Datenbanken 1

Übungsaufgaben

Prof. Dr.-Ing. Holger Vogelsang

Wintersemester 2024/2025

Don’t panic

Inhaltsverzeichnis

1. Einleitung

1.1 Organisation des Labors

Das Labor ist in mehrere Aufgabentypen aufgeteilt, die sich am Verlauf der Vorlesung orientieren. Aufgaben, die keine Abgabe einer Lösung erfordern, sind unter „Freiwillige Aufgaben“ zu finden. Die ersten drei großen Aufgabenblöcke behandeln SQL-Abfragen, angefangen von einfachen Abfragen über Joins bis hin zu Transaktionen.

Anschließend folgen Aufgaben, die den Umgang mit Datenbanken in der Java-Programmierung am Beispiel von JDBC zu trainieren. Die letzten beiden Pflichtaufgabenblöcke behandeln die ER-Modellierung und Normalisierung sowie die Programmierung einer Datenbank mit einem OR-Mapper.

Beachten Sie auch die Bonusausgabe 3, durch deren Lösung Sie bis zu 10% der Punkte des Datenbank-Teils der Klausur erzielen können. Auch ohne eine Abgabe dieser Lösung ist es möglich, in der Klausur alle Punkte zu erreichen. Um die Bonuspunkte zu erhalten, müssen Sie die Lösung dieser Aufgabe vor der Klausur abgegeben haben. Der Bonus gilt für alle möglicherweise erforderlichen Versuche, die Klausur zu bestehen.

Das Labor ist in mehrere Aufgabentypen aufgeteilt, die sich am Verlauf der Vorlesung orientieren. Aufgaben, die keine Abgabe einer Lösung erfordern, sind in Abschnitt 4 zu finden. Die Aufgaben 1 und 2 behandeln SQL-Abfragen, angefangen von einfachen Abfragen bis hin zu Joins. Aufgabe 3 lässt Sie Transaktionen verwenden.

Weitere Informationen finden Sie im ILIAS: `https://ilias.h-ka.de`. Dort liegt die Datei „SQL-Skripte.zip“, die SQL-Skripte zur Installation der Datenbank für Oracle, MS-SQL, PostgreSQL und MySQL umfasst. Weiter hinten in diesem Skript gibt es eine genauere Beschreibung der Datenbank.

1.2 Übersicht über die Pflichtaufgaben

Die folgenden Pflichtaufgaben müssen gelöst werden.

Kapitel	Art	Aufgaben
2.1	Abfragen	1.1 – 1.10
2.2	Joins und Views	2.1 – 2.11
2.3	Transaktionen	3
2.4	JDBC-Programmierung	4.1 – 4.4
2.5	ORM-Programmierung	5.1 – 5.3
2.6	ER-Modell	6.1 – 6.3
2.6	Normalisierung	6.4 – 6.5

1.3 Datenbanken

Sie müssen keinen der vorgegebenen Datenbank-Server verwenden, sondern dürfen auch einen eigenen installieren. Die Vorgaben der Aufgaben sollten prinzipiell mit MSSQL, MySQL/MariaDB, Oracle und PostgreSQL funktioniert, wobei die Tests nur unter Oracle und PostgreSQL stattfanden. Zu all diesen Datenbanken liefern die Anbieter eigene Clients für interaktive Zugriffe mit. Manchmal sind aber gerade Tools anderer Anbieter komfortabler und für den Einsatz in dieser Übung besser geeignet. Auf den Pool-PCs sind DBeaver und DBVisualizer bereits installiert, aber noch nicht konfiguriert, da die Einstellungen abhängig von Ihrer Gruppe sind. Die folgenden Abschnitt zeigen Ihnen einige dieser Tools sowie deren Konfiguration für den Zugriff auf die bereitgestellten Datenbanken.

1.3.1 Oracle XE

Ihnen steht eine Oracle-XE-Datenbank auf dem Server `iwi-i-db-01` über Port 1521 im Netz der Hochschule zur Verfügung.

1.3.1.1 DBeaver

DBeaver existiert sowohl als Eclipse-Plug-in als auch als eigenständige Anwendung. Wenn Sie bereits mit Eclipse entwickeln, dann bietet es sich an, das Eclipse-Plug-in zu verwenden. Auf Ihren eigenen PCs können Sie DBeaver über den Eclipse-Marketplace installieren.

- Zugriff **innerhalb** des Netzes der Hochschule:
  - Schalten Sie auf die DBeaver-Perspektive um.
  - Links im Datenbanknavigation fügen Sie eine neue Verbindung hinzu:
    Rechtsklick → Anlegen → Verbindung.
  - Wählen Sie den Datenbanktyp aus und klicken Sie auf „Next“.
  - Tragen Sie als Host `iwi-i-db-01` mit der Portnummer 1521 ein. Die Datenbank muss „FREE“ heißen, in der Auswahl rechts daneben muss „Servicename“ gewählt sein.
  - Die Authentifizierung ist „Database Native“, den Benutzernamen sowie Ihr Gruppenpasswort erhalten Sie per E-Mail.
  - Mit „OK“ beenden Sie die Eingabe. Sie können auch mit „Verbindung testen…“ die korrekte Eingabe der Daten prüfen.

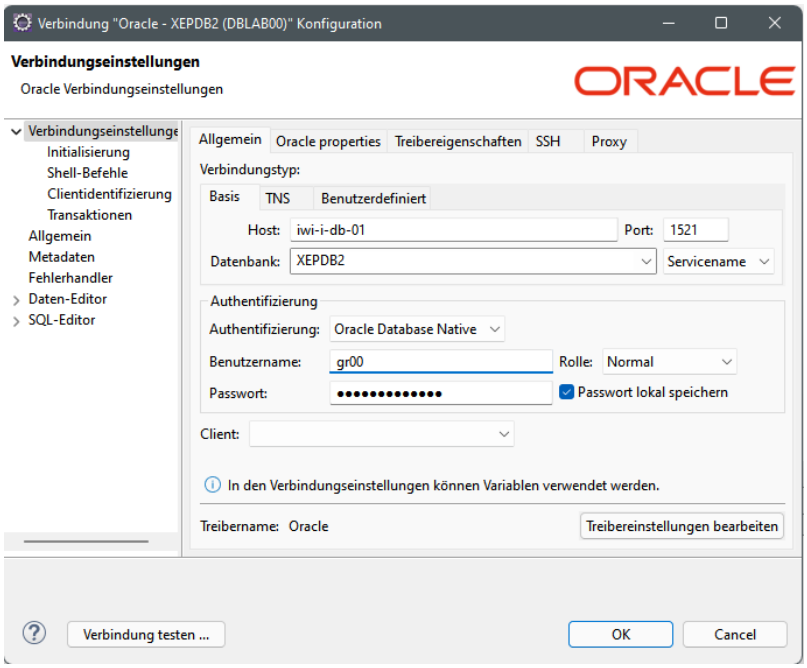

Abbildung 1.1: DBeaver-Verbindung für internen Zugriff auf die Oracle-Datenbank

- Zugriff **außerhalb** des Netzes der Hochschule: Wenn Sie von außerhalb des Hochschulnetzes auf die Oracle-Datenbank zugreifen wollen, dann müssen Sie im Tab „SSH“ zusätzlich den SSH-Tunnel konfigurieren:
  - Haken vor „Verwende SSH-Tunnel“ setzen.
  - „Host/IP“ ist `login.h-ka.de` auf Port 22.
  - „Benutzer“ ist Ihr RZ-Benutzername, „Passwort“ Ihr RZ-Passwort.

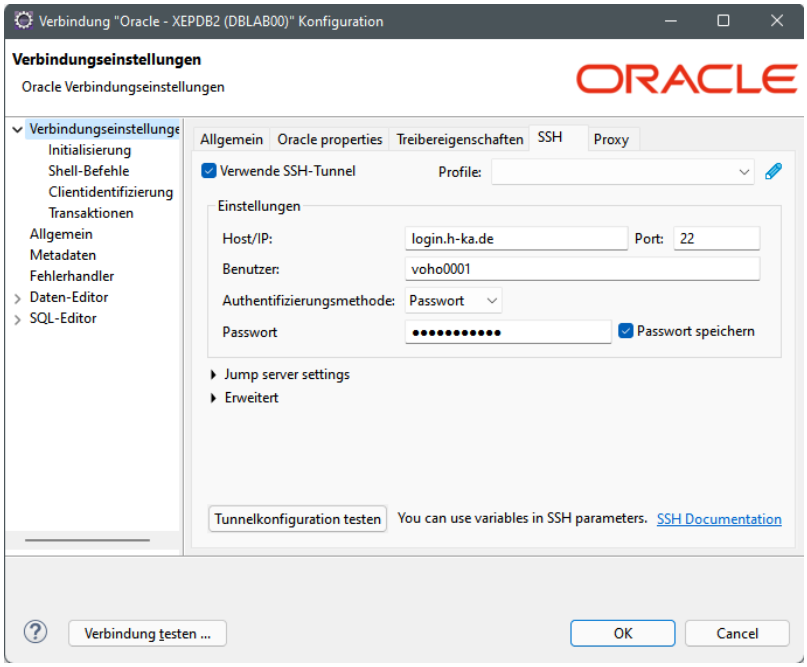

Abbildung 1.2: DBeaver-Verbindung für externen Zugriff auf die Oracle-Datenbank

In Abschnitt 1.3.2.1 zur Verwendung von DBeaver mit PostgreSQL ist beschrieben, wie die Anzahl an offenen Datenbankverbindungen eingeschränkt werden kann.

