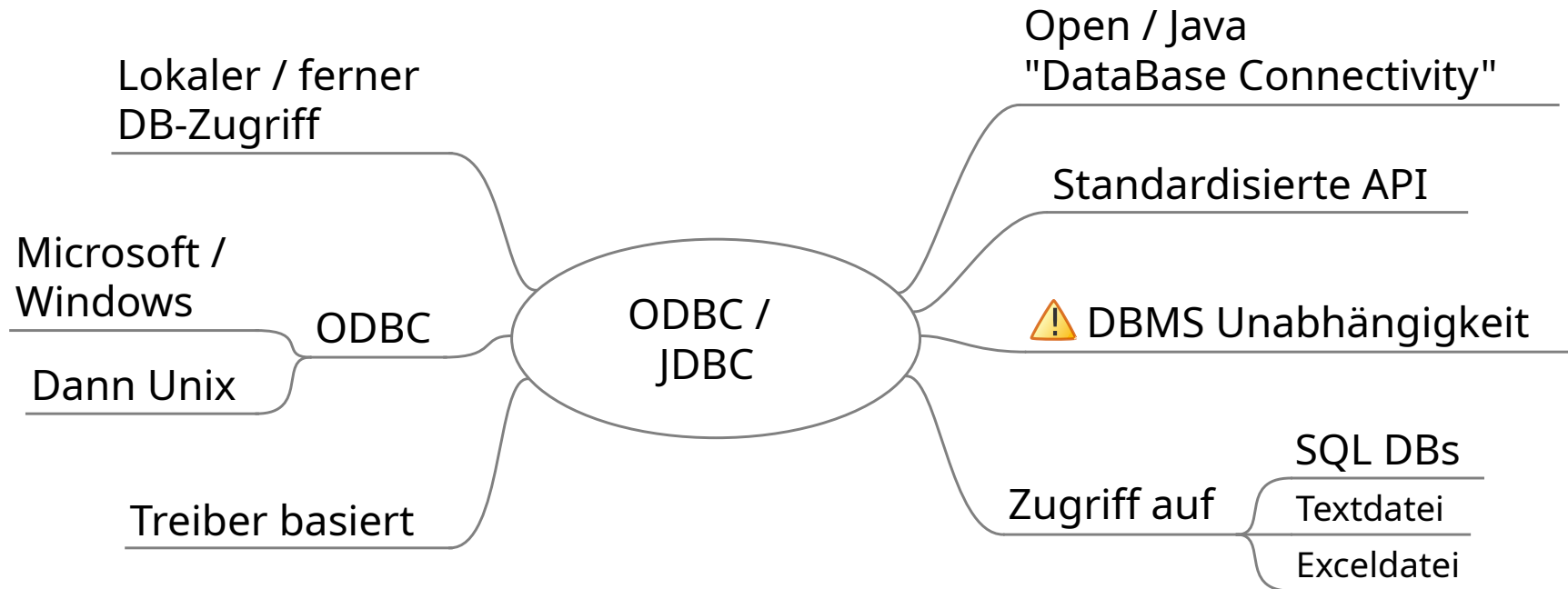


Tag 5

Inhaltsverzeichnis

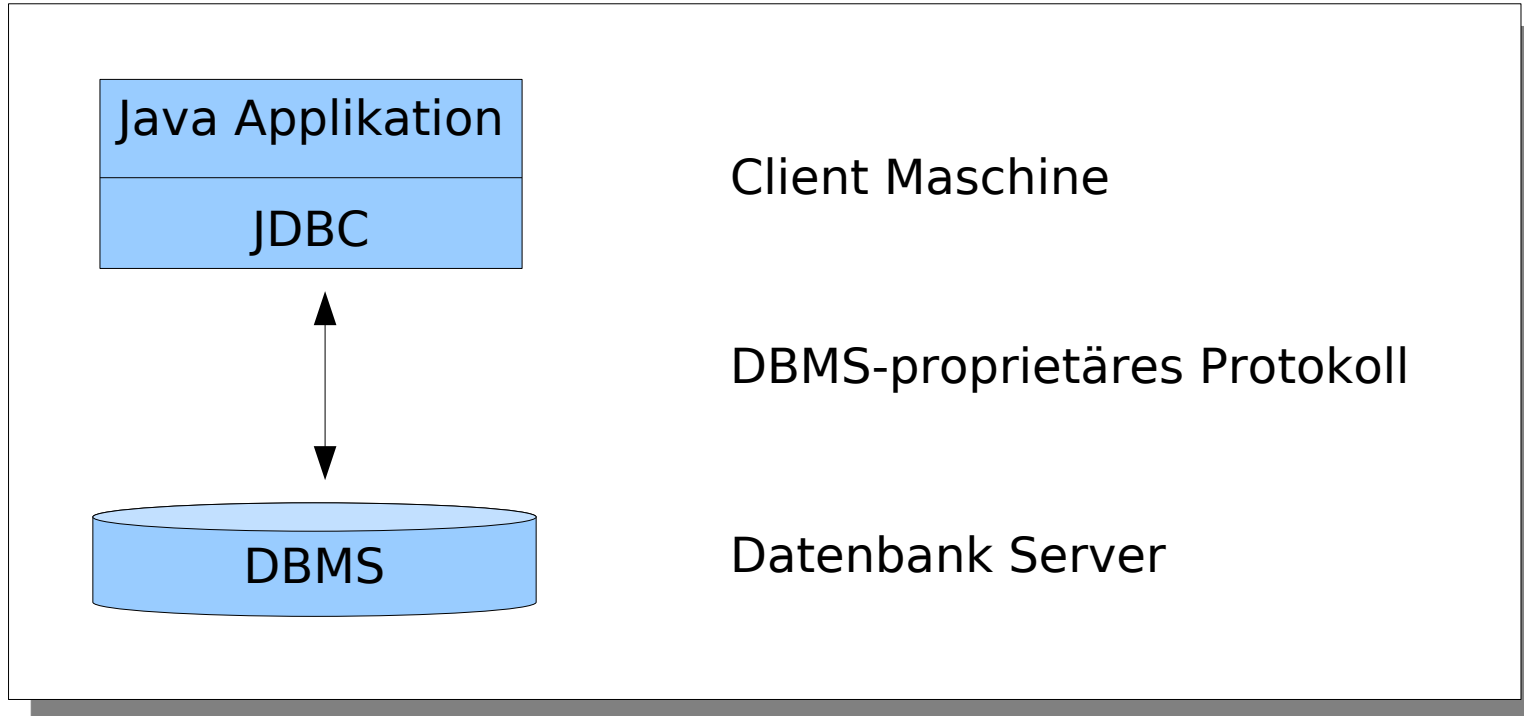
- ODBC / JDBC: Ziel und Prinzip
- JDBC Überblick
- Queries (Execute- und UpdateQuery)
- Der Typ "ResultSet"
- Diverses
 - Metadata
 - PreparedStatement
 - Transaktionen und Batches
- JavaDB
- JPA / ORM
- jOOQ
- Datenbank-Migration über Flyway
- Übungen

