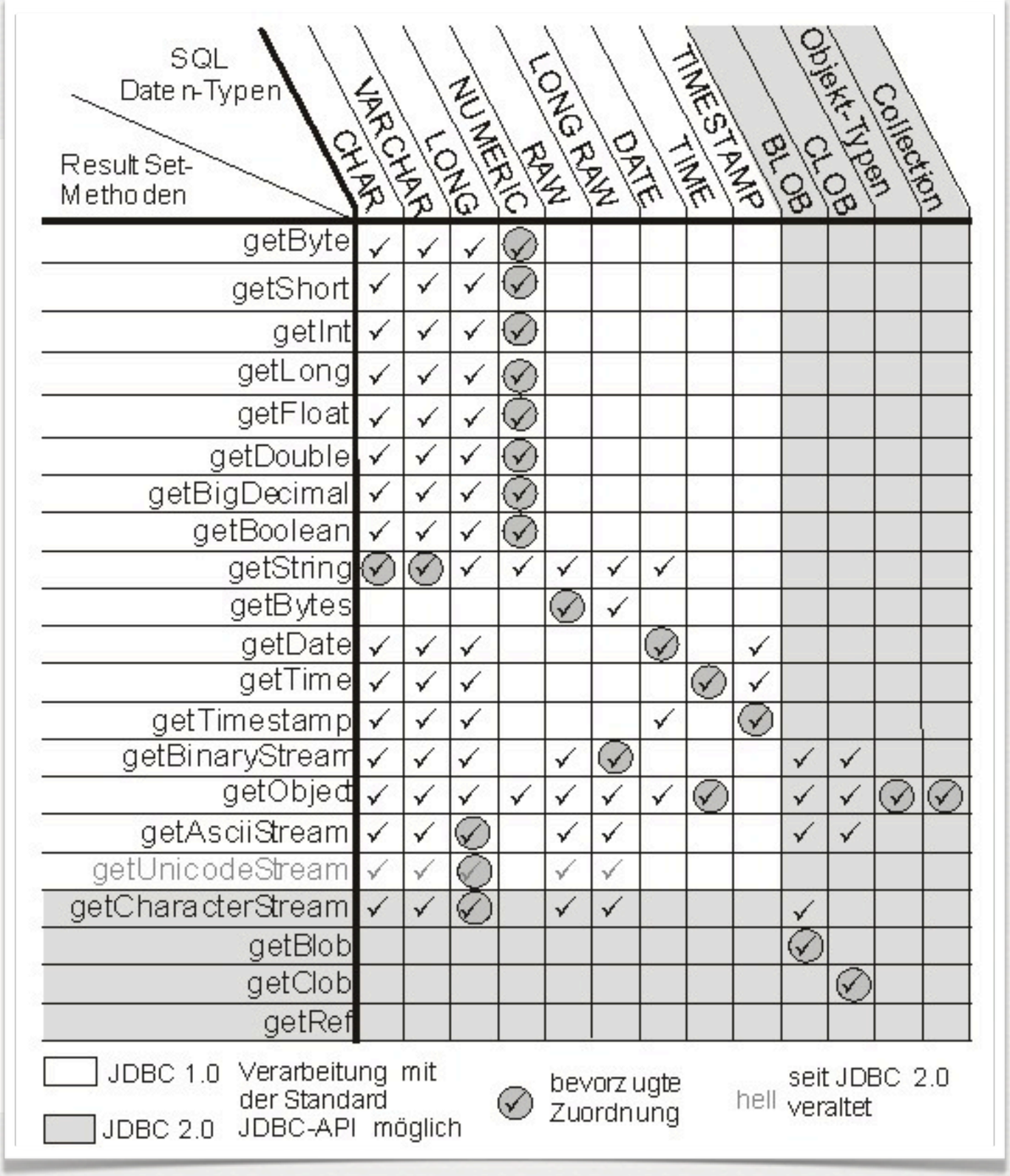
```
stmt.execute(„SELECT titel, preis FROM buecher;“)
              1      2
```



Datentypen

... werden von SQL-Datentypen wo möglich in Java-Datentypen übersetzt bzw. gecastet.

Für Datentypen die es in Java nicht gibt (oder anders funktionieren) definiert das Package java.sql eigene Klassen/ Interfaces.



RESULTSET

Nullwerte

Die Methode **getObject()** liefert **null** zurück, wenn in der Datenbank ein NULL-Wert gespeichert war.

Alle anderen getXXX()-Methoden ersetzen Nullwerte durch **Defaultwerte**: z.B. Zahlen NULL -> 0

Mit der Methode wasNull() kann der zuletzt geholte Wert auf NULL überprüft werden.

```
preis = rs.getFloat(„preis“);
if (rs.wasNull()) {
  System.out.println(„Preis noch nicht gesetzt“);
}
```

STRUKTUR EINER JDBC-ANWENDUNG

```
import java.sql.*;
public class JDBCTest {
 public static void main(String[] args) {
// Datenquelle erzeugen und konfigurieren
    DataSourceClass ds = new DataSourceClass();
    ds.setServerName(SERVER);
    ds.setUser(USER);
    ds.setPassword(PASSWORD);
// Verbindung herstellen
    Connection con = ds.getConnection();
// Abfrage vorbereiten und ausführen
    Statement st = con.createStatement(ERGEBNIS_PARAMETER);
    ResultSet rs = st.executeQuery(SELECT-QUERY);
// Ergebnisse verarbeiten
    while (rs.next()) { // Cursor bewegen
       TYPE wert = rs.getTYPE(ATTRIBUTNAME/INDEX);
    }
// aufräumen
    rs.close(); st.close(); con.close();
} }
```