1.3.1.2 DbVisualizer

Einstellungen der Verbindung:

- Zugriff **innerhalb** des Netzes der Hochschule:
  - Rechtsklick links in „Databases“ → „Create Database Connection“. Sie können hier die Unterstützung eines Wizards nutzen oder die Daten manuell erfassen. Letzteres ist in den folgen Punkten beschrieben.
  - Name: Frei vergebener Name für die Verbindung.
  - Database type: „Oracle“ auswählen.
  - Driver (JDBC): Oracle Thin
  - Connection Type: Service-Name
  - Database Server: `iwi-i-db-01`
  - Port: 1521
  - Service-Name: FREE
  - Databasde Userid: Ihr Gruppenname
  - Database Password: Ihr Gruppenpasswort
  - Autocommit: Kann vorerst eingeschaltet bleiben und muss später teilweise geändert werden, was aber in den Aufgaben vermerkt ist.
- Zugriff **außerhalb** des Netzes der Hochschule: Ein SSH-Tunnel ist nur in der kostenpflichtigen Pro-Version verfügbar. Sie können sich aber über VPN im Hochschulnetz anmelden und dann DbVisualizer wie oben beschrieben verwenden. Alternativ können Sie auch einen lokalen SSH-Client wie Putty oder MobaXTerm starten und darüber den Tunnel aufbauen, was hier nicht näher beschrieben wird.

1.3.1.3 Squirrel

Auch Squirrel lässt sich innerhalb und außerhalb des Hochschulnetzes verwenden. Dessen Konfiguration erfolgt mit denselben Einstellungen wie bei DBeaver. Beachten Sie aber bitte, dass Squirrel gerade beim Zugriff über das Internet recht langsam ist.

1.3.2 PostgreSQL

Neben der Oracle-Datenbank, die direkt nur aus dem Netz der Hochschule erreichbar ist, können Sie auch die PostgreSQL-Datenbank verwenden, die unter einer öffentlichen IP-Adresse direkt zugänglich ist. Auf dieser Datenbank sind dieselben Nutzer wie auf der Oracle-Datenbank vorhanden. Zugangsdaten:

- Server: `datenbanken1.ddns.net` bzw. IP-Adresse `193.196.37.16`
- Port: 3690 (Achtung: weicht vom Standard-Port ab, da dieser in der Firewall der Hochschule gesperrt ist)
- Datenbank: Ihr Gruppenname wie z.B. „g01“

- Nutzername: Ihr Gruppenname wie z.B. „g01“ (unbedingt Groß- und Kleinschreibung beachten)
- Passwort: Ihr Gruppenpasswort

1.3.2.1 DBeaver

DBeaver existiert sowohl als Eclipse-Plug-in als auch als eigenständige Anwendung. Wenn Sie bereits mit Eclipse entwickeln, dann bietet es sich an, das Eclipse-Plug-in zu verwenden. Auf Ihren eigenen PCs können Sie DBeaver über den Eclipse-Marketplace installieren. Die Konfiguration für den Zugriff aus dem Hochschulnetz heraus als auch von außerhalb sind identisch:

- Schalten Sie auf die DBeaver-Perspektive um.
- Links im Datenbanknavigation fügen Sie eine neue Verbindung hinzu: Rechtsklick → Anlegen → Verbindung.
- Tragen Sie die oben genannten Zugangsdaten ein.
- Wählen Sie den Datenbanktyp aus und klicken Sie auf „Next“.
- Mit „OK“ beenden Sie die Eingabe. Sie können auch mit „Verbindung testen...“ die korrekte Eingabe der Daten prüfen.

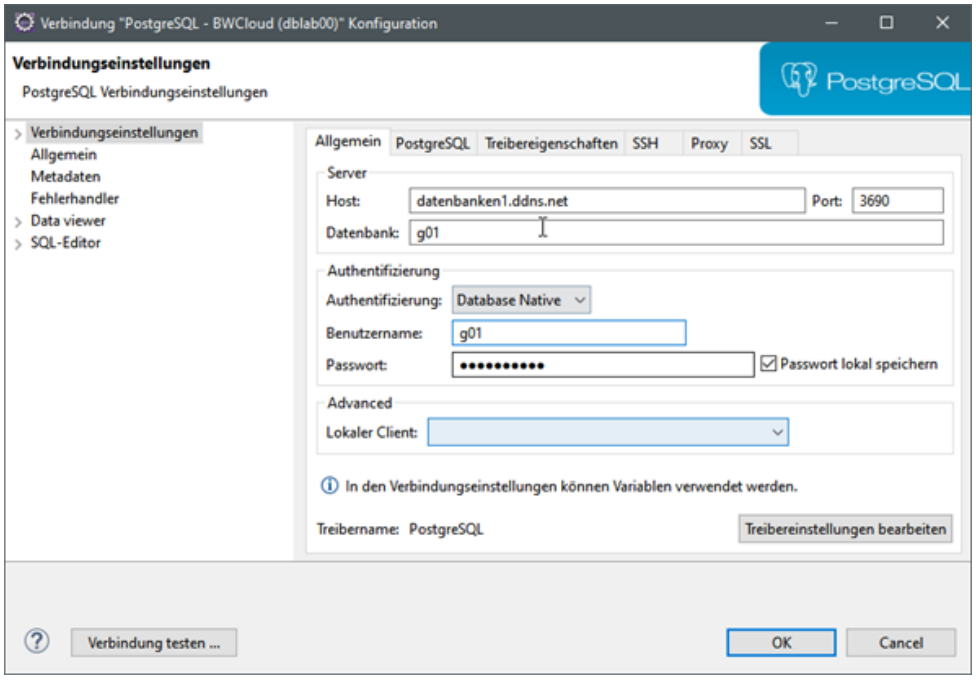

Abbildung 1.3: DBeaver-Verbindung für Zugriff auf die PostgreSQL-Datenbank

In der Standard-Einstellung verwendet DBeaver recht großzügig Datenbankverbindungen: Es beansprucht zwei Verbindungen intern und eine für jeden geöffneten SQL-Editor. Da jede geöffnete Verbindung aber Ressourcen auf dem Server benötigt, ist die Anzahl an offenen Verbindungen je Gruppe auf 20 limitiert. DBeaver kann so konfiguriert werden, dass es weniger Verbindungen offen hält, was für die Aufgabenstellungen keine Einschränkung darstellt. Öffnen Sie den Konfigurationsdialog mit „Windows“ → „Preferences“, und wählen Sie dort links im Baum den Eintrag „Database“.

- Sie können dort unter „Datenbankverbindungen“ und „Metadaten“ ein der beiden intern

genutzen Verbindungen verhindern, indem Sie den Haken vor „Eigene Verbindung für Metadatenabfragen öffnen“ entfernrn.

- Unter „Datenbankeditoren“ und „SQL-Editor“ entfernen Sie den Haken im rechten Dialog vor „Öffnen Sie eine separate Verbindung für jeden Editor“. Damit lassen sich zwar nicht mehr mehrere Transaktionen parallel in den Editoren starten, was aber für die Aufgabnstellungen auch nicht benötigt wird.

