Datenbank-Werkzeuge

Thema:	Datenbank-Werkzeuge
Autor:	Prof. Dr. Stephan Kleuker
Version/Datum:	1.1 / 7.11.2024
Empfänger:	Teilnehmer der Lehrveranstaltung Datenbanken

Inhaltsverzeichnis

0 KleukersSEU

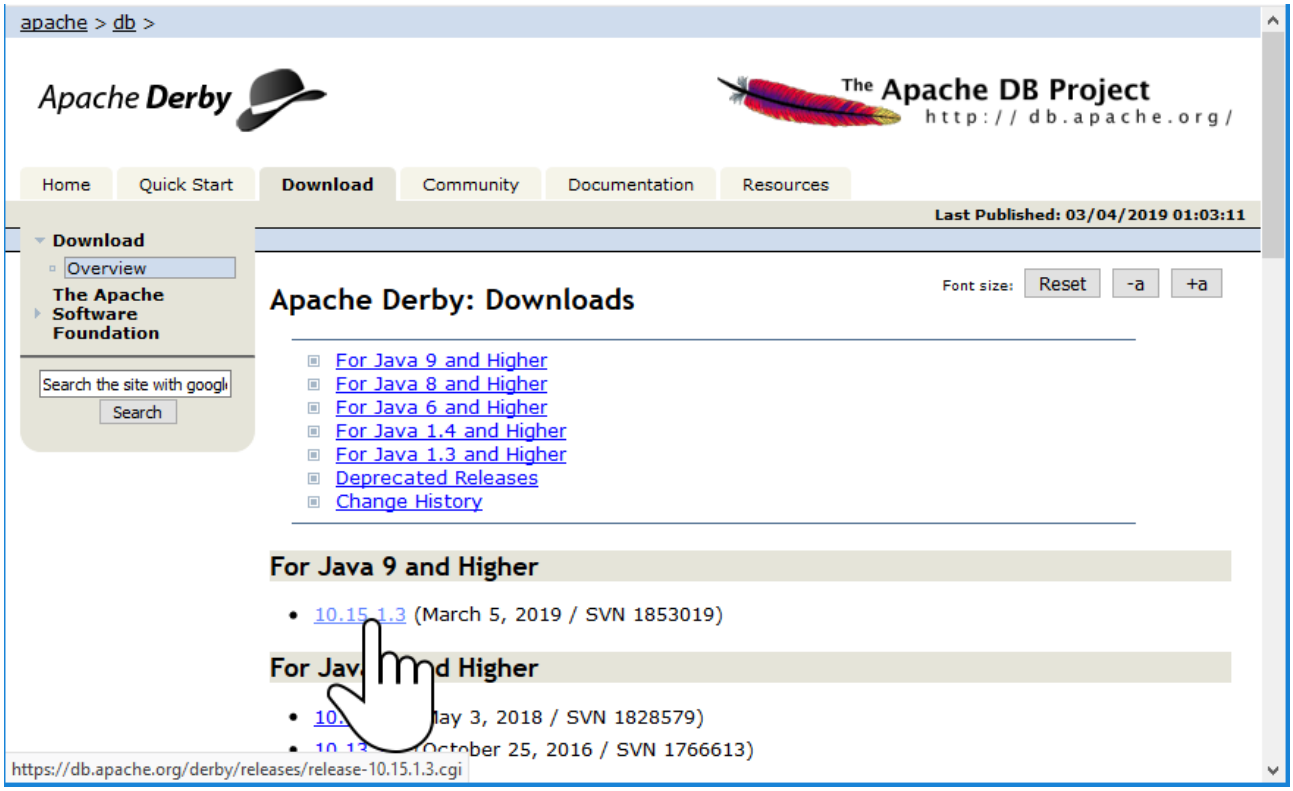
Um Konflikte mit anderen Software-Paketen zu vermeiden, bietet Prof. Dr. Kleuker Teilnehmern seiner Veranstaltungen ein Verzeichnis an, dass alle in seinen Veranstaltungen benötigte Software beinhaltet. Dieses Verzeichnis ist auf den Hochschulrechnern installiert und kann auf eigene Rechner oder einfach einen USB-Stick kopiert werden. Es ist dabei zu beachten, dass Einstellungen, die im Nutzerkonto gespeichert werden, auf jedem Rechner neu einzurichten sind. Weitere Hinweise und das Download-Paket können http://home.edvsz.hs-osnabrueck.de/skleuker/kleukersSEU/index.html entnommen werden.

1 Nutzung von Derby

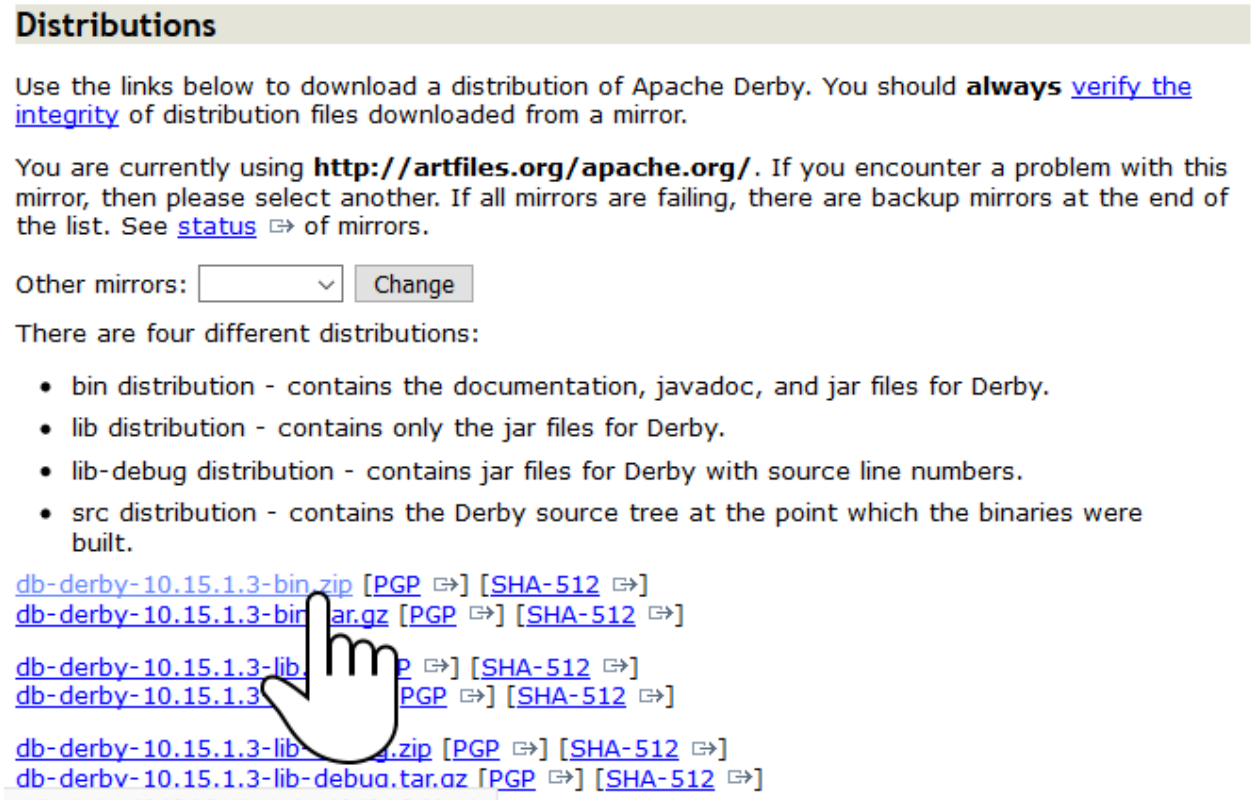
1.1 Installation von Derby

Die Installation beginnt mit einem Klick auf den Download-Reiter auf der Webseite https://db.apache.org/derby/.

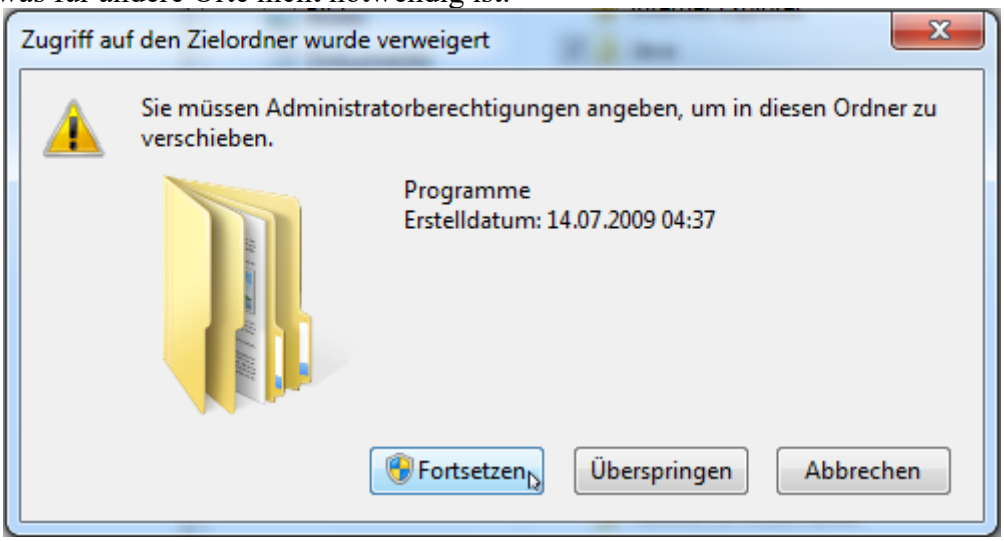
Es wird eine aktuelle zur installierten Java-Version passende Derby-Version angeklickt.



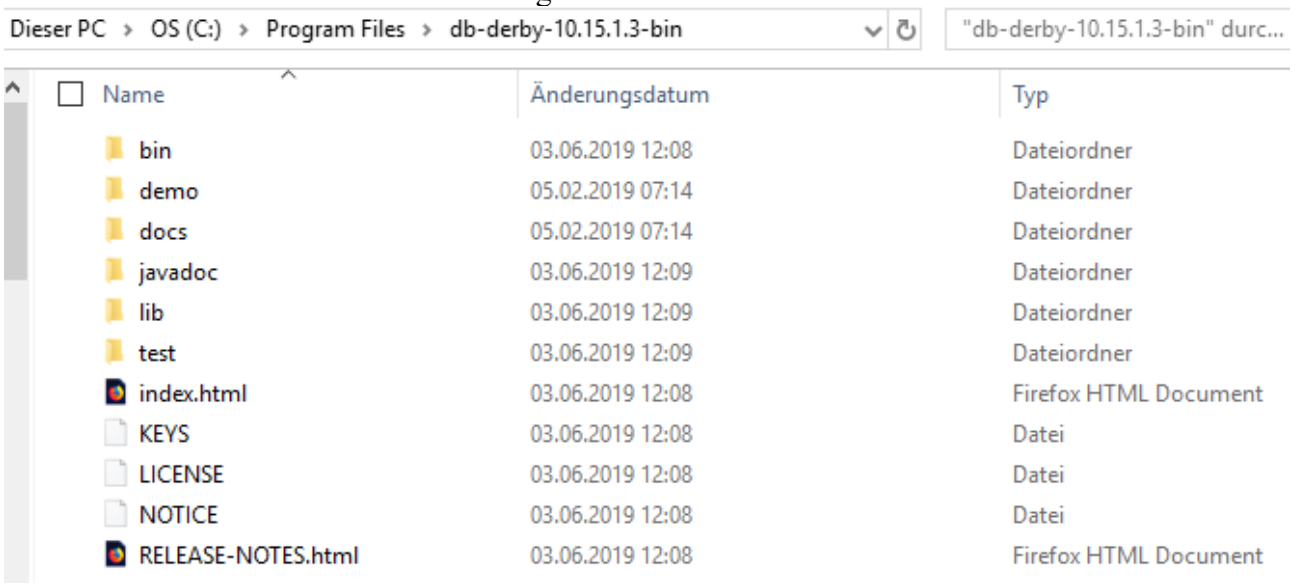
Es wird die aktuelle …bin.zip heruntergeladen.



Die Zip-Datei wird ausgepackt und das entstehende Verzeichnis an einen sinnvollen Ort geschoben, hier z. B. C:\Programme (oder C:\Program Files). Für diesen Zielort müssen Administatorrechte vorliegen, was für andere Orte nicht notwendig ist.



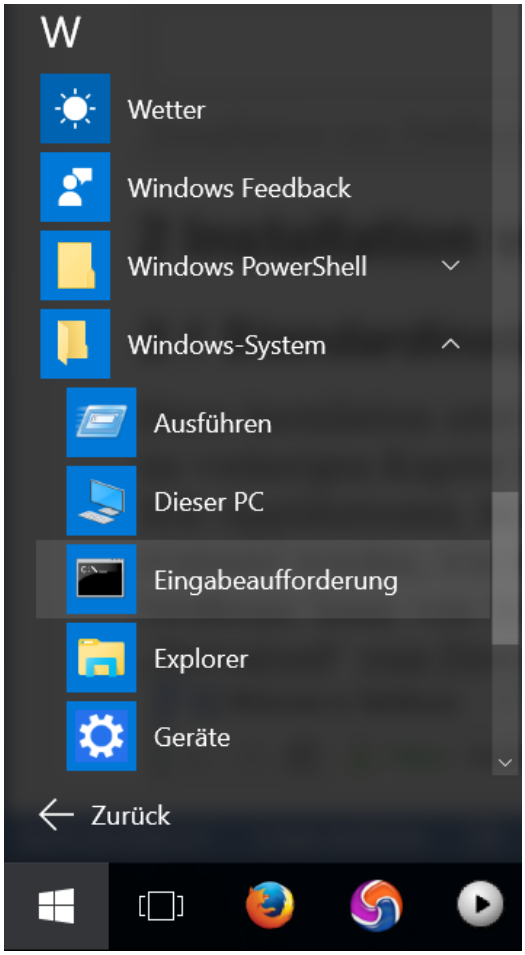
Das entstehende Verzeichnis sieht wie folgt aus.



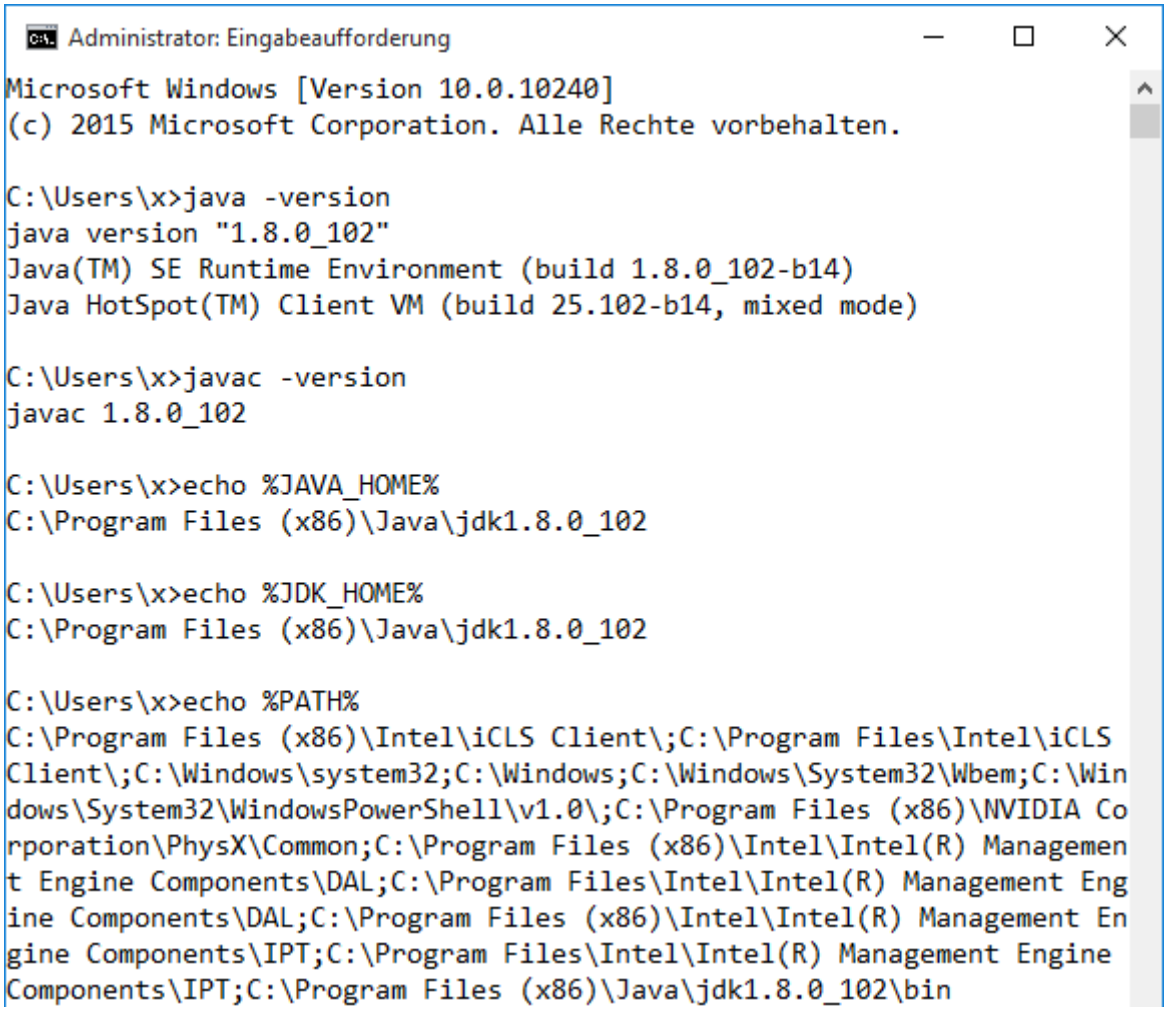
Soll intensiver direkt mit Derby gearbeitet werden, ist die Aufnahme des bin-Verzeichnisses in die PATH-Variable von Windows sinnvoll, wird aber sonst nicht benötigt. Bei direkter Nutzung muss die Systemvariable DERBY_HOME auf das Installationsverzeichnis gesetzt werden.

1.2 Überprüfung der Java-Version

Zur Validierung, welche Java-Version installiert ist, kann ein Konsolen-Fenster genutzt werden, das unter Windows 10 unter „Alle Apps“, „Windows-System“ und „Eingabeaufforderung“ erreichbar ist. Die Windows-PowerShell ist genauso nutzbar.



Die Ausgaben zu den Aufrufen `java –version` und `javac –version` geben die installierten Java-Versionen an. Sollte es eine Fehlermeldung geben, ist Java nicht korrekt installiert, was z. B. an einer Fehlerhaften PATH-Variablen liegen kann. Die PATH-Variable ist z. B. mit `echo %PATH%` ausgebbar.



1.3 Start und Stopp der Datenbank

Zur Nutzung muss Java installiert und in der PATH-Variablen eingetragen sein. Zum Starten wird in einem Konsolenfenster in das bin-Verzeichnis von Derby gesteuert und dort

```
startNetworkServer.bat -noSecurityManager
```

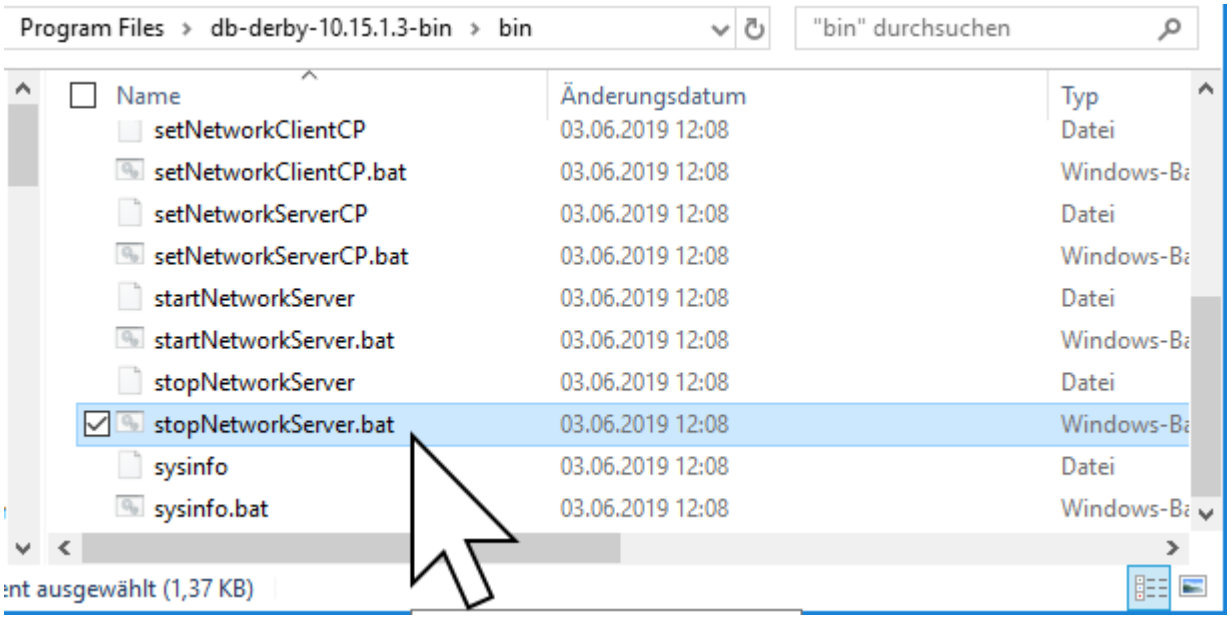
aufgerufen. Der Parameter ist relevant, da im späteren Verlauf Trigger für die Datenbank geschrieben werden sollen, die in die Datenbank integriert werden. Alternativ wären die Security-Einstellungen der Java-Installation anzupassen. Vor der Derby-Version 10.15 war der Parameter nicht notwendig. Für den Startbefehl kann eine eigene Batch-Datei, z. B. mit Verknüpfung zur Oberfläche angelegt werden.

```
Eingabeaufforderung - startNetworkServer.bat  -noSecurityManager
(c) 2018 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\skleuker>cd "\Program Files\db-derby-10.15.1.3-bin\bin"

C:\Program Files\db-derby-10.15.1.3-bin\bin>startNetworkServer.bat -noSecurityManager
Mon Jun 03 13:19:23 CEST 2019 Thread[main,5,main] java.io.FileNotFoundException: derby.
log (Zugriff verweigert)
Mon Jun 03 13:19:23 CEST 2019 : Apache Derby Network Server 10.15.1.3 - (1853019) wurde
 gestartet und ist bereit, Verbindungen auf Port 1527 zu akzeptieren.
Mon Jun 03 13:19:23 CEST 2019 : Apache Derby Network Server 10.15.1.3 - (1853019) wurde
 gestartet und ist bereit, Verbindungen auf Port 1527 zu akzeptieren.
```

Bei einer Meldung der folgenden Form ist die installierte Java-Version zu alt für die installierte Derby-Version. Java sollte dann aktualisiert werden.

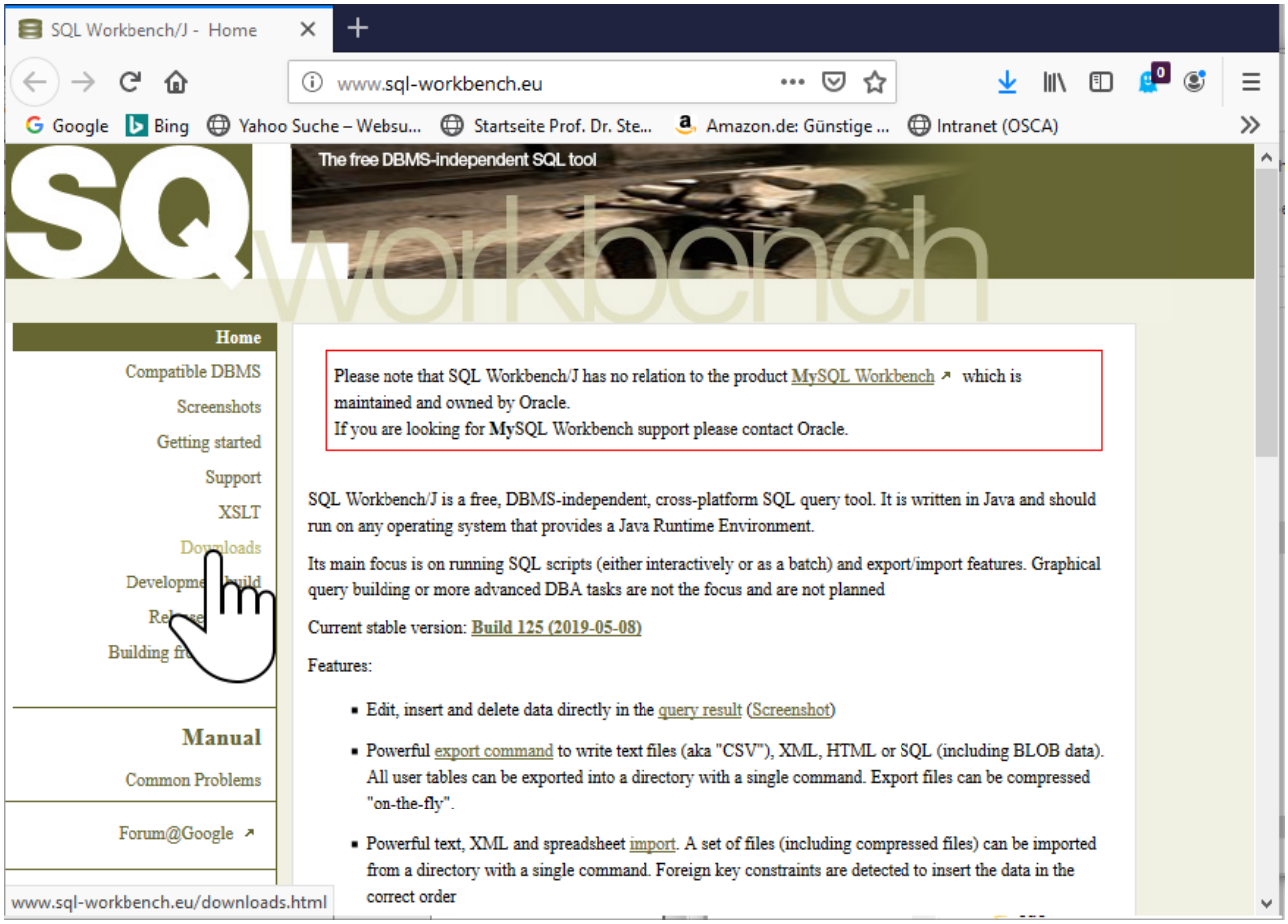
```
C:\Program Files\db-derby-10.15.1.3-bin\bin>startNetworkServer.bat -noSecurityManager
Error: A JNI error has occurred, please check your installation and try again
Exception in thread "main" java.lang.UnsupportedClassVersionError: org/apache/derby/drda/NetworkServe
rControl has been compiled by a more recent version of the Java Runtime (class file version 53.0), th
is version of the Java Runtime only recognizes class file versions up to 52.0
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:763)
        at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
```

Das Konsolenfenster darf nicht geschlossen werden, da sonst auch die Datenbank geschlossen wird. Sauberer wird die Datenbank durch die Ausführung von „stopNetworkServer.bat“ im bin-Verzeichnis beendet. Danach kann das Konsolenfenster geschlossen werden.

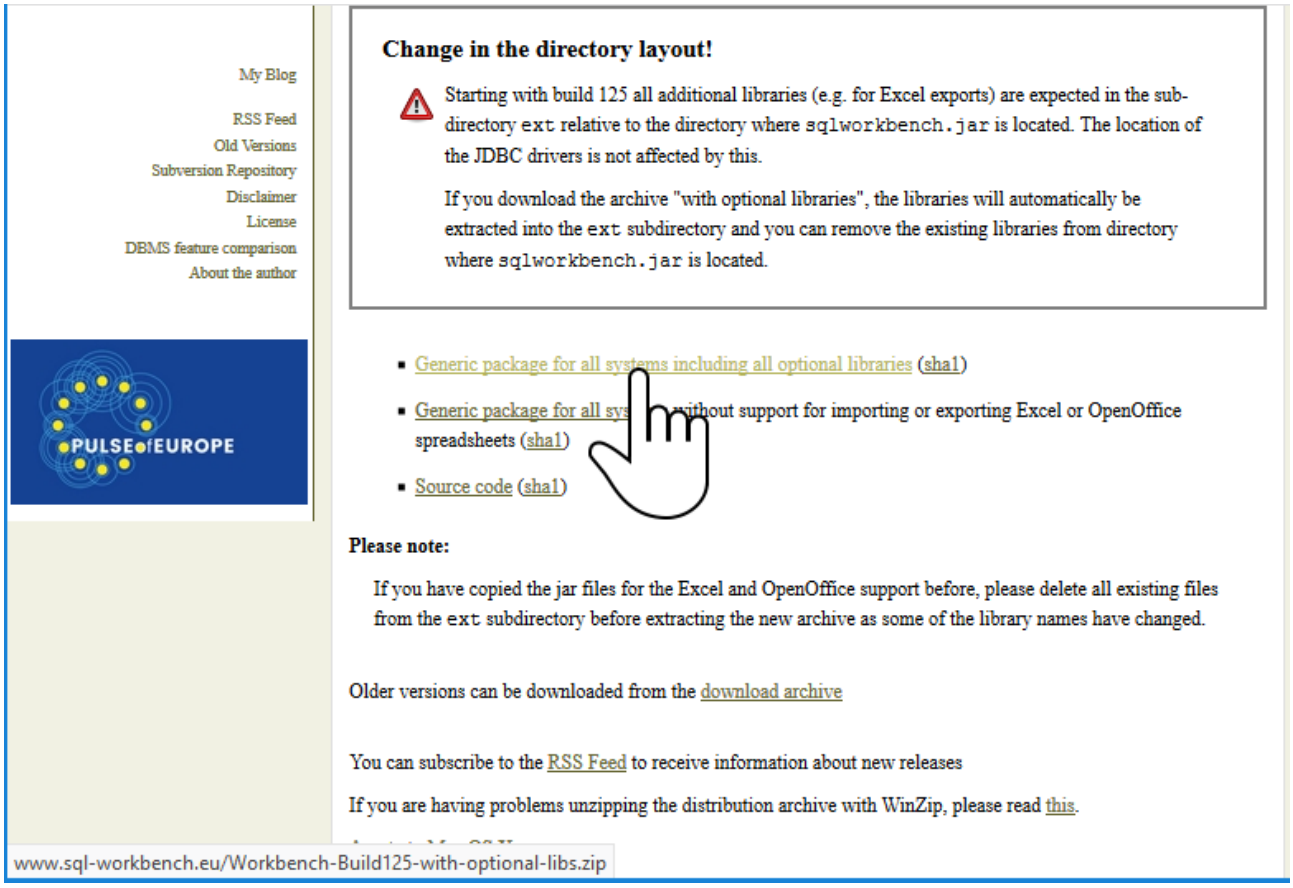


1.4 Installation von SQL-Workbench

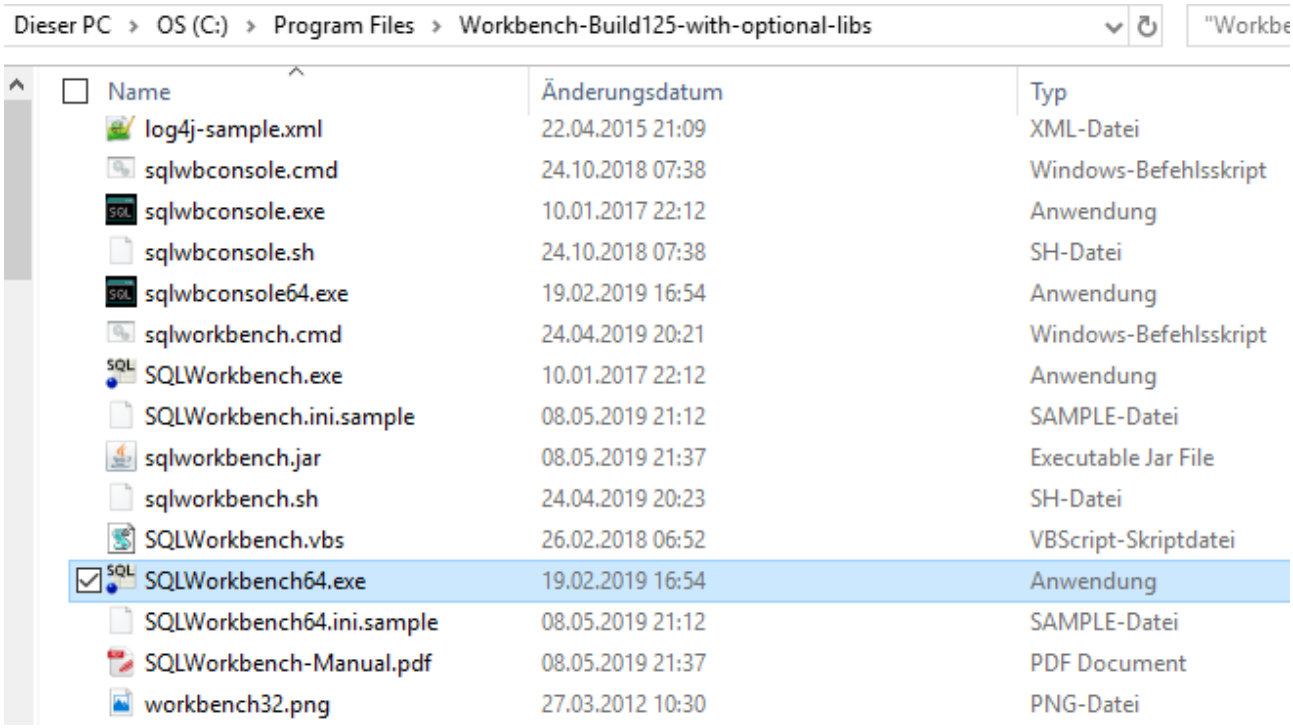
SQL Workbench / J (http://www.sql-workbench.net/) ist ein Datenbankwerkzeug, mit dem man sich u. a. recht einfach über existierende Tabellen und deren Inhalte informieren kann, was auch im Fokus dieser Beschreibung steht. Zum Download wird der Download-Link genutzt.



Es wird weiter unten die Variante „Generic package for all systems including all optional libraries“ heruntergeladen.

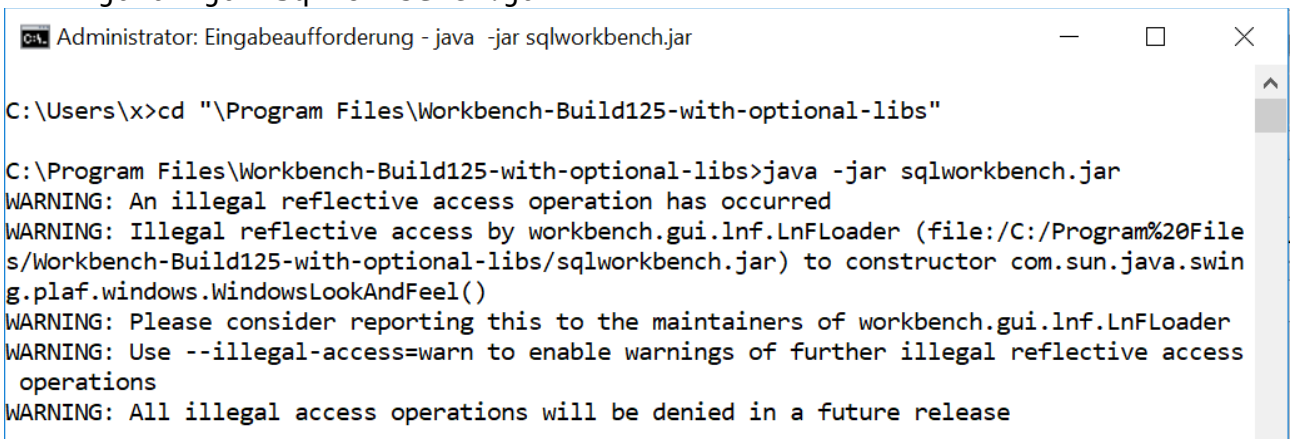


Das zip-Verzeichnis wird irgendwo, z. B. unter C:\Program Files oder C:\Programme ausgepackt. Bei diesen Verzeichnissen werden Administrator-Rechte benötigt, bei einem Verzeichnis, das man lesen und beschreiben kann, nicht. Eine Installation auf einem USB-Stick ist möglich.