ODBC / JDBC Prinzipien



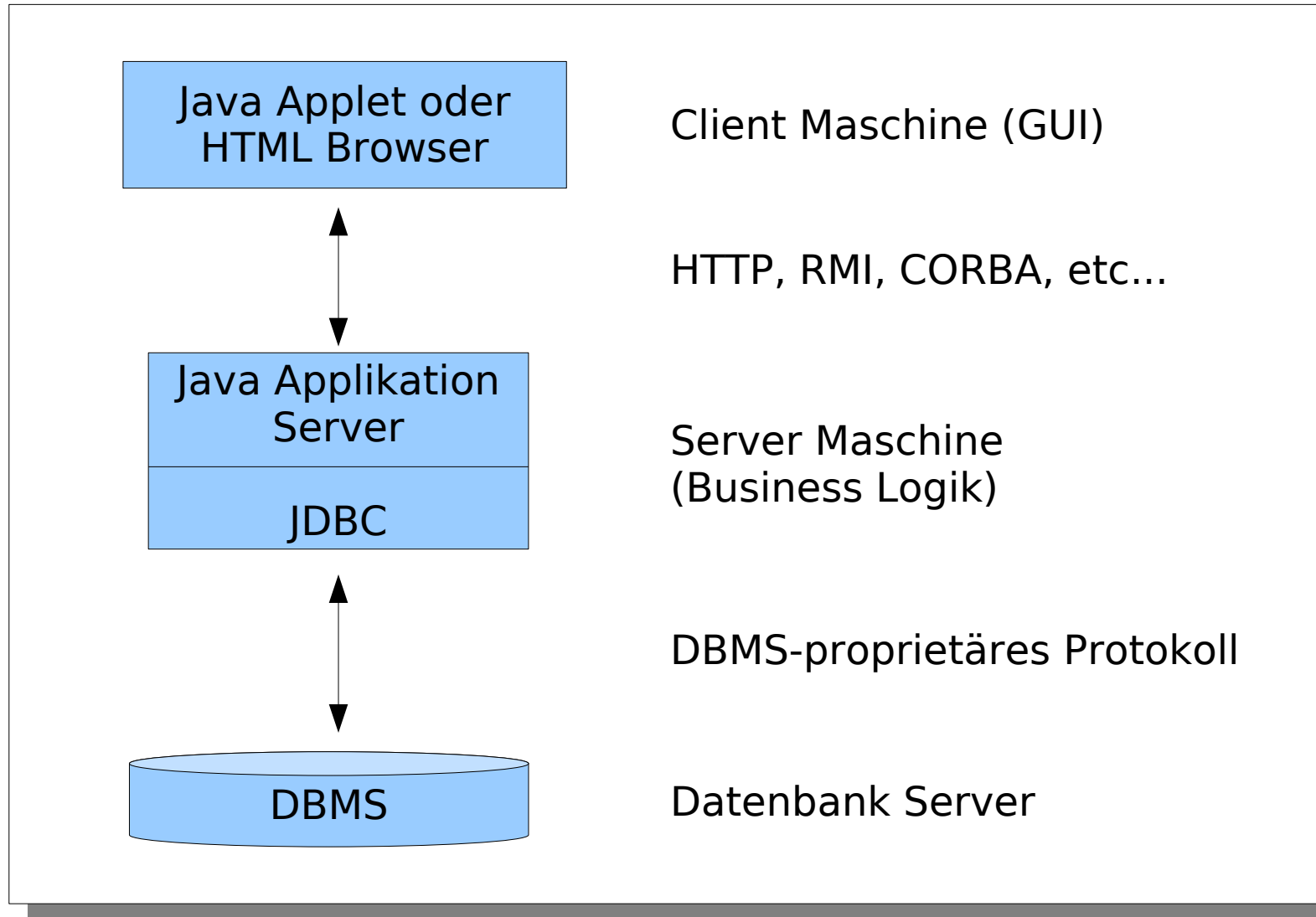
JDBC

"Two-tier" Architektur für Datenzugriff