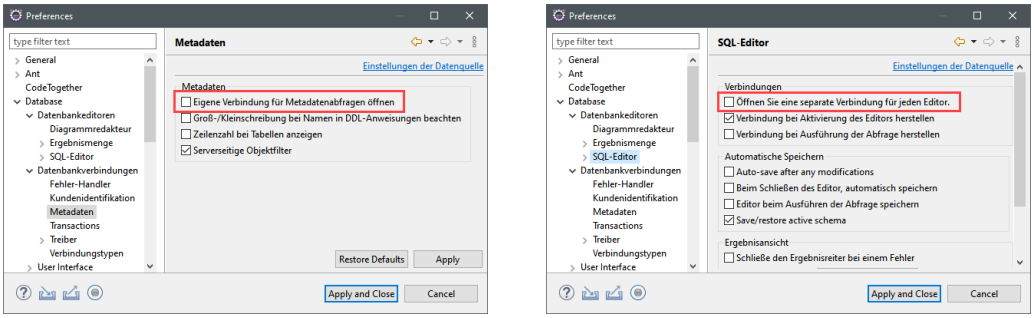

Abbildung 1.4: Datenbankverbindungen für Metadaten und SQL-Editor

1.3.2.2 DbVisualizer

Die Konfiguration für den Zugriff aus dem Hochschulnetz heraus als auch von außerhalb sind identisch:

- Rechtsklick links in „Databases“ → „Create Database Connection“. Sie können hier die Unterstützung eines Wizards nutzen oder die Daten manuell erfassen. Letzters wird in den folgen Punkten beschrieben.
- Tragen Sie die o.g. Daten ein.

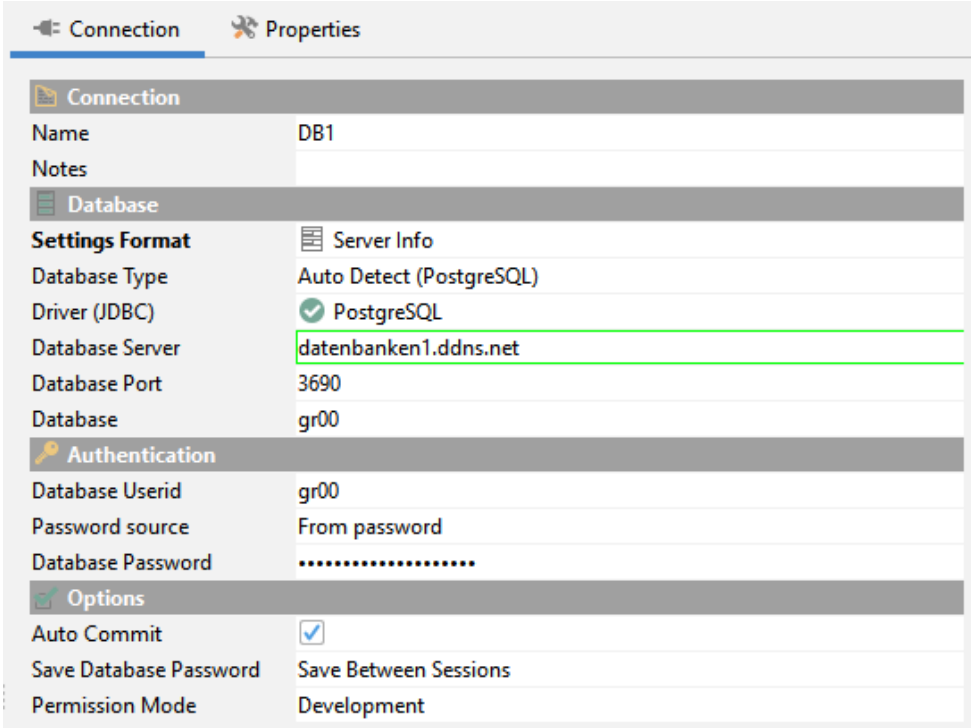

Abbildung 1.5: DbVisualizer-Verbindung für Zugriff auf die PostgreSQL-Datenbank

1.3.3 DBFiddle

Für die reinen SQL-Aufgaben können Sie DB-Fiddle https://www.db-fiddle.com/ verwenden und dort eine von mehreren vorhandenen Datenbanken auswählen. Spätestens für die JDBC- und Hibernate-Aufgaben benötigen Sie aber vollständigen Zugriff auf eine Datenbank.

2. Pflichtaufgaben

Diese Aufgaben müssen Sie lösen.

2.1 Abfragen

Laden Sie die vorgegebenen Skripte des Bike-Shops aus dem ilias herunter und lassen Sie diese einmal in Ihrer Datenbank laufen, damit alle Tabellen, die Sie in den folgenden Aufgaben benötigen, eingerichtet werden. Der Anhang 5 beschreibt die Tabellen.

1.1 Geben Sie alle Zeilen der Relation `teilestamm` aus.

1.2 Welche Zeilen in `teilestamm` haben in ihrem Namen das Wort „City“ in beliebiger Groß- oder Kleinschreibung?

1.3 Welches Teil hat den höchsten Umsatz in der Tabelle `Auftragsposten` erzielt?

1.4 Wie viele `kunden`-, `personal`-, `teilestamm`-Zeilen gibt es jeweils? Drei einzelne SELECT-Anweisungen reichen aus.

1.5 Aus welchem Zeitraum stammen die Aufträge? Beispielausgabe:

```
von         | bis
------------+------------
2008-08-04  | 2008-11-03
```

1.6 Schauen Sie sich die Relationen `kunde`, `personal` und `auftrag` an. Notieren Sie den Namen des Kunden sowie den Namen des zuständigen Mitarbeiters zu Auftrag 2. Welchen Namen hat der Vorgesetzte dieses Mitarbeiters? Lösen Sie diese Aufgabe mit vier einzelnen `SELECT`-Abfragen ohne komplizierte JOINs etc.

1.7 Geben Sie alle Teile, die sich mindestens einmal im Lager befinden, aufsteigend sortiert nach der Anzahl der Teile aus.

1.8 Finden Sie die Nummern aller Teile, die geliefert wurden, und geben Sie diese absteigend, anhand ihrer Nummer sortiert, und ohne Mehrfachnennung aus.

1.9 Zeigen Sie die Teilenummern, Bezeichnungen und (Brutto-)Preise für alle Teile, die brutto mehr als 30 Euro kosten. Geben Sie jeder Spalte der Ausgabe eine selbsterklärende Überschrift.

1.10 Von welchen Einzelteilen des Teils 300001 werden jeweils mehr als 100 cm benötigt? Ausgabe:

```
TEILENUMMER
-----------
500001
500011
```

2.2 Joins und Views

2.1 Ermitteln Sie die Auftragsnummern und das jeweilige Datum für alle Aufträge des Kunden „Fahrrad Shop“ mit Hilfe der `IN`-Anweisung. Beispielausgabe:

```
auftrnr | datum
--------+-----------
1       | 2008-08-04
6       | 2008-11-03
```

2.2 Formulieren Sie Ihre Anfrage aus 2.1 so um, dass Sie statt `IN` jetzt `SOME` verwenden.

2.3 Formulieren Sie die Anfrage aus 2.1 so um, dass Sie statt `IN` jetzt `EXISTS` verwenden.

2.4 Wie viele Aufträge gibt es je Kunde und in welchem Zeitraum lagen sie? Die Kundennummer reicht aus und soll gleichzeitig als Sortierschlüssel dienen. Beispielausgabe:

```
kundnr | anzahl | von        | bis
-------+--------+------------+-----------
1      | 2      | 2008-08-04 | 2008-11-03
3      | 1      | 2008-09-06 | 2008-09-06
4      | 2      | 2008-10-07 | 2008-10-22
6      | 1      | 2008-10-18 | 2008-10-18
```

2.5 Ändern Sie Aufgabe 2.4 so ab, dass nur Kunden mit einem Auftrag gewählt werden.

```
kundnr | anzahl | von        | bis
-------+--------+------------+-----------
3      | 1      | 2008-09-06 | 2008-09-06
6      | 1      | 2008-10-18 | 2008-10-18
```

2.6 Ermitteln Sie mit Hilfe der `JOIN`-Anweisung alle Kunden mit ihren zugehörigen Auftragsnummern. Beispielausgabe:

```
nr | name               | auftrag
---+--------------------+--------
1  | Fahrrad Shop       | 1
1  | Fahrrad Shop       | 6
3  | Maier Ingrid       | 2
4  | Rafa - Seger KG    | 3
4  | Rafa - Seger KG    | 5
```

```
6   | Fahrräder Hammerl | 4
```

2.7 Ändern Sie die Lösung zu Aufgabe 2.6 so ab, dass Sie zusätzlich auch noch den Namen des Mitarbeiters ausgeben. Beispielausgabe:

```
nr | name               | auftrag | mitarbeiter
---+--------------------+---------+---------------
1  | Fahrrad Shop       | 1       | Anna Kraus
1  | Fahrrad Shop       | 6       | Anna Kraus
3  | Maier Ingrid       | 2       | Johanna Köster
4  | Rafa - Seger KG    | 3       | Anna Kraus
4  | Rafa - Seger KG    | 5       | Anna Kraus
6  | Fahrräder Hammerl  | 4       | Johanna Köster
```

2.8 Welcher Kunde hat für den höchsten Gesamtpreis Aufträge erteilt? Mit anderen Worten: Wer ist der beste Kunde? Geben Sie den Namen des Kunden inkl. seiner Gesamtauftragssumme aus.