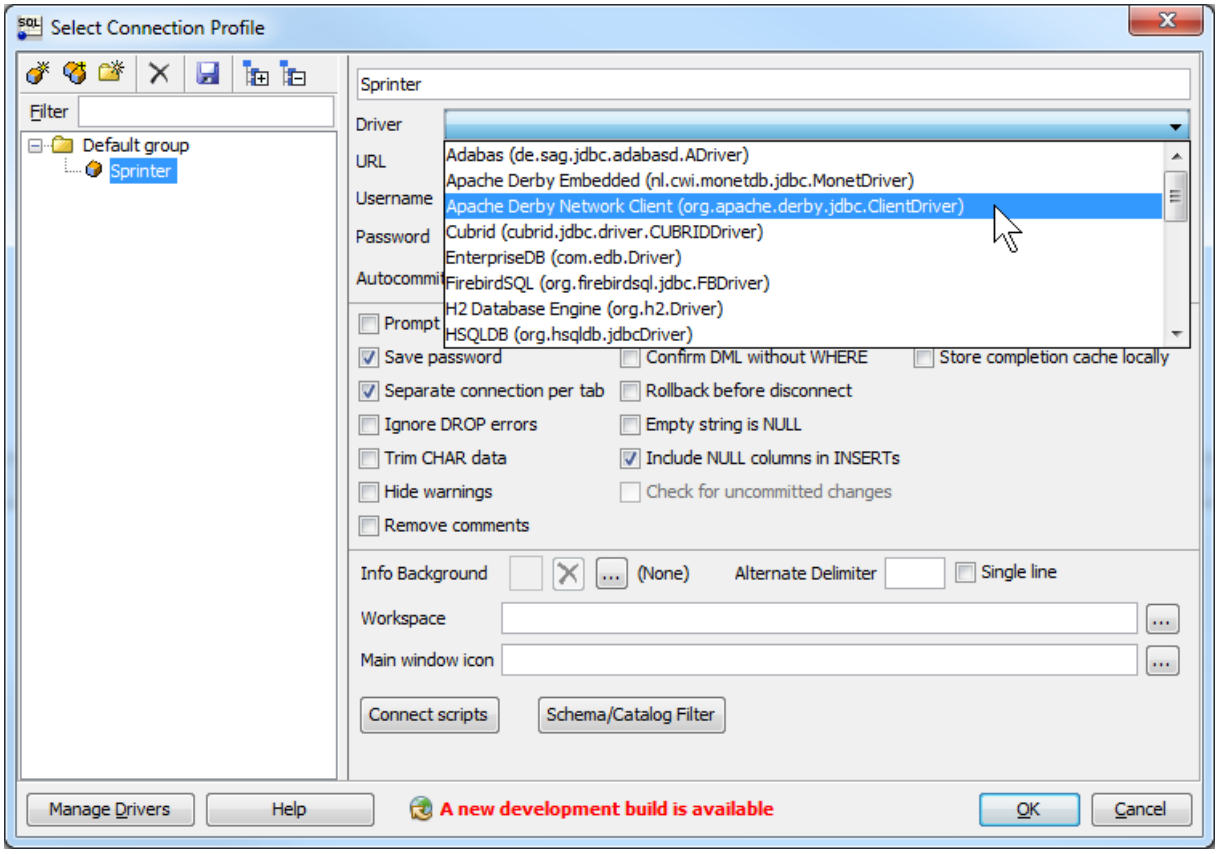


Der Start erfolgt über ein Konsolenfenster, die Nutzung von SQLWorkbench64.exe hatte zumindest Probleme mit einzelnen Java-Versionen. Der Aufruf kann natürlich auch in einer Batch-Datei stehen.

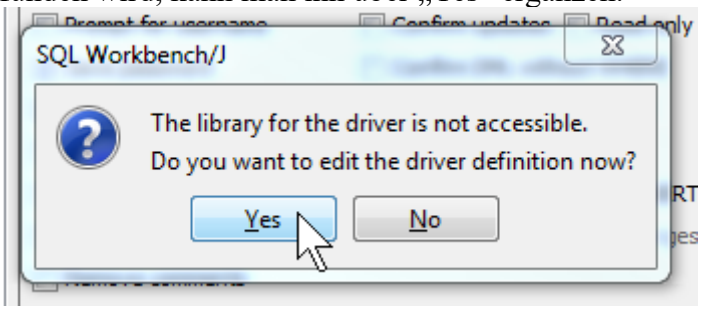
```
java -jar sqlworkbench.jar
```



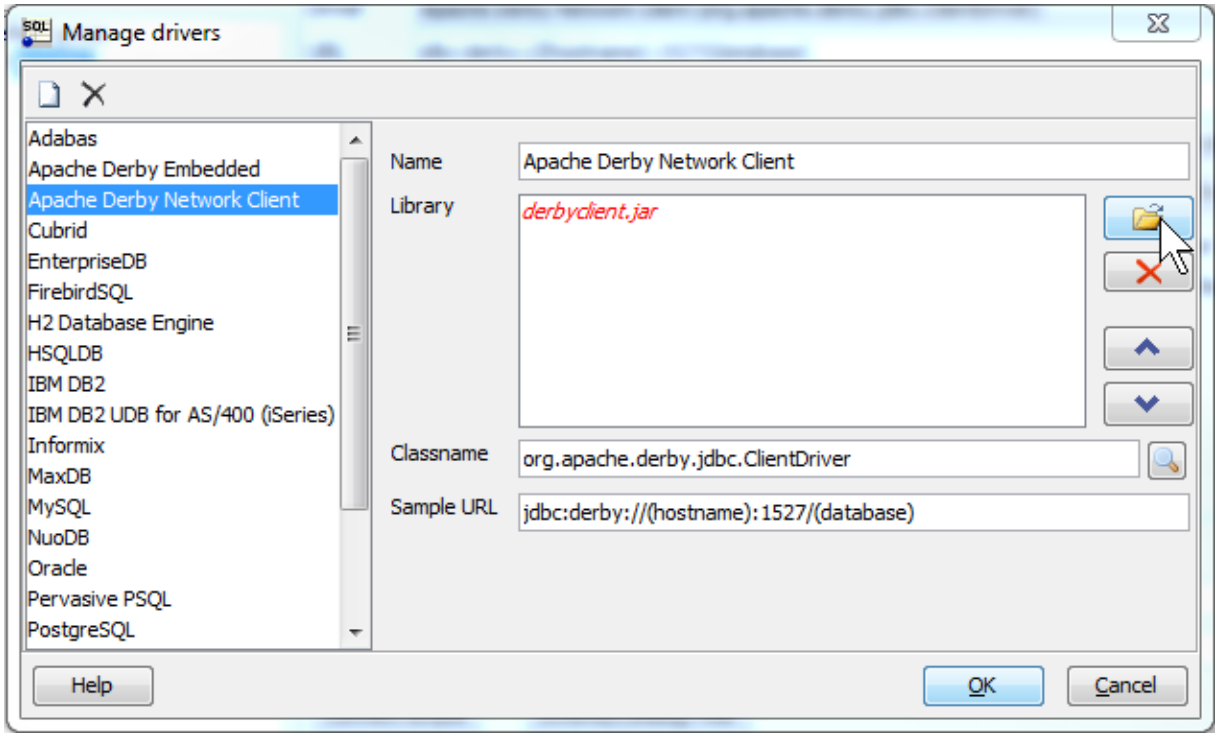
Nach dem ersten Start muss eine Verbindung aufgebaut werden. Dazu sollte bekannt sein, wo sich der JDBC-Treiber der genutzten Datenbank befindet. Das ist bei Derby das lib-Verzeichnis der Installation, benötigt wird derbyclient.jar. Da Derby in der Version 10.15 an das MNodul-Konzept von Java angepasst wurde, empfiehlt es sich die Bibliotheken derbyshared.jar und derbytools.jar immer mitzukopieren. Man klappt zunächst das Feld Driver aus und wählt die genutzte Datenbank „Apache Derby Network Client“.



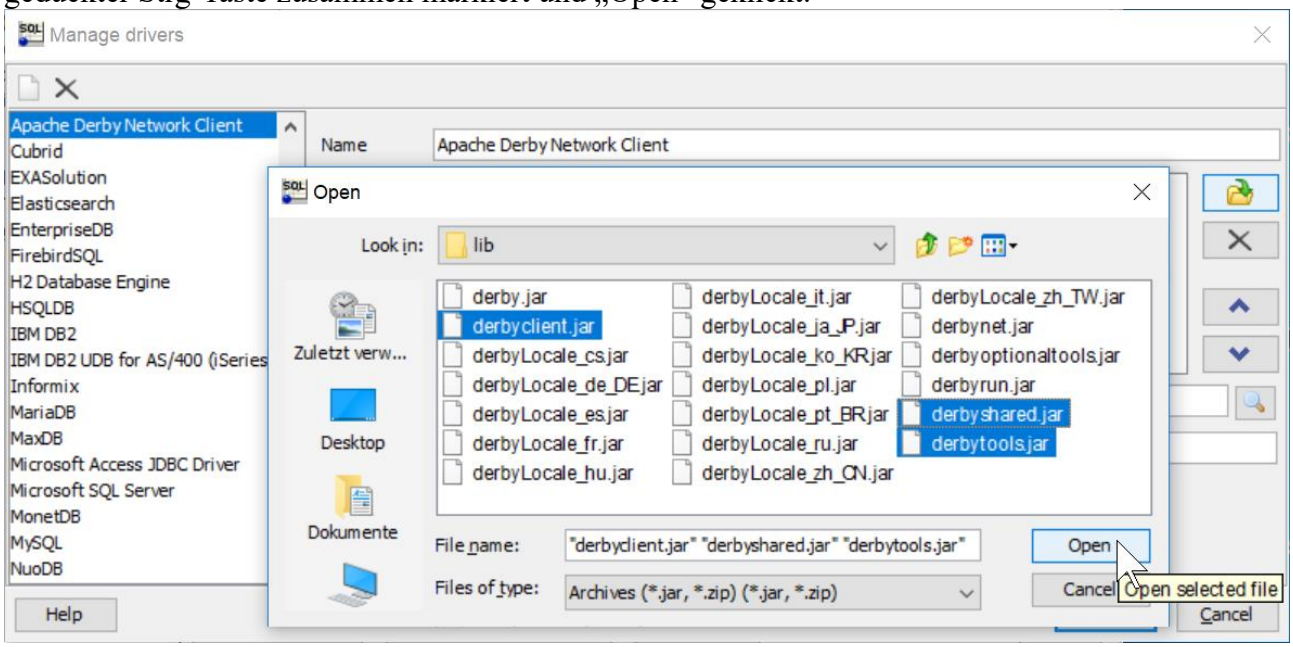
Da der Treiber nicht gefunden wird, kann man ihn über „Yes“ ergänzen.



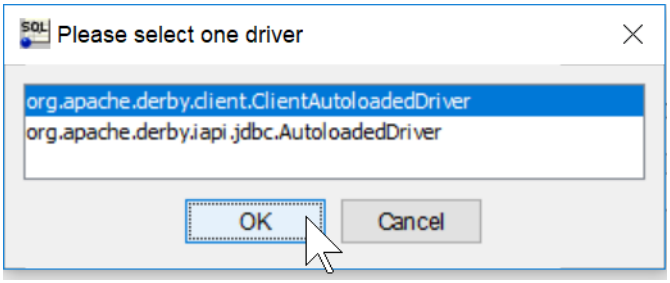
Über das Ordner-Symbol rechts-oben wird dann zum passenden JDBC-Treiber mit den zusätzlichen Dateien manövriert.



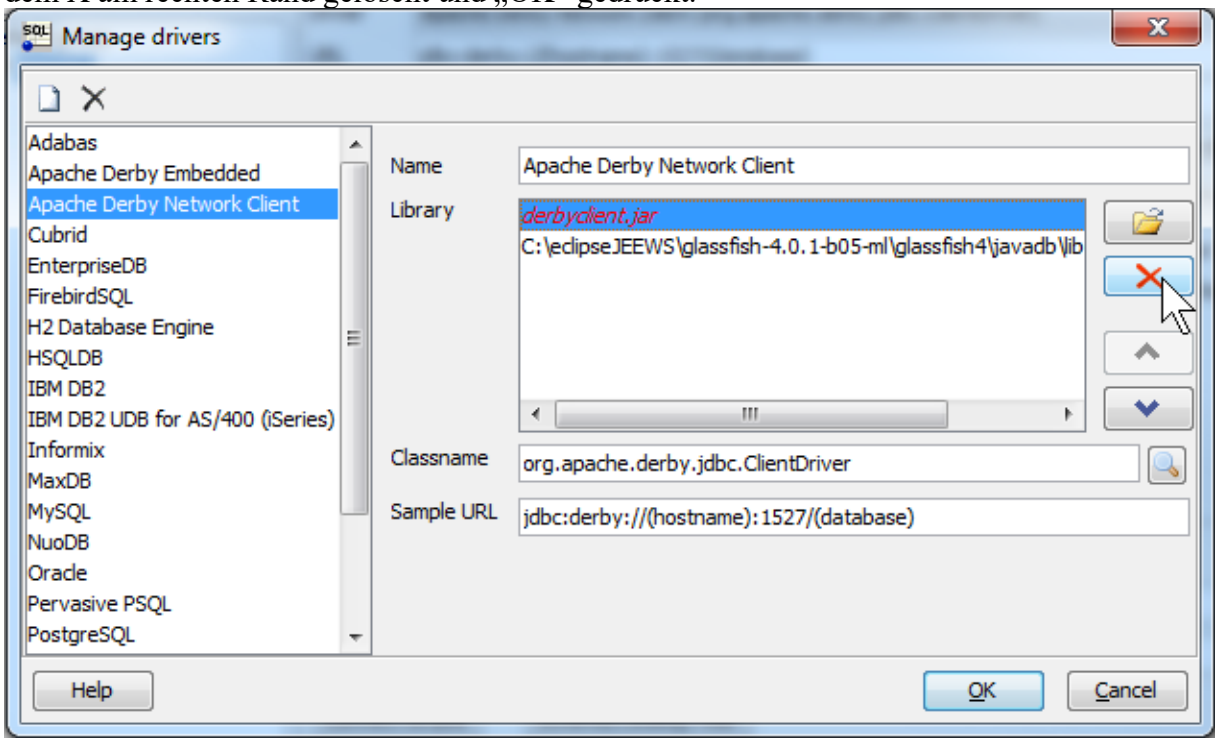
Im Installationsordner gibt es ein Verzeichnis lib mit dem Treiber. Es werden die drei Dateien mit gedückter Strg-Taste zusammen markiert und „Open“ geklickt.



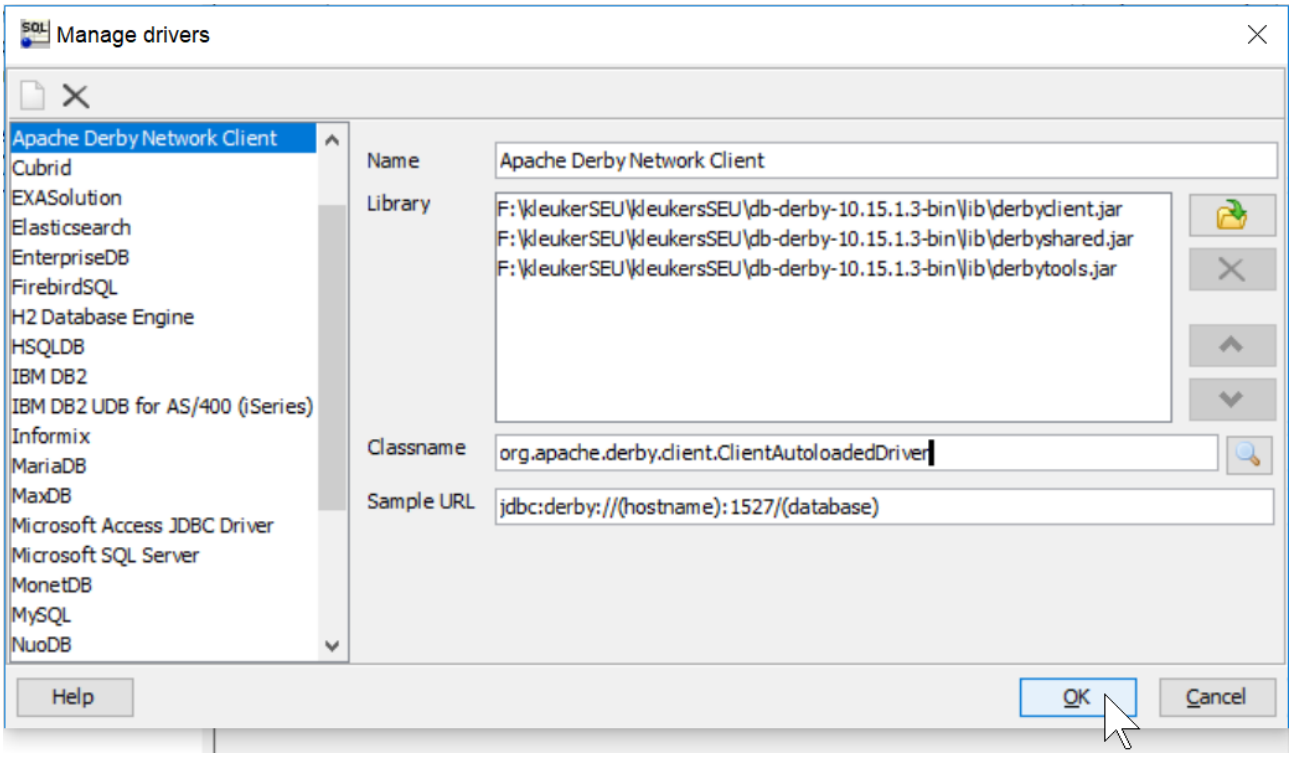
Falls eine Auswahl angeboten wird, wird der erste Treiber ausgewählt. Die Auswahl sollte nur für Derby-Versionen vor 10.15 angeboten werden. Ab 10.15 sollte der Eintrag automatisch „org.apache.derby.client.ClientAutoloadedDriver“ sein.



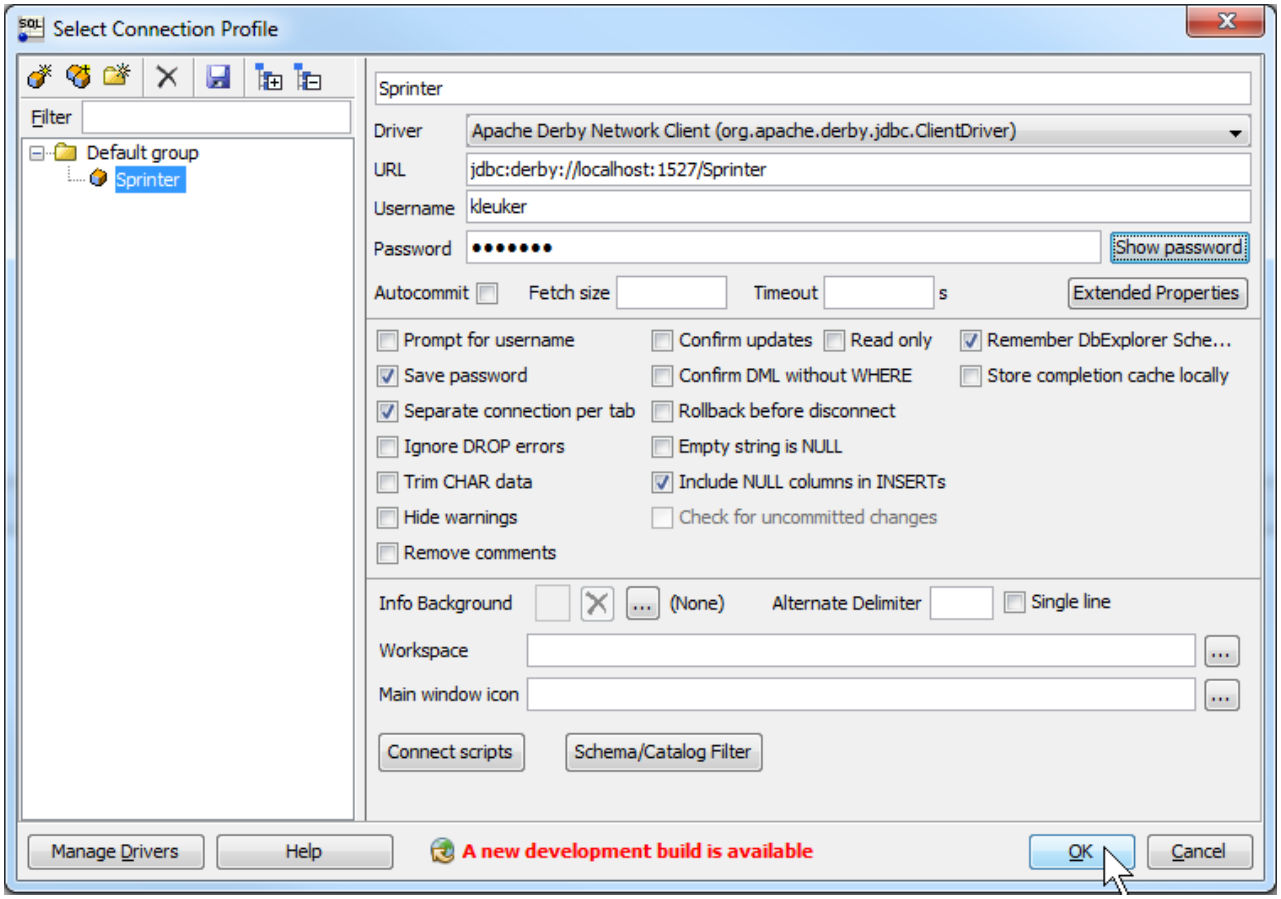
Falls der Eintrag „derbyclient.jar“ in roter kursiver Schrift noch angezeigt wird, wird der Eintrag mit dem X am rechten Rand gelöscht und „OK“ gedrückt.



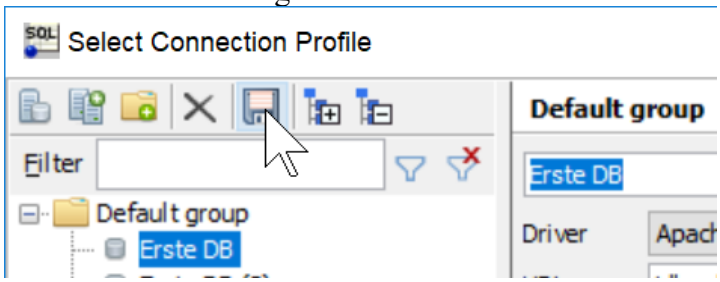
Bei einer Beispiel-Installation sieht die dann mit „OK“ zu bestätigenden Auswahl wie folgt aus.



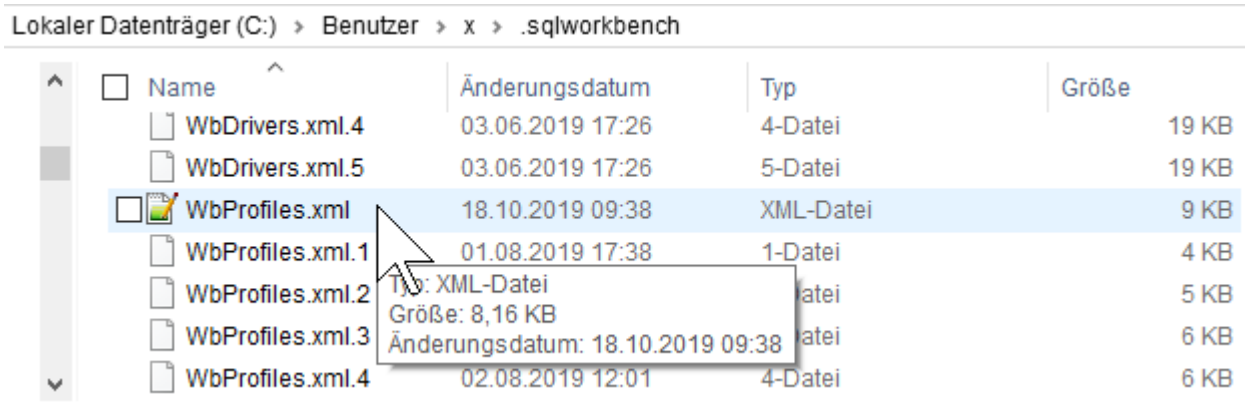
Soll eine bereits existierende Datenbank genutzt werden, wird die URL aktualisiert und Username sowie Password eingetragen. Die anderen Einstellungen hängen davon ab, wie man mit der Datenbank arbeiten möchte. Zum einfachen Lesen können die Einstellungen so übernommen werden. In der Kopfzeile wird der Verbindung ein sprechender Name gegeben. Es kann für eine Datenbank mehrere Verbindungseinträge geben, um z. B. mit Einstellungsparametern zu experimentieren.
Eine neue Datenbank wird im folgenden Unterkapitel angelegt.



Eine erfolgreich genutzte Verbindung muss über den oberen „Disketten“-Knopf gespeichert werden, da sie sonst verloren geht. Im oberen Textfeld ist der Verbindung ein sinnvoller Name zu geben.



Die Einstellung der SQL-Workbench werden standardmäßig im Nutzerverzeichnis im Unterordner „.sqlworkbench“ in der Datei „WbProfiles.xml“ gespeichert. Weiterhin werden ältere Versionen dieser Datei gesichert.



Ein Blick in die Datei „WbProfiles.xml“ kann sinnvoll sein, da so u. a. Passwörter und Connection-Strings eingesehen werden können. Das Passwörter im Klartext gespeichert sind, müsste die Datei in der Praxis besonders gesichert werden. Für die Datenbankexperimente in der Hochschule ist das kein Problem.

```
252   <void method="add">
253    <object class="workbench.db.ConnectionProfile">
254     <void property="autocommit">
255      <boolean>true</boolean>
256     </void>
257     <void property="driverName">
258      <string>Apache Derby Network Client</string>
259     </void>
260     <void property="driverclass">
261      <string>org.apache.derby.client.ClientAutoloadedDriver</string>
262     </void>
263     <void property="name">
264      <string>Mondial</string>
265     </void>
266     <void property="password">
267      <string>kleuker</string>
268     </void>
269     <void property="storeExplorerSchema">
270      <boolean>true</boolean>
271     </void>
272     <void property="url">
273      <string>jdbc:derby://localhost:1527/F:\workspaces\datenbanken\Mondial;create=true</string>
274     </void>
275     <void property="useSeparateConnectionPerTab">
276      <boolean>true</boolean>
277     </void>
278     <void property="username">
279      <string>kleuker</string>
280     </void>
281    </object>
282   </void>
```

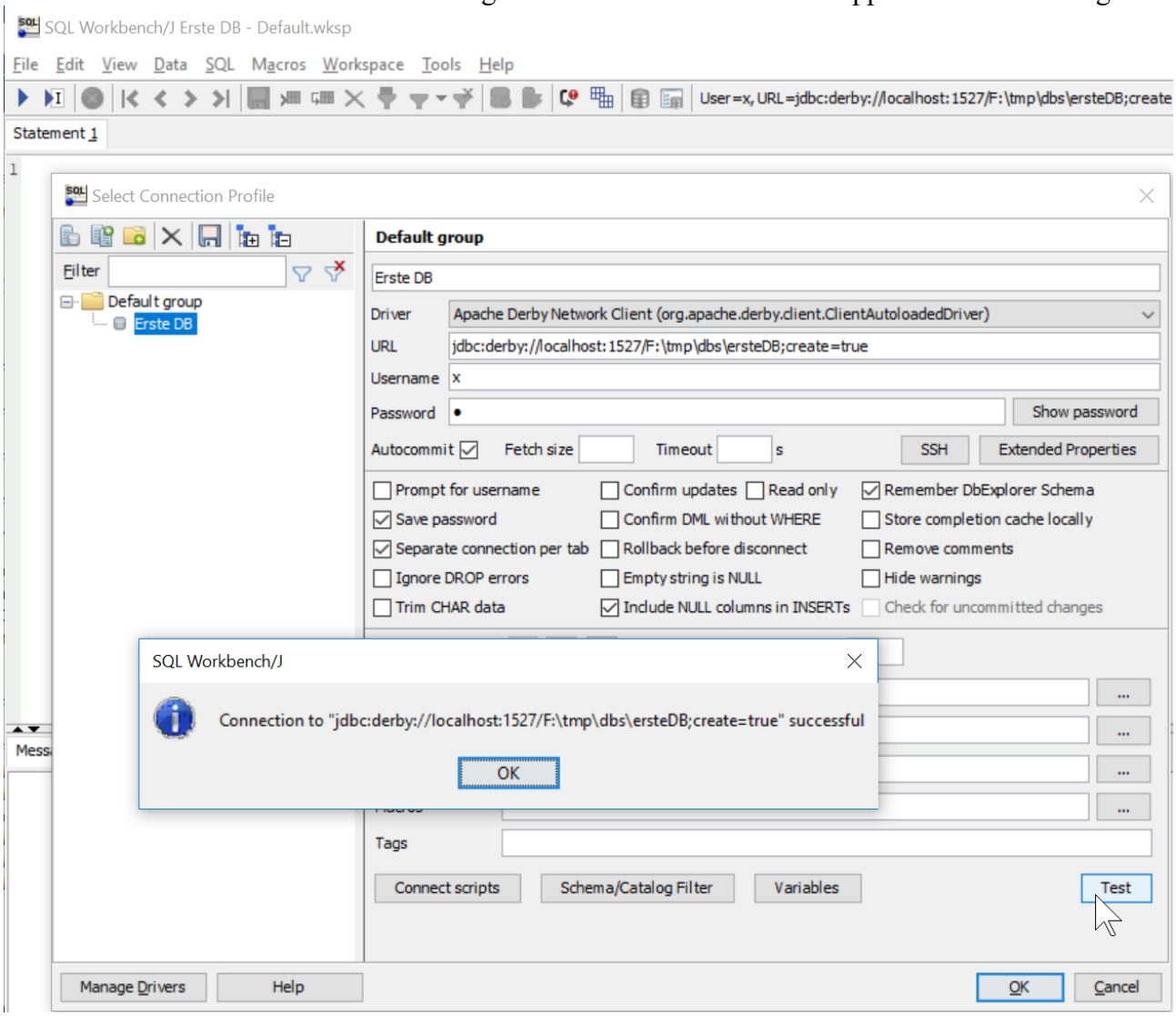
Durch einen Klick auf „OK“ im Fenster „Select Connection Profile“ wird die Verbindung aufgebaut und es öffnet sich eine Arbeitsfläche, in der oben Befehle eingeben und unten die Ergebnisse angesehen werden können.

1.5 Erstellung einer Datenbank

Die Datenbankerstellung ist direkt in der SQL-Workbench für Derby möglich. Der sogenannte Connection-String, der im Feld URL steht, kann um einen Parameter

```
;create=true, z. B.
jdbc:derby://localhost:1527/F:\tmp\dbs\ersteDB;create=true
```

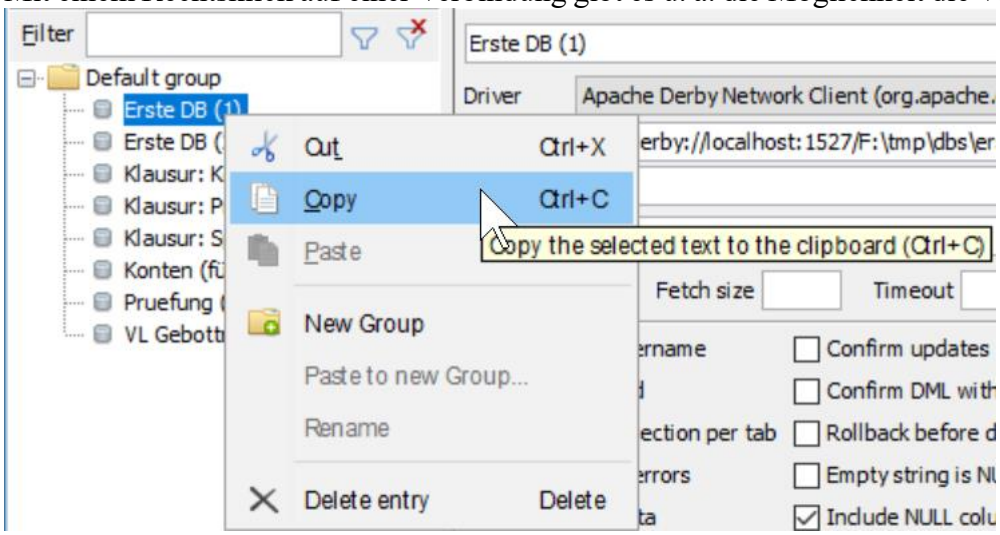
ergänzt werden. Sollte die Datenbank nicht existieren wird sie angelegt. Der Parameter kann für existierende Datenbanken stehen bleiben, da diese nicht davon betroffen sind. Der angegebene Speicherort sollte für alle Datenbanken genutzt werden, ist aber generell frei wählbar. Er muss nur lesbar und beschreibbar sein. Die Datenbank ist auch auf einem USB-Stick erstell- und nutzbar. Generell sollte immer ein Username und ein Password vergeben werden. Weiterhin wird am Anfang das Kreuz bei „Autocommit“ gesetzt. Generell kann eine Verbindung über den Knopf „Test“ rechts-unten überprüft werden, im konkreten Fall wird beim ersten „Test“ die Datenbank angelegt.
Links-unten ist über „Manage Drivers“ wieder das Fenster erreichbar, mit dem der Treiber ausgewählt wird, was bei Problemen hilfreich sein kann. Die üblichen Probleme beim ersten Start sind aber eher, dass die Datenbank-Software noch nicht gestartet wurde oder es einen Tippfehler in der URL gibt.



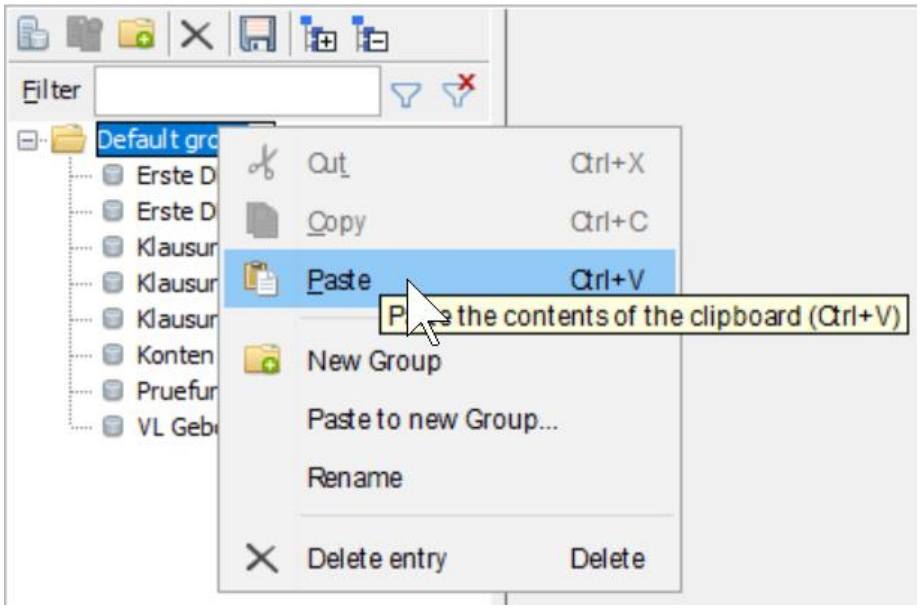
Soll mit unterschiedlichen Derby-Installationen, z. B. in der Hochschule und zu Hause oder mit unterschiedlichen Nutzern auf die Datenbank zugegriffen werden, sind die Sicherheitseinstellungen

für den entstandenen Datenbankordner abzuschwächen. Dies wird in „1.11 Datenbanken auf unterschiedlichen Rechnern öffnen“ beschrieben. Es sei daran erinnert die Verbindungen abzuspeichern.