JDBC

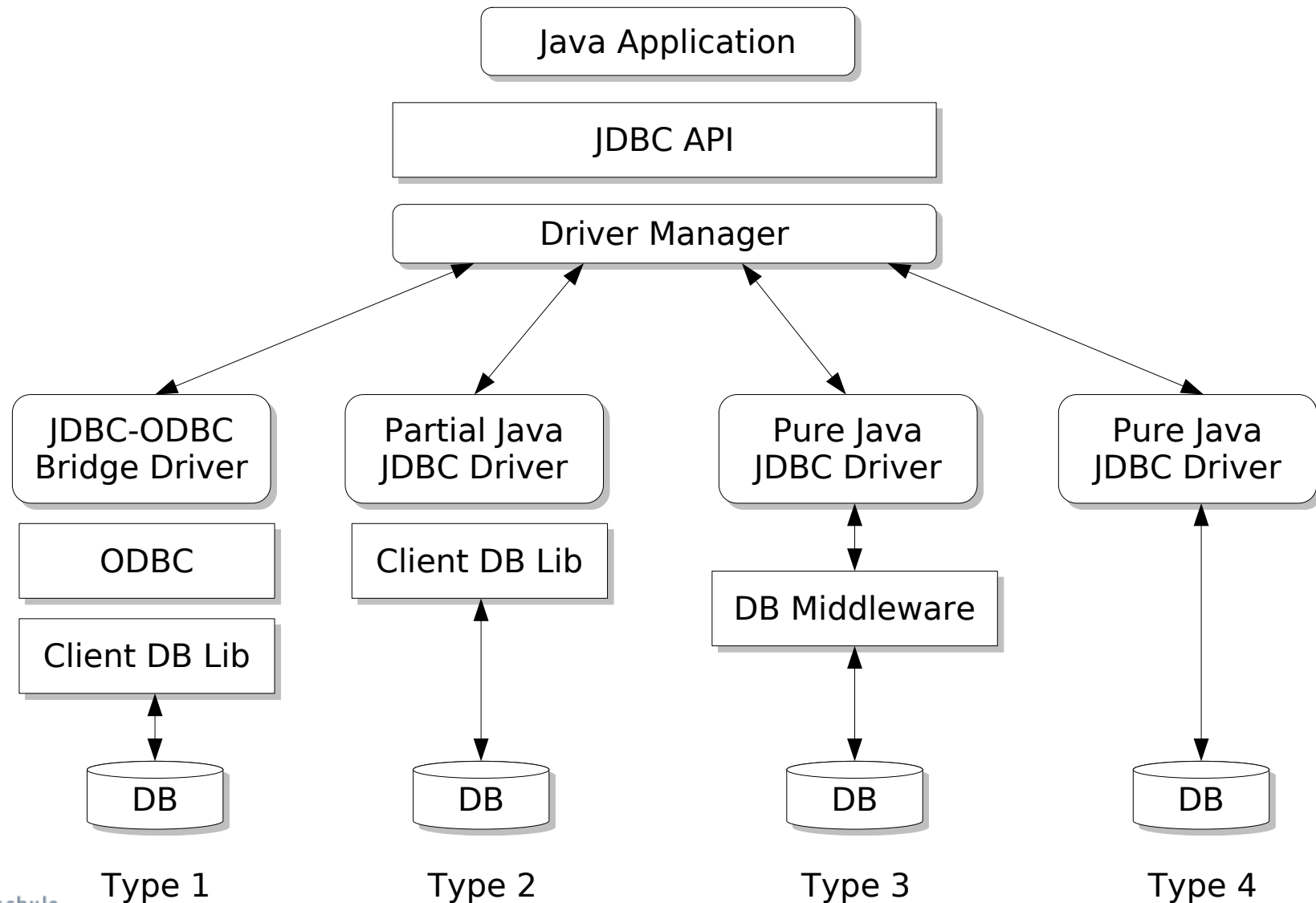
"Three-tier" Architektur für Datenzugriff