2.9 Schreiben Sie Ihre Lösung der Aufgabe 2.8 so um, dass Sie zunächst eine ungeordnete Hilfstabelle aller Kunde mit ihren Gesamtauftragssummen mit Hilfe der WITH-Anweisung und lesen Sie aus dieser Tabelle den Kunden mit dem höchsten Umsatz aus.

2.10 Erzeugen Sie eine View (Ansicht) `KundenUmsatz` mit allen Kunden und den Summen Ihrer Aufträge. Lesen Sie alle Daten der View aus. Beispielausgabe:

```
name               | summe
-------------------+------
Fahrrad Shop       |  4468
Fahrräder Hammerl  |   824
Maier Ingrid       |  2350
Rafa - Seger KG    |  3286
```

2.11 Löschen Sie die Ansicht (View) aus Aufgabe 2.10 wieder.

2.3 Transaktionen

3. Gehen Sie wie folgt vor:
   - Öffnen Sie eine Transaktion.
   - Bestimmen Sie die Zahl der Lieferungen.
   - Löschen Sie alle Lieferanten.
   - Bestimmen Sie die Zahl der Lieferungen. Warum sind welche verschwunden? Konsultieren Sie zur Beantwortung auch die Datei `hska_oracle_bike.sql`.
   - Stellen Sie durch Beenden der Transaktion den ursprünglichen Zustand wieder her.
   - Bestimmen Sie die Zahl der Lieferungen.

2.4 JDBC-Programmierung

In dieser Aufgabe des Datenbanklabors geht es um Teile einer Implementierung eines Backends für die Warenwirtschaft des Fahrradhändlers. Damit Sie kein komplett neues Projekt erstellen müssen, können Sie das Maven-Projekt `DBLab-JDBC-Projekt` als Basis verwenden. Alle Datenbank-Treiber werden automatisch per Maven heruntergeladen. Laden Sie das Projekt herunter, entpacken Sie die Dateien und importieren Sie es in Eclipse oder eine andere IDE. Sie können statt dessen auch das Projekt `DBLab-JDBC-Projekt Libs.zip` verwenden, das auf Maven verzichtet und alle Bibliotheken direkt eingebunden hat.

2.4.1 SSH-Tunnel

Wenn Sie die Oracle-Datenbank der Hochschule verwenden wollen, dann stehen Sie vor dem Problem, dass diese nur innerhalb des Netzes der Hochschule erreichbar ist. Um trotzdem z.B. von zuhause darauf zugreifen zu können, besteht die Möglichkeit, einen SSH-Tunnel zu verwenden (nähere Informationen dazu in der Vorlesung "Kommunikationsnetze"). Sie haben prinzipiell zwei Varianten zur Verfügung:

1. Verwendung einer separaten Software, die den Tunnel aufbaut.
2. Einbindung einer Java-Bibliothek in Ihr JDBC-Projekt, die den Tunnel Client-seitig implementiert.

Hier wird Ihnen Variante 2 vorgestellt. Die Java-Bibliothek JSch (http://www.jcraft.com/jsch/) implementiert einen SSH-Tunnel. Diese Bibliothek wird intern auch von DBeaver verwendet.

- Binden Sie die Bibliothek JSch in Ihr Projekt ein (siehe auch Vorlesungsbeispiel-Projekte `JDBC-Example` für den Einsatz mit Maven und `JDBC-Example Libs` für die Verwendung ohne Maven).
- Über den SSH-Server `login.h-ka.de` bauen Sie in Ihrem Programm einen Tunnel in das Netz der Hochschule zum Oracle-Server `iwi-i-db-01` auf.

Beispiel für einen lokalen SSH-Client auf Port 22222. Die Parameter `adsName` und `adsPassword` sind Ihr IZ-Account.

```
// SSH-Tunnel aufbauen
JSch jsch = new JSch();
sshSession = jsch.getSession(adsName,
                             "login.h-ka.de", 22);
sshSession.setPassword(adsPassword);
Properties config = new Properties();
config.put("StrictHostKeyChecking", "no");
sshSession.setConfig(config);
sshSession.connect();
sshSession.setPortForwardingL(22222, "iwi-i-db-01", 1521);
```

Jetzt bauen Sie Ihre Datenbank-Verbindung über den SSH-Tunnel auf Ihrem lokalen PC auf, wobei `databaseUser` und `databasePassword` Ihr Datenbank-Account ist. `databaseName` ist der Name der Datenbankinstanz, der Ihrer Gruppe zugewiesen wurde.

```
// Oracle-Treiber
Class.forName("oracle.jdbc.OracleDriver");
// 2. Verbinden mit Anmelde-Daten
Properties props = new Properties();
props.put("user", databaseUser);
```

```
props.put("password", databasePassword);
Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:22222:" + databaseName,
    props);
```

Eine komplette Implementierung finden Sie in der Klasse `DBUtils` des Projekts `JDBC-Example` aus der Vorlesung.

2.4.2 Aufgabenstellung

Die im Projekt vorhandene Methode `reInitializeDB()` der Klasse `JDBCBikeShop` benötigen Sie erst in der letzten Teilaufgabe.

4.1 Datenbank-Verbindung: Stellen Sie per JDBC eine Verbindung zur Datenbank her, geben Sie den Namen der Datenbank sowie Informationen zum JDBC-Treiber auf der Konsole aus. Schließen Sie die Datenbank-Verbindung wieder. Kapseln Sie dabei das Öffnen, den Abruf der Informationen sowie das Schließen der Verbindung in separaten Methoden einer Klasse.

4.2 Arbeit mit dem `ResultSet`: Nehmen Sie Aufgabe 4.1 als Basis und lesen Sie in einer weiteren Methode mit
`SELECT persnr, name, ort, aufgabe FROM personal`
Daten aus der Tabelle `personal` aus. Geben Sie diese Daten zeilenweise in dieser Form aus:

```
persnr   | name             | ort            | aufgabe
number   | char             | char           | char
--------+------------------+----------------+----------
      1 | Maria Forster    | Regensburg     | Manager
      2 | Anna Kraus       | Regensburg     | Vertreter
            ...
      9 | Ernst Pach       | Stuttgart      | Azubi
```

Hinweise zur Formatierung der Spalten:

- Ermitteln Sie die Spaltennamen und -typen dynamisch anhand der Metadaten des `ResultSet`s. Sie dürfen diese Kopfzeilen nicht als feste Zeichenketten in den Quelltext einprogrammiert haben.
- Ermitteln Sie mit den Metadaten aus dem `ResultSet` die maximalen Breiten aller Spalten mit `getColumnDisplaySize`. Diese maximalen Breiten verwenden Sie in der Ausgabe für die einzelnen Spalten.
- Zur Formatierung einer Ausgabe in der festen Spaltenbreite gibt es im JDK bereits Lösungen. Um die Trennlinie zwischen Kopf und Rumpf der Tabelle zu erzeugen, können Sie folgende Codezeile verwenden, wobei `colWidth` die Breite der Spalte enthält:

  ```
  String result = String.format("%-" + colWidth + "s", "-")
                        .replace(' ', '-');
  ```

  Um die ganze Zahl 42 linksbündig in der Spaltenbreite auszugeben, verwenden Sie diese Codezeile:

```
String result = String.format("%-" + colWidth + "d", 42);
```

Für eine rechtsbündige Ausgabe lassen Sie das Minuszeichen weg. Wollen Sie Zeichenketten ausgeben, dann verwenden Sie `"s"` statt `"d"`, für `Double`-, `Float`- und `BigDecimal`-Zahlen `"f"`. Nähere Hinweise zur Formatierung finden Sie in der API-Dokumentation der Klasse `java.util.Formatter`.