Mit einem Rechtsklick auf einer Verbindung gibt es u. a. die Möglichkeit die Verbindung zu kopieren.



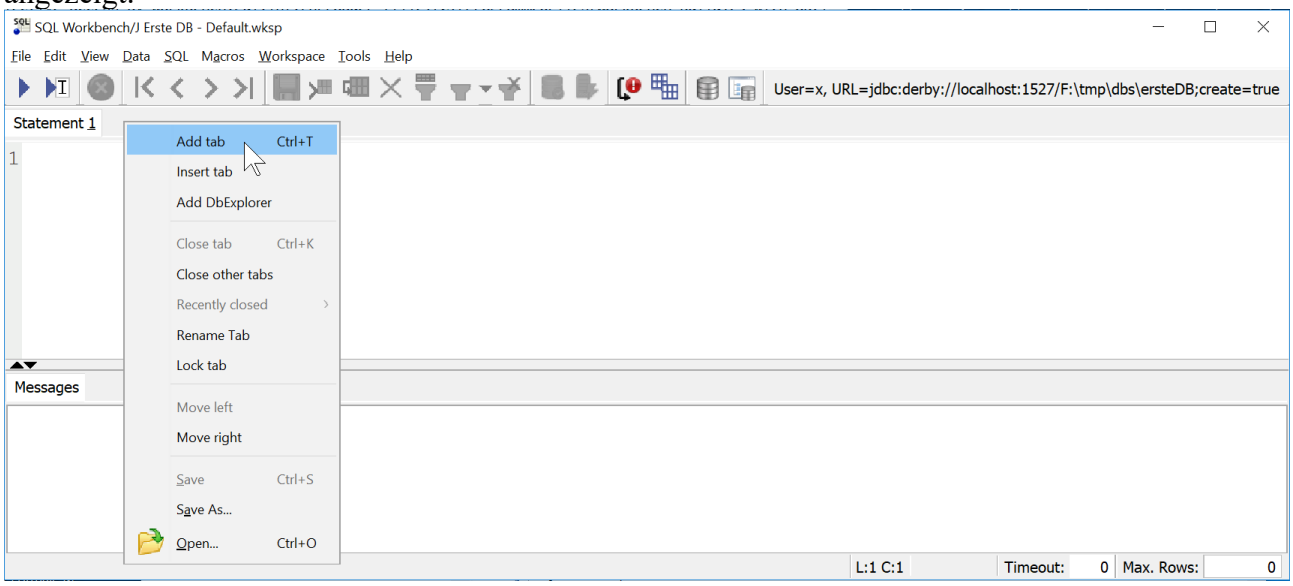
Zum Einfügen wird ein Rechtsklick auf einer Gruppe gemacht. Am Anfang existiert nur die Gruppe „Default group“. In dem Menü besteht auch die Möglichkeit eine neue Gruppe anzulegen.



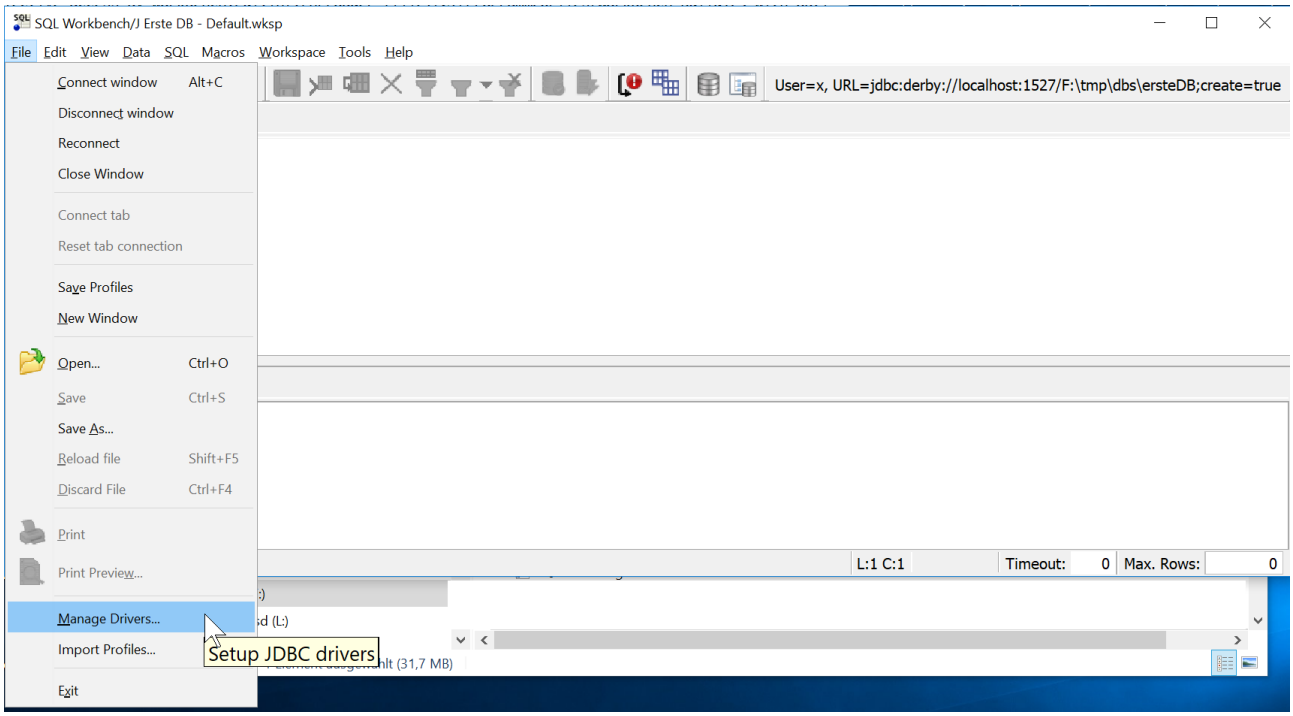
1.6 Ausführen von SQL-Befehlen

Die eigentliche Arbeitsumgebung ist zweigeteilt. Oben erfolgt die Eingabe von Befehlen, unten die Ausgabe, ggfls. mit Fehlermeldungen. Mit einem Rechtsklick neben dem ersten Reiter sind weitere

Reiter hinzufügbar, die alle auf der Datenbank arbeiten. Die genutzte Datenbank wird links-oben angezeigt.

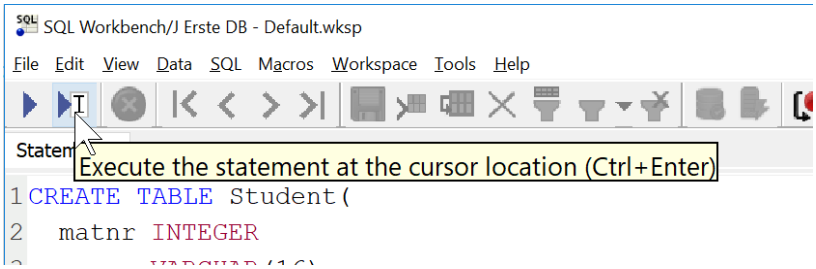


Unter dem File-Menü kann die Datenbank-Verbindung ein- bzw. ausgeschaltet werden. Weiterhin besteht die Möglichkeit Fensterinhalte zu speichern, was sehr wichtig ist, damit erreichte Ergebnisse nicht einfach verschwinden. Im unteren Bereich ist mit „Manage Drivers“ die Treiber-Verwaltung aufrufbar.

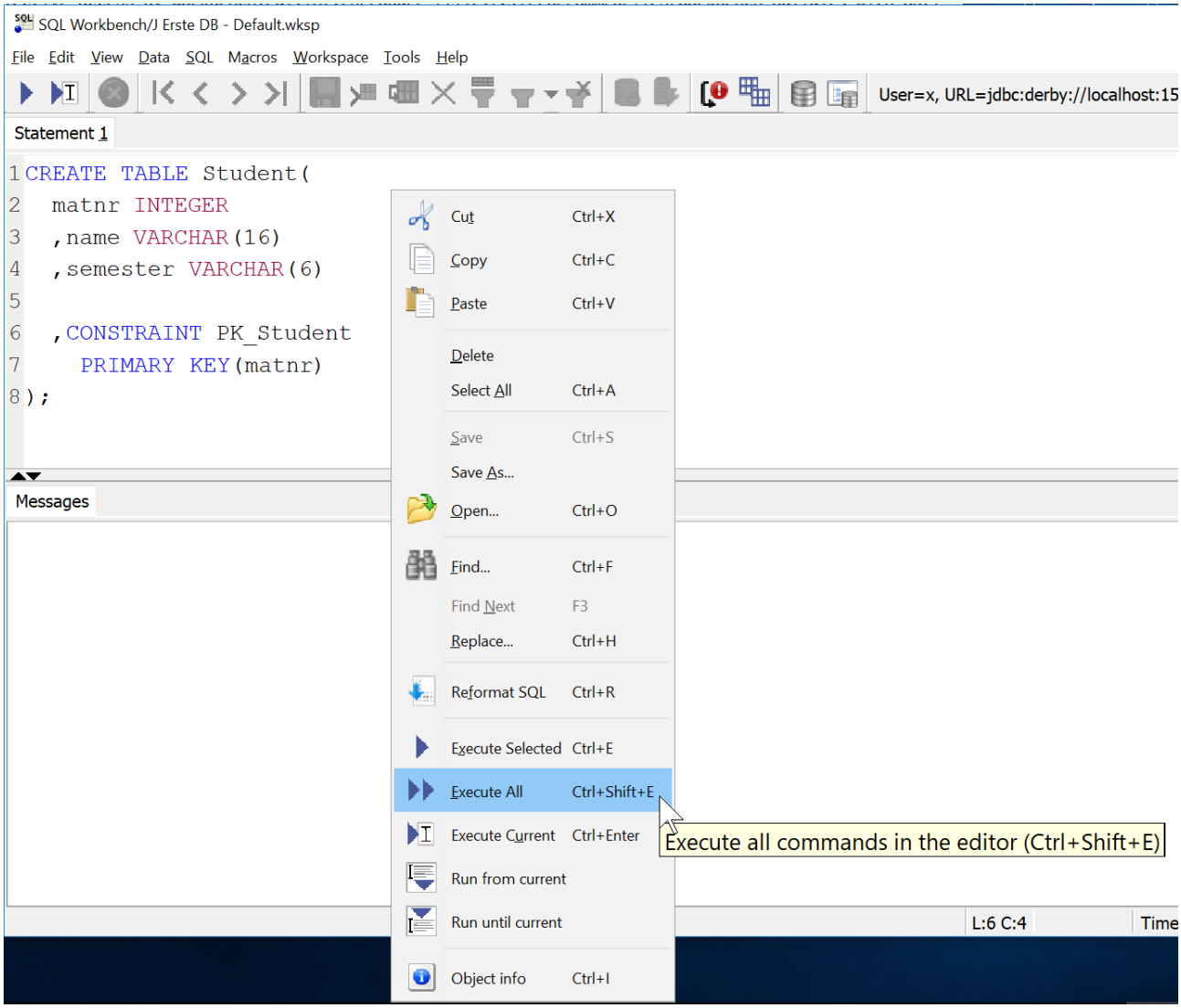


Zum Ausführen von SQL-Befehlen stehen mehrere Möglichkeiten zur Verfügung, die wichtigsten sind die Folgenden.
Mit dem zweiten Knopf der Knopfleiste wird der Befehl ausgeführt, auf oder bei dem der Cursor

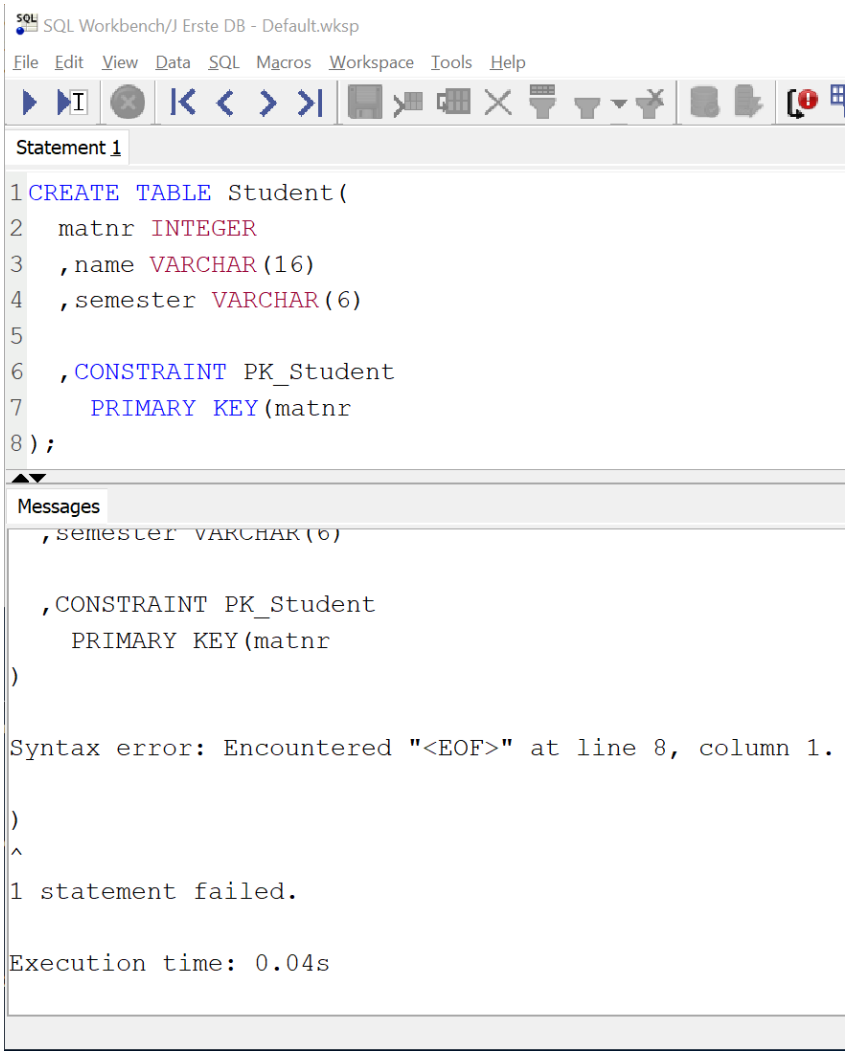
steht. Wichtig ist nur, dass die vorherigen Befehle syntaktisch korrekt sind, insbesondere mit einem Semikolon abschließen.



Nach einem Rechtklick im Ausführungsfenster stehen mehrere Varianten der Ausführung zur Verfügung. Beim Einspielen neuer Tabellen oder Einträge oder generell SQL-Skripten mit mehreren Befehlen, wird oft „Execute All“ genutzt. Sollte es sich um sehr große Datenmengen handeln, wie beim Einspielen von Mondial, ist es ratsam, während der Ausführung das Fenster zu minimieren, da Ausgaben generell enorm viel Zeit verbrauchen. Zwischenzeitlich nachzuschauen wie weit die Abarbeitung ist, ist natürlich sinnvoll.



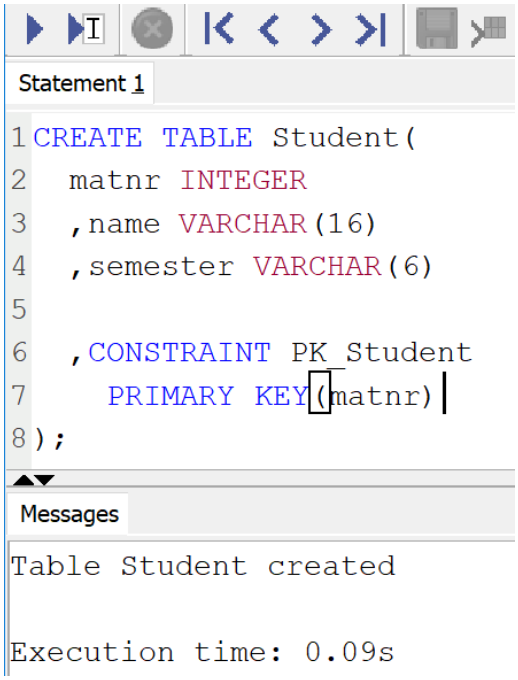
Fehler werden im Output-Bereich unter Nennung des Fensternamens ausgegeben. Im folgenden Beispiel fehlt eine schließende Klammer.



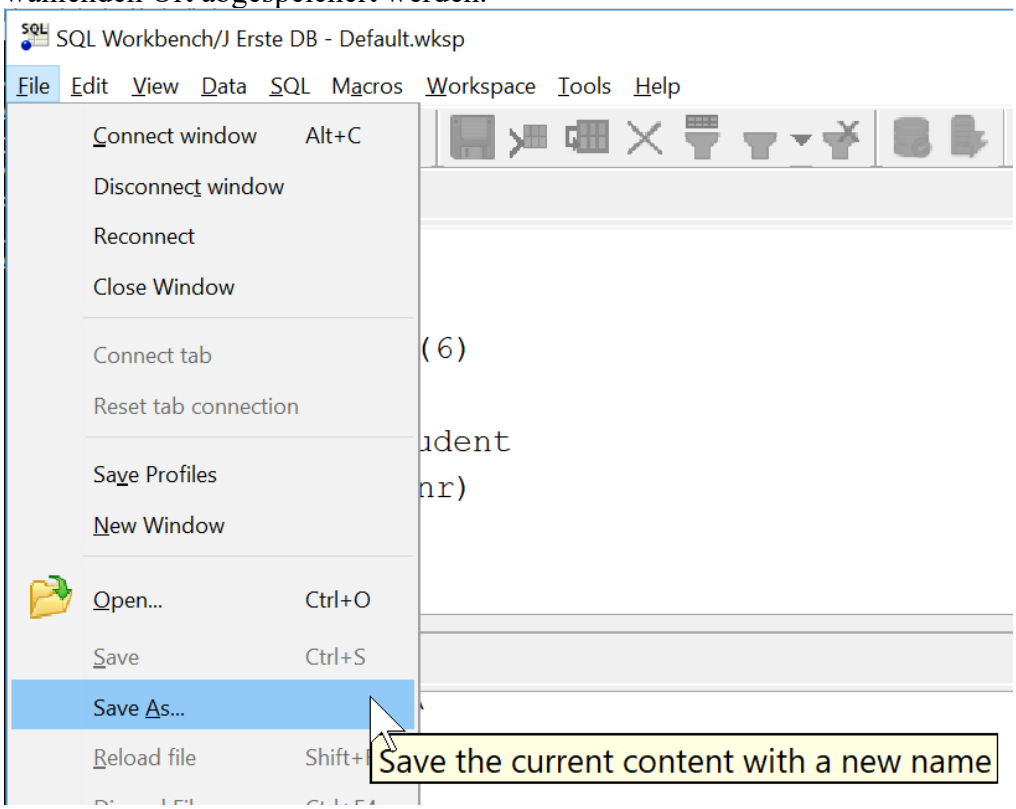
Das Skript wird wie folgt korrigiert.

```
CREATE TABLE Student(
  matnr INTEGER,
  name VARCHAR(16),
  semester VARCHAR(6),

  CONSTRAINT PK_Student
    PRIMARY KEY(matnr)
);
```

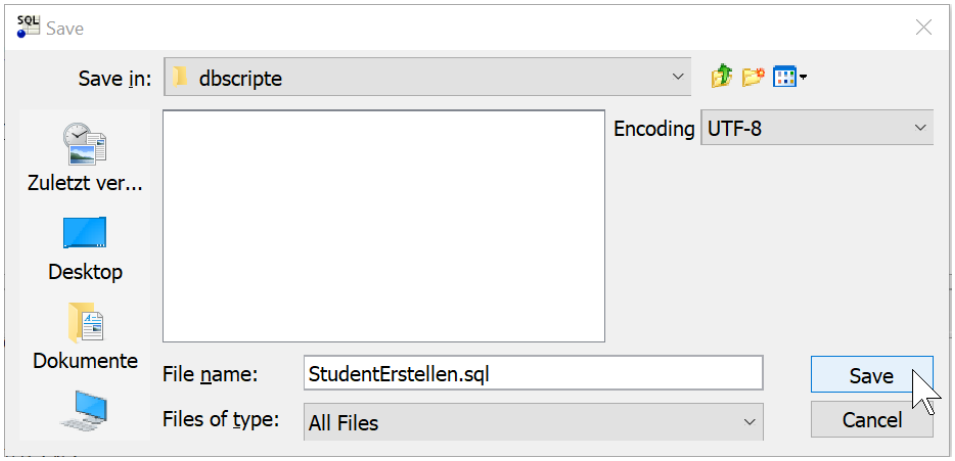
Nachdem die Tabellenerzeugung erfolgreich abgeschlossen wurde, wird eine Information im Output-Fenster ergänzt.



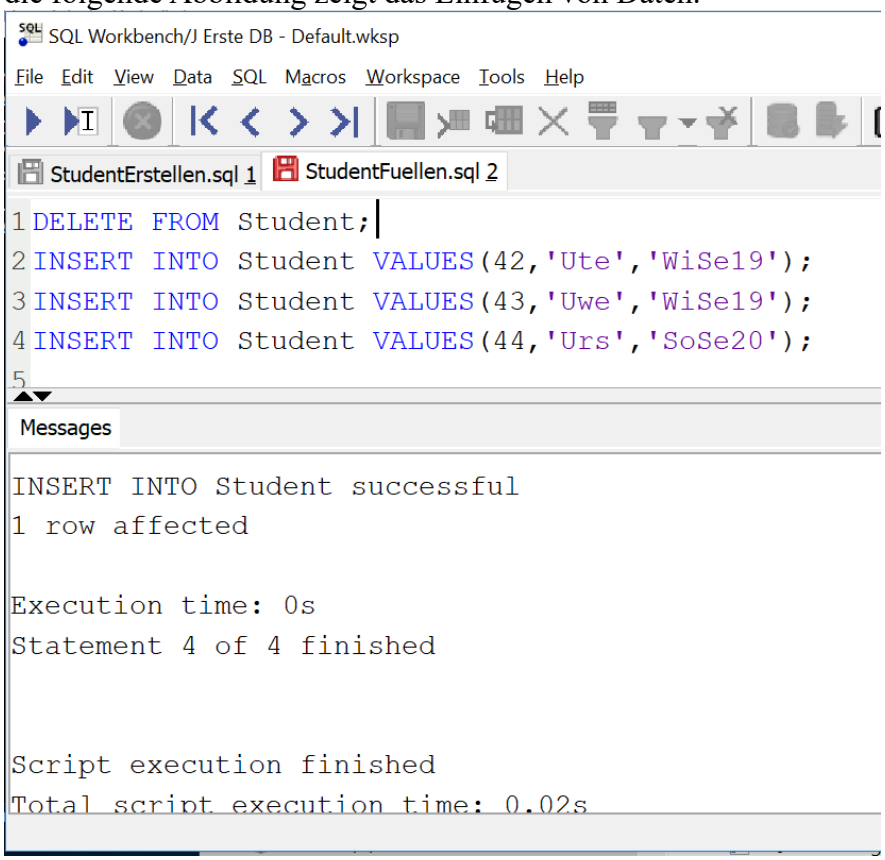
Das Skript kann mit „File > Save As“ als SQL-Skript mit der Endung „.sql“ an einem sinnvoll zu wählenden Ort abgespeichert werden.



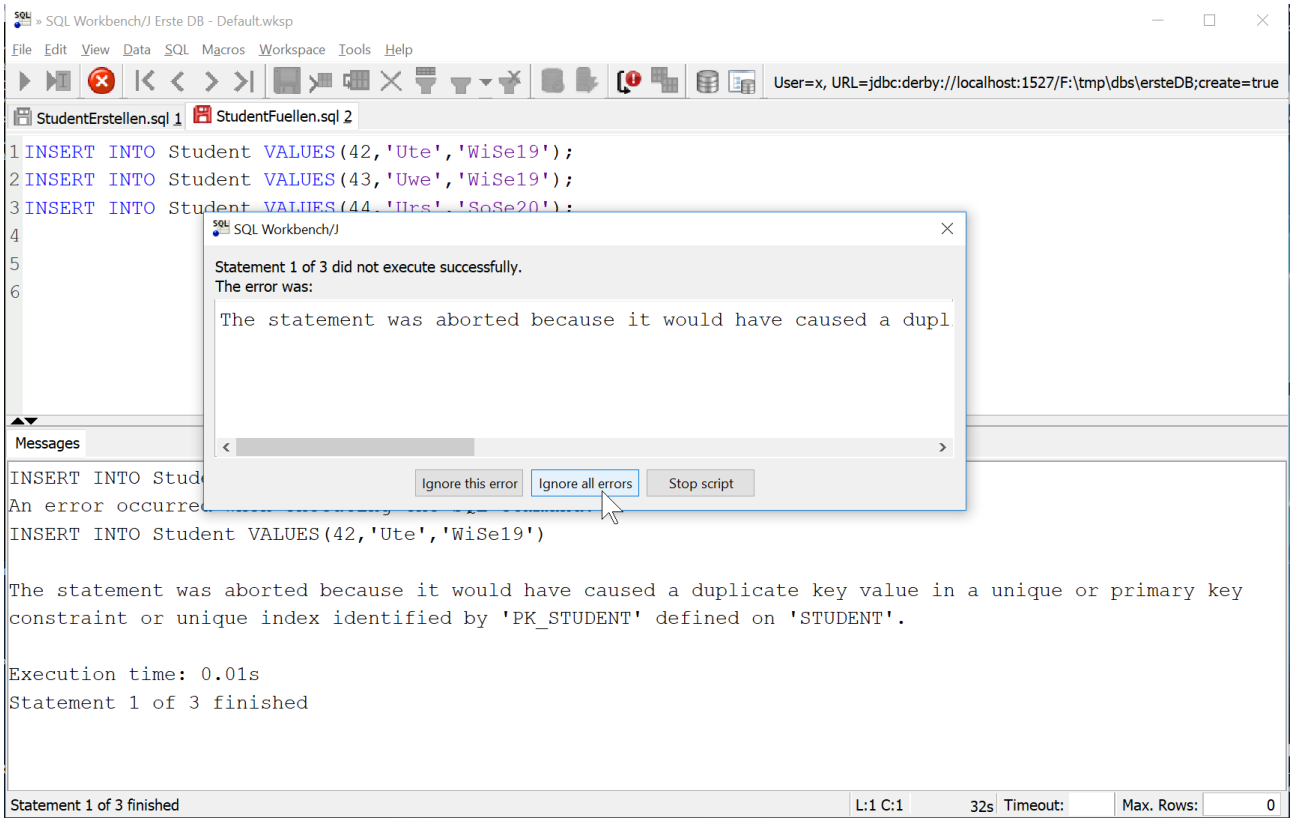
Beim Speichern kann das Encoding geändert werden, sinnvoll ist z. B. UTF-8.



CREATE, INSERT, UPDATE, DELETE und ähnliche Befehle laufen nach dem gleichen Prozess ab, die folgende Abbildung zeigt das Einfügen von Daten.

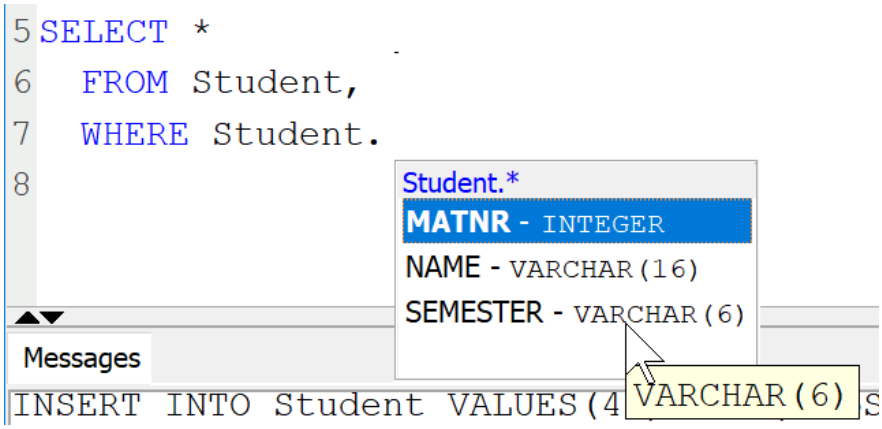


Sollten Fehler in Skripten mit mehreren Anweisungen auftreten, kann das Verhalten von SQL-Workbench konfiguriert werden. Im folgenden Beispiel werden nur die INSERT-Befehle nochmals ausgeführt. Da dies gegen den Primary Key verstößt, wird der erste Befehl als fehlerhaft erkannt und der Nutzer erhält die Möglichkeit zu entscheiden, wie weiter vorgegangen werden soll.

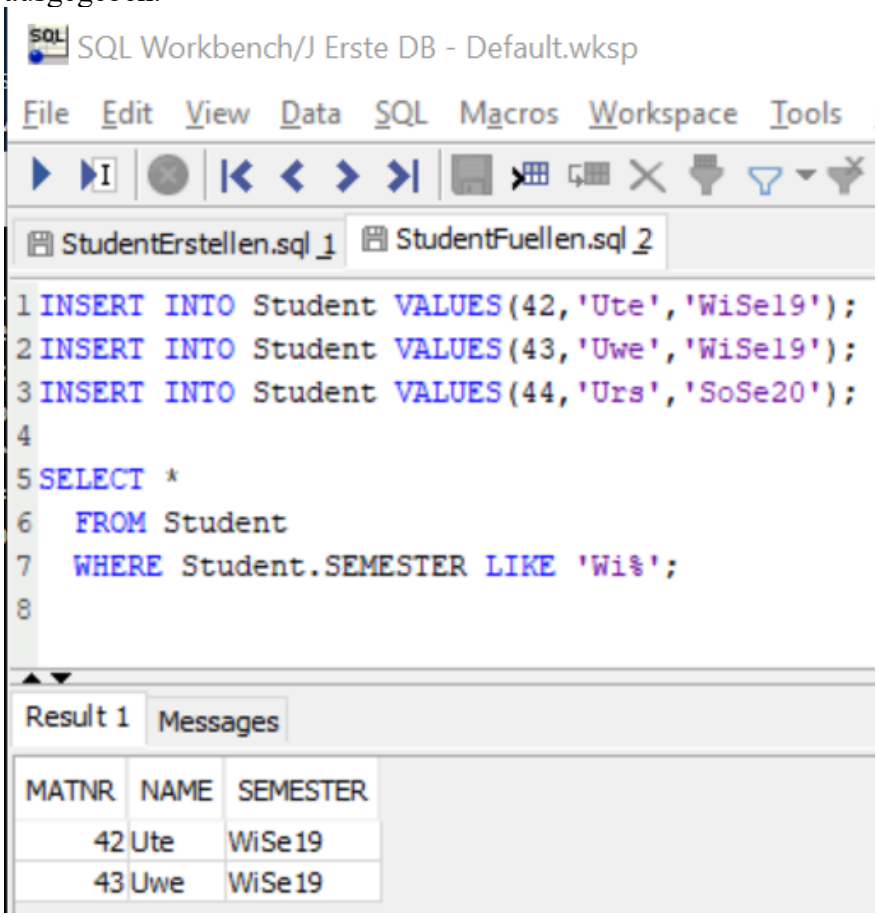


Andere Werkzeuge lassen Skripte einfach durchlaufen und ignorieren dabei die Fehler, die nur gezählt werden. Ist diese Eigenschaft gewünscht kann dies z. B. über einen Toggle-Button in der Knopf-Leiste eingestellt werden.

Bei Select-Befehlen kann z. B. bei der Auswahl von Attributen die Code-Completion genutzt werden, die über „STRG + Leertaste“ Angebote macht,



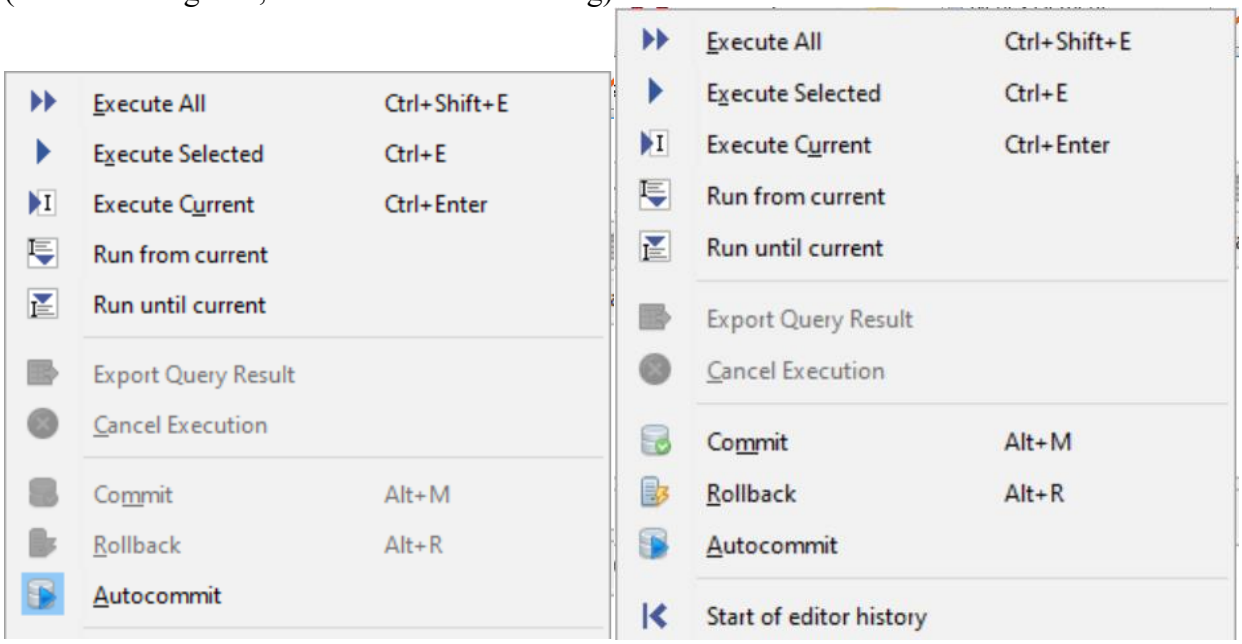
Bei SELECT-Befehlen wird im unteren Teil die resultierende Tabelle in einem eigenen Reiter ausgegeben.



1.7 Transaktionssteuerung

Transaktionen können beliebig viele SQL-Befehle der Form INSERT, UPDATE, DELETE und SELECT enthalten, die von der Datenbank ganz in einem Ablauf oder gar nicht durchgeführt werden.

Um die bei der Erstellung genutzte Einstellung AUTOCOMMIT zu nutzen, wird die Transaktionssteuerung ausgeschaltet, da nach jedem einzelnen Befehl automatisch ein COMMIT geschickt wird. Ob dies sinnvoll ist, hängt von dem Einsatzbereich der Datenbank ab.
Generell kann man mit SQLWorkbench für jeden Reiter die Transaktionssteuerung an- und ausschalten. Unter Data gibt es dazu die Option „Autocommit“, wobei ein gehighlightetes Symbol (blauer Hintergrund, linke Seite der Abbildung) für true steht.