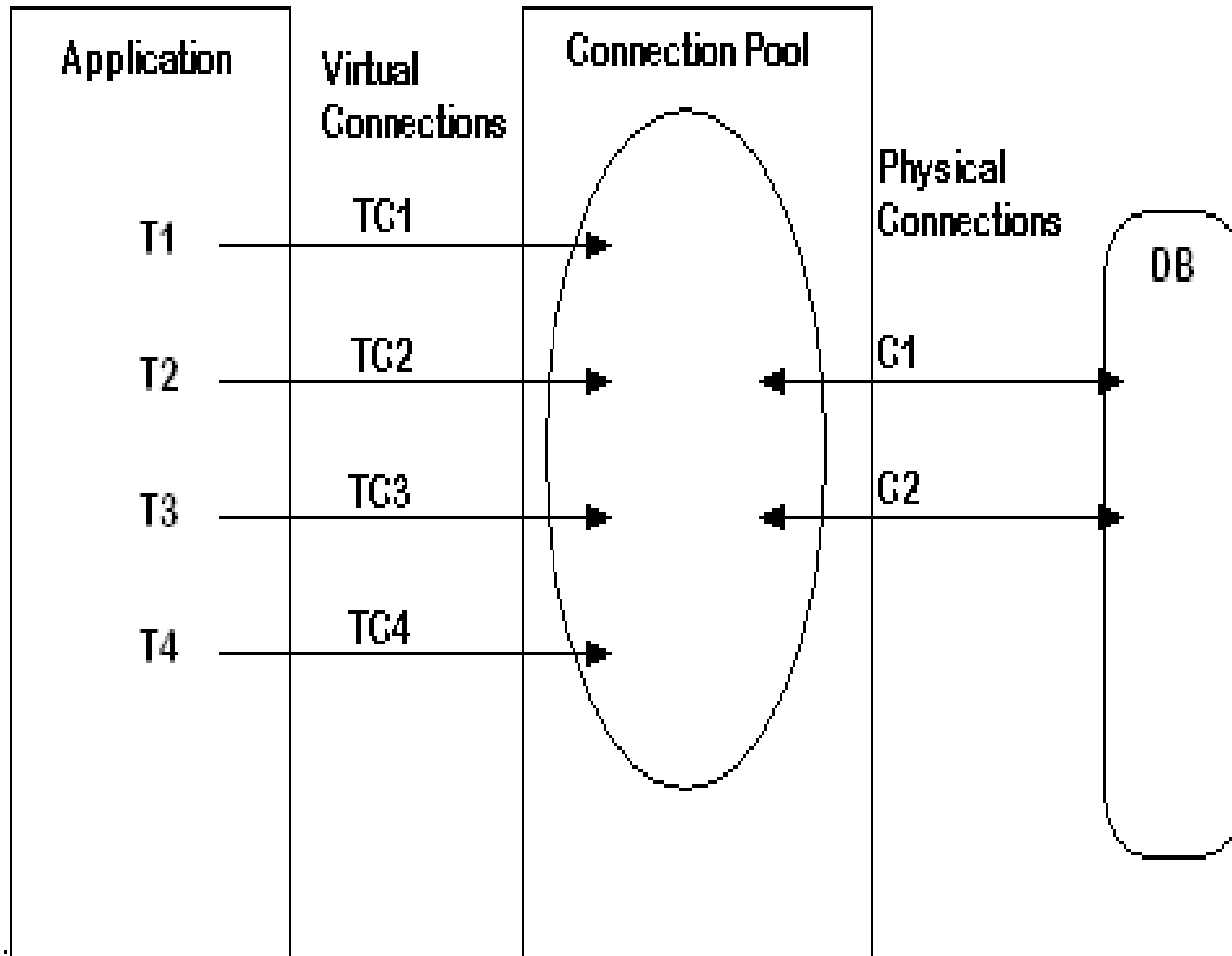
JDBC Komponenten

- JDBC API
 - Teil von JavaSE & JavaEE
- JDBC Driver Manager
 - DriverManager
 - DataSource + Java Naming and Directory Interface (JNDI)
- JDBC Test Suite
- JDBC-ODBC Bridge

JDBC Treiber Typen



JDBC Connection Pool



JDBC Installation

Treiber für MySQL (Connector/J)

- Treiber und Dokumentation auf <https://dev.mysql.com/doc/connector-j/en/connector-j-installing.html>
- Empfohlene Installation durch Maven pom.xml

```
<dependencies>  
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  
  <dependency>  
    <groupId>mysql</groupId>  
    <artifactId>mysql-connector-java</artifactId>  
    <version>8.0.????</version>  
  </dependency>  
</dependencies>
```


JDBC, erstes Beispiel

Main Programm

```
public class JdbcTest {

    private static final Logger LOGGER = Logger.getLogger(JdbcTest.class.getName());
    private static final String CONNECTION_STRING =
        "jdbc:mysql://localhost/gmcd?user=gmaitre&password=*****";

    public static void main(String[] cmdArg) {
        // Try with Resources Block
        // https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html
        try (Connection connection = DriverManager.getConnection(CONNECTION_STRING);
            Statement stm = connection.createStatement()) {
            displayResult(stm);
        } catch (SQLException ex) {
            LOGGER.log(Level.SEVERE, ex.getLocalizedMessage());
        }
    }
}
```

JDBC, erstes Beispiel

Abfrageresultate anzeigen

```
private static void displayResult(Statement stm) throws SQLException { 1usage
    final String QUERY_STR = "select * from cd";
    try (ResultSet rs = stm.executeQuery(QUERY_STR)) {
        while (rs.next()) {
            int cdid = rs.getInt(s: "ID");
            String datum = rs.getString(s: "Datum");
            String beschreibung = rs.getString(s: "Beschreibung");
            System.out.println(MessageFormat.format(
                pattern: "CDID: '{0}', Datum: '{1}', Beschreibung: '{2}'",
                cdid, datum, beschreibung));
        }
    }
}
```

CDID: '1', Datum: '1968-01-13', Beschreibung: 'Yellow Submarine'
CDID: '2', Datum: '1975-01-24', Beschreibung: 'The Koeln Concert'
CDID: '3', Datum: '2009-09-04', Beschreibung: 'Dresden'

JDBC, erstes Beispiel

Welcher Port?

- ◆ `--port=port_num, -P port_num`

Option Sets Variable	Yes, port
Variable Name	port
Variable Scope	Global
Dynamic Variable	No
Value Set	Type numeric
	Default 3306



The port number to use when listening for TCP/IP connections. The port number must be 1024 or higher unless the server is started by the **root** system user.