- Testen Sie Ihre Ausgabemethode auch mit der folgenden SQL-Anweisung:
  `SELECT * FROM kunde`

4.3 Nehmen Sie die Lösung der vorherigen Aufgabe als Grundlage. In dieser Aufgabe über Sie die Verwendung von `JOIN-Operationen` mit JDBC. Zunächst lesen Sie alle Kunden-Lieferantenbeziehungen in dieser Form aus, wobei Sie die Ausgabemethode aus Aufgabe 4.2 verwenden:

```
kunde              | knr    | lieferant               | lnr
char               | number | char                    | number
-------------------+--------+-------------------------+-------
Fahrrad Shop       |      1 | Firma Gerti Schmidtner  |      1
Fahrräder Hammerl  |      6 | Firma Gerti Schmidtner  |      1
Fahrräder Hammerl  |      6 | MSM GmbH                |      5
Maier Ingrid       |      3 |                         |
Rafa - Seger KG    |      4 | Firma Gerti Schmidtner  |      1
Rafa - Seger KG    |      4 | Rauch GmbH              |      2
                 ...
```

Erweitern Sie jetzt Ihre Methode, so dass Sie die Kunden anhand ihres Namens filtern können. Sie wollen also z.B. alle Beziehungen zwischen Kunden und Lieferanten ermitteln, bei denen der Kundenname einem bestimmten Suchmuster wie z.B. `Rafa%` entspricht. Hinweise:

- Sie dürfen für das Auslesen in beiden Varianten nur eine einzige SQL-Abfrage absetzen und nicht mit mehreren Anfragen das Ergebnis in Java zusammenbasteln.
- Warum müssen Sie mindestens für den 2. Fall ein `PreparedStatement` verwenden?

4.4 Hier üben Sie das Ändern von Zeilen in einer Tabelle.

- Einfügen: Fügen Sie einen neuen Kunden in die Datenbank ein, und weisen Sie ihm einen neuen Auftrag. Zu diesem Auftrag ergänzen Sie einen Auftragsposten, der auf ein bereits vorhandenes Teil im Teilestamm verweist. Dieses Teil können Sie direkt durch Angabe seiner ID referenzieren. Ebenso verfahren Sie beim Auftrag, indem Sie dem Auftrag eine feste ID eines vorhandenen Mitarbeiters zuweisen. Sie können für die neuen Objekte ebenfalls fest in das Programm eingetragene IDs verwenden und müssen nicht per `SELECT` die jeweils höchsten vergebenen IDs ermitteln (Sie dürfen es aber). Verwenden Sie auch hier `PreparedStatement`s. Zur Kontrolle geben Sie nach dem Einfügen mit Ihrer Methode aus Aufgabe 4.2 die Tabellen auf der Konsole aus, in die Sie neue Werte eingefügt haben.
- Aktualisieren: Tragen Sie jetzt eine Sperre in den neuen Kunden mit Hilfe eines `UPDATE`-Befehls ein und prüfen Sie das Ergebnis, indem Sie die Tabelle erneut auf der Konsole ausgeben.
- Löschen: Löschen Sie den eingefügten Kunden wieder aus der Datenbank und geben Sie die Tabellen `kunde`, `auftrag` und `auftragsposten` aus. Und Sie stellen fest: So

einfach ist das Löschen hier doch nicht, weil die Datenbank beim Löschen mitteilt, dass die Tabelle `auftrag` vom Kunden abhängt (und kein `ON DELETE CASCADE` definiert wurde). Wie können Sie den Kunden doch löschen, ohne die Definitionen der Tabellen zu ändern? Implementieren Sie die Lösung.

Hinweis: Setzen Sie die Methode `reInitializeDB()` ein, um die Datenbank beim Testen immer wieder neu aufzusetzen. Die eventuell auftretende Fehlermeldung, dass die Tabelle „farbe“ nicht existiert, können Sie ignorieren, weil diese Tabelle in einer optionalen Aufgabe in Kapitel 4 erstellt wird.

2.5 ORM-Programmierung

In dieser Aufgabe setzen Sie Hibernate als OR-Mapper ein und implementieren eine Anwendung, in der Kunden Flüge buchen können. Im Ilias finden Sie dazu das Projekt „DBLab-ORM-Projekt.zip“, das ebenso wie das JDBC-Projekt auf Maven basiert und alle Datenbank-Treiber sowie die Hibernate-Abhängigkeiten automatisch auflöst. Sie können das Projekt in Eclipse direkt mit „Import existing Project“ einbinden. Das Projekt „DBLab-ORM-Projekt Libs.zip“ verzichtet auf Maven und beinhaltet alle erforderlichen Bibliotheken.

5.1 Entitäten

- **Kunde**: Fügen Sie in das Projekt eine Klasse für Kunden ein, wobei ein Kunde einen Vornamen, einen Nachnamen sowie eine E-MailAdresse besitzt. Keine dieser Angaben darf `null` sein. Verwenden Sie JPA-Annotationen zur Abbildung der Klassen auf die Tabellen. Sie können die Tabellen automatisch anhand der Annotationen erzeugen lassen. Dieses Verhalten ist in der Projektvorlage bereits eingestellt.
- **Flug**: Ergänzen Sie eine Klasse für Flüge. Jeder Flug eine eine Nummer als Kombination aus Buchstaben und Zahlen, eine Startzeit sowie einen Startflughafen. Auch diese Angaben dürfe nicht `null` sein.
- **Buchung**: Kunden können beliebig viele Flüge buchen. Jeder Buchung sind genau ein Kunde sowie ein Flug zugeordnet. Buchungen enthalten die Anzahl der gebuchten Plätze in dem Flug (muss größer oder gleich 1 sein) sowie das Datum, an dem Buchung erfolgte. Das Datum darf nicht `null` sein.

5.2 Erstellen Sie zwei Kunden- sowie drei Flugobjekte und lassen Sie die Kunden jeweils 2 Plätze in zwei Flügen buchen.

5.3 Lesen Sie alle Buchungen eines Kunden aus, indem Sie nach dem Kunden anhand seines Nachnamens suchen. Duplikate der Nachnamen können Sie ignorieren.

Hinweis: Denken Sie auch in dieser Aufgabe daran, am Ende sauber aufzuräumen.

2.6 Datenbankmodellierung

In dieser Aufgabe werden Sie aus einer textuellen Beschreibung ein ER-Modell erzeugen und eine Schema normalisieren.

2.6.1 ER-Modell

Die folgenden Beschreibungen geben Ihnen die Struktur in einem einfachen, fiktiven Programm zur Verwaltung von Stundenplänen an einer Hochschule wieder.

- Ein Stundenplan gehört zu einem Studiengang, wobei jeder Studiengang einen Namen und ein eindeutiges Kürzel hat.

- Eine Veranstaltung wird von einem oder zwei Dozenten betreut. Es kann sich dabei um Professoren, Lehrbeauftragte oder Mitarbeiter der Hochschule handeln.
- Dozenten haben diese Eigenschaften: Name, Vorname, akademischer Grad, Mail-Adresse
- Professoren und Mitarbeiter haben außerdem ein Büro in einem Gebäude sowie Sprechzeiten.
- Jedes Gebäude und jeder Raum in dem Gebäude haben eine Nummer und einen Namen.
- Eine Veranstaltung hat einen Namen und ist entweder ein Labor, eine Übung oder eine Vorlesung.
- Eine Veranstaltung hat die folgenden Eigenschaften:
  - Wochentag, Startzeit, Endzeit
  - ein oder zwei Räume
  - maximal zwei durchführende Dozenten
  - Studiengang, in dem die Veranstaltung angeboten wird
  - Fachsemester, dem die Veranstaltung zugeordnet ist
  - Häufigkeit der Durchführung (z.B. „wöchentlich“, „Blockveranstaltung“ oder „vierzehntägig“)
  - Name
- Ein Stundenplan besteht aus einer Menge von Veranstaltungen.

Aufgaben:

6.1 Dokumentieren Sie den Stundenplan durch ein konzeptionelles ERM-Diagramm.

6.2 Erstellen Sie dazu das logische Modell und führen Sie dort bitte alle Attribute auf.

6.3 Erstellen Sie aus dem physikalischem Modell eine Beschreibung der Tabellen in SQL (DDL) inklusive der Foreign-Key-Beziehungen. Surrogatschlüssel verwenden Sie dabei bitte spärlich Es reicht aus, wenn Sie sich auf die Tabellen für die Veranstaltungen sowie die Räume beschränken.

2.6.2 Normalisierung

Gegeben sei folgende Relation zur Auftragsverwaltung:

MitarbeiterNr	Abteilung	ChefNr	Kunde	Ware	Anzahl
11	VT	5	Miele, Bosch, Miele	Dichtung, TFT, Drehschalter	15, 51, 55
21	VT	5	DB, Miele	Trafo, Dichtung	58, 55
28	EN	8	SEW	Wicklung	70

`MitarbeiterNr` und `ChefNr` seien dabei Fremdschlüssel in eine Personaltabelle, die nicht weiter betrachtet werden muss. Aufgaben:

6.4 Führen Sie die Tabelle für die Auftragsverwaltung in die erste Normalform über.

6.5 Erstellen Sie daraus bitte Tabellen für die Auftragsverwaltung in der dritten Normalform.

3. Bonusaufgabe zur JDBC-Programmierung

Mit der Lösung dieser Aufgabe können Sie im Datenbank-Teil der Klausur maximal 6 Bonuspunkte erreichen, was in dem Teil 10% der möglichen Punkte entspricht. Dazu muss die Lösung vor der Klausur abgegeben werden.