Bei der Transaktionssteuerung muss wie für alle Datenbanken üblich, die genaue Art der Transaktionssteuerung mit dem ersten Befehl der Transaktion festgelegt werden. Dies ist nach jedem COMMIT und ROLLBACK zu wiederholen. Für Derby stehen vier Modi zur Verfügung.

```
SET ISOLATION READ UNCOMMITTED;
SET ISOLATION READ COMMITTED;
SET ISOLATION REPEATABLE READ;
SET ISOLATION SERIALIZABLE;
```

Durch die Nutzung mehrerer Reiter, die jeweils automatisch eine eigene Transaktion haben, sind Situationen mit mehreren Nutzern simulierbar. Generell sollten solche Experimente aber besser in eigenen Konsolenfenstern stattfinden, da dort besser zu beobachten ist, wann die Datenbank eine Transaktion automatisch abbricht und wie sich dann andere Transaktionen verhalten.

SQL Workbench/J Erste DB - Default.wksp

File Edit View Data SQL Macros Workspace Tools Help

StudentErstellen.sql 1 | StudentFuellen.sql 2 | Trigger.sql 3 | Statement 4 | Statement 5 | Database Explo

```
1 SET ISOLATION SERIALIZABLE;
2
3 INSERT INTO Student VALUES(445,'Urs','SoSe20');
4 SELECT * FROM Student;
```

Messages

```
An error occurred when executing the SQL command:
SELECT * FROM Student

A lock could not be obtained due to a deadlock, cycle of locks and waiters is:
Lock : TABLE, STUDENT, Tablelock
  Waiting XID : {646, S} , X, SELECT * FROM Student
  Granted XID : {644, IX}
Lock : TABLE, STUDENT, Tablelock
  Waiting XID : {644, S} , X, SELECT * FROM Student
  Granted XID : {644, IX} , {646, IX}
. The selected victim is XID : 646.
1 statement failed.

Execution time: 3.03s
```

SQL Workbench/J Erste DB - Default.wksp

File Edit View Data SQL Macros Workspace Tools Help

StudentErstellen.sql 1 | StudentFuellen.sql 2 | Trigger.sql 3 | Statement 4 | Statement 5 | Database Explo

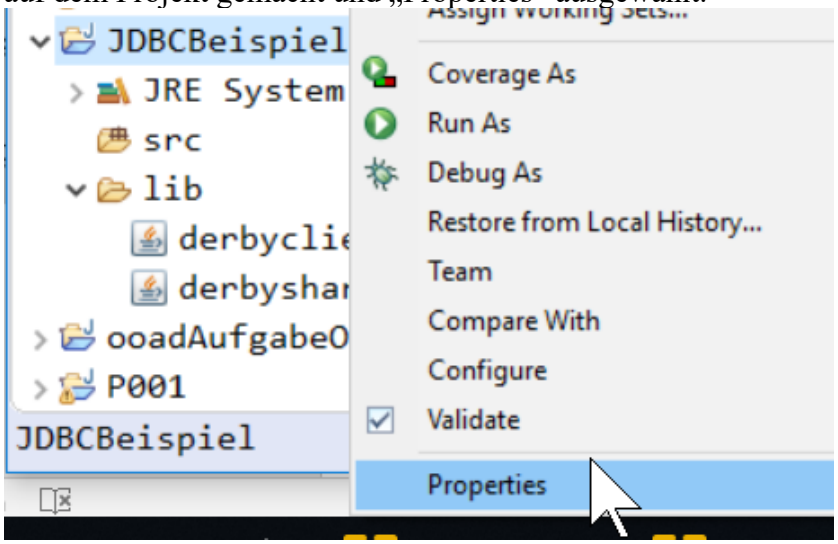
```
1 SET ISOLATION SERIALIZABLE;
2
3 INSERT INTO Student VALUES(445,'Urs','SoSe20');
4 SELECT * FROM Student;
```

Messages

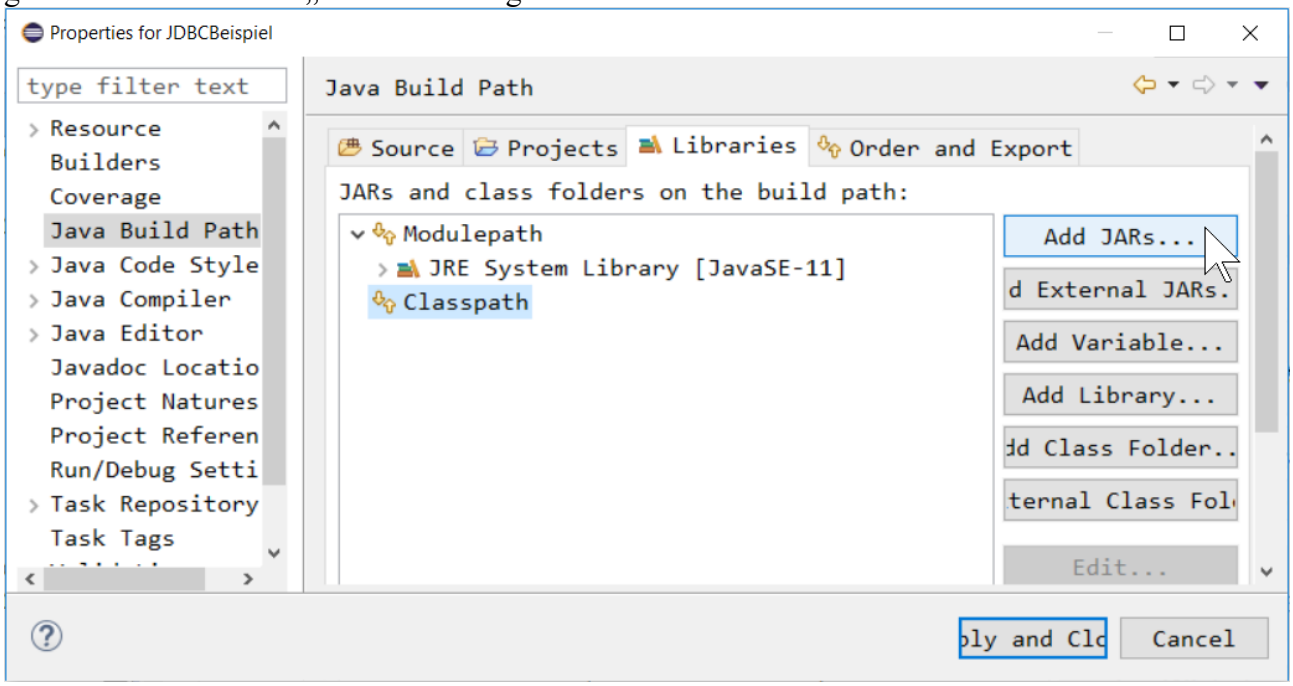
```
INSERT INTO Student not successful
An error occurred when executing the SQL command:
INSERT INTO Student VALUES(445,'Urs','SoSe20')

A lock could not be obtained within the time requested
1 statement failed.

Execution time: 1m 0s
```

1.8 Zugriff mit JDBC

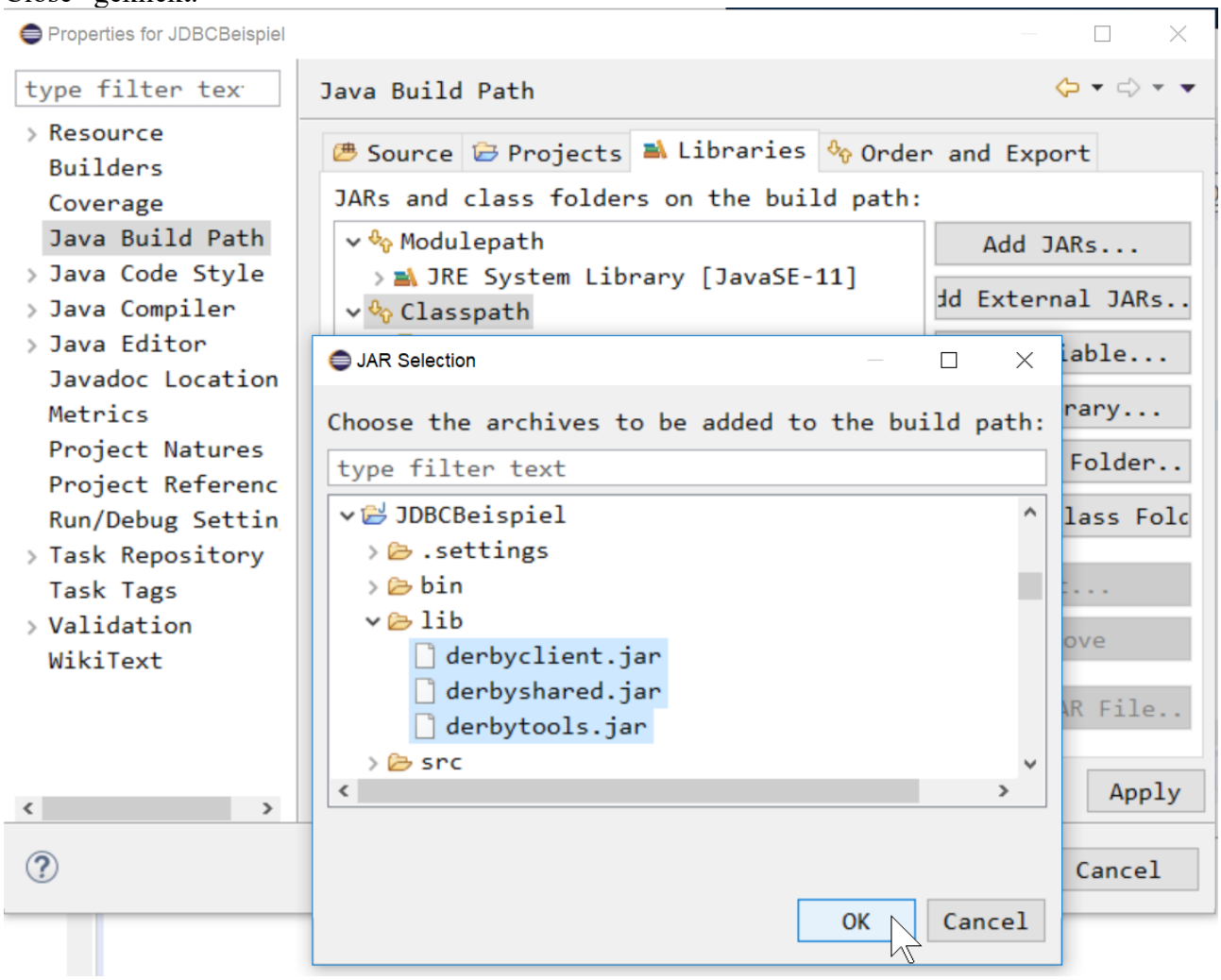
Der Zugriff aus einem Java-Programm mit Hilfe von JDBC erfolgt mit den zugehörigen Sprachkonstrukten, der „Connection-String“ wurde bereits beim Aufbau der Verbindung angegeben. Zum Aufbau der Datenbankverbindung werden drei jar-Dateien aus dem lib-Verzeichnis der Derby-Installation benötigt. Dies sind derbyclient.jar, derbyshared.jar und derbytools.jar. Diese Dateien können direkt genutzt werden, alternativ werden sie in ein lokales Projektverzeichnis kopiert, wodurch das Projekt einfacher kopiert werden kann.
Im Beispiel werden die Dateien in einen Projektunterordner lib kopiert. Dann wird ein Rechtsklick auf dem Projekt gemacht und „Properties“ ausgewählt.



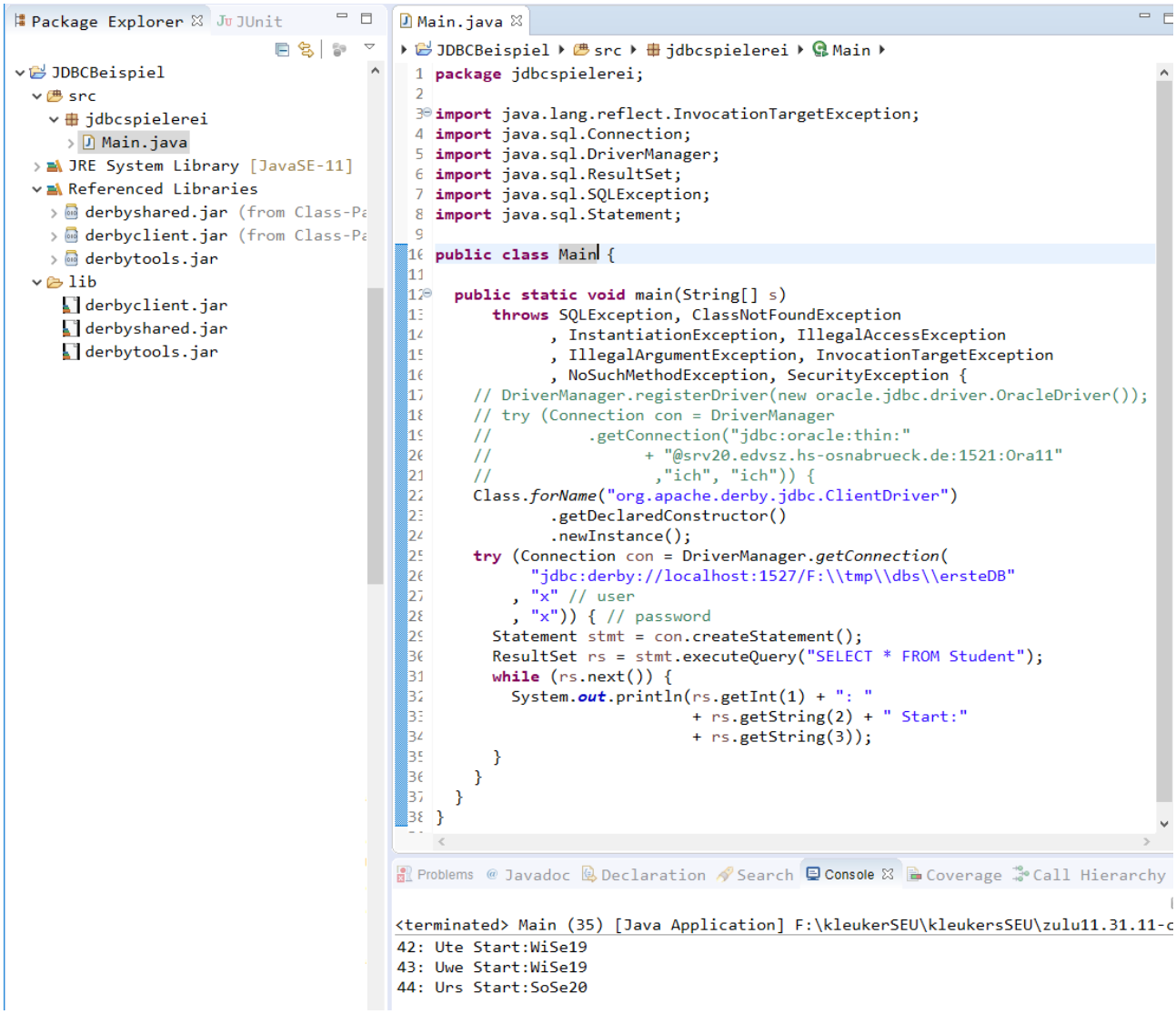
Links wird „Java Build Path“ angeklickt, dann der „Libraries“ ausgewählt, unten auf „Classpath“ geklickt und rechts auf „Add JARs…“ geklickt.



Es werden mit gedrückter „Strg“-Taste die jar-Dateien ausgewählt, „OK“ und dann „Apply and Close“ geklickt.



Ein Beispielprogramm sieht mit zugehöriger Ausgabe wie folgt aus. Man beachte, dass in den Kommentaren die Verbindung zu einer externen Oracle-Datenbank beschrieben ist, für die man zunächst einen JDBC-Treiber herunterladen muss.




```
package jdbcspielerei;

import java.lang.reflect.InvocationTargetException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Main {

  public static void main(String[] s)
      throws SQLException, ClassNotFoundException
```

```
            , InstantiationException, IllegalAccessException
            , IllegalArgumentException, InvocationTargetException
            , NoSuchMethodException, SecurityException {
    // DriverManager.registerDriver(
           new oracle.jdbc.driver.OracleDriver());
    // try (Connection con = DriverManager
    //          .getConnection("jdbc:oracle:thin:"
    //                + "@srv20.edvsz.hs-osnabrueck.de:1521:Ora11"
    //                 ,"ich", "ich")) {
    Class.forName("org.apache.derby.jdbc.ClientDriver")
            .getDeclaredConstructor()
            .newInstance();
    try (Connection con = DriverManager.getConnection(
          "jdbc:derby://localhost:1527/F:\\tmp\\dbs\\ersteDB"
        , "x" // user
        , "x")) { // password
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery("SELECT * FROM Student");
      while (rs.next()) {
        System.out.println(rs.getInt(1) + ": "
                           + rs.getString(2) + " Start:"
                           + rs.getString(3));
      }
    }
  }
}
```

1.9 Stored Procedures und Trigger

Mit Hilfe von Stored Procedures kann man Datenbanken „serverseitig“, also innerhalb der Datenbank erweitern. Trigger ermöglichen es, auf INSERT, UPDATE und DELETE-Befehle zu reagieren und diese z. B. nach einer Prüfung der Aktion abzulehnen. In Derby werden die Erweiterungen in Java basierend auf Klassenmethoden umgesetzt.
Das hier gezeigte Beispiel nutzt die folgende Gebotstabelle.

```
CREATE TABLE Gebot(
  mnr INTEGER -- Mitgliedsnummer
 ,ware INTEGER -- Identifikator der Ware
 ,gebot DECIMAL(8, 2) -- gebotener Preis
 ,PRIMARY KEY(mnr,ware,gebot)
);
```

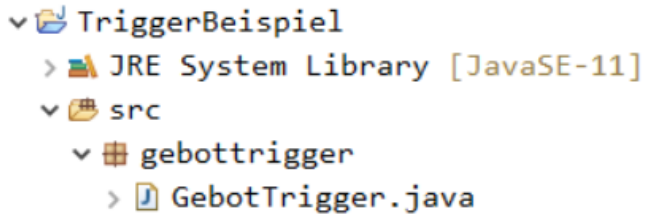
Umzusetzen ist die Anforderung, bei neuen Geboten (insert oder update erlaubt) für die gleiche Ware muss das eigene Gebot (gleiche mnr) erhöht werden.
Zunächst wird eine Java-Methode geschrieben, die folgende Randbedingungen erfüllt:

- es ist eine Klassenmethode (static)
- der Rückgabetyp ist void

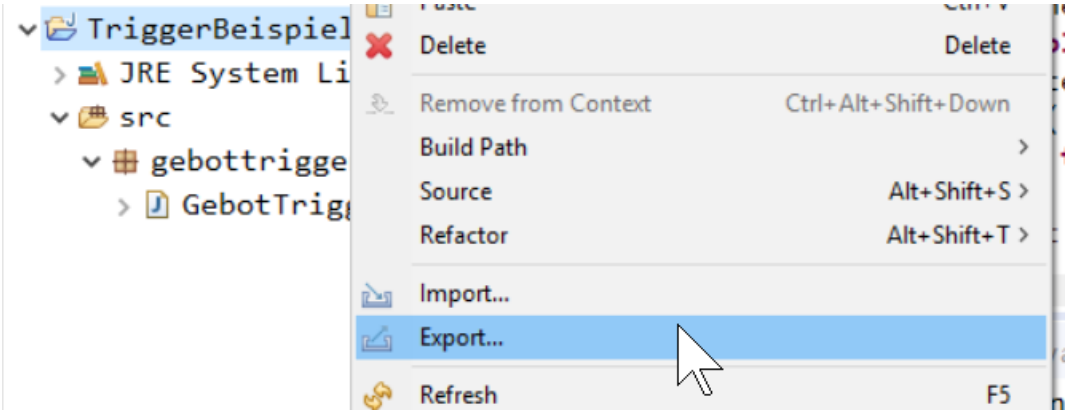
- die Parameter entsprechen der zu untersuchenden Aufgabe, hier wird ein Gebot bestehend aus seinen Parametern übergeben
- im Fehlerfall wird eine eigene SQLException geworfen, die als ersten Parameter den Grund und als zweiten Parameter ein individuell aus dem Intervall 30000 bis 38000 gewählten SQL-State hat.

```
package gebottrigger;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class GebotTrigger {

  public static void gebotErhoehen(int mnr, int ware, double gebot) throws SQLException {
     // hole aktuelle Verbindung, genauer laufende Transition
     Connection con = DriverManager
           .getConnection("jdbc:default:connection");
     Statement stmt = con.createStatement();
     ResultSet rs = stmt.executeQuery("SELECT MAX(gebot) "
           + "FROM Gebot "
           + "WHERE mnr=" + mnr + " AND ware=" + ware);
     rs.next();
     double max = rs.getDouble(1);
     System.out.println(rs.wasNull() + " " + max);
     if (!rs.wasNull() && max >= gebot) {
        throw new SQLException("Gebot erhöhen!", "30009");
     }
     stmt.close();
  }
}
```

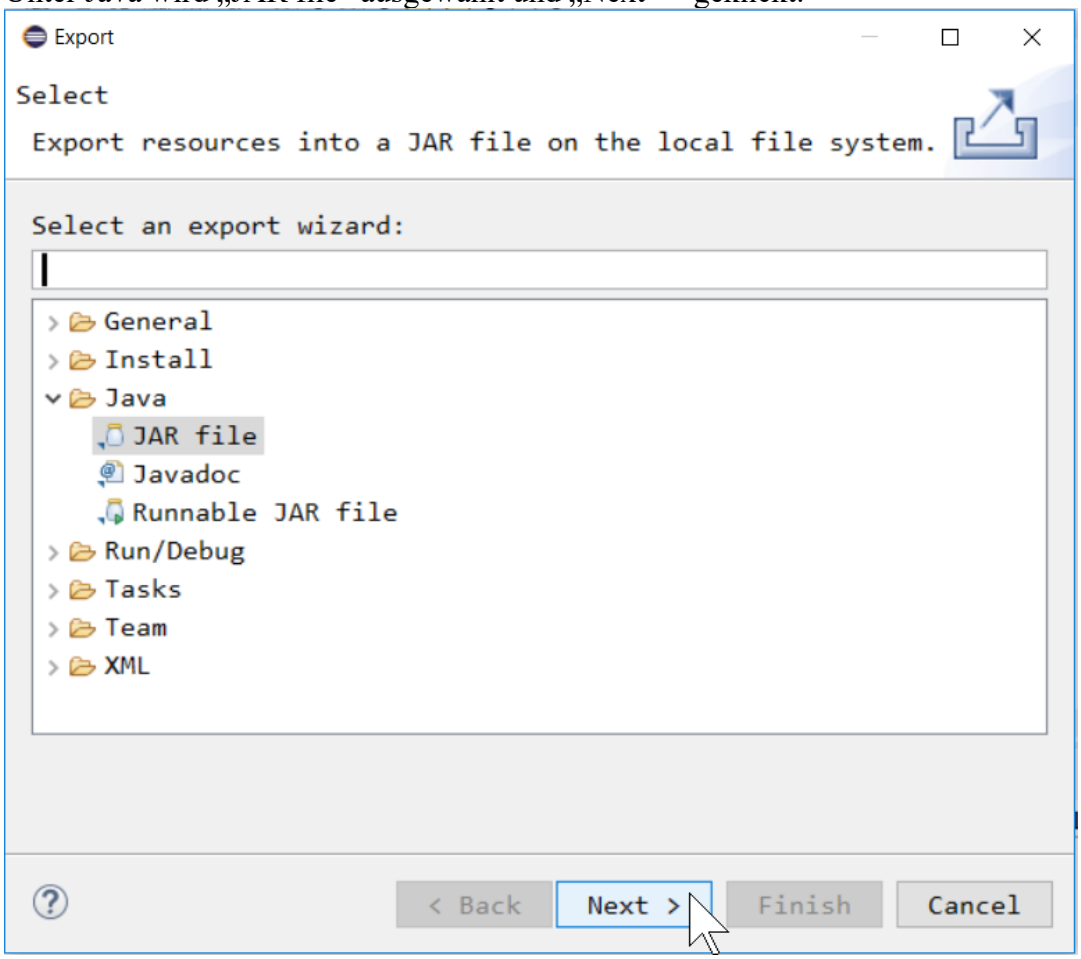
Diese Java-Methode steht in einem normalen Eclipse-Java-Projekt in einer frei gewählten Klasse, hier GebotTrigger.java im Paket gebottrigger. Das Projekt hat keine main-Methode, natürlich könnten Tests ergänzt werden. Weiterhin werden keine Bibliotheken genutzt, da nur die Standard-JDBC-Schnittstellen auftreten.



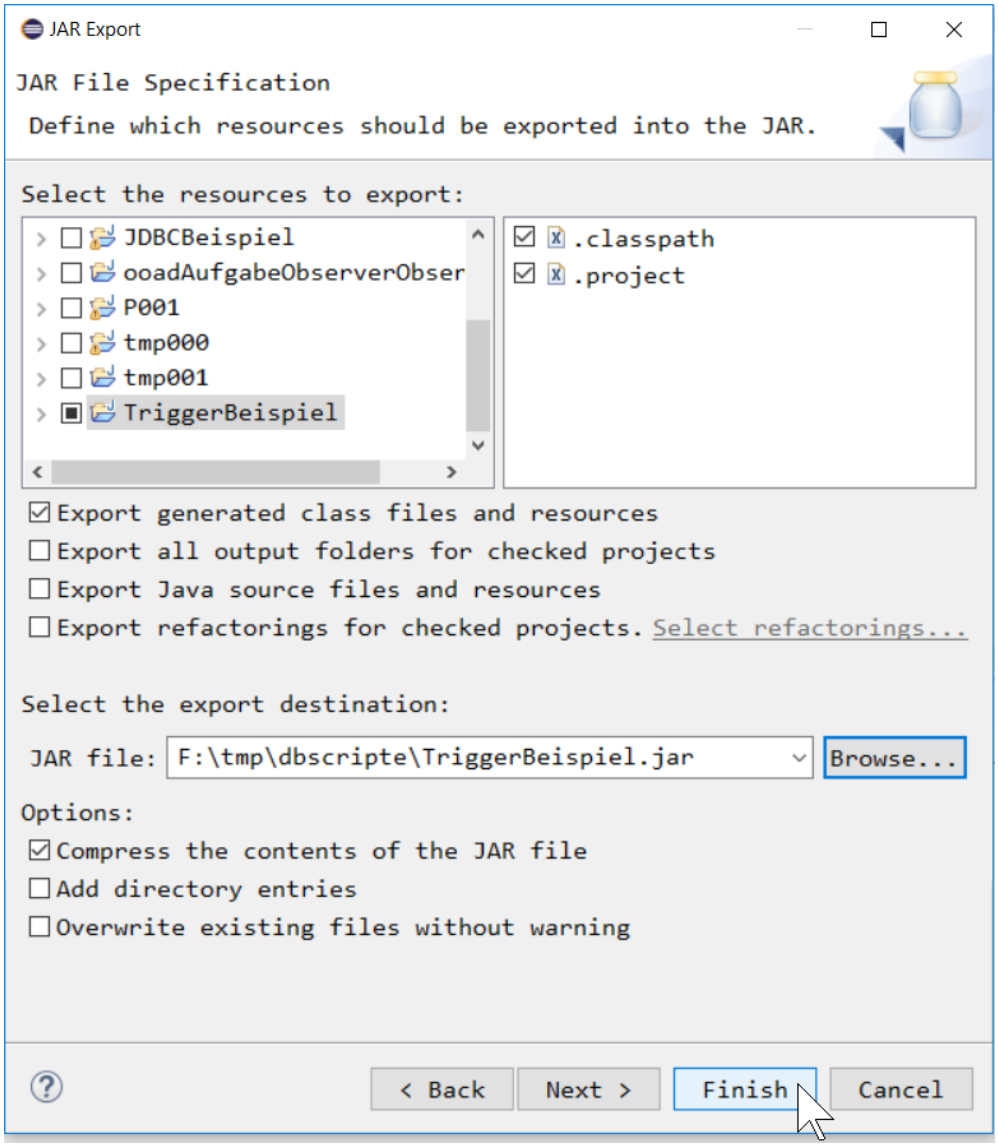
Nun muss eine jar-Datei erzeugt werden. Dazu wird ein Rechtsklick auf dem Projekt gemacht und „Export…“ ausgewählt.



Unter Java wird „JAR file“ ausgewählt und „Next >“ geklickt.

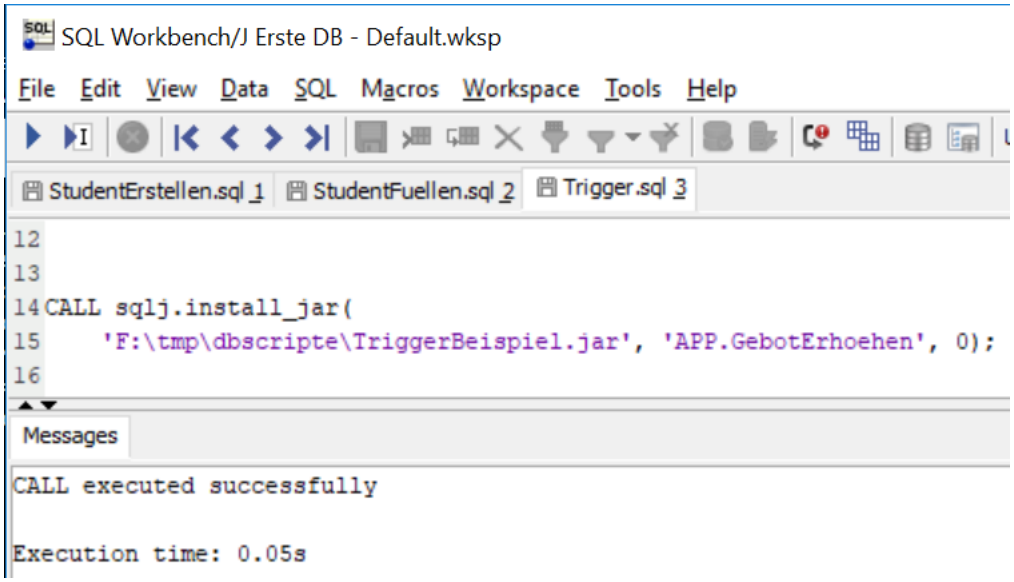


Es muss unter „JAR file“ mit Hilfe des Knopfs „Browse…“ ein passender Speicherort und Name vergeben werden. Zum Abschluss wird „Finish“ gedrückt.



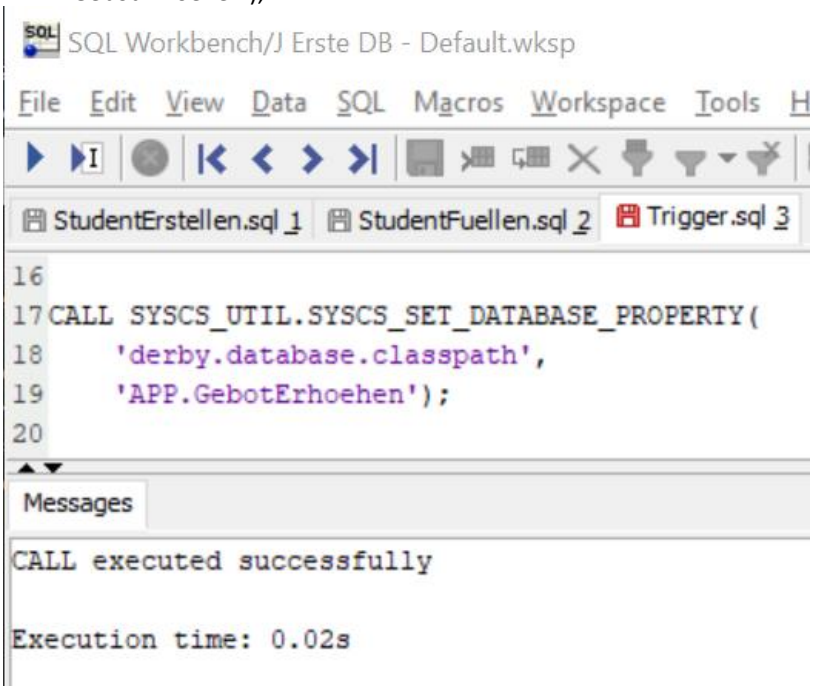
Im nächsten Schritt wird die jar-Datei in die Datenbank geladen, dazu wird ein Befehlsfenster für die Datenbank aufgemacht und der folgende Befehl eingegeben, bei dem der vorher gemerkte Pfad eine Rolle spielt. Der weiterhin angegebene Name „APP.GebotErhoehen“ ist ein relativ willkürlicher Bezeichner, der nur eindeutig in der Datenbank sein muss. Durch den Präfix „APP.“ findet eine Zuordnung zu einem Schema statt, die hier keine Rolle spielt.

```
CALL sqlj.install_jar(
  'F:\tmp\dbscripte\TriggerBeispiel.jar', 'APP.GebotErhoehen', 0);
```



Sollte es eine AccessControlException geben wurde Derby wahrscheinlich ohne Parameter gestartet und die Datenbank ist zunächst zu stoppen und dann mit Parameter zu starten.
Wurde die jar-Datei erfolgreich zur Datenbank hinzugefügt, muss die Datei zu den ausführbaren Dateien der Datenbank, genauer zu deren Pfad, hinzugefügt werden, was mit dem folgenden Befehl erfolgt. Der vorher ausgewählte Name „APP.GebotErhoehen“ ist hier wieder zu nutzen. Es wird nur der letzte Befehl im Fenster ausgeführt.