Siehe auch `/etc/mysql/my.cnf`

JDBC, erstes Beispiel

Installationsprobleme mit IntelliJ

- 1) Passen Sie auf, dass Ihre lokale Firewall die Aktualisierung Ihrer IntelliJ-Umgebung nicht verhindert.
- 2) Schreiben Sie in Ihrer Maven pom.xml Datei die Java JDK-Version, die Sie verwenden (als Property).
Überprüfen:
 - a) Unter Menü Project structure / Project sind die Werte von "Project SDK" sowie "Project Language Level", gleich dem pom.xml Wert.
 - b) Unter Menü Project structure / Modules ist der Wert von "Language Level", gleich dem pom.xml Wert.
- 3) Falls Sie wegen der TimeZone vom MySQL-Server eine Runtime-Exception bekommen, am einfachsten im MySQL Connection-String folgendes anhängen: "&serverTimezone=UTC".

JDBC, erstes Beispiel

Verbindungsprobleme

Häufige Fehler:

- Error 2003: Can't connect to MySQL server
=> Zugriff nicht möglich, siehe my.cnf
(Problem mit Hostname, Port oder DBName?)
- Error 1130: Host 'xyz' is not allowed to connect to this MySQL server
=> Siehe mysql.user, Spalte "hostname"
- Error 1045: Access denied for user 'gugus@xyz'
=> Falscher Name oder Passwort ;-)

JDBC

Java, JDBC und MySQL Typen (1)

Java Data Type	MySQL Column Type
<u>boolean</u> , <u>Boolean</u>	<u>BIT (1)</u>
<u>byte</u> , <u>Byte</u>	<u>BIT (1)</u> to <u>BIT (8)</u> , <u>TINYINT</u>
<u>short</u> , <u>Short</u>	<u>BIT (1)</u> to <u>BIT (16)</u> , <u>SMALLINT</u> , <u>YEAR</u>
<u>int</u> , <u>Integer</u>	<u>BIT (1)</u> to <u>BIT (32)</u> , <u>INT</u>
<u>long</u> , <u>Long</u>	<u>BIT (1)</u> to <u>BIT (64)</u> , <u>BIGINT</u> , <u>BIGINT UNSIGNED</u>
<u>float</u> , <u>Float</u>	<u>FLOAT</u>
<u>double</u> , <u>Double</u>	<u>DOUBLE</u>
<u>java.math.BigDecimal</u>	<u>NUMERIC</u> , <u>DECIMAL</u>
<u>java.math.BigInteger</u>	<u>NUMERIC</u> (precision = 0), <u>DECIMAL</u> (precision = 0)

JDBC

Java, JDBC und MySQL Typen (2)

Java Data Type	MySQL Column Type
<u>Java.util.Date</u>	<u>DATETIME</u> , <u>TIMESTAMP</u> , <u>TIME</u> , <u>DATE</u>
<u>Java.sql.Date</u>	<u>DATE</u>
<u>Java.sql.Time</u>	<u>TIME</u>
<u>Java.sql.Timestamp</u>	<u>DATETIME</u> , <u>TIMESTAMP</u>

Java Data Type	MySQL Column Type
<u>String</u>	<u>CHAR</u> , <u>VARCHAR</u> , <u>TEXT</u>
<u>byte []</u>	<u>BINARY</u> , <u>VARBINARY</u> , <u>BLOB</u>

JDBC Grundlagen

ExecuteQuery

```
/**
 * Executes the given SQL statement, which returns a single
 * ResultSet object.
 */
ResultSet executeQuery(String sql) throws SQLException;

// Beispiel
ResultSet rs = stm.executeQuery
    ("SELECT PersNr, Name FROM Autor");
```


JDBC Grundlagen

ExecuteUpdate

```
/**
 * Executes the given SQL statement, which may be an INSERT,
 * UPDATE, or DELETE statement or an SQL statement that returns
 * nothing, such as an SQL DDL statement.
 * @return either
 *   (1) the row count for (DML) statements
 *   (2) 0 for SQL statements that return nothing
 */
int executeUpdate(String sql) throws SQLException;

// Beispiel
int nbrOfRows = stm.executeUpdate
    ("update autor set vorname='Gilles J.F.' " +
     "where vorname='Gilles';");
```

JDBC Grundlagen

Werte aus einem ResultSet extrahieren

```
ResultSet rs = stm.executeQuery("select persnr, name from autor");
while (rs.next()) {
    int key = rs.getInt("persnr");
    String name = rs.getString("name"); ←
}
rs.close();
```

```
ResultSet rs = stm.executeQuery("select persnr, name from autor");
while (rs.next()) {
    int key = rs.getInt(1);
    String name = rs.getString(2); ←
}
rs.close();
```

Anderer Weg:

- getString + explizite Konvertierung
- getObject + Casting

JDBC Grundlagen

AUTO_INCREMENT Werte finden

```
/**  
 * Retrieves any auto-generated keys created as a result of  
 * executing this Statement object.  
 * If this Statement object did not generate any keys,  
 * an empty ResultSet object is returned.  
 */  
ResultSet getGeneratedKeys() throws SQLException;
```

JDBC Grundlagen

Metadaten

- Wenn "ResultSet" nicht voraussehbar ist
- Klasse "ResultSetMetaData" studieren
- All what you ever wanted to know about a "ResultSet"...
- Es gibt auch eine Klasse *DatabaseMetaData*

JDBC Grundlagen

0, null und NULL

- "executeQuery" gibt ein "ResultSet" zurück
 - Kann leer sein
 - next() verwenden
- In einem "ResultSet" kann ein Element 0 oder NULL sein
 - 0 ist 0 ;-)
 - NULL => mit wasNull() testen