Hier üben Sie das Aktualisieren eines Datenbank mit JDBC. Nehmen Sie dazu das Projekt „DBLab-Weather-Reader“ für Maven bzw. „DBLab-Weather-Reader Labs“ ohne die Verwendung von Maven aus dem Ilias als Basis. Es stellt einen HTTPS-Client sowie einen Parser bereit, um Wetterdaten vom Server des Deutschen Wetterdienstes im JSON-Format herunterzuladen und vereinfacht in Java-Objekten zu speichern. In der Aufgabenstellung wurde dieser Dienst gewählt, weil er ohne API-Key und damit ohne eine Registrierung auskommt. Die Beispielklasse `DemoWeather` zeigt exemplarisch die Verwendung:

1. Erzeugen Sie sich ein `WeatherReader`-Objekt. Das Objekt kapselt die komplette Kommunikation mit dem Server. Für den Zugriff aus dem Hochschulnetz heraus auf den Wetterdienst muss der Proxy-Server verwendet werden. Dazu gibt es einen zweiten Konstruktor, der die Proxy-Zugangsdaten entgegennimmt (Proxy-Adresse, Proxy-Port, RZ-Nutzername und RZ-Passwort).
2. Die Wetterdaten werden für den Ort einer Wetterstation anhand der ID der Station ermittelt. Die IDs sind hier `www.dwd.de/DE/leistungen/klimadatendeutschland/statliste/statlex_html.html` dokumentiert. Verwenden Sie nur solche IDs, in deren Zeile die letzte Spalte leer ist. Diese Spalte enthält nämlich das Datum, an dem die Station außer Betrieb genommen wurde. Für den Einstieg können Sie gerne die Beispiele `10519` für Karlsruhe-Durlach und `10727` für Karlsruhe verwenden. Da sich die verfügbaren IDs immer wieder einmal ändern und die o.g. Liste nicht immer aktuell ist, finden Sie eine Liste der IDs im Labor-Team von Mattermost.
3. Die Stations-ID werden dem Aufruf `readWeatherForecast` übergeben. Das Ergebnis ist eine Vorhersage für mehrere Tage. Auch hier findet keine saubere Fehlerbehandlung statt, und es wird statt dessen `null` zurückgegeben.

Alle Klassen besitzen Javadoc-Kommentare. Die Aufgabe besteht aus vier Teilen:

1. Erstellen Sie eine Tabelle, um die Vorhersagen zusammen mit dem Tag, an dem die Vorhersage abgerufen wurde, zu speichern. Die Daten, die Sie speichern müssen, können Sie aus den Attributen der Klassen `Weather` zusammen mit dem aktuellen Datum ableiten. Verwenden Sie bitte SQL-Datentypen, die zu den Java-Typen der Attribute passen und denken Sie daran, Primärschlüssel zu verwenden. Es soll immer möglich sein, nachträglich

festzustellen, von welcher Wetterstation die Vorhersagen stammten.
2. Lesen Sie die Informationen von 5 verschiedenen Orten aus und speichern Sie diese in der Datenbank. Verwenden Sie dazu Batch-Anweisungen.
3. Durchlaufen Sie die Tabelle mit den Vorhersagen und lesen Sie zu einer Stations-ID die gespeicherten Vorhersagen aus.
4. Geben Sie alle Stations-IDs aus, deren Temperatur an einem bestimmten Datum zwischen zwei Werten, die Sie ebenso wie das Datum frei auswählen können, liegen.

4. Freiwillig zu bearbeitende Aufgaben

Die Lösung dieser Aufgaben erfolgt auf freiwilliger Basis.

4.1 Fahrradshop

O.1 Ändern Sie die Lösung von Aufgabe 2.7 so ab, dass zusätzlich auch die jeweiligen Chefs ausgegeben werden (ist hier aus Platzgründen abgekürzt dargestellt). Beispielausgabe:

```
nr | name               | auftrag | mitarbeiter    | chef
---+--------------------+---------+----------------+---------
1  | Fahrrad Shop       | 1       | Anna Kraus     | Maria...
1  | Fahrrad Shop       | 6       | Anna Kraus     | Maria...
3  | Maier Ingrid       | 2       | Johanna Köster | Maria...
4  | Rafa - Seger KG    | 3       | Anna Kraus     | Maria...
4  | Rafa - Seger KG    | 5       | Anna Kraus     | Maria...
6  | Fahrräder Hammerl  | 4       | Johanna Köster | Maria...
```

O.2 Ändern Sie die Lösung von Aufgabe O.1 so ab, dass Sie zusätzlich die Anzahl der Auftragsposten angeben. Beispielausgabe:

```
nr ... posten
--     ------
1      1
...
4      3
6      3
```

O.3 Ändern Sie die Lösung von Aufgabe O.2 so ab, dass Sie zusätzlich die minimalen und maximalen Preise (aus `teilestamm`) in den Auftragsposten angeben. Beispielausgabe:

```
nr ... posten | minpreis | maxpreis
--     ------ +----------+---------
1      1      | 400      | 400
...
```

```
4       3       | 20        | 94
6       3       | 7,5       | 700
```

4.2 Universität

Diese Datenbank der TU München beschreibt eine kleine Universität und wird hier leicht modifiziert eingesetzt (https://db.in.tum.de/teaching/bookDBMSeinf/daten/index.shtml).

4.2.1 Beschreibung

Zum Ausprobieren der Abfragen laden Sie die Beispielskripte „Universitaet (PostgreSQL).sql“ für eine PostgreSQL-Datenbank bzw. „Universitaet (Oracle).sql“ für eine Oracle-Datenbank von den Übungsseiten aus dem Ilias herunter und erzeugen Sie damit die Datenbanken. Das folgende Diagramm gibt die logische Datenbankstruktur wieder.

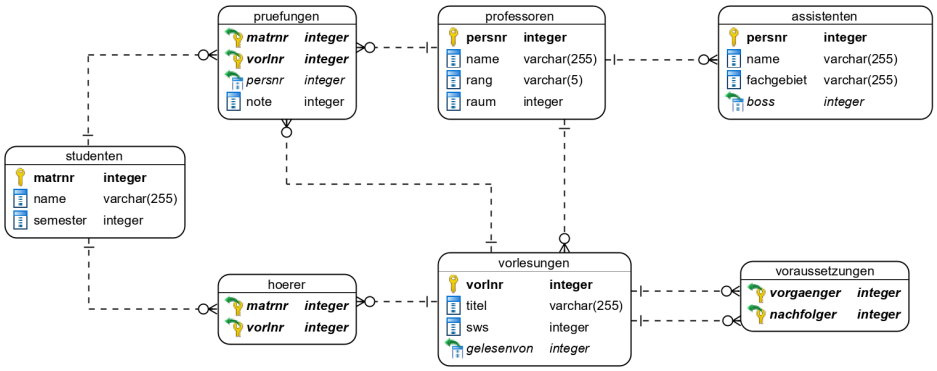

Abbildung 4.1: Universität

- studenten: Informationen zu Studierenden
- professoren: Informationen zu Professoren, `rang` ist die Gehaltsstufe (W1 bis W3).
- vorlesungen: Relation mit allen Vorlesungen. `gelesenvon` ist eine Fremdschlüsselbeziehung zu dem Professor, der die Vorlesung hält.
- assistenten: Assistenten des Professors und diesem über den Fremdschlüssel `boss` zugeordnet.
- hoerer: Zuordnung der Vorlesungen zu den Studierenden, die diese Vorlesungen besuchen.
- pruefungen: Prüfungsleistungen. `persnr` ist der Professor als Prüfer.
- voraussetzungen: Die Vorlesungen haben inhaltliche Voraussetzungen. Der Fremdschlüssel `vorgaenger` verweist auf die Vorlesung, die vor Besuch der Vorlesung `nachfolger` besucht werden solle.

4.2.2 Aufgaben

P.1 Geben Sie die Namen aller Professoren sowie die Namen der Räume, in denen die Professoren ihre Büros haben, aus.

P.2 Geben Sie die Namen der Assistenten zusammen mit den Namen ihrer jeweiligen Bosse aus.