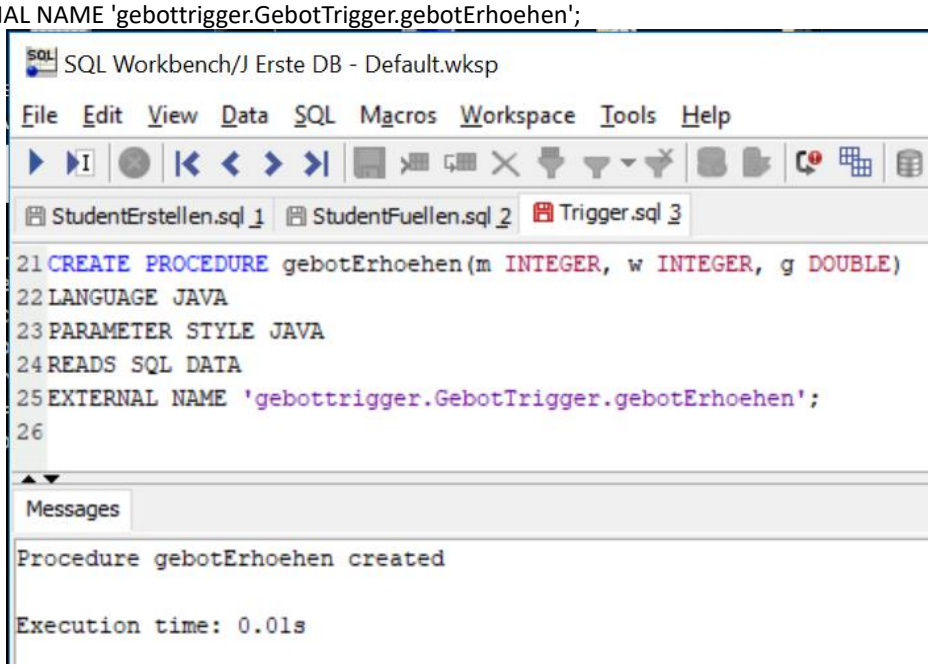
```
CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY(
  'derby.database.classpath',
  'APP.GebotErhoehen');
```



Nun muss eine Prozedur angelegt werden, mit der die Java-Methode aufrufbar wird. Dabei müssen

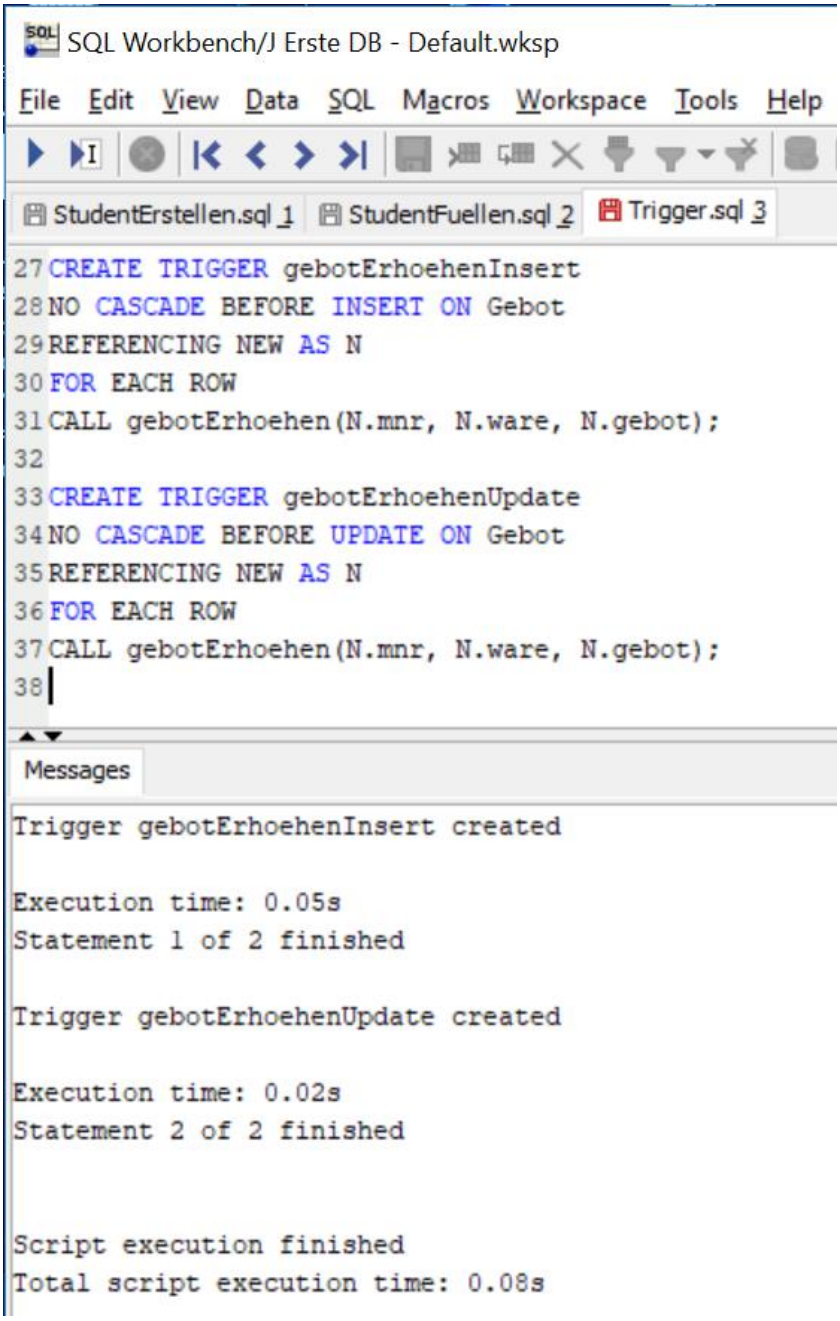
die Parameter vom Typ zu den Parametern der Java-Methode passen. Der Name der Prozedur ist frei wählbar, der externe Name ergibt sich aus dem vollqualifizierten Klassennamen, hier gebottrigger.GebotTrigger und dem Methodennamen. Die Ergänzung „READS SQL DATA“ ist wichtig und verweist darauf, dass in der Prozedur in der Datenbank gelesen, die Daten aber nicht modifiziert werden. Auf etwaige Tippfehler in Namen wird man an dieser Stelle leider nicht hingewiesen.

```
CREATE PROCEDURE gebotErhoehen(m INTEGER, w INTEGER, g DOUBLE)
LANGUAGE JAVA
PARAMETER STYLE JAVA
READS SQL DATA
EXTERNAL NAME 'gebottrigger.GebotTrigger.gebotErhoehen';
```



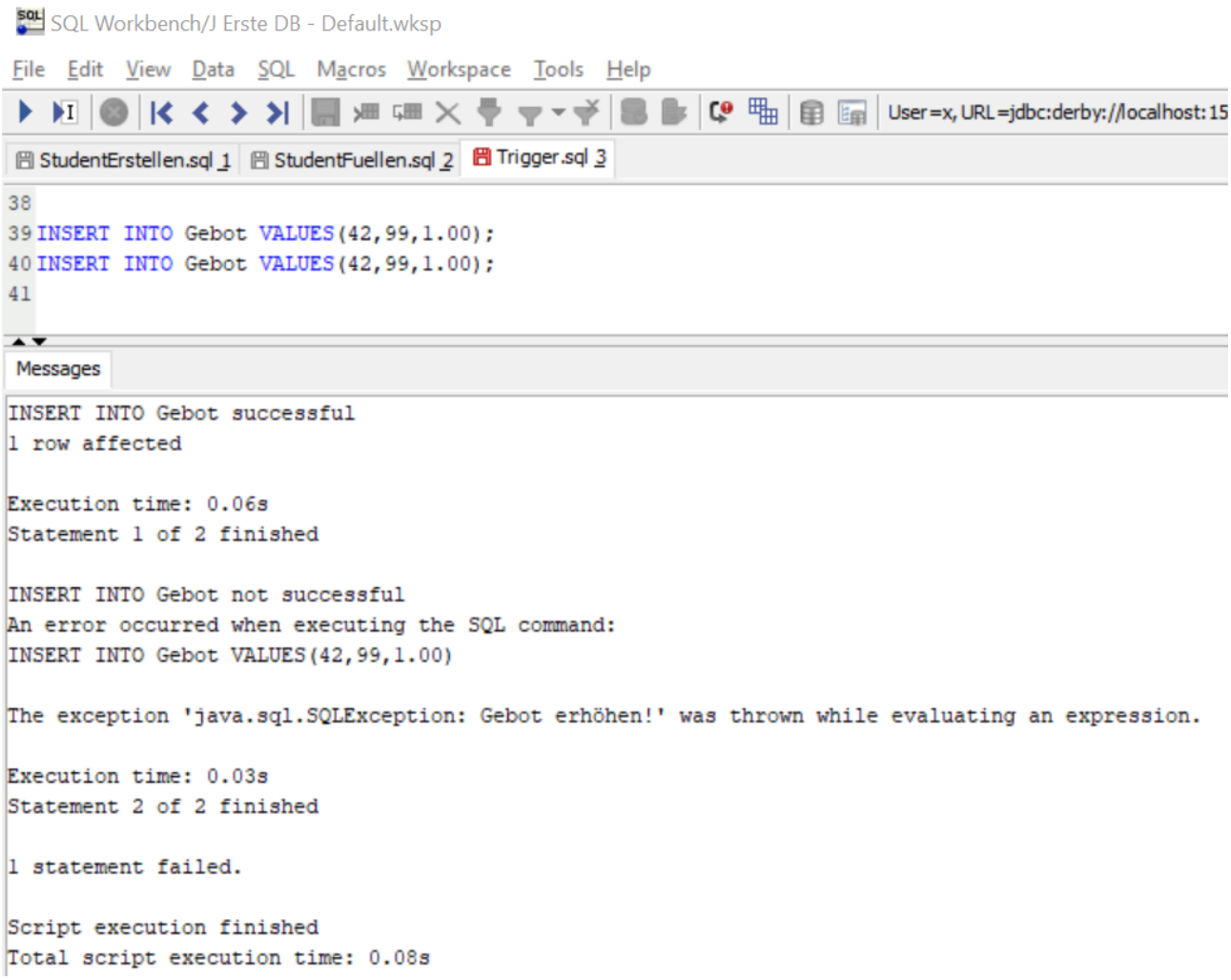
Abschließend wird der Trigger mit dem dazu passenden Event angelegt. Im konkreten Fall wird die vorher angelegte Prozedur aufgerufen.

```
CREATE TRIGGER gebotErhoehenInsert
NO CASCADE BEFORE INSERT ON Gebot
REFERENCING NEW AS N
FOR EACH ROW
CALL gebotErhoehen(N.mnr, N.ware, N.gebot);

CREATE TRIGGER gebotErhoehenUpdate
NO CASCADE BEFORE UPDATE ON Gebot
REFERENCING NEW AS N
FOR EACH ROW
CALL gebotErhoehen(N.mnr, N.ware, N.gebot);
```

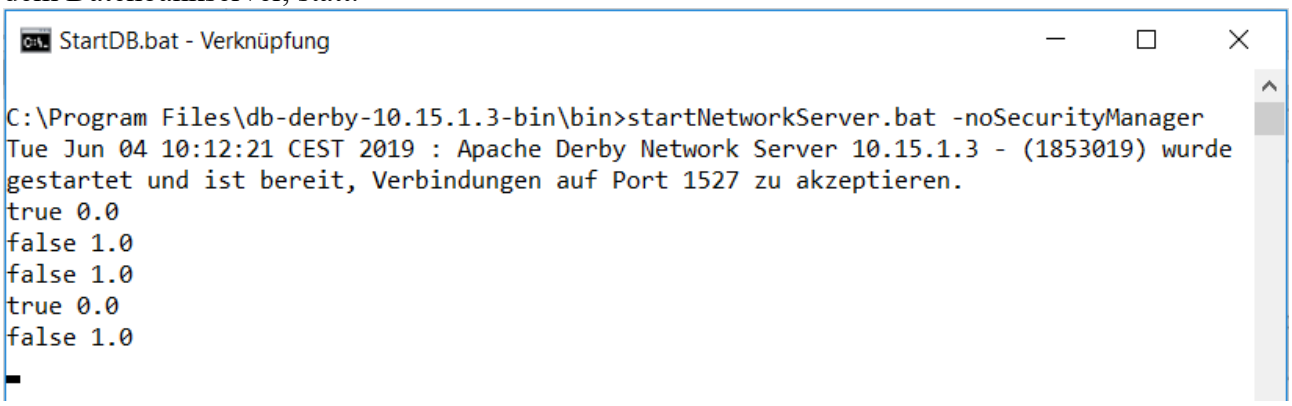


Danach können die Trigger ausprobiert werden. Der zweite Befehl erzeugt dabei den erwarteten Abbruch. Nebenbei wird deutlich, dass die Trigger-Prüfung vor der Constraint-Prüfung, hier dem Primary Key, erfolgt.

```
INSERT INTO Gebot VALUES(42,99,1.00);
INSERT INTO Gebot VALUES(42,99,1.00);
```



Die Ausgaben von Prozeduren und Funktionen finden in der Konsole der Datenbank, vereinfacht auf dem Datenbankserver, statt.



In der Realität ist es sehr unwahrscheinlich, dass die erste gewählte Lösung funktioniert, oft muss dabei das Java-Programm angepasst werden. Dazu sind nicht genau die bisher beschriebenen Schritte zu nutzen. Es sind folgende Schritte durchzuführen.

1. Änderung des Java-Programms
2. Erstellung einer neuen jar-Datei
3. Die jar-Datei muss in der Datenbank ersetzt werden, dazu wird der folgende Befehl genutzt

```
CALL sqlj.replace_jar(
  'F:\tmp\dbscripte\TriggerBeispiel.jar'
  , 'APP.GebotErhoehen');
```

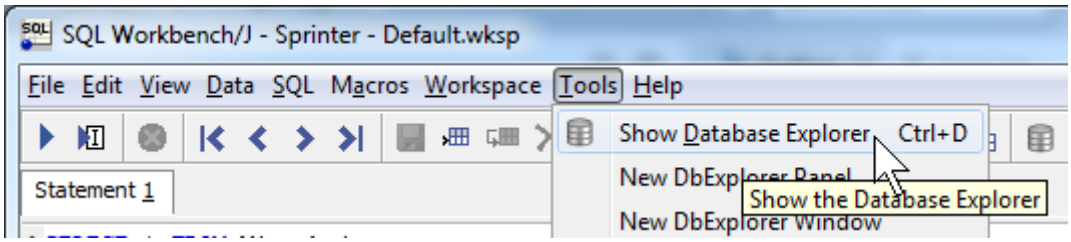
4. Die anderen Schritte sind nicht zu wiederholen, der Trigger kann direkt neu getestet werden, wobei eventuell die vorher eingefügten Daten zu löschen sind.

Generell sollte während der Arbeit einfach eine Datei mit allen wichtigen Datenbankbefehlen in einem Reiter jederzeit verfügbar sein, so dass die Befehle schnell ausführbar sind. Will man eine Prozedur oder einen Trigger in der Struktur verändern, müssen diese vorher mit einem DROP-Befehl gelöscht und dann wieder angelegt werden.

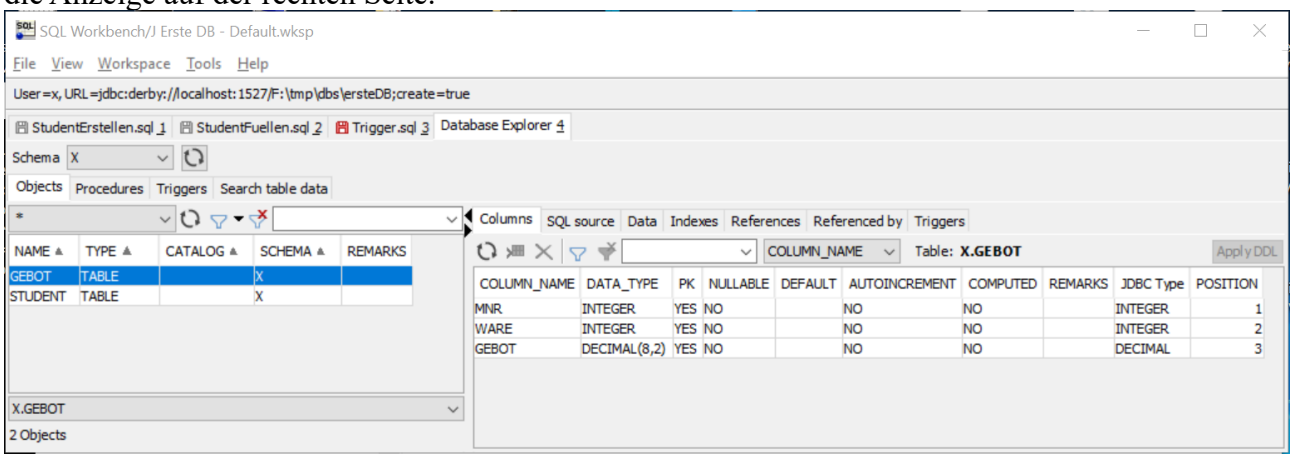
```
DROP Trigger gebotErhoehenInsert;
DROP Trigger gebotErhoehenUpdate;
```

1.10 Weitere Möglichkeiten von SQL-Workbench

Generell bietet SQL-Workbench eine Vielzahl von Funktionalitäten, die man sich selbst erschließen kann. Auf eine Wichtige wird hier genauer hingewiesen.
Zum reinen Betrachten der Tabellen wird unter „Tools“ dann „Show Database Explorer“ genutzt, für den es auch einen Knopf in der Knopfleiste gibt.

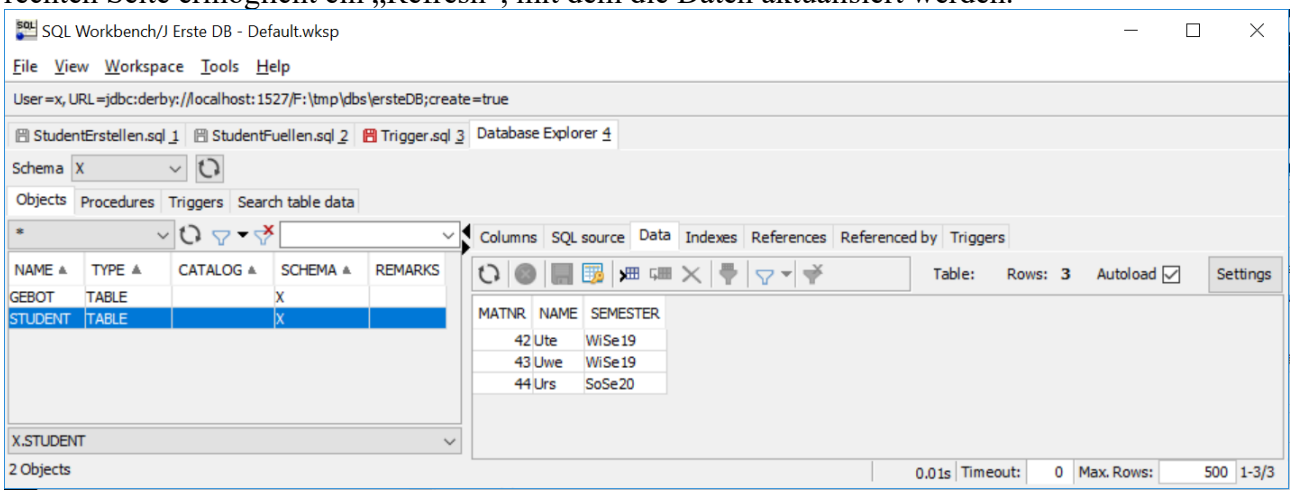


Auf der linken Seite werden alle Tabellen angezeigt. Klickt man auf eine der Tabellen, ändert sich die Anzeige auf der rechten Seite.



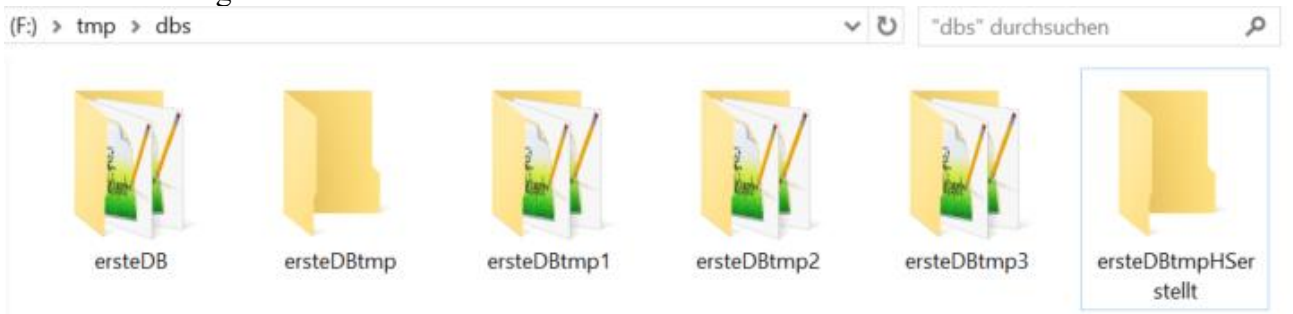
Der mittlere Balken ist etwas holperig über die kleinen Pfeile steuerbar, so dass auch die folgende Abbildung möglich wird. Dabei kann man auf der rechten Seite durch die oberen Reiter die

Informationen auswählen, die man betrachten möchte. Das kleine kreisartige Symbol links auf der rechten Seite ermöglicht ein „Refresh“, mit dem die Daten aktualisiert werden.

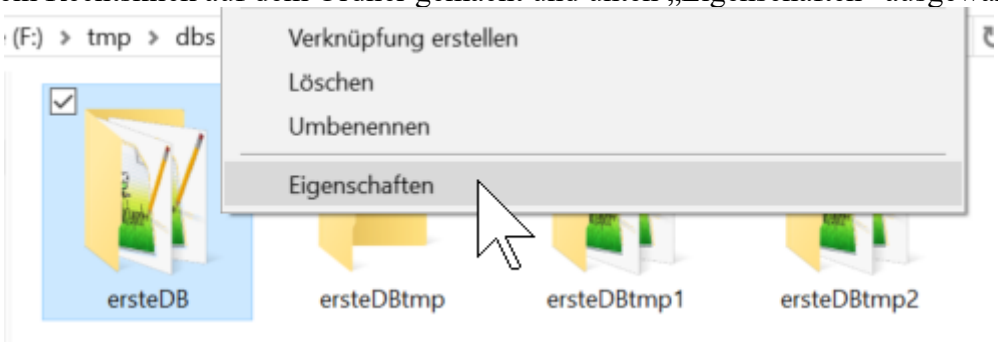


1.11 Datenbanken auf unterschiedlichen Rechnern öffnen

Wird eine Datenbank mit einem Nutzeraccount erstellt und soll diese Datenbank unter einem anderen Nutzer geöffnet werden, verweigert Derby den Zugriff. Ein guter Indikator, ob ein Zugriff möglich ist, ist die Ansicht der Ordner der Datenbanken als Graphiken im Datei-Browser. Das folgende Bild zeigt sechs Beispieldatenbanken, von denen nur vier potenziell nutzbar sind, da kein weiterer Ordnerinhalt dargestellt wird.

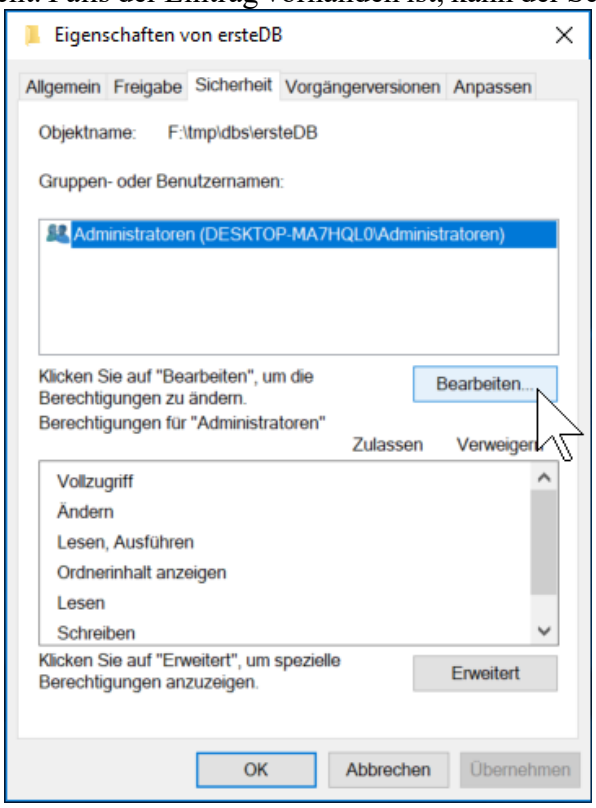


Es müssen durch den *Ersteller der Datenbank* die Rechte an dem Datenbankordner geändert werden. Dazu wird ein Rechtsklick auf dem Ordner gemacht und unten „Eigenschaften“ ausgewählt.

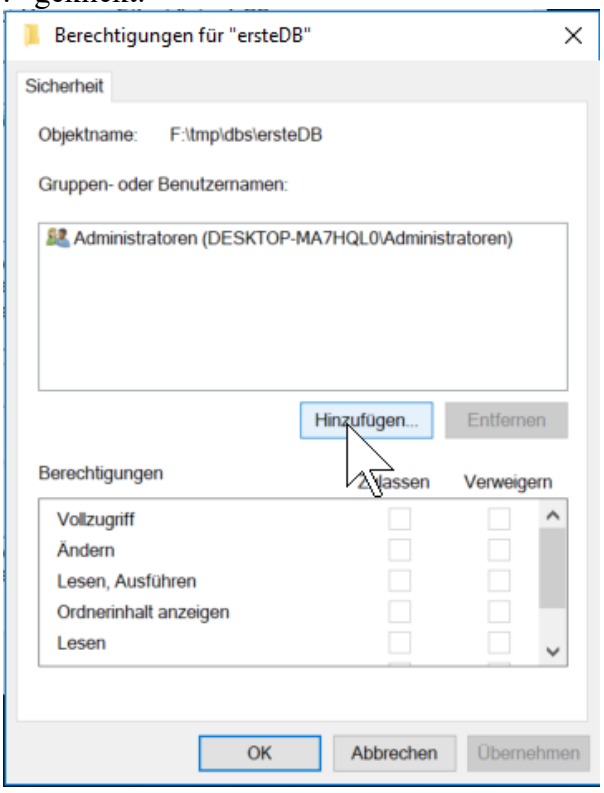


Es wird der Reiter „Sicherheit“ angeklickt und falls im oberen Feld kein Eintrag „Jeder“ steht, ein

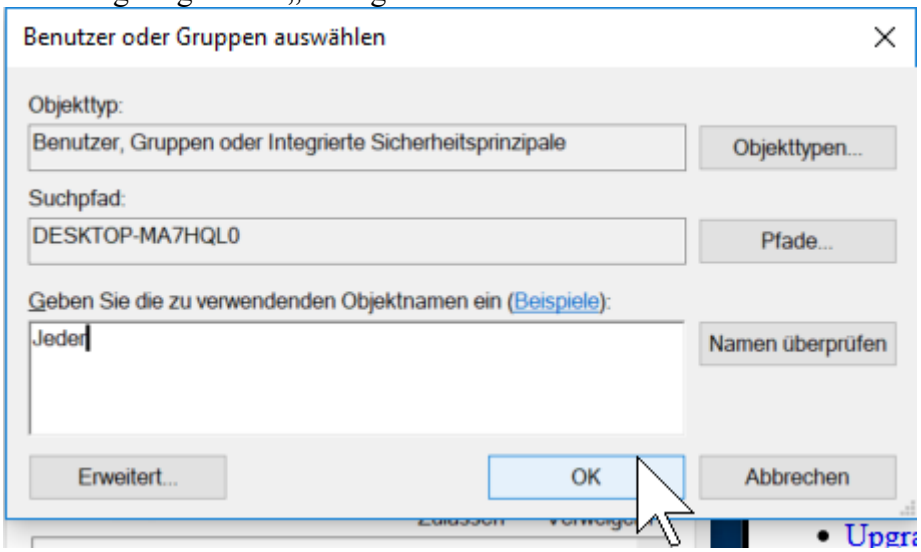
Klick auf Bearbeiten gemacht. Falls der Eintrag vorhanden ist, kann der Schritt übersprungen werden.



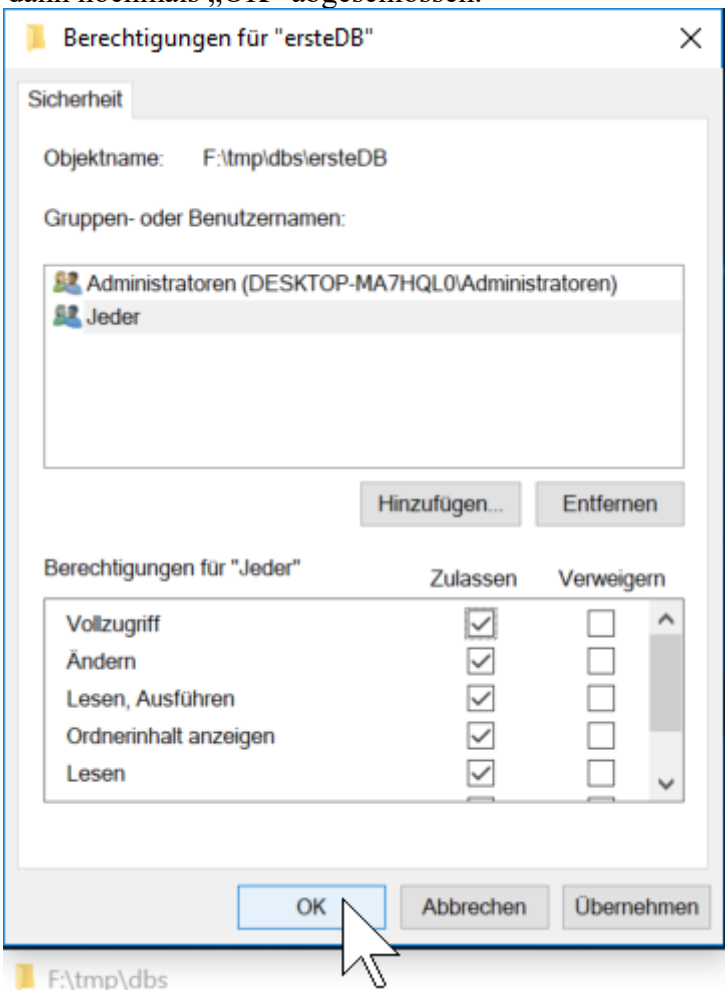
Es wird auf „Hinzufügen…“ geklickt.



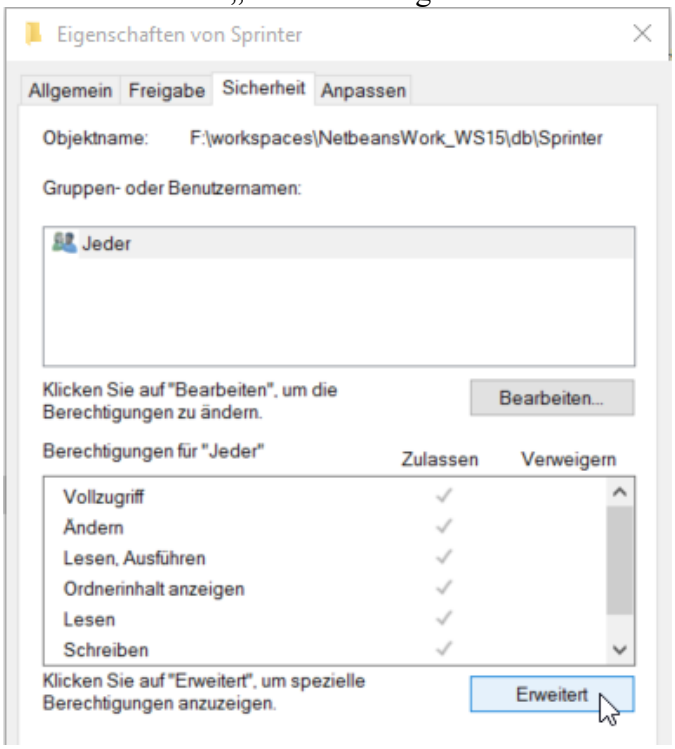
Unten wird „Jeder“ eingetragen und „OK“ geklickt.



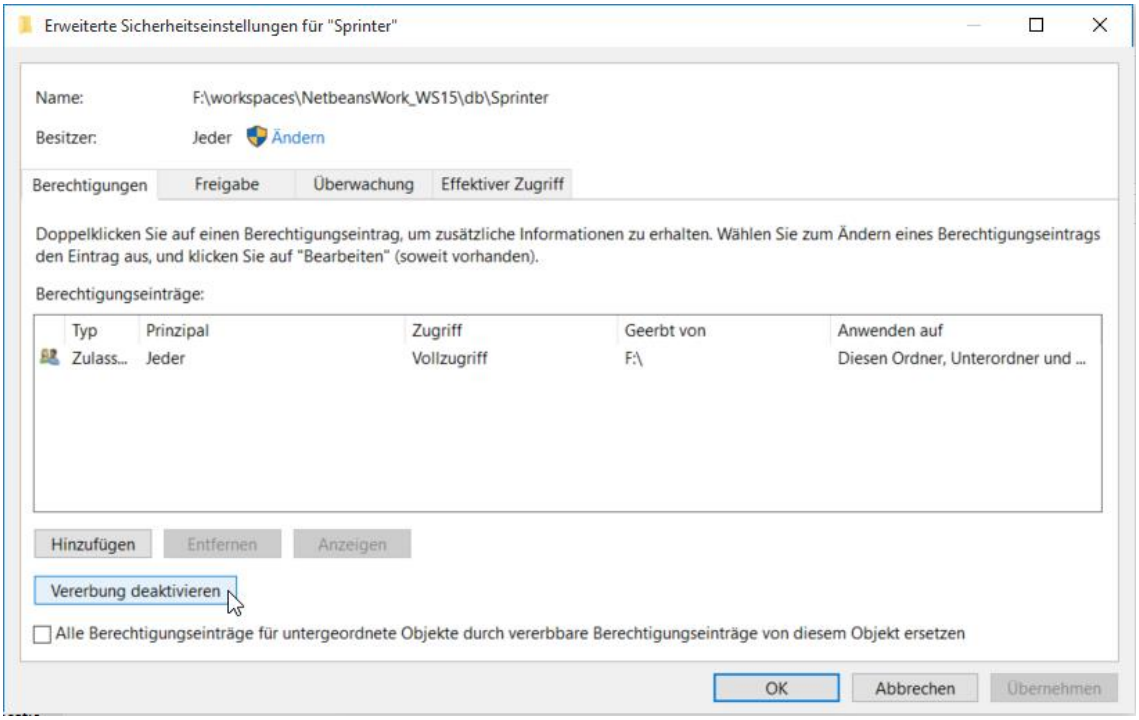
Es wird oben „Jeder“ angeklickt“ und unten ein Kreuz bei „Vollzugriff Zulassen“ gesetzt und die Aktion mit „OK“ und dann nochmals „OK“ abgeschlossen.

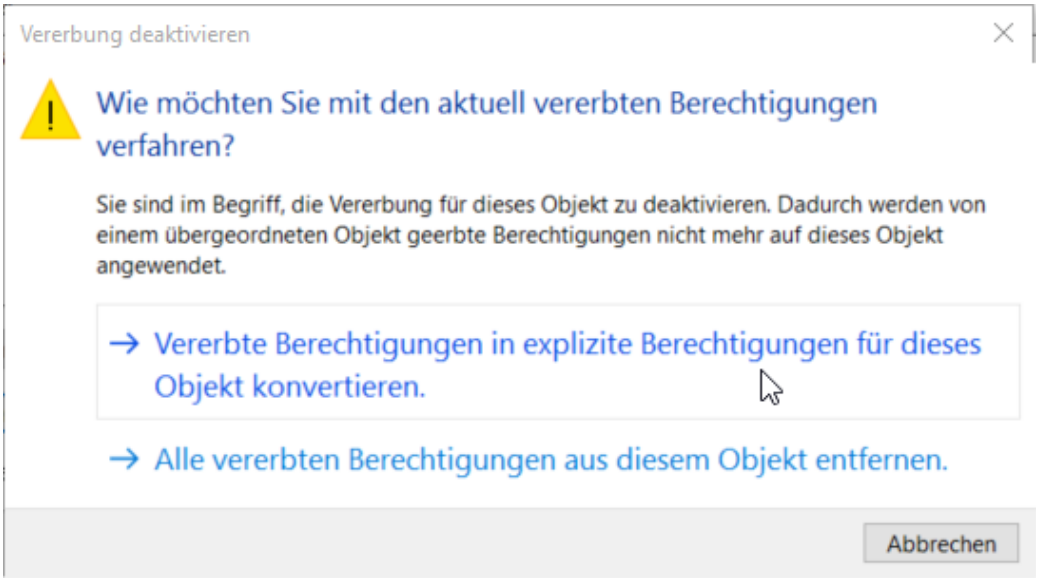


Es wird der Reiter „Sicherheit“ und dann „Erweitert“ angeklickt.