JDBC Grundlagen

Navigation im ResultSet

- next(), previous()
- first(), last(), isFirst(), isLast()
- beforeFirst(), afterLast()
- getRow()
- absolute(int row) // (1 <=> first; -1 <=> last)

JDBC Grundlagen

PreparedStatement (1)

- Wenn Abfrage mehrmals ausgeführt werden muss
- Wird (*eventuell*) vom Server "vor-kompiliert"
=> effizienter
- 1) Abfrageparameter mit "?" setzen
- 2) Parameter mit Werten setzen und Statement ausführen

JDBC Grundlagen

PreparedStatement (2)

```
PreparedStatement ps = con.prepareStatement
    ("select * from autor where vorname= ? ");

ps.setString(1, "Michael");

ResultSet rs = ps.executeQuery();
while (rs.next()) {
    String name = rs.getString("Name");
    System.out.println("Name: " + name);
}
rs.close();
```


JDBC Grundlagen

Transaktionen

- Reihenfolge von zusammengehörigen Operationen
- Wechsel zwischen konsistenten Zuständen
- Muss ACID-Eigenschaften erfüllen

```
con.setAutoCommit(false);
```

```
Statement stm = con.createStatement();
```

```
try {
```

```
    stm.executeUpdate
```

```
        ("insert autor (persnr, name, vorname) " +  
         "values (78, 'Wall', 'Larry')");
```

```
    stm.executeUpdate
```

```
        ("insert autor (persnr, name, vorname) " +  
         "values (90, 'Schneier', 'Bruce')");
```

```
    con.commit();
```

```
} catch (Exception e) {
```

```
    e.printStackTrace();
```

```
    con.rollback(); // Nach Savepoint 'X' wäre möglich
```

JDBC Grundlagen

Batches

Effizienter als einzelne Befehle, *im Prinzip*

```
Statement stm = con.createStatement();
```

```
stm.addBatch("update autor set persnr='21' where persnr='12'");  
stm.addBatch("update autor set persnr='43' where persnr='34'");  
stm.addBatch("update autor set persnr='65' where persnr='56'");  
stm.addBatch("update autor set persnr='87' where persnr='78'");
```

```
int [] changes = stm.executeBatch();
```

```
for (int i = 0; i < changes.length; i++) {  
    System.out.println  
        ("Columns changed by stm: " + i + ": " + changes[i]);  
}
```

JDBC Grundlagen

Batches *und* Transaktionen

```
con.setAutoCommit(false);
```

```
Statement stm = con.createStatement();
```

```
try {  
    stm.addBatch("update autor set persnr='12' where persnr='21'");  
    stm.addBatch("update autor set persnr='34' where persnr='43'");  
    stm.addBatch("update autor set persnr='56' where persnr='65'");  
    stm.addBatch("update autor set persnr='78' where persnr='87'");  
    stm.executeBatch();  
    con.commit();  
} catch (Exception e) {  
    e.printStackTrace();  
    con.rollback();  
}
```

JDBC Grundlagen

Mit BLOBs arbeiten

- In Selbststudium
- Ein gutes Beispiel ist hier zu finden

<http://www.java2s.com/Code/Java/Database-SQL-JDBC/InsertpicturetoMySQL.htm>

JavaDB

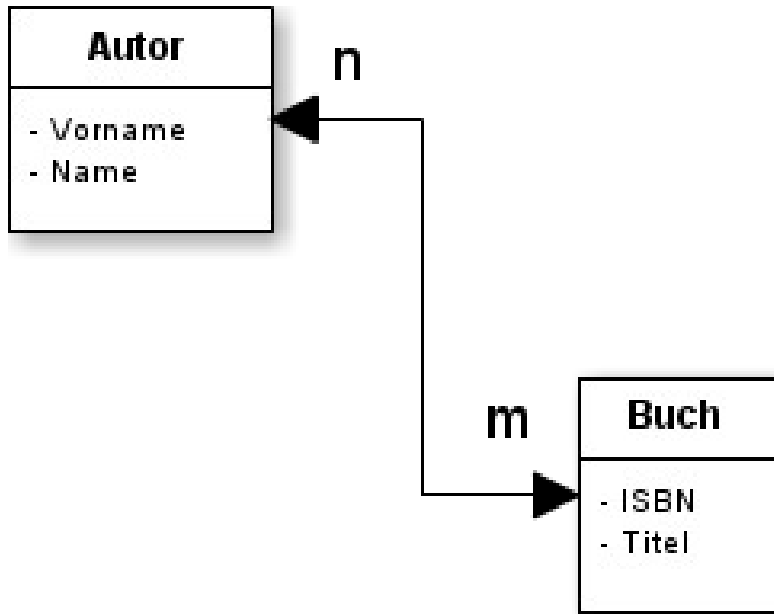
Was ist das?