P.3 Geben Sie die Namen der Vorlesungen aus, in denen die Note 1 vergeben wurde.

P.4 Erweitern Sie die Lösung aus der vorherigen Aufgabe, indem Sie auch die Namen der Studierenden, die eine 1 geschrieben haben, ausgeben.

P.5 Geben Sie die Namen aller Vorlesungen mit ihren SWS aus, die einen Umfang von 3 oder 4 SWS haben. Verwenden Sie den `BETWEEN`-Operator.

P.6 Geben Sie die Namen aller Vorlesungen mit ihren SWS aus, die einen Umfang von 3 oder 4 SWS haben. Verwenden Sie den `UNION`-Operator. Verwenden Sie den `BETWEEN`-Operator.

P.7 Wie hoch ist der durchschnittliche Zeitumfang aller Vorlesungen in SWS?

P.8 Geben Sie die Namen und Personalnummern aller Professoren und Assistenten als eine gemeinsame Ergebnistabelle aus.

P.9 Geben sie das Ergebnis aus der vorherigen Aufgabe aufsteigend sortiert anhand des Namens und bei gleichem Namen zusätzlich anhand der Personalnummer aus.

P.10 Geben Sie die Matrikelnummer aller Studierenden aus, die noch keine Prüfung mitgeschrieben haben.

P.11 Geben Sie die Titel aller Vorlesungen aus, die keine anderen Vorlesungen als Voraussetzungen haben.

P.12 Geben sie die Namen aller Professoren mit den Titeln ihrer Vorlesungen aus.

P.13 Geben Sie die Namen aller Vorlesungen zusammen mit den Namen der Vorlesungen aus, von denen diese abhängen. Wie berücksichtigen sie Vorlesungen, die keine Voraussetzungen haben?

P.14 Geben Sie die Namen aller Vorlesungen zusammen mit den Studierenden aus, die diese Vorlesungen besuchen. Lassen Sie Vorlesungen ohne Zuhörer weg.

P.15 Ändern Sie die Lösung aus der vorherigen Aufgabe so ab, dass auch Vorlesungen ohne Zuhörer ausgegeben werden.

P.16 Geben Sie die Namen aller Professoren mit dem jeweiligen Umfang ihrer Lehre aus („wieviel SWS Lehrumfang hat der Professor“).

P.17 Geben Sie den Namen des Professors aus, der die höchste Lehrbelastung hat, dessen Vorlesungen also die höchste Anzahl an SWS aufweist. Sie dürfen dazu keine Sortierung der Daten verwenden. Tipp: Erzeugen sie mit `WITH` eine temporäre Hilfstabelle.

P.18 Schreiben Sie eine Abfrage, die genau einmal `true` ausgibt, wenn alle Professoren mit Gehaltsstufe W3 mehr Lehre in Form von SWS leisten als alle Professoren mit Gehaltsstufe W2. Ist das nicht der Fall, dann geben Sie gar nichts aus.

4.3 Mondial

Diese Aufgaben basieren auf der aus der Vorlesung bekannten Mondial-Datenbank.

Q.1 Geben Sie die Namen der Länder aus, in denen mehr als 50% der Bevölkerung deutsch spricht. Ergänzen Sie für jedes Land auch den Prozentsatz.

Q.2 Welche Seen liegen in Norwegen (Ländercode „N“)?

Q.3 Ermitteln Sie in drei separaten Anfragen die Seen mit der größten Fläche, der größten Tiefe sowie den am tiefsten gelegenen See.

Q.4 Ermitteln Sie den See, an den die größte Anzahl an Ländern angrenzt. Sie dürfen keine Sortierung verwenden.

Q.5 Geben Sie die Namen der Länder aus, die politisch von Großbritannien („GB“) abhängig sind.

Q.6 Ermitteln Sie alle politisch von Großbritannien („GB“) abhängigen Länder, deren Industrien (Attribut `industry` in Tabelle `economy`) einen Anteil von weniger als 20% am Bruttosozialprodukt haben.

Q.7 Ermitteln Sie die 10 höchsten Berge, die auf Inseln liegen. Geben Sie die Namen der Berge, deren Höhen, die Namen der Inseln sowie die Namen der Länder, zu denen die Inseln gehören, aus. Sie dürfen hier Sortierung verwenden. Beachten Sie, dass Inseln und Berge zu mehr als einem Land gehören können.

Q.8 Ermitteln Sie die Länder, in der die Summe aller Berghöhen größer als die Summe aller Flusslängen in dem Land ist (wozu auch immer). Geben Sie den Namen des Landes sowie die Summe seiner Berghöhen aus. Das klappt elegant mit Hilfe von `WITH`.

Q.9 Geben Sie Namen des Flusses mit dem längsten Namen aus, der nicht mit „B“ anfängt. Verwenden Sie keine Sortierung..

4.4 ER-Modell

Sie erhalten von einem Kunden den Auftrag, ein Informationssystem zu erstellen. Leiten Sie aus der folgenden Beschreibung des Kunden ein Datenmodell ab, das die verwendeten Geschäftsregeln wiedergibt.

Abbildung 4.2: Entenrennen

- Bei Richmond Arms findet jährlich ein großes Ereignis statt. Richmond Arms ist der Name einer Kneipe direkt neben dem Mühlengraben in West Ashling, einem Dorf in der Grafschaft Sussex. Der Kneipenwirt veranstaltet jedes Jahr ein Entenrennen zugunsten wohltätiger Hilfsorganisationen.

- Schon beim ersten Rennen wurde dem Wirt bewusst, dass lebende Enten für ein organisiertes Rennen nicht ruhig genug sind. Daher kam er auf die Idee, Enten aus Holz zu benutzen. Die Enten werden in den Mühlengraben gesetzt und schwimmen mit der Strömung vom Start bis zur Southbrook-Brücke.
- Gäste der Kneipe, die mit einer Ente am Rennen teilnehmen wollen, der Wirt nennt sie Sponsoren, müssen einen Geldbetrag von mindestens 100 Pfund zu Gunsten einer Hilfsorganisation ihrer Wahl setzen, bevor sie eine Ente bekommen. Die Ente muss in ordentlicher aufrechter Position schwimmen. Veränderungen zum besseren Schwimmverhalten sind erwünscht, mit Ausnahme jeglicher Form von Antrieb.
- Die Sponsoren schmücken mit großem Aufwand ihre Enten. Dabei muss die offizielle Nummer an der vorgeschriebenen Stelle gut sichtbar bleiben. Der Wirt teilt die Enten in verschiedene Dekorationsgruppen ein. Die örtlichen Unternehmen spenden Preise für diese Gruppen. Diese Preise verleiht der Wirt an den Sponsor der originellsten Ente jeder Gruppe.
- Der Sponsor darf die Ente während des Rennens nicht anfassen. Der Wirt hat diese Regel wohlweislich eingeführt, nachdem mehrere übereifrige Sponsoren ins Wasser gesprungen waren, um ihre Enten zu unterstützen. Sie, die Sponsoren, wurden nicht ganz unversehrt stromabwärts in der Gegend von Oakwood Weir an Land gezogen. Die Enten werden immer noch vermisst.
- Jede Ente erhält eine Startnummer und eine Startposition. Die Startpositionen der Enten werden durch das Los bestimmt. Eine Seite der Absperrung wird beim Start geöffnet. Enten mit niedrigen Positionsnummern sind dabei im Vorteil.
- Ein Sponsor kann mehrere Enten auswählen, wenn er unsicher ist, welche Bauweise die Strömung am besten ausnutzt. Um seine Gewinnchancen zu erhöhen, wählt er mehrere Enten aus und verändert jeweils deren Schwimmverhalten im Rahmen der zulässigen Regeln.
- Der Wirt von Richmond Arms will nicht mehr Personen als nötig nachrennen, um deren Einsätze einzusammeln, Er hat auch keine Lust, mit mehreren Sponsoren je Ente zu verhandeln. Deswegen können nicht mehrere Sponsoren gemeinsam auf dieselbe Ente setzen.
- Der Wirt notiert Datum, Zeit und Namen eines Zeugen, wenn einer seiner Gäste eine Ente sponsern möchte. Er tut dies, damit er die Sponsoren an ihre Verpflichtung erinnern kann, falls sie vergessen sollten, ihre Ente am Tag des Rennens abzuholen.
- Die Summe der Einsätze beim Entenrennen wird der Hilfsorganisation übergeben, deren Ente das Rennen gewinnt. Der glückliche Sponsor der Gewinnerente bekommt offiziell nichts für den Sieg seiner Ente, aber der Wirt spendiert ihm das eine oder andere Freibier.