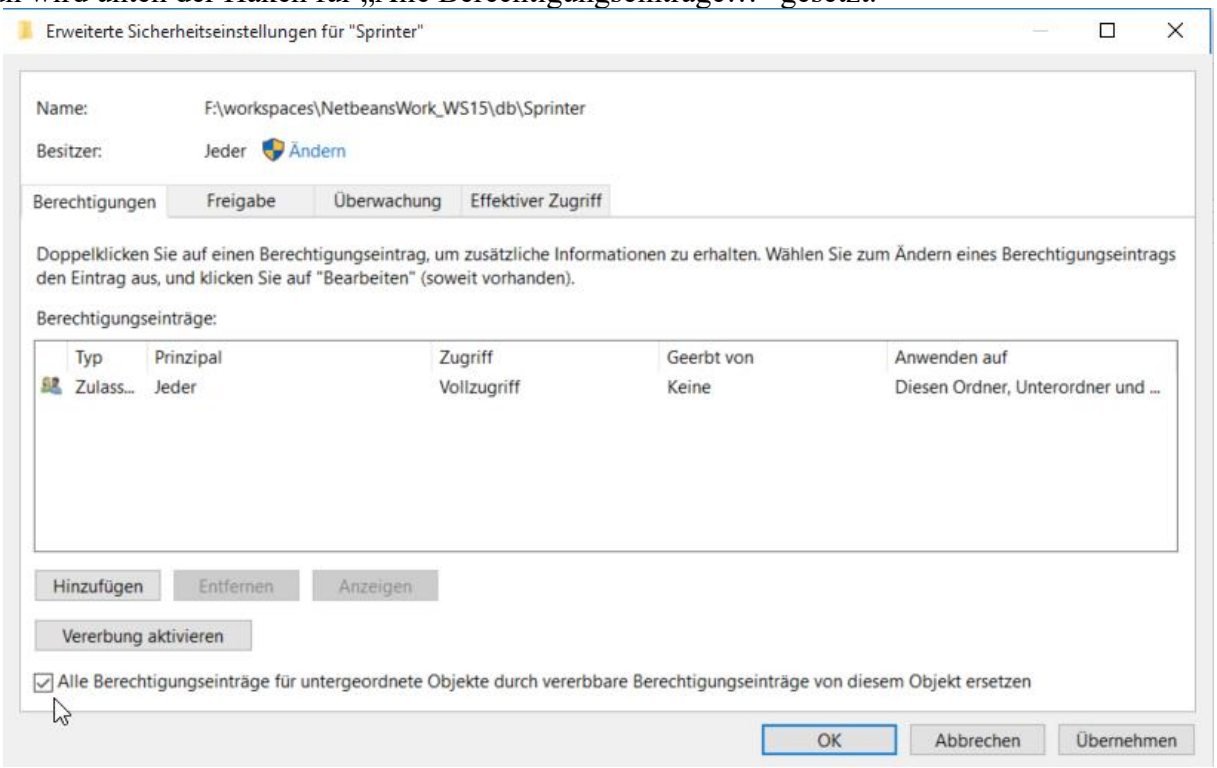


Insofern unten „Vererbung deaktivien“ auf dem Knopf steht, wird zunächst ein Klick auf „Vererbung deaktivieren“ gemacht und danach der erste Eintrag mit „konvertieren“ gewählt. Steht auf dem Knopf bereits „Vererbung aktivieren“ wird der Schritt übersprungen und mit dem „Kreuz“ links-unten fortgesetzt.

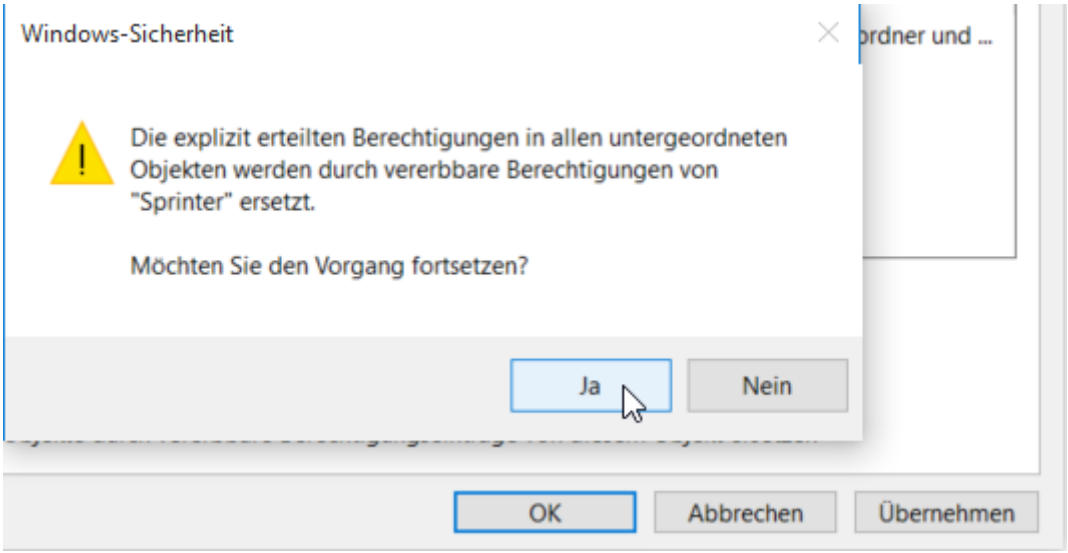




Nun wird unten der Haken für „Alle Berechtigungseinträge…“ gesetzt.



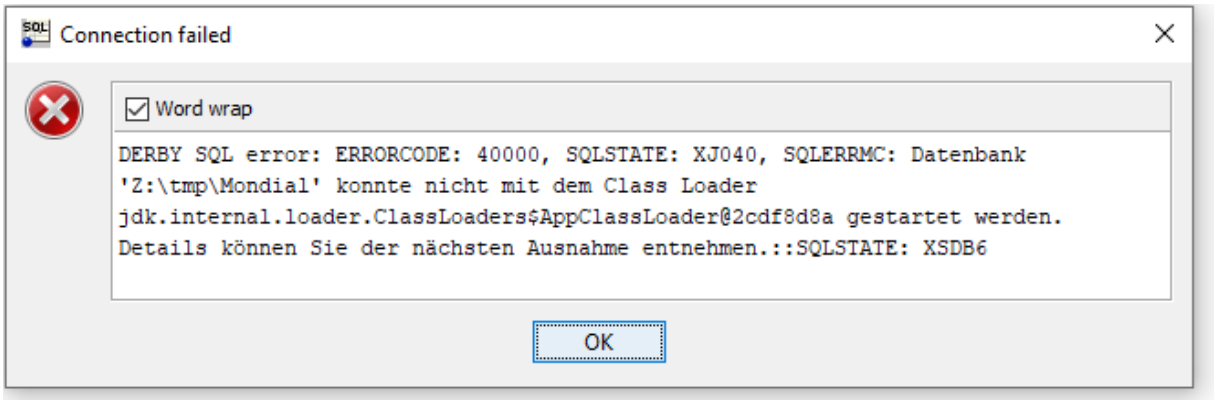
Nach einem Klick auf „Ok“ müssen die Änderungen mit „Ja“ bestätigt werden.



Danach sollte die Datenbank für beliebige Nutzer zugreifbar sein.

1.12 Versuch existierende Datenbank zu öffnen scheitert

Bevor der hier dargestellte Ansatz genutzt wird, solle zunächst geprüft werden, ob die Verbindungsdaten, insbesondere der Connection-String in Ordnung sind. Trotzdem kann folgende Fehlermeldung auftreten.

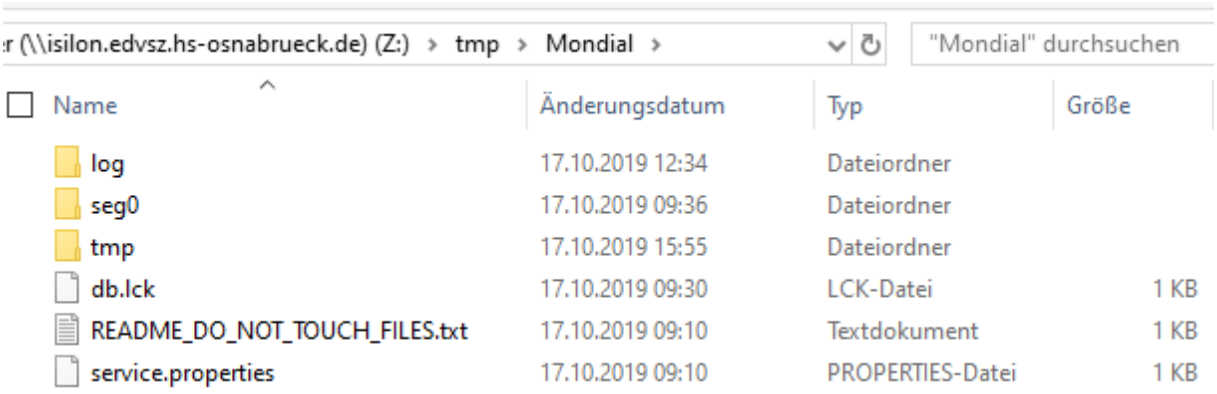


Der Hintergrund dieses Problems beim DB-Start ist, dass angeblich eine andere DB-Software auf die DB zugreift. Die genauen Fehlermeldungen sind

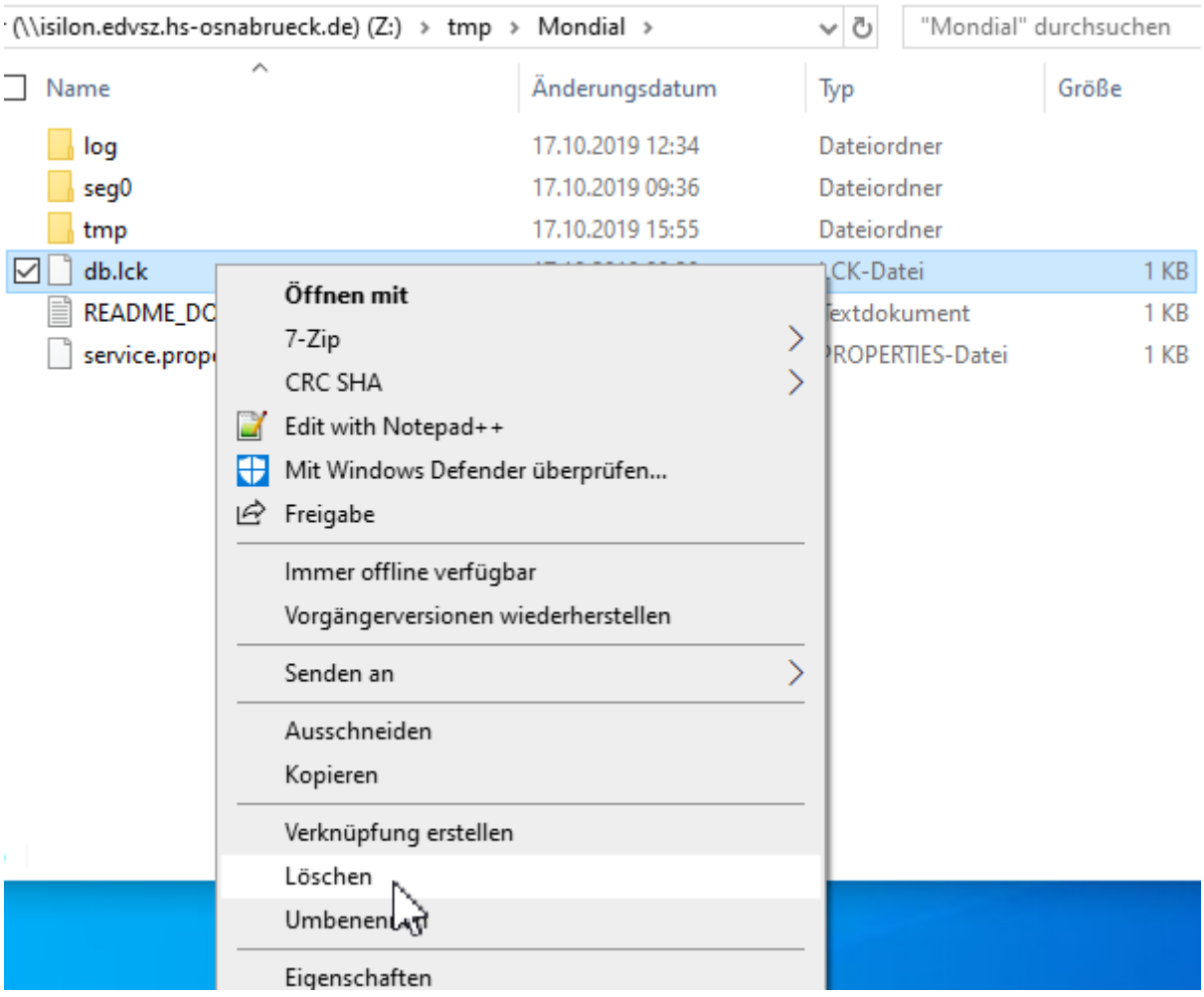
XJ040 Failed to start database '<databaseName>', see the next exception for details.

XSDB6 Another instance of Derby may have already booted the database <value>.

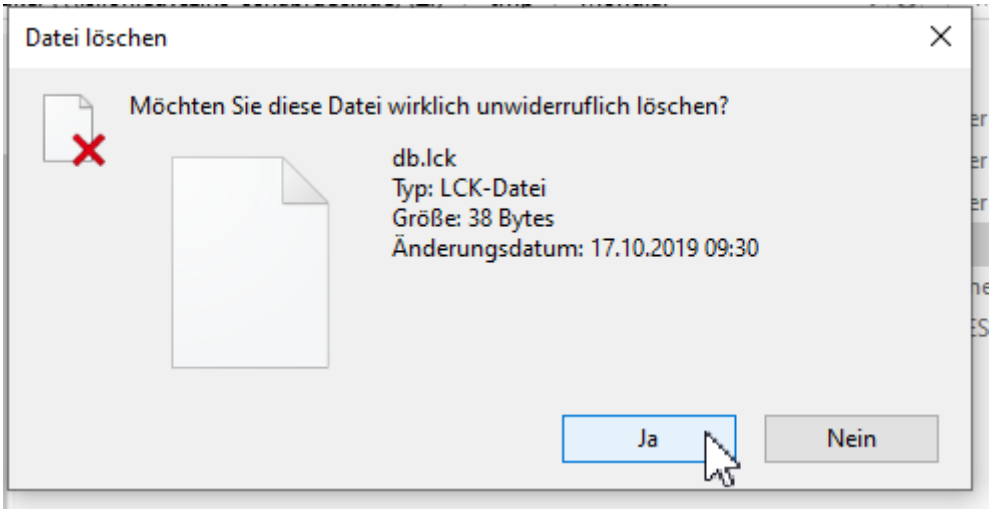
Fehlermeldungen können unter https://db.apache.org/derby/docs/10.1/ref/rrefexcept71493.html nachgelesen werden. Es muss die Datei db.lck im Verzeichnis der Datenbank (Datenbankname entspricht Dateiordnername) gelöscht werden, dabei ist vorher die Datenbank zu stoppen und ggfls. sind über den Task-Manager alle Java-Prozesse zu terminieren.



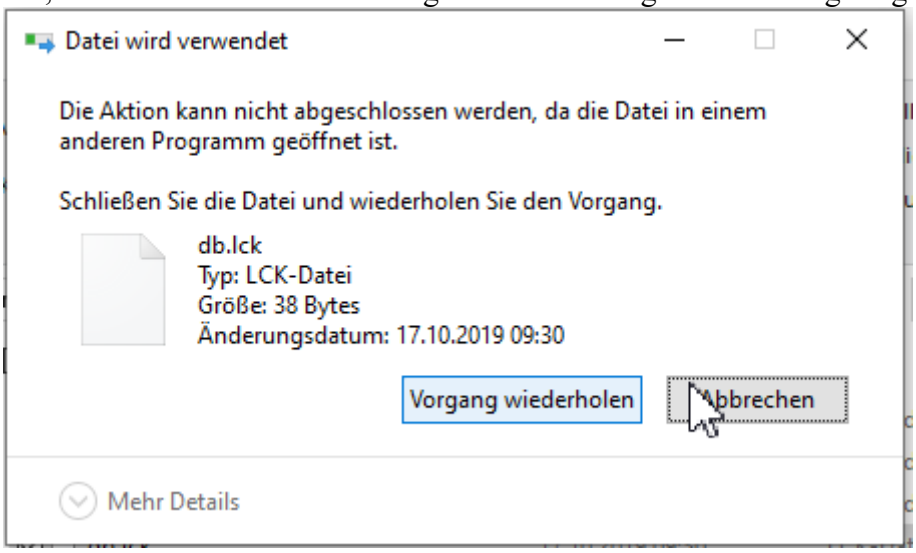
Zunächst wird direkt versucht die kritische Datei über einen Rechtsklick zu löschen.



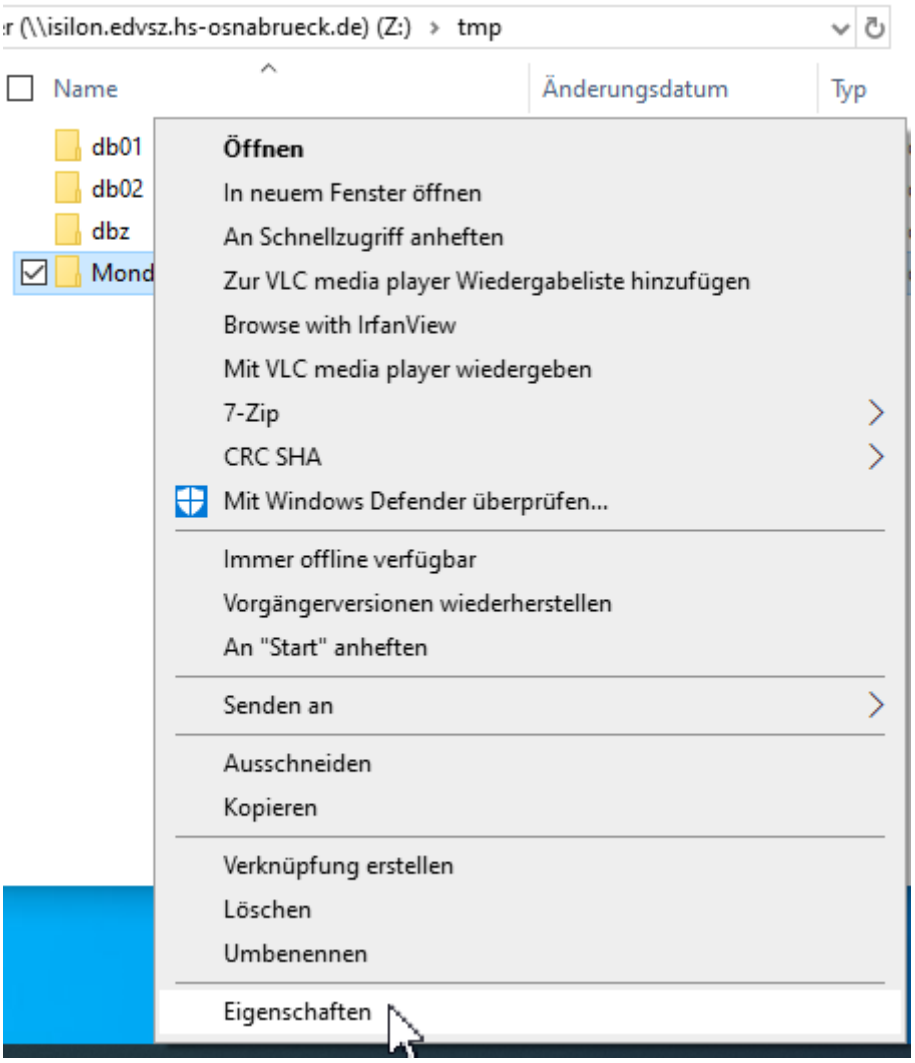
Das Löschen wird mit „Ja“ bestätigt.



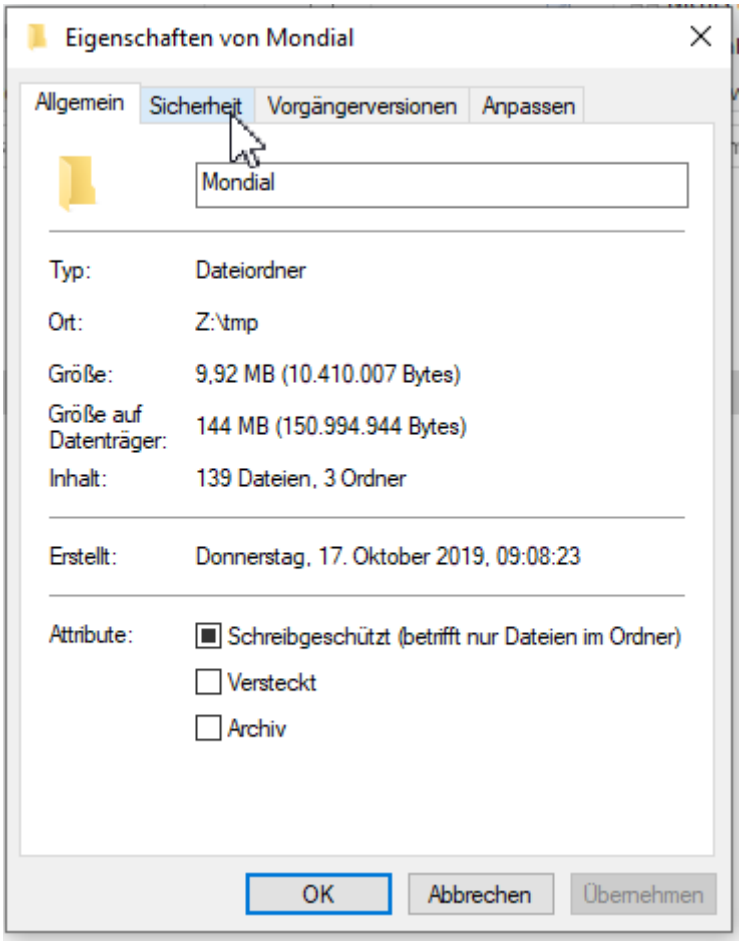
Es kann passieren, dass das Löschen nicht erfolgreich ist und folgende Meldung ausgegeben wird.



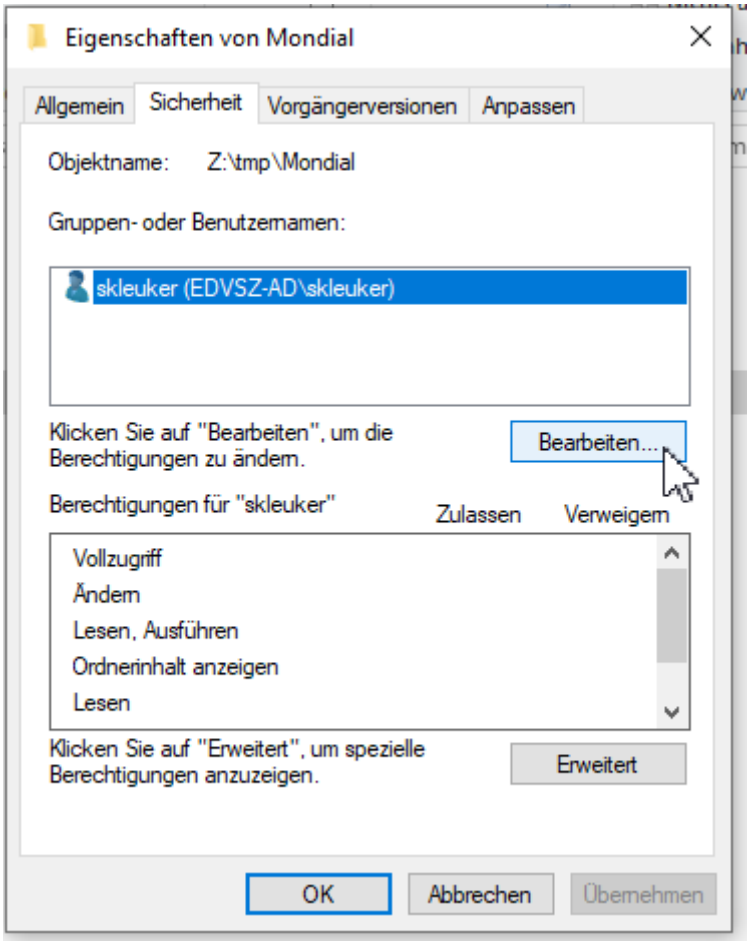
Bei solchen Problemen sollten zunächst die Rechte auf dem Datenbankverzeichnis geklärt werden. Dies erfolgt durch einen Rechtsklick auf dem Ordner und der Auswahl „Eigenschaften“.



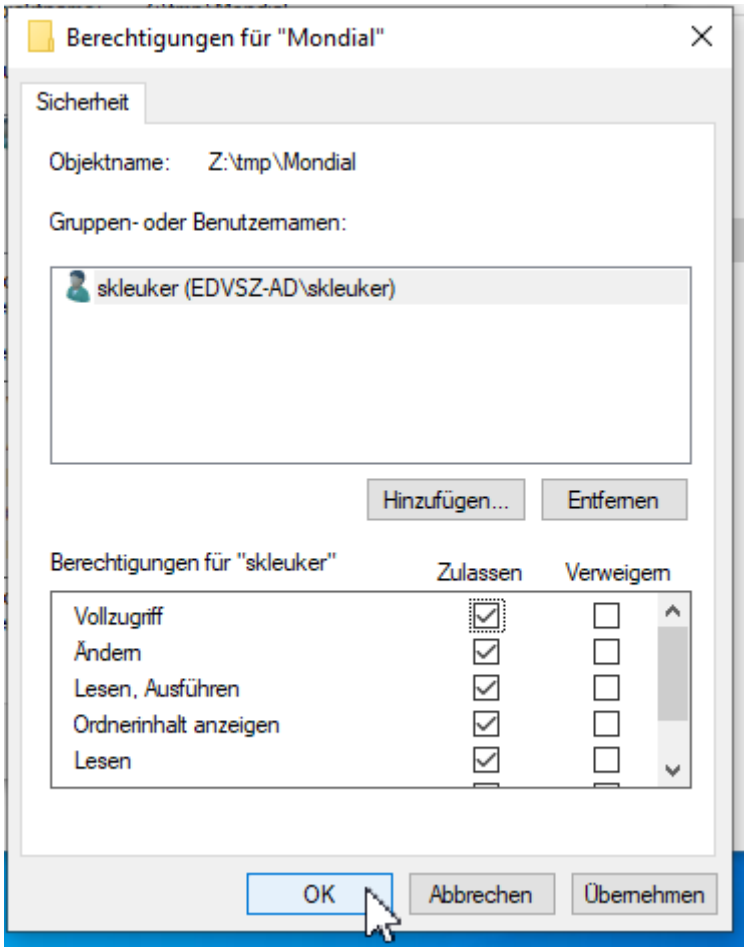
Das angeklickte Feld „Schreibgeschützt“ sieht schon „nicht gut“ aus. Zur Änderung wird oben der Reiter „Sicherheit“ genutzt.



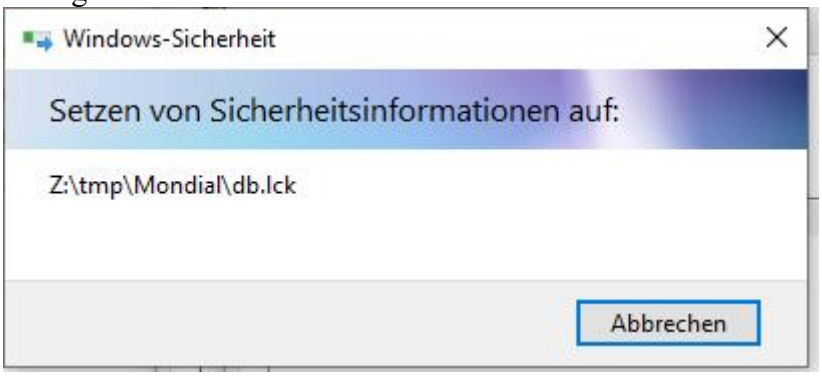
Es kann etwas dauern, bis ein kryptischer String im oberen Feld beim Nutzer in den Nutzernamen aufgelöst wird. Es wird auf „Bearbeiten…“ geklickt.



Es wird ein Haken bei „Vollzugriff“ „Zulassen“ gesetzt und „OK“ geklickt. Die anderen Haken werden dadurch automatisch gesetzt.



Es wird eine Meldung ausgegeben, was passiert. Zum momentanen Zeitpunkt erfolgen die Änderungen extrem langsam.



Danach sollte die Datei db.lck löschbar sein.

2 UMLet

UMLet ist ein recht intuitiv zu bedienendes Werkzeug zur Erstellung von UML-Diagrammen, das auch für ER-Diagramme genutzt werden kann. Dabei handelt es sich um ein Skizzenwerkzeug, das z. B. keine Code-Erzeugung oder Reverse-Engineering ermöglicht. Der wesentliche Vorteil neben der recht schnellen Erstellung, ist die Möglichkeit beliebige Diagrammelemente in einem Diagramm zu kombinieren, was gerade bei frühen Projektüberlegungen sinnvoll sein kann. UMLet gibt es als Eclipse-Plugin und als Standalone-Version, die hier betrachtet wird.

2.1 Installation

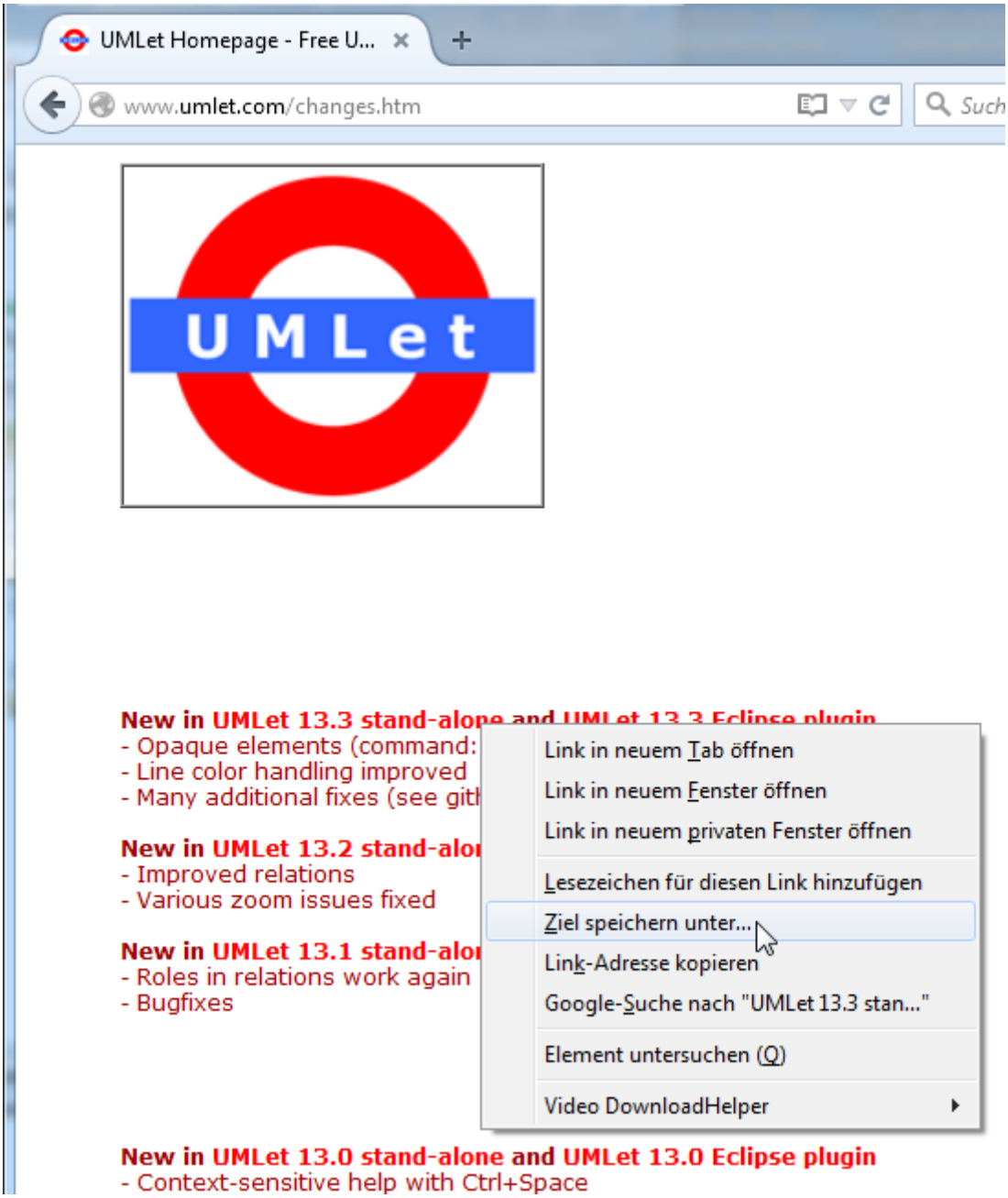
Hinweis: Im Text wird die Version 13.3 von UMLet genutzt.

In der Veranstaltung wird das Werkzeug UMLet für die Erstellung von Entity-Relationship-Diagrammen genutzt. Das Werkzeug kann von der Web-Seite http://www.umlet.com/ geladen werden.

Auf der Startseite wird der Download-Link genutzt.

Die Download-Seite ist http://www.umlet.com/changes.htm.

Hier wird die aktuelle Stand-Alone-Version mit einem Rechtsklick heruntergeladen.



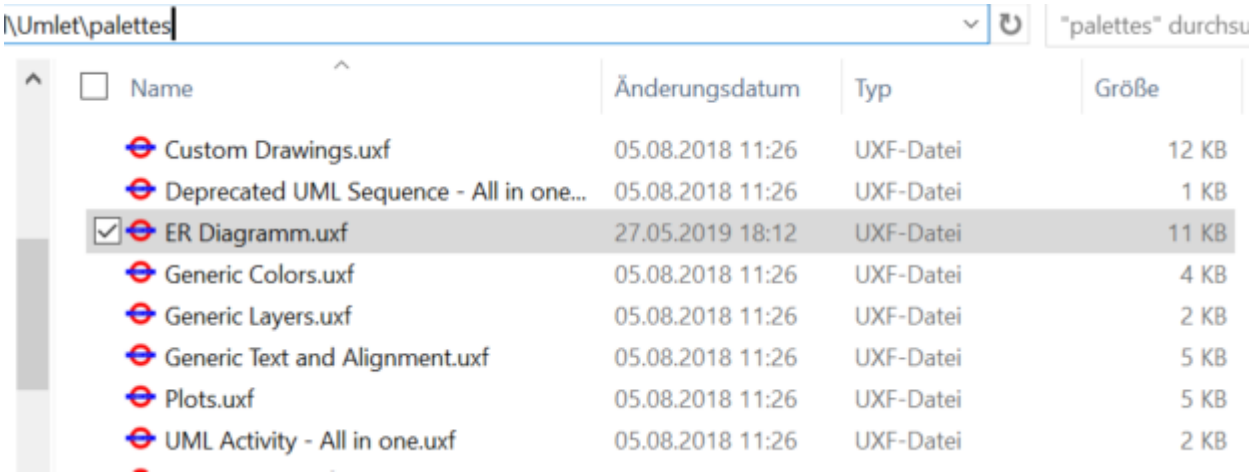
2.2 Installation der ER-Erweiterung

Da es keine direkt nutzbaren ER-Diagrammsymbole gibt, wurde eine Palette mit Symbolen ergänzt. Dies kann als Datei von

http://home.edvsz.hs-osnabrueck.de/skleuker/querschnittlich/ER%20Diagramm.uxf

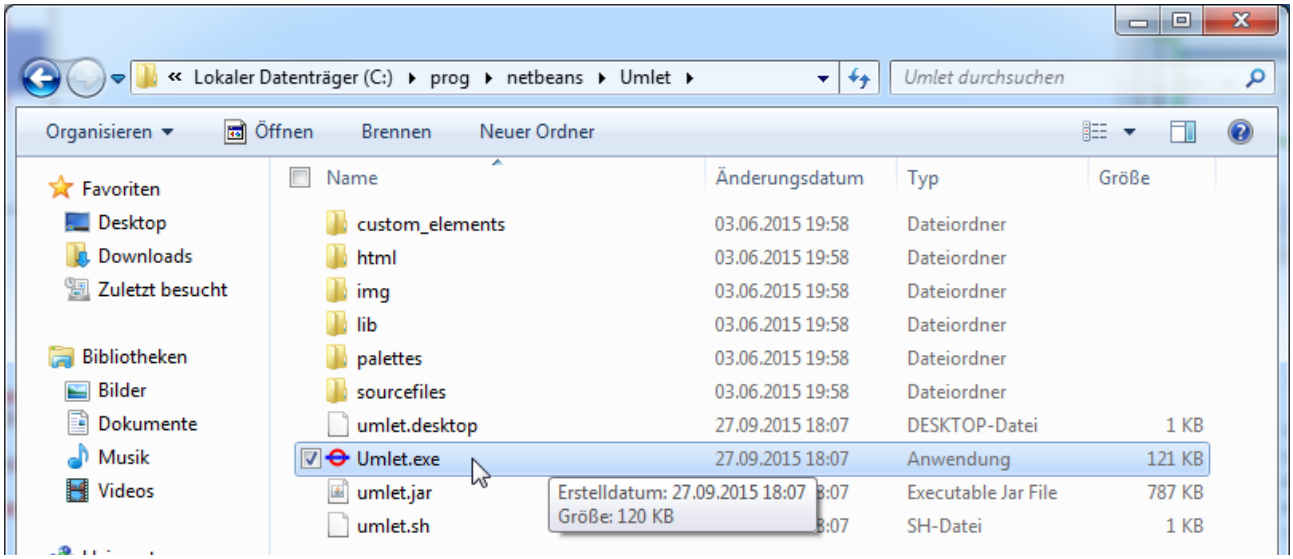
heruntergeladen werden.