- Siehe <https://db.apache.org/derby/>
- Praktisch für "kleine" Projekte, die eine SQL-DB brauchen
- JavaDB = Apache Derby (nicht mehr im Java JDK integriert, seit Java 7)
- Tools:
 - `java -jar $DERBY_HOME/lib/derbyrun.jar sysinfo`
 - `java -jar $DERBY_HOME/lib/derbyrun.jar ij`
 - `java -jar $DERBY_HOME/lib/derbyrun.jar dblook -d 'jdbc:derby:firstdb'`

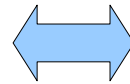
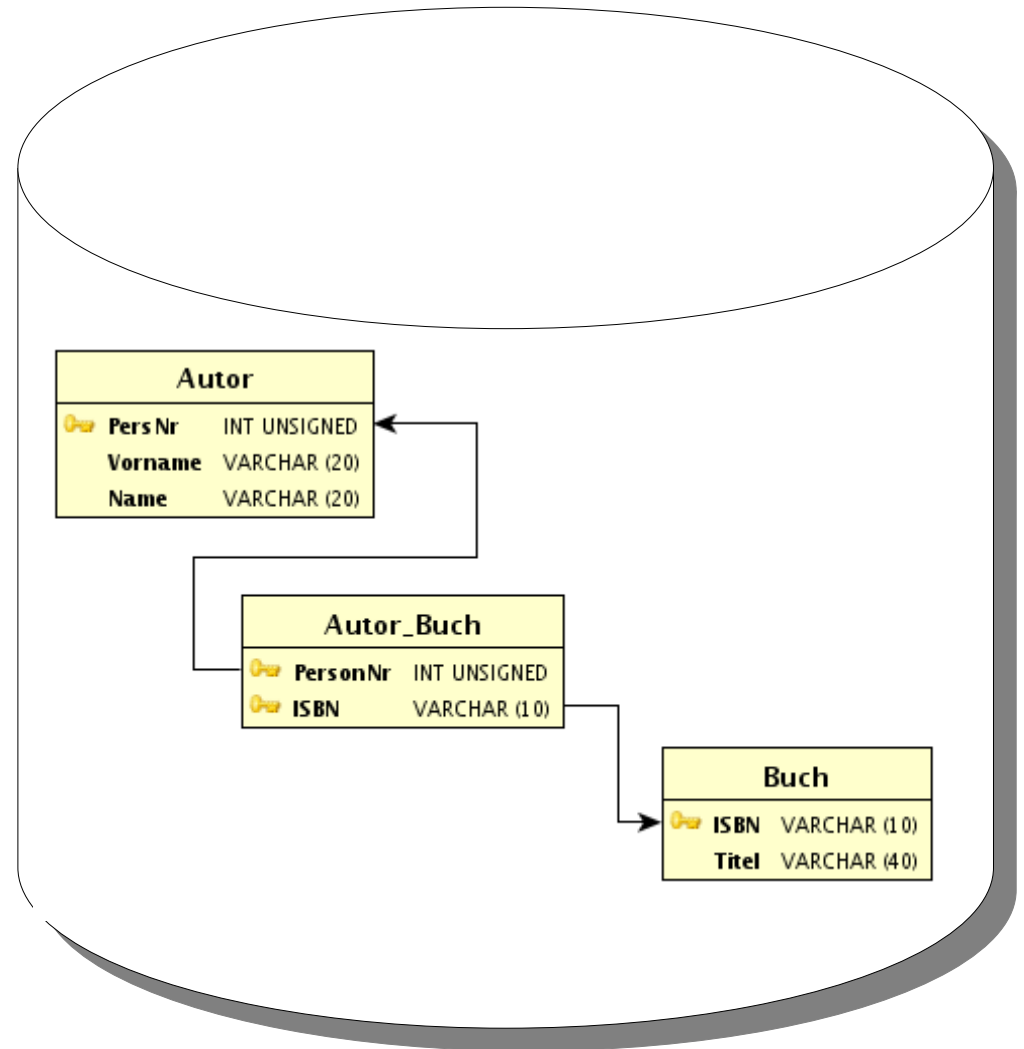
JPA / ORM

Was ist das?

Applikation



Relationale Datenbank



SQL in Java schreiben jOOQ

- Ziel: SQL Code in Java einfach schreiben.
- Ein Weg dazu: jOOQ verwenden.
Siehe <https://www.jooq.org/>
- Beispiel daraus:

```
SELECT AUTHOR.FIRST_NAME, AUTHOR.LAST_NAME, COUNT(*)
FROM AUTHOR
JOIN BOOK ON AUTHOR.ID = BOOK.AUTHOR_ID
WHERE BOOK.LANGUAGE = 'DE'
AND BOOK.PUBLISHED > DATE '2008-01-01'
GROUP BY AUTHOR.FIRST_NAME, AUTHOR.LAST_NAME
HAVING COUNT(*) > 5
ORDER BY AUTHOR.LAST_NAME ASC NULLS FIRST
LIMIT 2
OFFSET 1
```

```
create.select(AUTHOR.FIRST_NAME, AUTHOR.LAST_NAME, count())
    .from(AUTHOR)
    .join(BOOK).on(AUTHOR.ID.equal(BOOK.AUTHOR_ID))
    .where(BOOK.LANGUAGE.eq("DE"))
    .and(BOOK.PUBLISHED.gt(date("2008-01-01")))
    .groupBy(AUTHOR.FIRST_NAME, AUTHOR.LAST_NAME)
    .having(count().gt(5))
    .orderBy(AUTHOR.LAST_NAME.asc().nullsFirst())
    .limit(2)
    .offset(1)
```

Datenbank-Migration Flyway

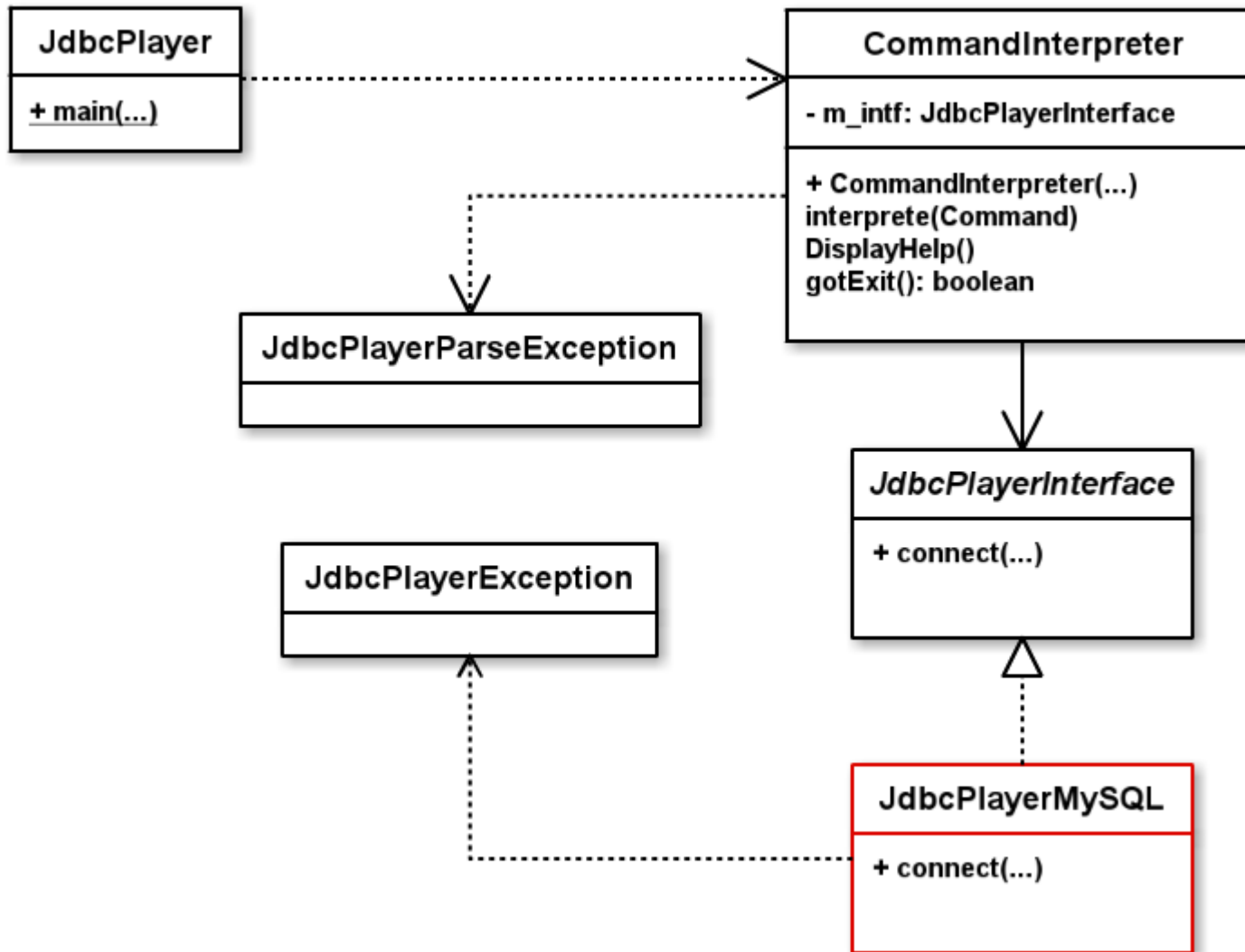
- Ziel: Konsistenz zwischen Code und DB-Schema sicherstellen
- Weg dazu: DB-Schema versionieren und mit dem Code verknüpfen
- Das Tool Flyway hilft
 - Why database migrations?
Siehe <https://flywaydb.org/getstarted/why>
 - How Flyway works?
Siehe <https://flywaydb.org/getstarted/how>

Übungen

- 1) Bringen Sie mein JdbcTest-Programm zum Laufen.
 - a) Den Code finden Sie unter <https://web.mtg1.bfh.science/SD-RDB/JdbcTest.zip>.
 - b) Importieren Sie das Projekt in IntelliJ.
- 2) Erweitern Sie die Klasse "JdbcPlayerMySQL", die die "JdbcPlayerInterface" für GMCD-Datenbank realisiert (siehe nächste Seite).
 - a) Den Code finden Sie unter <https://web.mtg1.bfh.science/SD-RDB/JdbcPlayer.zip>.
 - b) Importieren Sie das Projekt in IntelliJ.
 - c) Kopieren Sie die Klasse "JcbcPlayerMySQLDebug.java" nach "JcbcPlayerMySQL.java" und implementieren Sie die JDBC-Calls, um auf die GMCD Datenbank zuzugreifen.
 - d) Fügen Sie neue Daten in die GMCD-Datenbank mit JdbcPlayer hinzu.

Das Programm JdbcPlayer

UML Klassendiagramm



Das Programm JdbcPlayer

Anwendungsbeispiel

JdbcPlayer Command Interpreter V1.2

Type 'help' to list the commands

JdbcPlayer> *help*

List of available commands (case-insensitive)

connect "<jdbcConnectionString>"

disconnect

show Tables

execute query "<QueryString>"

execute update "<UpdateString>"

exit

help

JdbcPlayer> *connect "jdbc:mysql://localhost/gmcd?user=...&password=..."*

INFO: New connection established

JdbcPlayer> *execute query "select * from autor"*

```
+-----+-----+-----+
| id | datum      | beschreibung      |
+-----+-----+-----+
|  1 | 1968-01-13 | Yellow Submarine  |
|  2 | 1975-01-24 | The Koeln Concert |
|  3 | 2009-09-04 | Dresden           |
+-----+-----+-----+
```

JdbcPlayer> *exit*