Diese Datei wird in den Unterordner palettes der UMLet-Installation kopiert, um die Installation abzuschließen.

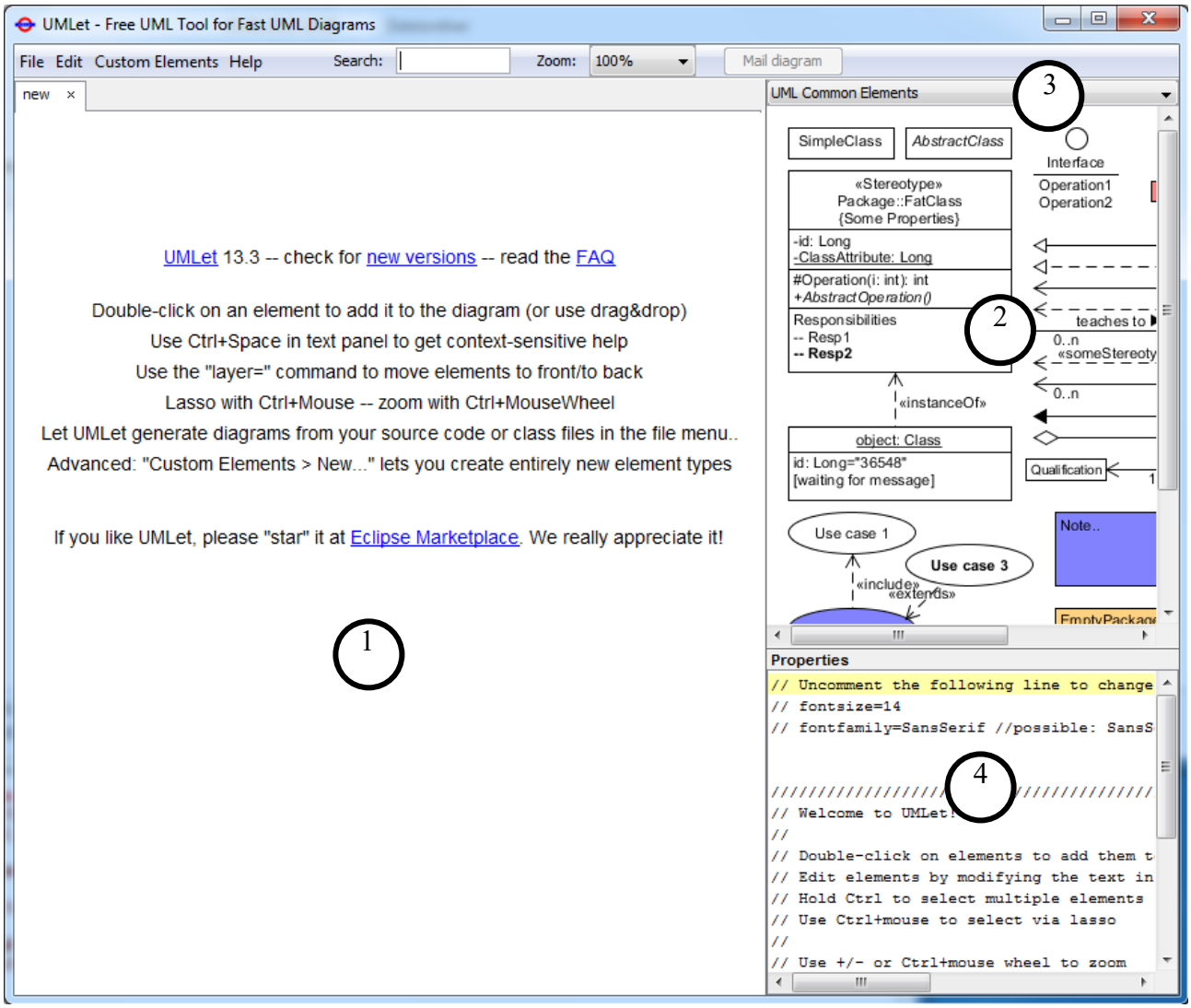


2.3 Erste Nutzung

Zum direkten Aufruf des Werkzeugs wird die zip-Datei ausgepackt und Umlet.exe mit einem Doppelklick gestartet. Alternativ kann in das Verzeichnis mit einer „Dos-Box“ (cmd-Werkzeug) oder unter anderen Betriebssystemen mit einem Konsolen-Fenster gegangen und das jar-File von Hand mit java -jar umlet.jar gestartet werden.



Da das Fenster relativ klein zum Arbeiten ist, sollte dieses Fenster vergrößert werden.

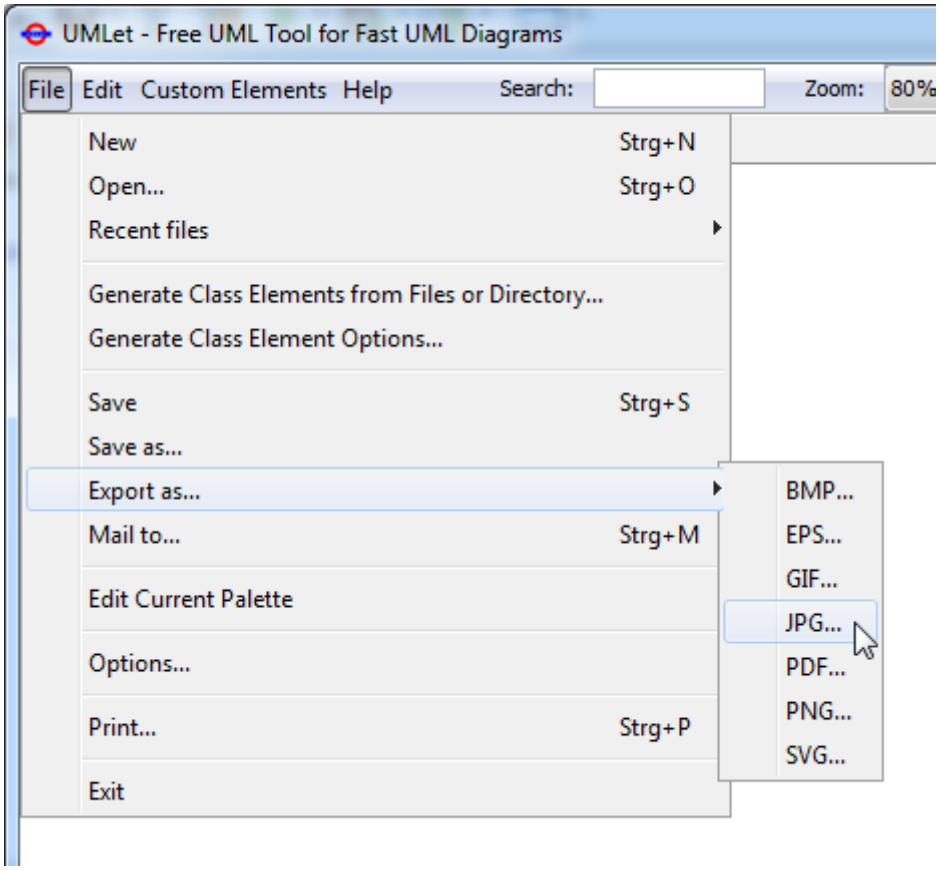


Im Werkzeug gibt es neben der üblichen Dateibehandlung die vier markierten wichtigen Bereiche.

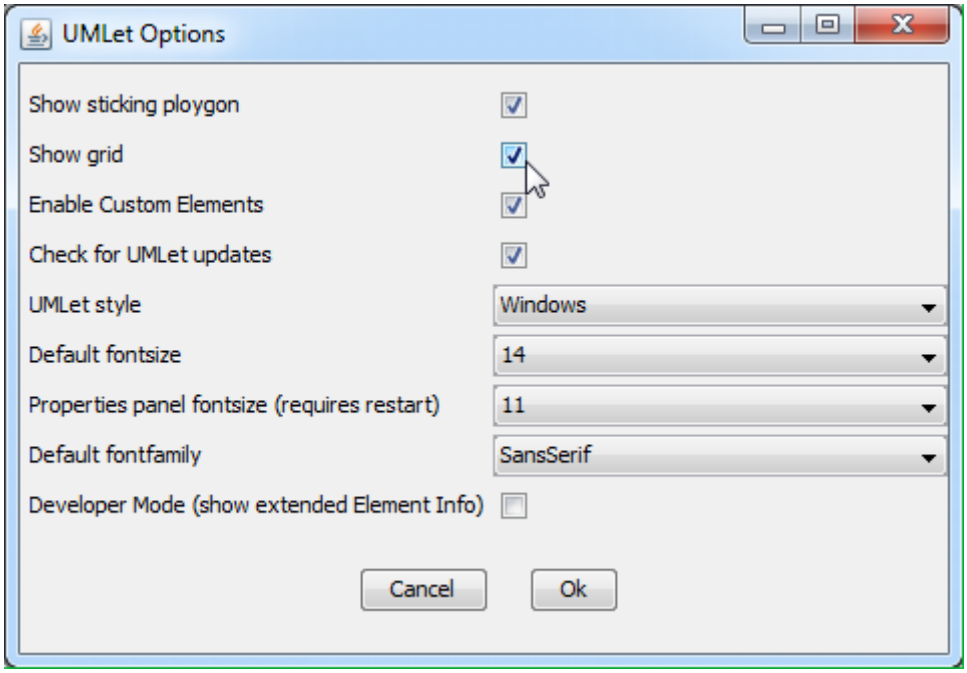
1. Dies ist der Zeichenbereich, in dem alle Elemente verknüpft und angezeigt werden.
2. In diesem Bereich wird eine Palette vom UML-Zeichenelementen angezeigt, die durch einen Doppelklick in den Zeichenbereich übernommen werden. (Diese Art der Steuerung ist etwas gewöhnungsbedürftig, aber recht effizient.)
3. In dieser Drop-Down-Box kann man unterschiedliche Paletten auswählen, die für unterschiedliche UML-Diagramme verschiedene Zeichenelemente anbieten.
4. Wenn man Beschriftungen z. B. von Aktivitäten oder Inhalte von Klassen ändern möchte, passiert dies immer in dieser Textbox.

Das File-Menü bietet die Möglichkeit die Datei unter einem eigenen Namen unter „Save As“ abzuspeichern. Weiterhin gibt es hier die Möglichkeit, das Diagramm in verschiedene graphische

Formate zu exportieren. Dabei ist es immer sinnvoll, das Diagramm möglichst genau links-oben im Editor zu platzieren.



Weiterhin gibt es hier das Options-Menü, bei dem u. a. Gitterlinien eingeblendet werden können, die bei der Ausrichtung von Elementen helfen.

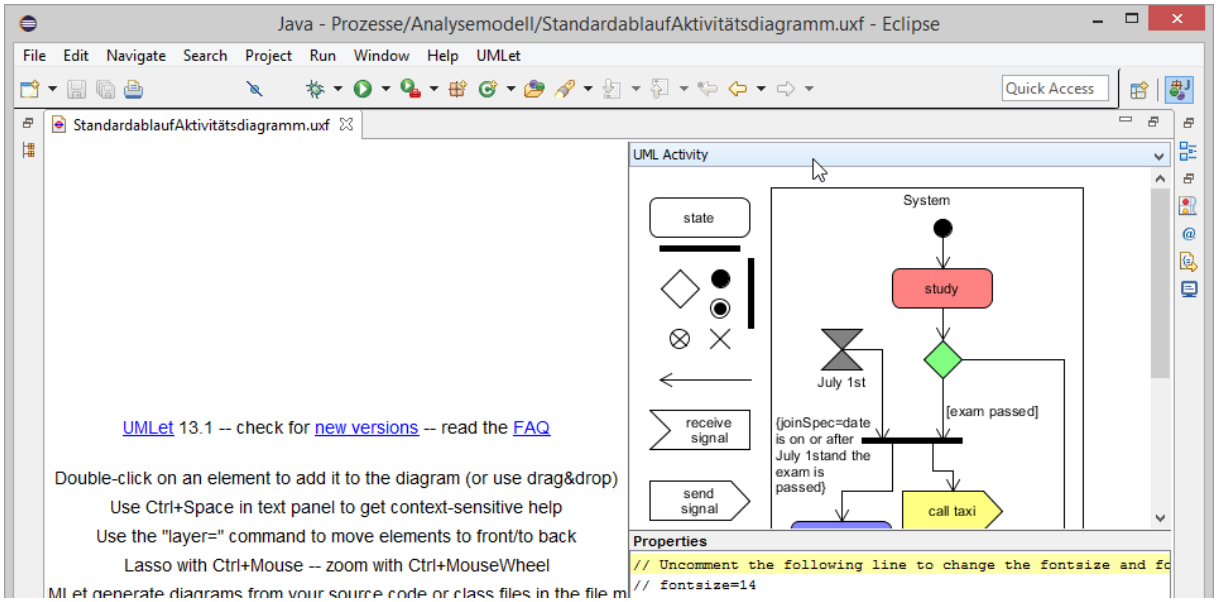


UMLet macht keine Syntaxprüfung, dass bedeutet, dass man beliebigen „Unsinn“ in die Diagramme zeichnen kann. Dies ist aber auch ein wichtiger Vorteil für die Analysephase, da man sich hier frei für eine Darstellungsform entscheidet. Wichtig ist nur, dass sie von jedem Projektbeteiligten gelesen werden kann und dass die Grundideen der Darstellungsweise dokumentiert sind.

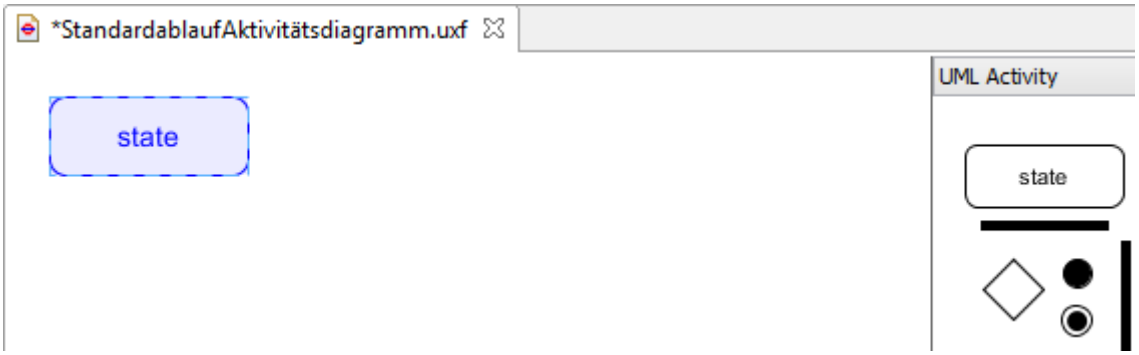
Die folgenden Anmerkungen gelten für die hier betrachtete Version und das Eclipse-Plugin. Einzelne Abbildungen stammen aus der Nutzung des Eclipse-PlugIns, sind aber einfach auf die andere Version übertragbar.

Beim Arbeiten mit UMLet ist zu beachten, dass man mit einem Doppelklick auf ein Zeichenelement, egal ob rechts in der Palette oder links in der Zeichenfläche, das ausgewählte Element verdoppelt. Dies ist sehr hilfreich, wenn man z. B. mehrere Aktivitäten oder auch Pfeile im Zeichenbereich organisieren möchte. Bei Anfängern führt dieser Ansatz aber ab und zu, zu Problemen, so dass auch ein Drag-and-Drop von Elementen von der rechten Seite in das Zeichenfeld auf der linken Seite möglich ist. Da dabei allerdings auch Elemente der rechten Seite unabsichtlich verschoben oder gelöscht werden können, ist der Weg mit dem Doppelklick vorzuziehen.

Im Folgenden ist skizziert, wie man ein Aktivitätsdiagramm anlegen kann, dabei ist jeder Nutzer zum selbst Experimentieren aufgerufen. Zunächst wird die zu Aktivitätsdiagrammen gehörende Palette „UML Activity“ ausgewählt.

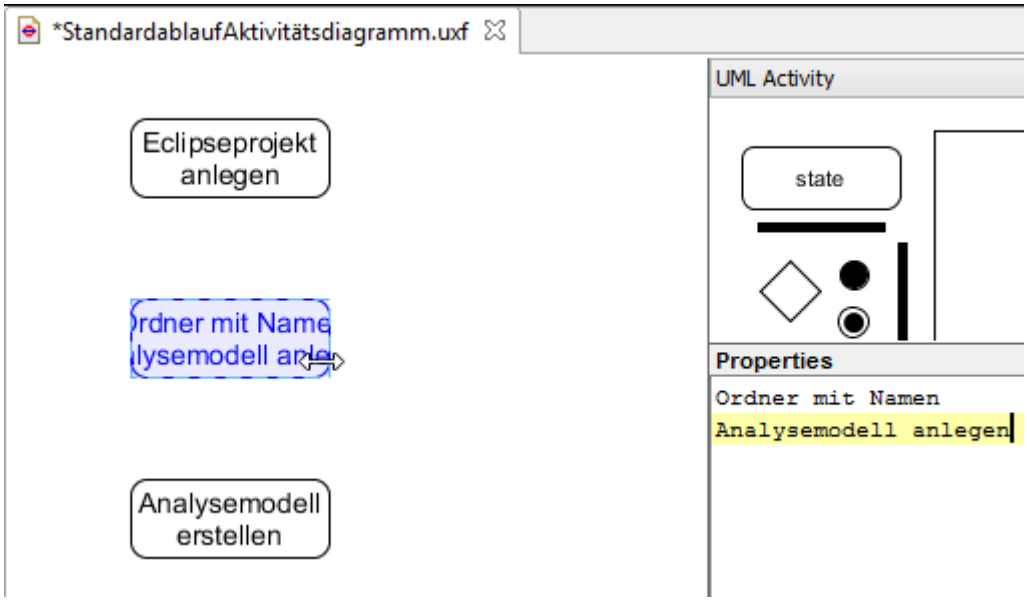


Danach wird auf der rechten Seite die Aktion, etwas unsauber als „state“ bezeichnet, doppelt angeklickt und so in den Zeichenbereich übernommen.

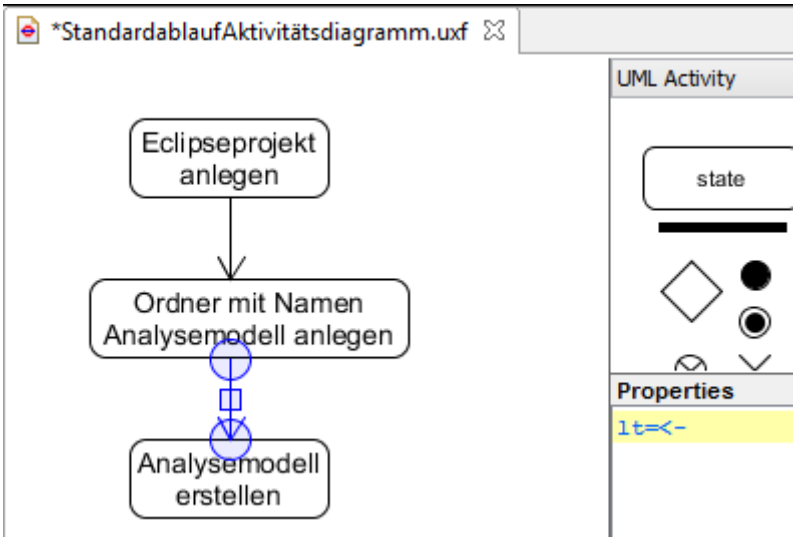


Um mehrere Aktionen zu erzeugen, wird dann ein Doppelklick auf der Aktion im Zeichenbereich ausgeführt. Graphische Elemente können generell mit gedrückter linker Maustaste verschoben und durch einen Doppelklick kopiert werden.

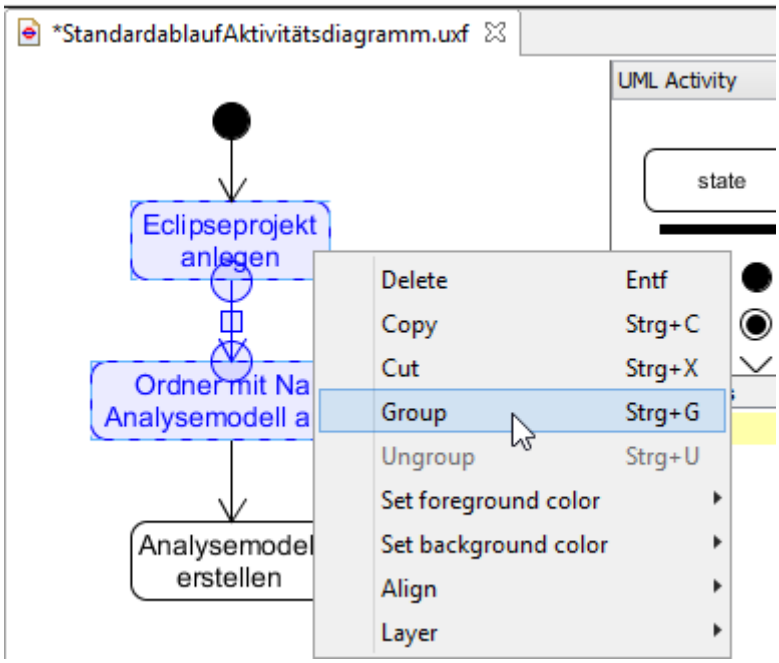
Die Beschriftung wird im erwähnten vierten Bereich geändert. Wenn man die Maus über ein Element schiebt, kann man dessen Größe anpassen.



Danach werden die anderen Elemente eingefügt und miteinander verbunden, Pfeile werden ebenfalls mit einem Doppelklick kopiert. Die mit Kreisen markierten Enden können über einen Linksklick verschoben werden. Das Quadrat in der Mitte des Pfeils ermöglicht die Verschiebung des gesamten Pfeils. Sitzen Pfeilende am Rand eines Objektes, werden später mit verschoben.

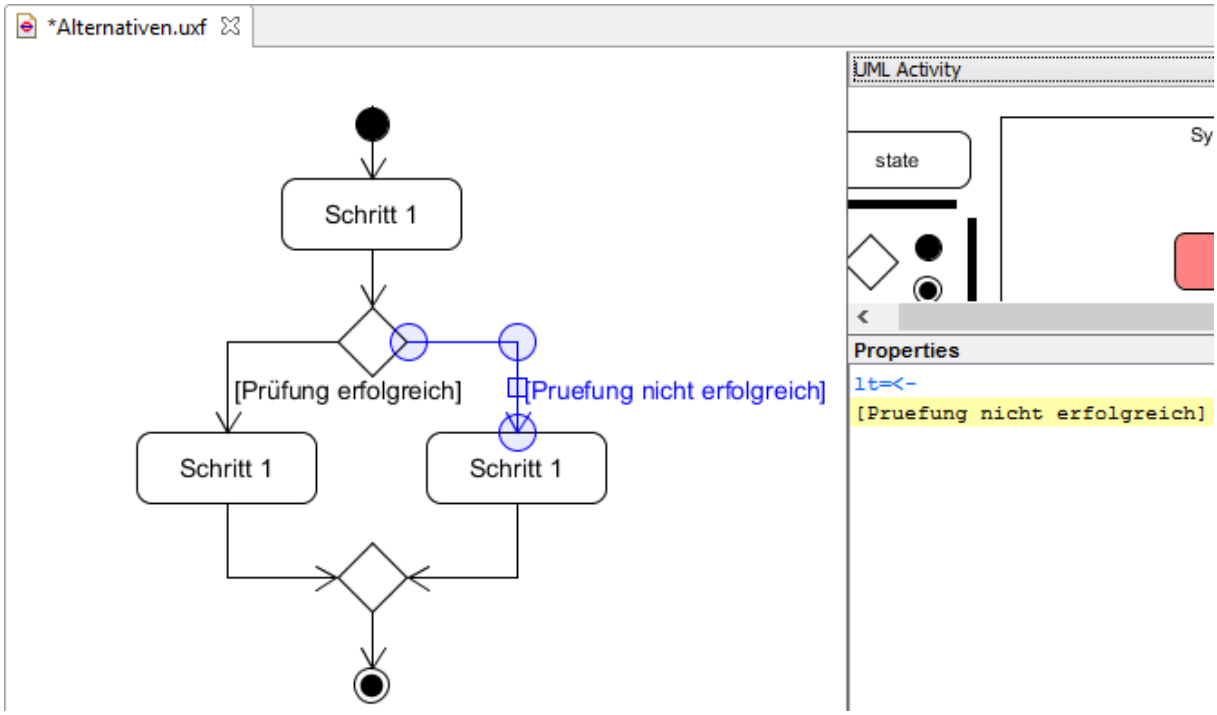


Klickt man neben das Diagramm und verschiebt bei gedrückter linker Maustaste die Maus, so wird das gesamte Diagramm verschoben. Man kann mit gedrückter Strg-Taste mehrere Elemente selektieren und diese dann mit einem Rechtsklick im dann sichtbaren Menü zu einer Gruppe zusammenfassen. Danach wird diese Gruppe immer zusammenbehandelt, was z. B. für das Verschieben einer Teilmenge der Diagrammelemente sehr hilfreich ist.

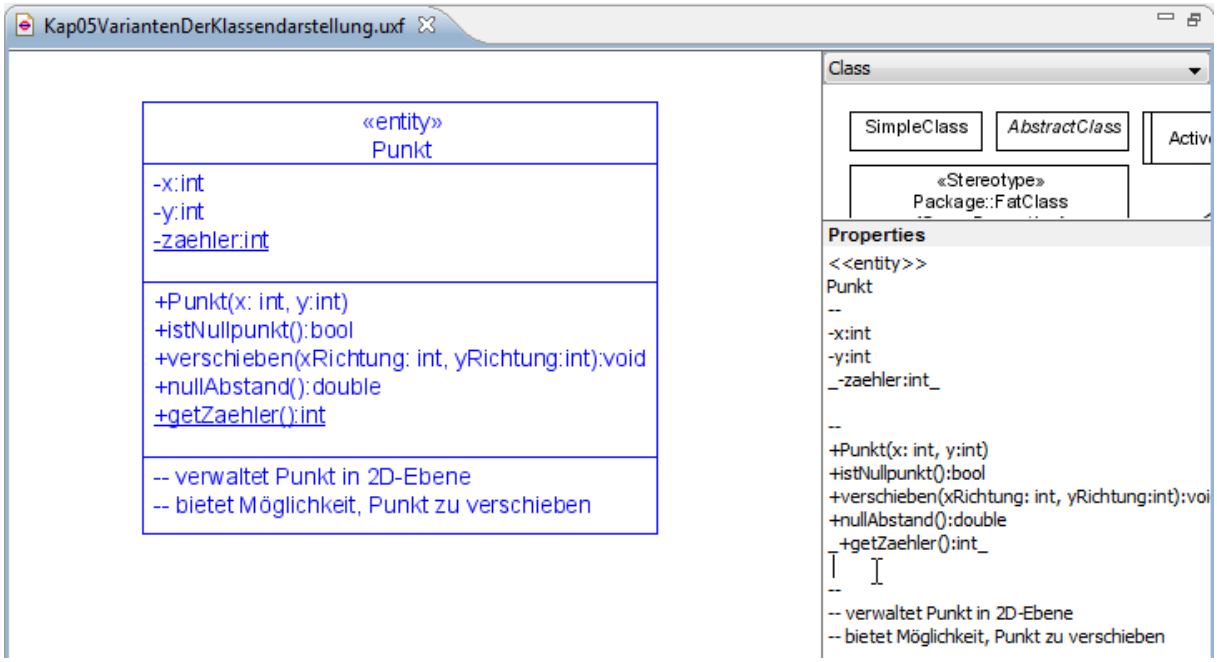


Nachdem ein Modell vollständig eingegeben wurde, muss es mit dem Punkt „Save“ im File-Menü gespeichert werden.

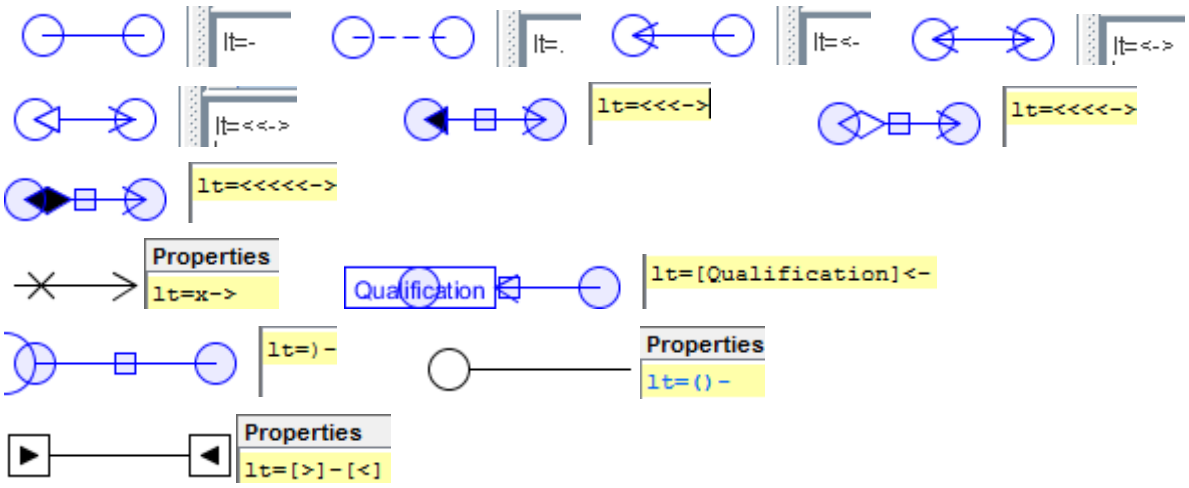
Möchte man Kanten z. B. bei Entscheidungen beschriften, ist zu beachten, dass die erste Zeile in der Beschriftungszeile gleich bleibt, da mit ihr angegeben wird, an welchen Enden Pfeilspitzen stehen sollen. Die folgende Abbildung zeigt links eine selektierte Kante und rechts das Textfeld, in dem die Beschriftung eingetragen wird. Knicke in Kanten erzeugt man, wenn man auf einer selektierten Kante an dem Punkt, an dem ein neuer Knickpunkt entstehen soll, an der Kante mit gedrückter linker Maustaste zieht. Der markierte Mittelpunkt der Kante dient dazu, die gesamte Kante zu verschieben und kann nicht zur Knickerstellung genutzt werden.



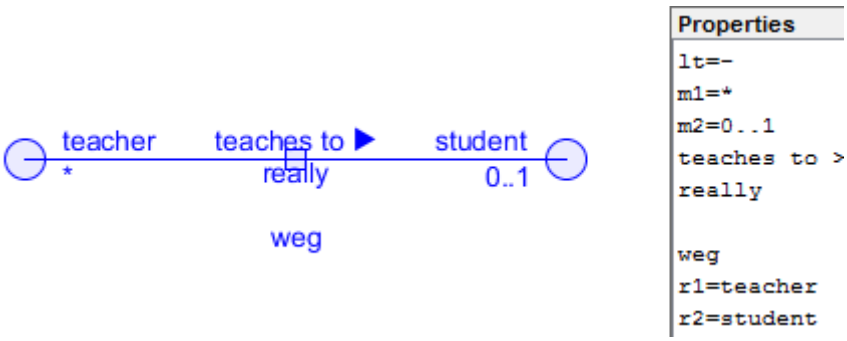
Bei der Bearbeitung von Klassen ist noch zu beachten, dass bei Texteingaben eine leere Zeile mit zwei Minuszeichen dazu führt, dass im Klassendiagramm ein Strich gemalt wird. Möchte man Leerzeilen einfügen, müssen diese mindestens ein Leerzeichen enthalten. Klassenvariablen und Klassenmethoden werden in der UML unterstrichen, dies ist durch das Voranstellen und Abschließen mit einem Unterstrich möglich. Es gibt einige weitere Steuerungsbefehle, von denen einige im Folgenden betrachtet werden.



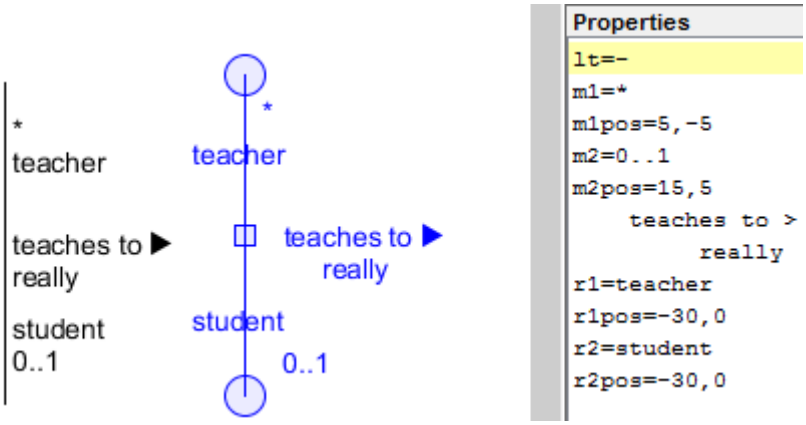
Beschäftigt man sich etwas genauer mit UMLet, gibt es noch einige wenige Steuerbefehle, die in der Beschriftungsbox eingegeben werden können. Die folgenden Bilder zeigen zunächst links die Linienart und rechts den eingegebenen Text, die Kreise zeigen nur, dass die Linie selektiert wurde.



Weiterhin gibt es verschiedene Plätze, an denen Eigenschaften an Linien notiert werden können.

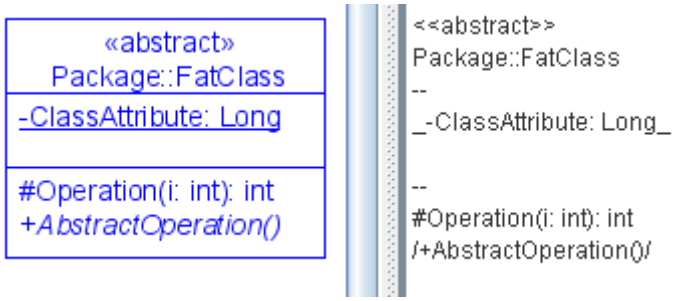


Da die Platzierung nicht immer optimal ist, kann man die Texte mit einzelnen Leerzeichen oder der Angabe von Pixelverschiebungen platzieren.



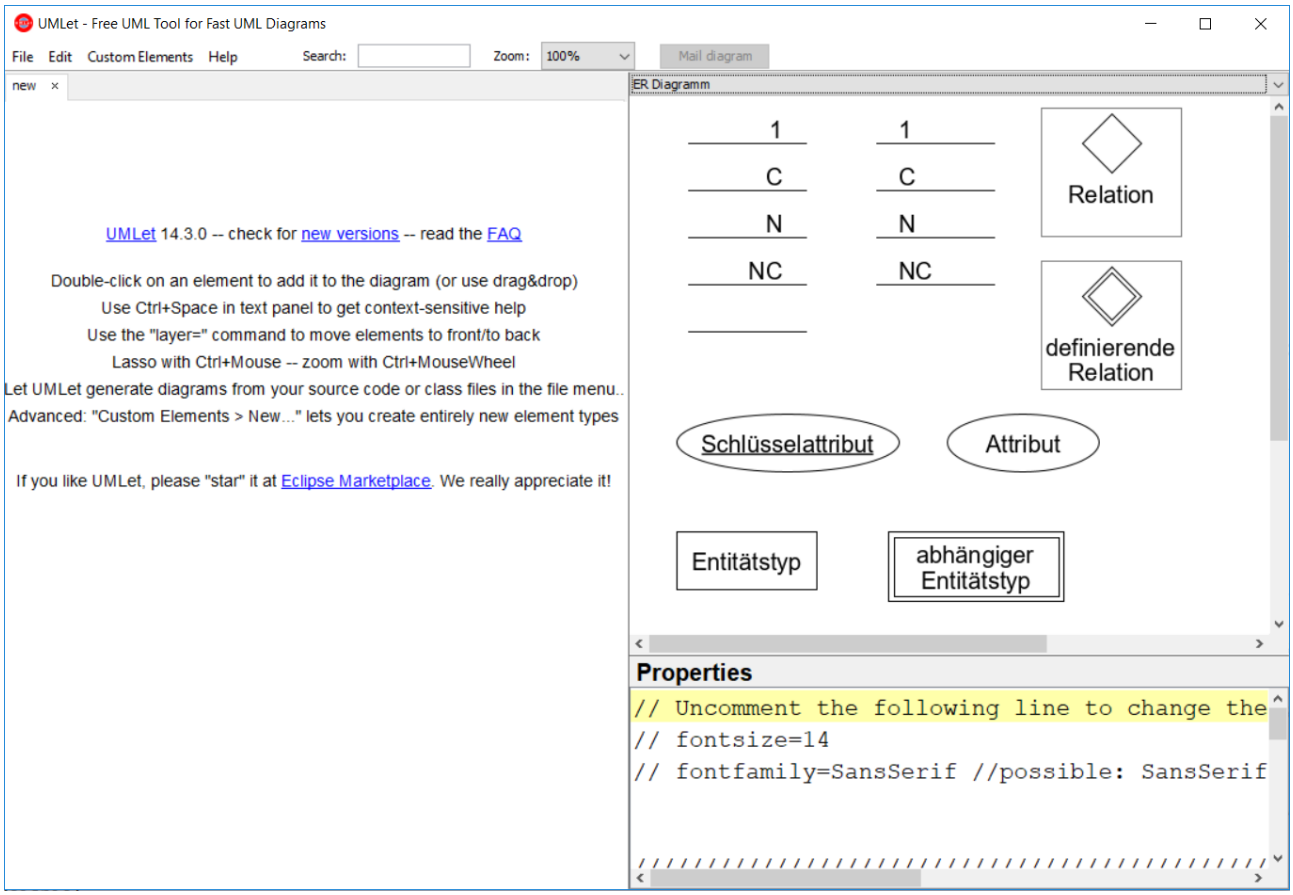
Ausgefüllte Elemente können eine Farbe haben.

Für abstrakte Methoden müssen Schrägstriche um den Namen herum stehen, damit dieser kursiv dargestellt wird. Beo Klassenmethoden und Klassenvariablen steht unmittelbar vor und nach dem Namen ein Unterstrich.

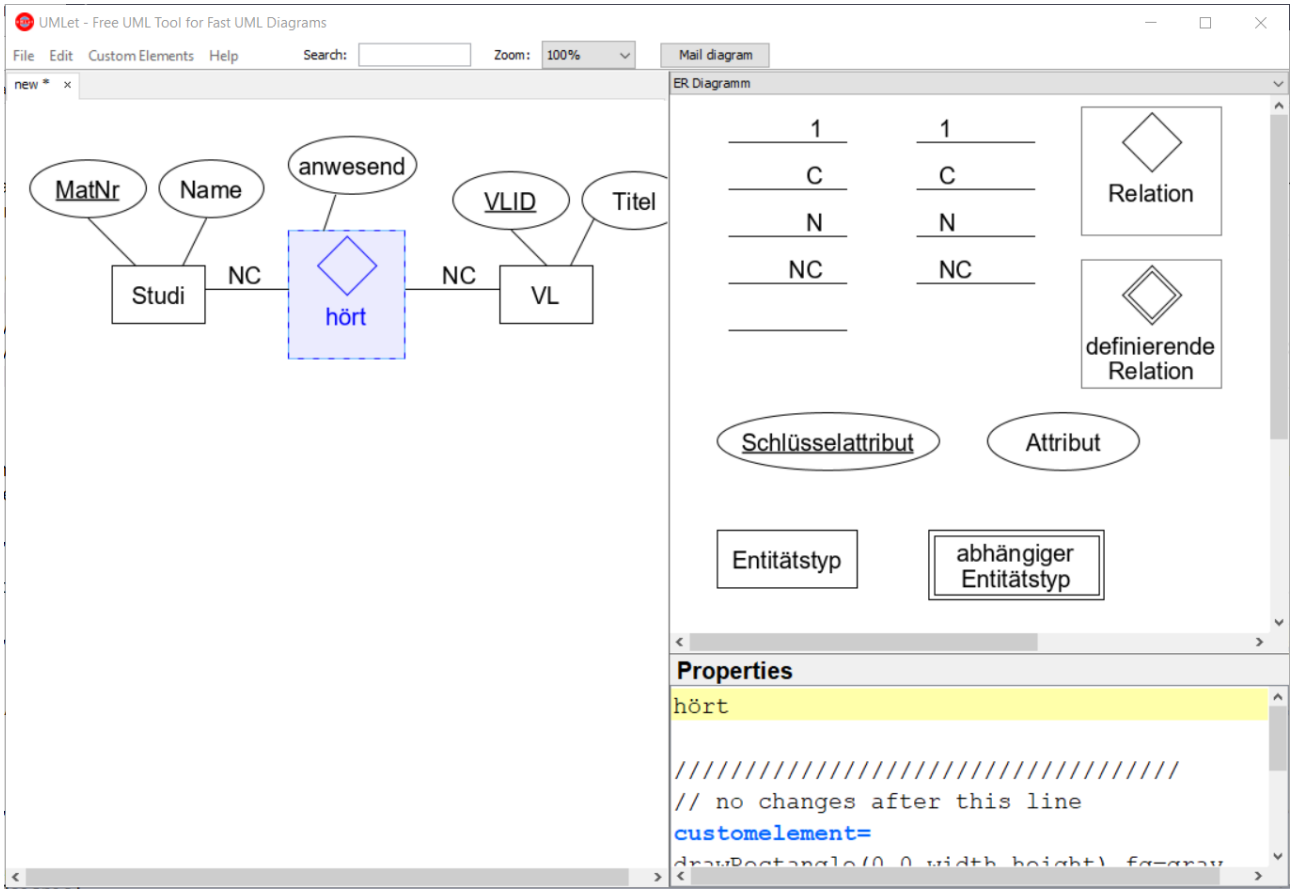


2.4 Erstellung von ER-Diagrammen

Das vorherige Unterkapitel hat die generelle Nutzung beschrieben, die vollständig auf die Erstellung von ER-Diagrammen übertragbar ist. Zur Erstellung muss die Palette „ER-Diagramm ausgewählt werden.

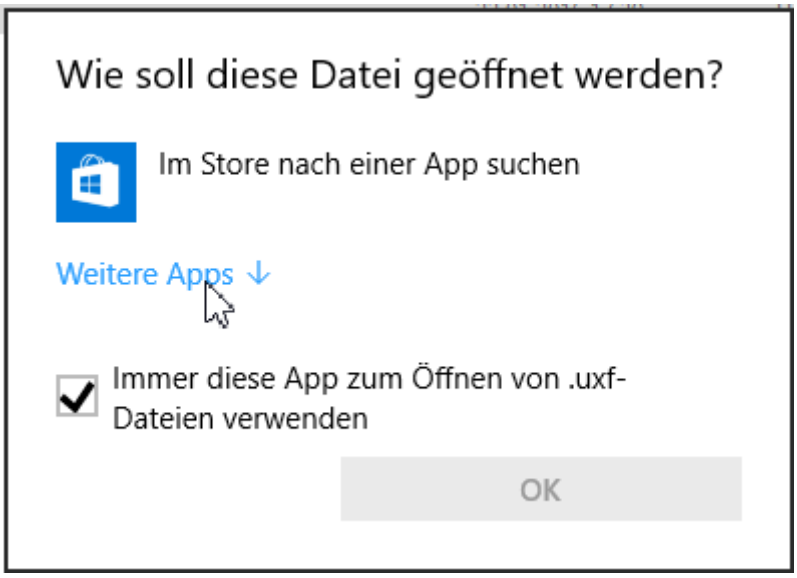


Alle ins Modell kopierten Elemente sind wieder über die Properties veränderbar. Hinweise, dass Inhalte nicht geändert werden sollten, sind ernst zu nehmen.

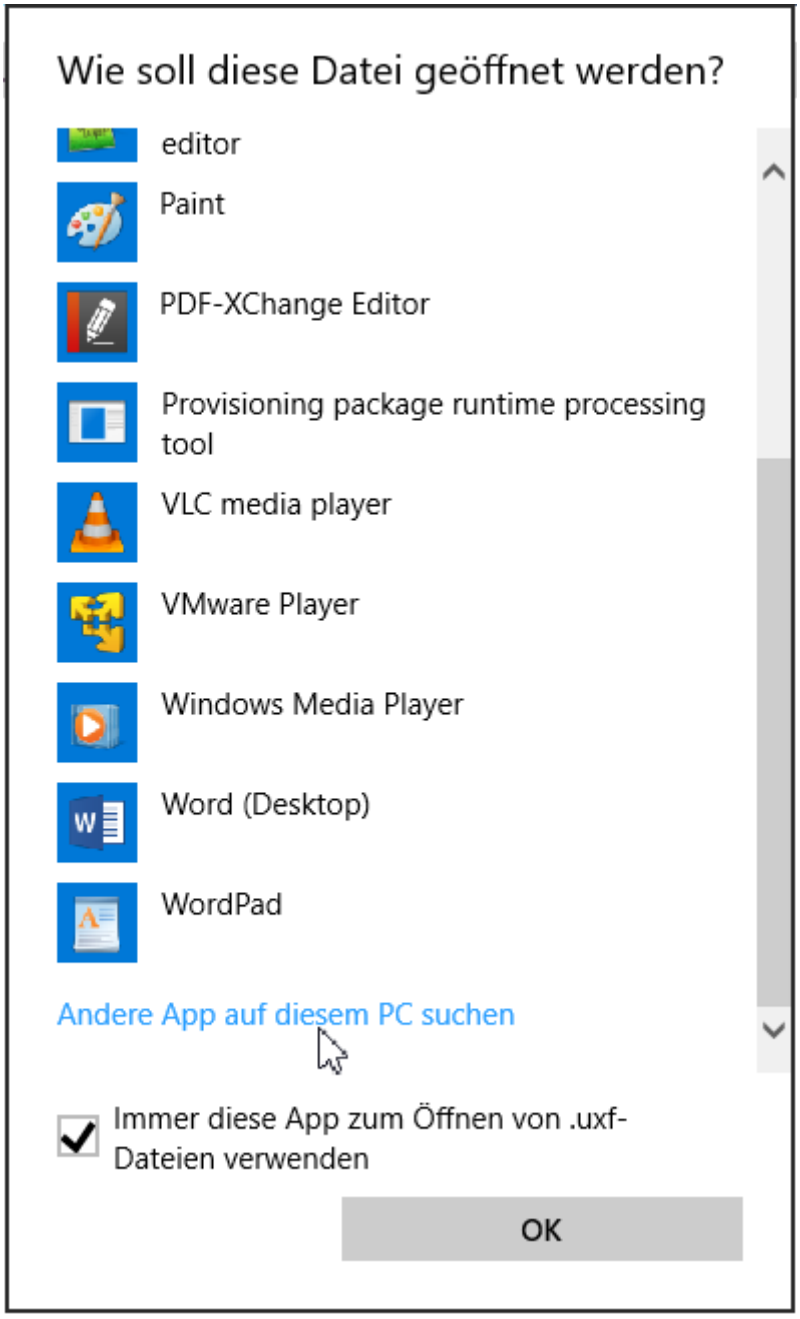


2.5 Verknüpfung von uxf-Dateien mit UMLet

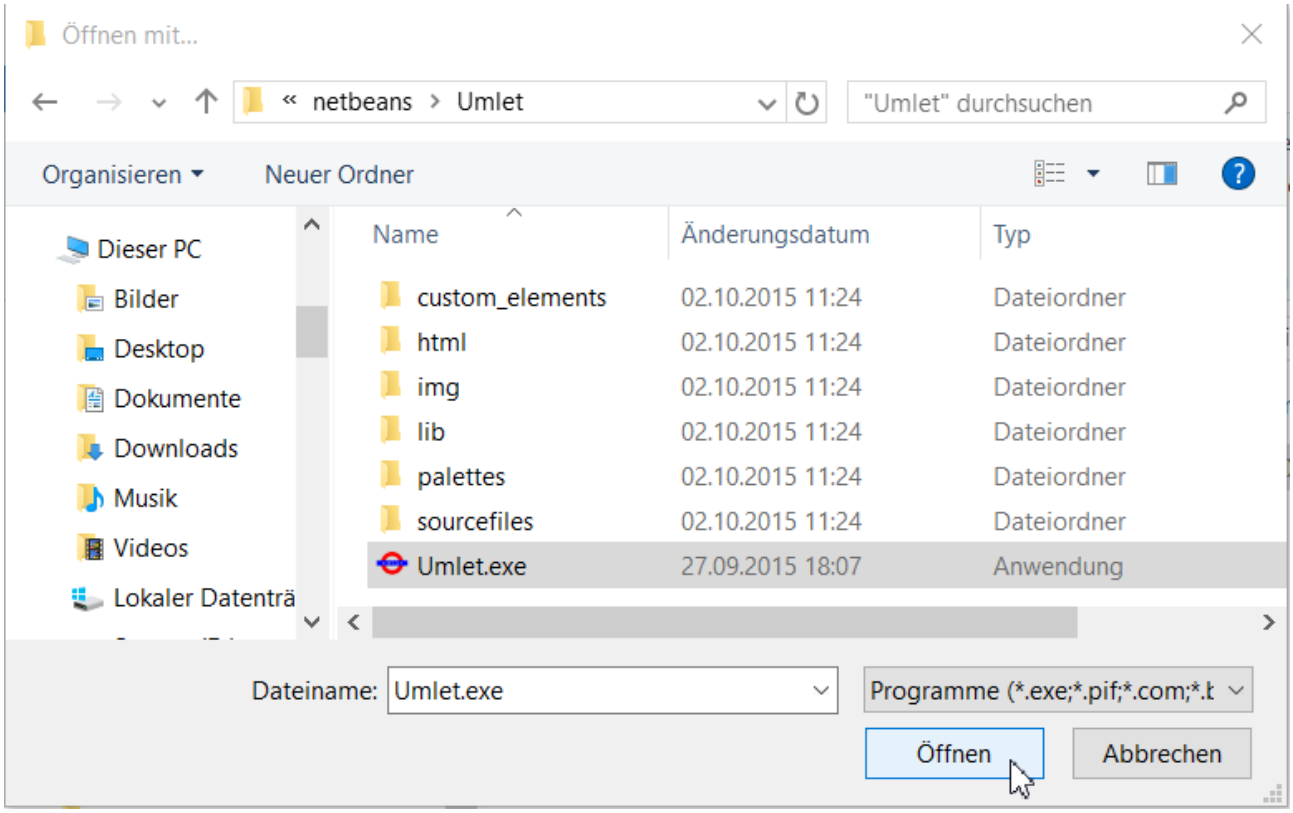
UMLet ist alleine nutzbar und hat keine echte Integration in NetBeans. Hier wird gezeigt, wie eine Verknüpfung mit den zugehörigen .uxf-Dateien und dadurch eine Integration in NetBeans erfolgt. Nach einem Doppelkick auf einer uxf-Datei öffnet sich die Frage nach dem zu nutzenden Programm. Es wird auf „Weiter Apps“ geklickt.



Es wird „Andere App auf diesem PC suchen“ geklickt.



Nun wird zur ausführbaren Datei umlet.exe gesteuert, diese ausgewählt und „Öffnen“ geklickt.



Danach öffnet sich UMLet und die Datei kann bearbeitet werden.

3 SQLChecker

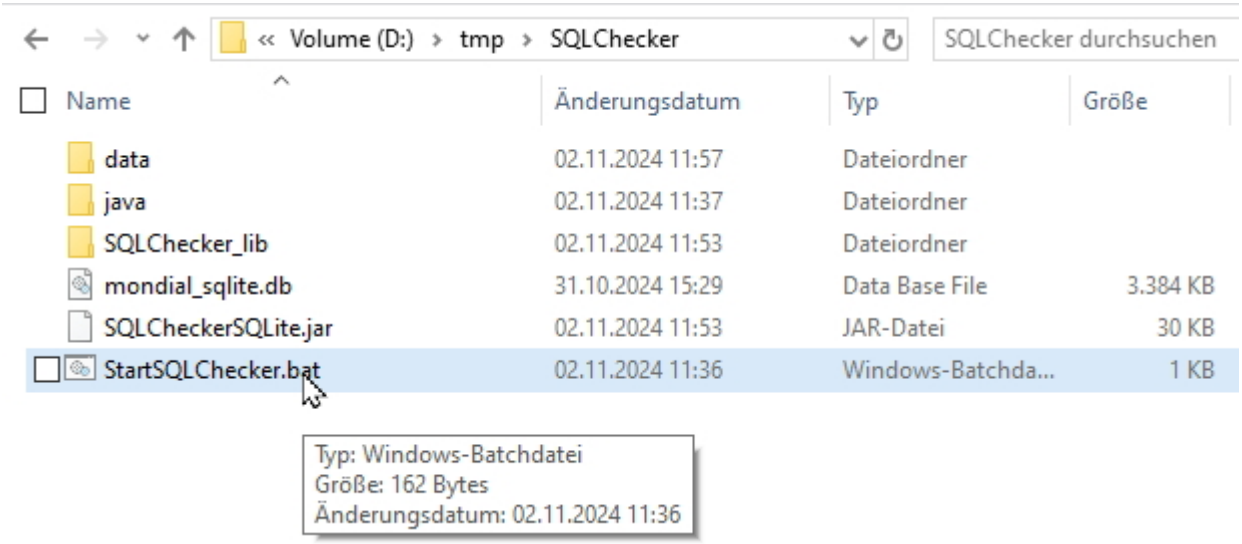
Der SQLChecker ist ein kleines Programm, mit dem Lösungen zu SQL-Aufgaben lokal überprüft werden können. Dazu werden die Ergebnisse des eigenen Lösungsversuchs mit dem erwarteten Ergebnis verglichen. Es kann damit nicht geprüft werden, ob die Lösung sinnvoll oder effizient ist, nur dass die erreichten Ergebnisse stimmen.

Es ist sehr sinnvoll, SQL-Anfragen zunächst in einem anderen Werkzeug auszuprobieren, da diese meist Syntax-Highlighting und etwas genauere Fehlermeldungen bieten. Die Lösung ist dann zur Überprüfung in dieses Werkzeug zu kopieren.

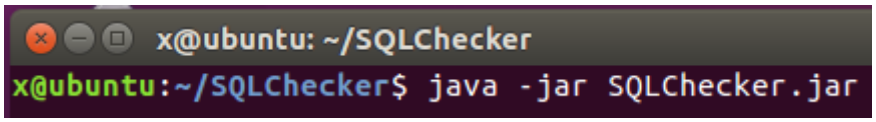
Der SQLChecker kann unter Windows mit einem 64-Bit-System direkt genutzt werden.

Die Datei SQLChecker.zip kann an einem fast beliebigen Ort mit „Hier entpacken“ ausgepackt werden, es muss nur sichergestellt sein, dass der Nutzer in dem entstehenden Verzeichnis Lese- und Schreibrechte hat, da ein lokaler Ordner data zum Schreiben von Daten angelegt wird.

Der Start des SQLCheckers erfolgt über einen Doppelklick auf der Datei StartSQLChecker.bat.

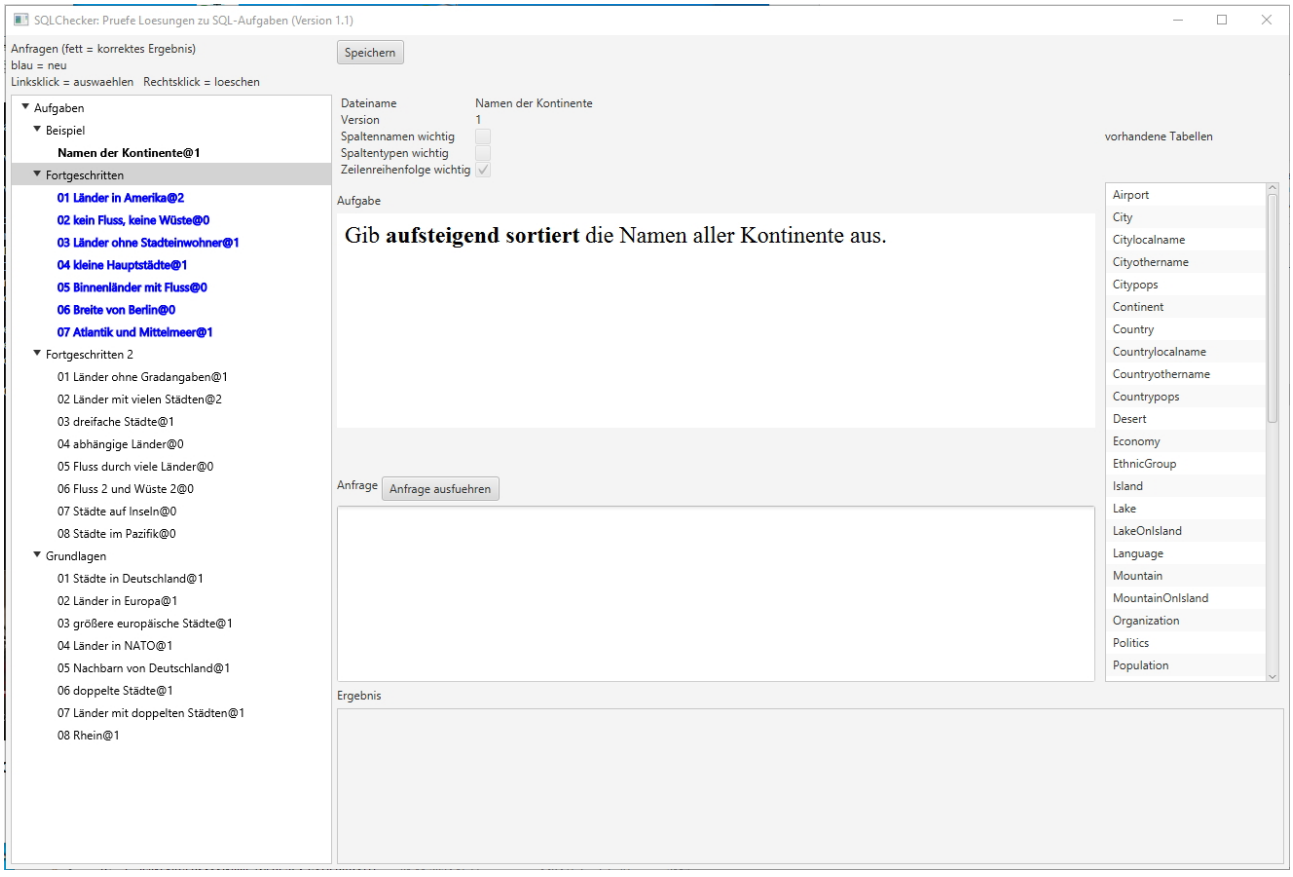


Falls Jar-Dateien unter Windows mit Java verknüpft sind, ist alternativ SQLChecker.jar direkt durch einen Doppelklick startbar. Unter anderen Systemen muss in einer Konsole in das Verzeichnis manövriert und `java –jar SQLChecker.jar` aufgerufen werden. Dies gilt auch unter Ubuntu, damit das Verzeichnis data im passenden Ordner angelegt wird. Generell bietet der Aufruf unter der Konsole den Vorteil, dass der virtuellen Java-Maschine mehr Speicher über Parameter zugeordnet werden kann.

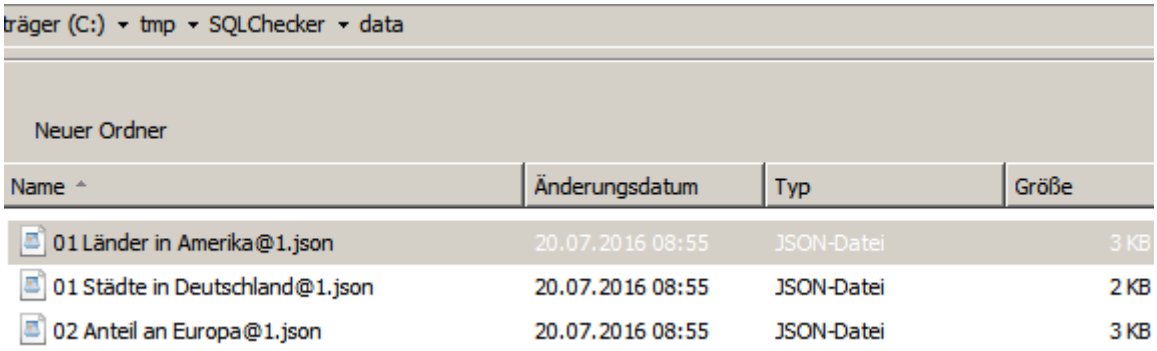


Das Werkzeug lädt zunächst alle lokal vorhandenen Aufgaben und sucht dann im Internet nach neueren Aufgaben, die ebenfalls geladen werden. Die Aufgaben werden in einem Baum auf der linken Seite angezeigt. Schwarze Dateinamen stehen für lokal geladene Aufgaben, blaue für Dateien, die bei diesem Aufruf aus dem Internet geladen wurden und jetzt lokal zur Verfügung stehen. Ist der Text fetter als normal handelt es sich um eine gelöste Aufgabe. Mit einem Links-Klick wird eine Aufgabe ausgewählt, mit einem Rechts-Klick wird sie endgültig gelöscht.

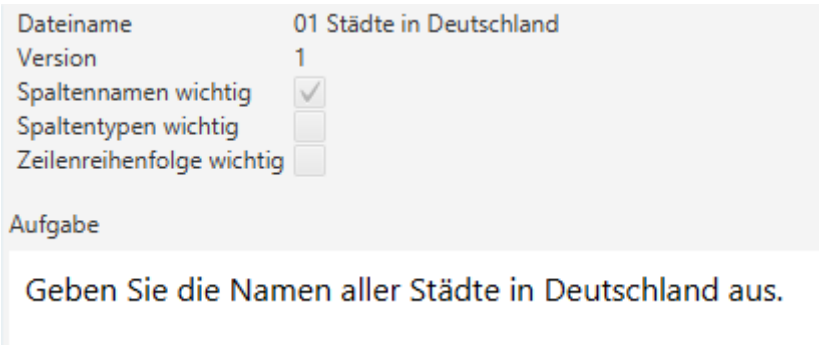
Die Namen der Aufgaben enthalten am Ende immer ein @-Symbol und eine Versionsnummer. Sollte es zwei Anfragen im Baum mit gleichen Namen geben, ist davon auszugehen, dass es mehrere Versionen der Aufgabe gibt, von denen üblicherweise die neueste bearbeitet wird.



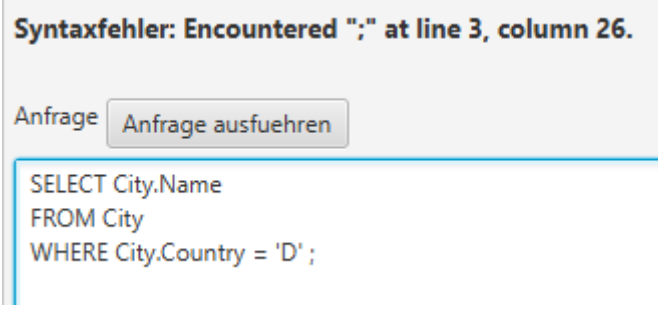
Alle vorhandenen Aufgaben stehen im Unterverzeichnis „data“ und haben die Endung „.json“. Anders heruntergeladene Aufgaben sind zur Nutzung einfach in dieses Verzeichnis zu kopieren.



Im Aufgabe-Feld steht die zu bearbeitende Aufgabe, dabei können die oberen drei CheckBoxen interessant sein. Ein Haken bei „Spaltennamen wichtig“ bedeutet, dass der genaue Spaltenname zur Lösung passen muss. Steht dazu nichts in der Aufgabenstellung, sind typischerweise die Spaltennamen der genutzten Tabellen gemeint, so dass hier keine besondere Beachtung notwendig ist. Ein Haken bei „Spaltentypen wichtig“ bedeutet, dass konkrete Typen für die Ergebnisspalten gefordert sind, so dass z. B. nicht eine Spalte mit korrektem Namen, aber statt INTEGER dem Typ VARCHAR in der Lösung genutzt wird. Für die konkrete Aufgabenstellung ist aber ein solcher Haken meist vernachlässigbar, da er eine Einschränkung auf ein konkretes Datenbanksystem sein kann, da z. B. Oracle mehr Zahlentypen mit etwas anderen Wertebereichen unter dem Namen Number anbietet, als es in Derby der Fall ist. Ein Haken bei „Zeilenreihenfolge wichtig“ bedeutet, dass die Reihenfolge in der Ergebnisaushabe relevant ist. Da SQL mengenorientiert arbeitet, muss die gleiche Anfrage nicht zur gleichen Reihenfolge der Ergebniszeilen führen.

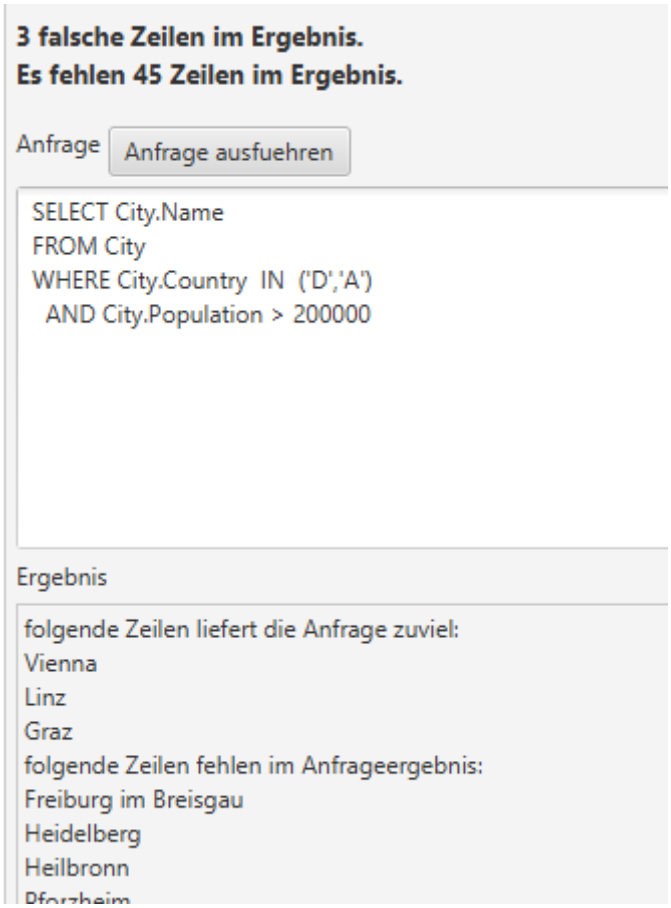


Der eigentliche Lösungsversuch wird im Feld „Anfrage“ eingegeben und mit dem Knopf „Anfrage ausfuehren“ ausgeführt. In dem Informationsfeld darüber werden etwaige Hinweise ausgegeben. Im folgenden Beispiel liegt ein Syntaxfehler vor, da das Semikolon kein Bestandteil von SQL ist.

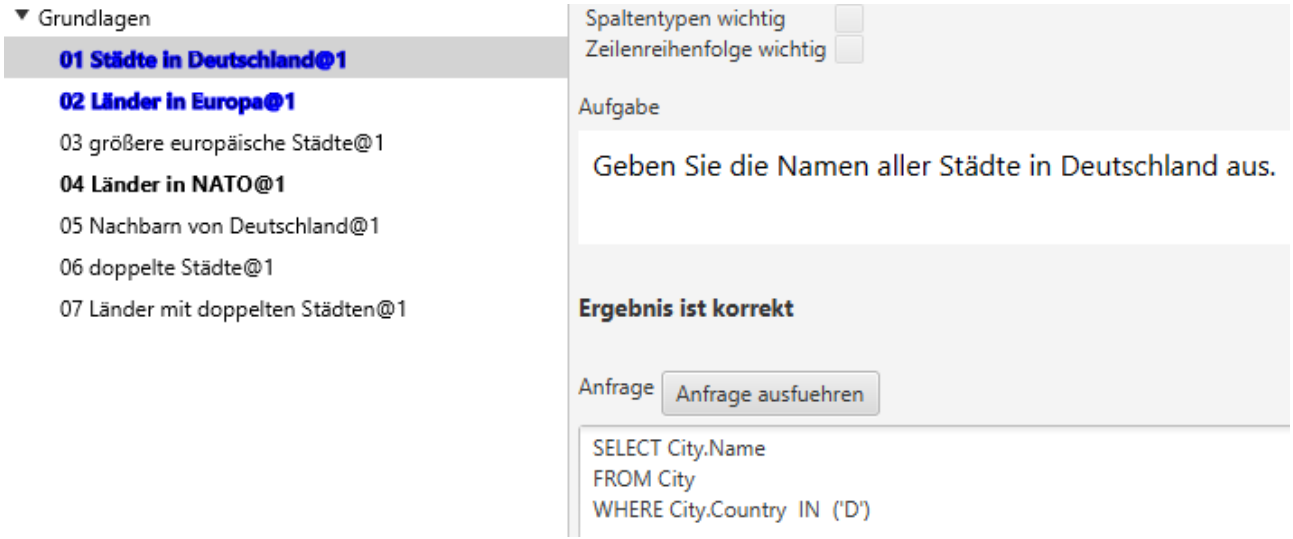


Sollte die Anfrage Zeilen enthalten, die nicht zum Ergebnis gehören oder es Zeilen im gewünschten Ergebnis geben, die das Ergebnis der Anfrage nicht liefert, werden diese im unteren Ergebnisfenster angezeigt. Das Ergebnisfenster zeigt damit das Ergebnis der Korrektheitsprüfung und nicht das Ergebnis der Anfrage.

Es ist zu beachten, dass die Auswahl einer neuen Aufgabe nur durch Anklicken mit der Maus möglich ist. Es kann etwas dauern, da eventuell eine neue Datenbankverbindung aufgemacht werden muss.



Liefert die Anfrage das korrekte Ergebnis, wird dies im Informationsfeld ausgegeben und die Anfrage wird im Baum fetter dargestellt. Weiterhin wird eine korrekte Lösung automatisch gespeichert. Alle anderen Lösungsversuche müssen über den Knopf „Speichern“ explizit abgespeichert werden, ansonsten gehen alle eingetragenen Informationen beim Aufruf einer neuen Aufgabe endgültig verloren.


Grundlagen
01 Städte in Deutschland@1
02 Länder in Europa@1
03 größere europäische Städte@1
04 Länder in NATO@1
05 Nachbarn von Deutschland@1
06 doppelte Städte@1
07 Länder mit doppelten Städten@1
Spaltentypen wichtig
Zeilenreihenfolge wichtig
Aufgabe
Geben Sie die Namen aller Städte in Deutschland aus.
Ergebnis ist korrekt
Anfrage
Anfrage ausfuehren
SELECT City.Name
FROM City
WHERE City.Country IN ('D')