Aufgaben:

O.4 Dokumentieren Sie Ihr Ergebnis durch ein logisches ERM-Diagramm.

O.5 Erstellen Sie ein physikalisches Datenbankschema in SQL inklusive der Fremdschlüssel-Beziehungen. Surrogatschlüssel verwenden Sie dabei bitte spärlich, falls doch, nennen Sie diese bitte „id“, darauf zeigende Fremdschlüssel bitte „xxx_id“.

5. Anhang Bike-Shop

Der Anhang beschreibt die Datenmodelle, die sie zur Lösung der meisten Aufgaben benötigen. Die Aufgaben des Datenbanklabors basieren auf der Bike-Datenbank eines Fahrradhändlers von E. Schicker. Diese Datenbank ist der Praxis entnommen und voll funktionsfähig, mit sowohl einfachen als auch komplexen Relationen. Durch diesen Aufbau ist es möglich, in den Übungen einen Eindruck über die Leistungsfähigkeit von relationalen Datenbanken zu bekommen.

An der Hochschule stehen ihnen die Zugänge zu einer Oracle- und einer PostgreSQL-Datenbank zur Verfügung. Das Übungspaket enthält aber auch SQL-Skripte der Bike-Datenbank für MSSQL und MySQL. Sie können das Installationsskript ausführen, indem Sie den Inhalt der Datei in einen SQL-Editor (z.B. DBeaver) laden und ablaufen lassen. Lesen Sie bitte unbedingt vorher etwaige Kommentare in den Köpfen der jeweiligen Installationsskripte.

Datei	Datenbank	Hinweis
`hska_pgsql_bike.sql`	PostgreSQL	Vorgabe-Datenbank
`hska_pgsql_bike2.sql`	PostgreSQL	optionale Erweiterung
`hska_mysql_bike.sql`	MySQL/MariaDB	Vorgabe-Datenbank
`hska_mysql_bike2.sql`	MySQL/MariaDB	optionale Erweiterung
`hska_oracle_bike.sql`	Oracle	Vorgabe-Datenbank
`hska_oracle_bike2.sql`	Oracle	optionale Erweiterung

Entstanden ist die Bike-Datenbank zur Verwaltung der Warenwirtschaft von Fahrradhändlern. Folgende Überlegungen liegen dabei zugrunde:

- Es existiert eine bestimmte, aber mit der Zeit variable Anzahl von Teilen, die Einzelteile, Zwischenteile sowie Endprodukte umfassen.
- Zwischenteile und Endprodukte bestehen aus einfacheren Teilen. Die Datenbank soll diese Struktur wiedergeben.
- Der aktuelle Lagerbestand muss festgehalten werden. Für geplante Arbeiten benötigte Teile werden reserviert.
- Für die Bestellung neuer Teile werden Lieferanten benötigt. Es muss auch erkennbar sein, welcher Lieferant welches Teil liefern kann.
- Um dem Geschäft nachgehen zu können, müssen Aufträge verwaltet werden, welche

wiederum von Kunden kommen. Es soll auch vermerkt werden, welche Arbeiten zu einem Auftrag anfallen und welcher Mitarbeiter für diese zuständig ist. Da hierfür Teile reserviert werden, besteht auch ein Zusammenhang zum Lager.

- Die Mitarbeiter werden damit auch in der Datenbank geführt. Für Aufträge soll erkennbar sein, welcher Mitarbeiter welchen Auftrag entgegen nahm.
- Um die Datenbank einfach zu halten, wird auf ein Rechnungs- und Mahnwesen verzichtet.

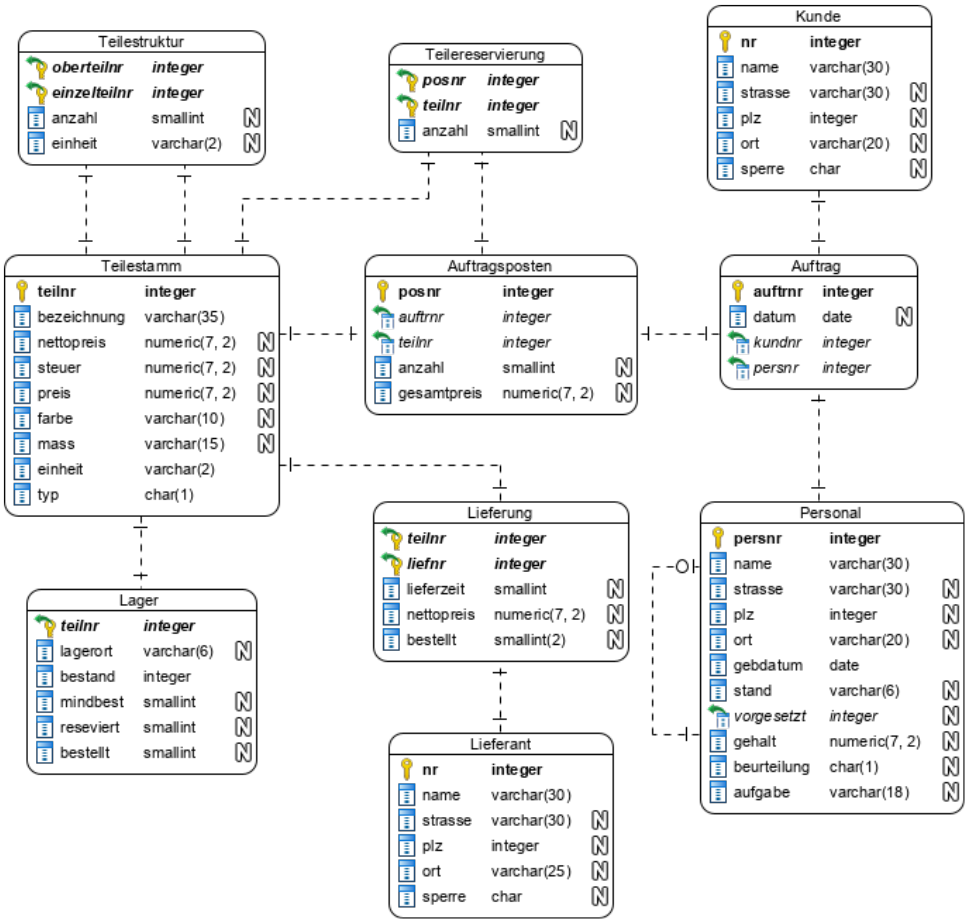

Abbildung 5.1: Logisches ER-Modell

Aufgrund der Vorüberlegungen ist schnell erkennbar, dass die Teileverwaltung und das Auftragswesen eine zentrale Rolle spielen. Dazu gibt es Lieferanten, Kunden sowie Personal. In den folgenden Zeilen sind Schlüsselattribute unterstrichen.

5.1 Teilestamm

Zur Verwaltung der Teile gibt es die Entität `Teilestamm`. Hier finden sich alle Teile, mit denen der Händler in Berührung kommt, vom Einzelteil bis zum fertigen Fahrrad. Dazu wird eine Entität `Lager` benötigt, welche den aktuellen Bestand speichert. Diese ist schwach bezüglich `Teilestamm`, da Teile, die es nicht gibt, auch nicht gelagert werden können. Der Aufbau komplexer Teile wird über eine Beziehungsentität, die `Teilestruktur`, realisiert. `Teilestamm` enthält also detaillierte Informationen über jedes Teil.

Spaltenname	Bedeutung
teilnr	Nummer des Teils
bezeichnung	Bezeichnung des Teils
nettopreis	Preis des Teils ohne Mehrwertsteuer
steuer	Mehrwertsteuer
preis	Preis inkl. Mehrwertsteuer
farbe	Farbe des Teils
mass	Maßangabe
einheit	Einheit, in der das Teil verkauft wird (z.B. ST = Stück)
typ	E = Endprodukt, Z = zusammengesetztes Teil, F = Fremdteil

5.2 Teilestruktur

Die Stückliste Teilestruktur ist eine Beziehungsrelation, die beschreibt, welche zusammengesetzten Teile aus welchen Einzelteilen bestehen. Für jedes Tupel einer Beziehung von komplexem Teil (oberteilnr) zu einem Einzelteil (einzelteilnr) existiert ein Eintrag mit der benötigten Anzahl für das jeweilige Einzelteil und der entsprechenden Einheit. Wird ein Teil beispielsweise aus 10 Einzelteilen zusammengesetzt, gibt es in Teilestruktur 10 Einträge zu diesen Teilen.

Spaltenname	Bedeutung
oberteilnr	Nummer des komplexen Teils
einzelteilnr	Nummer des Einzelteils, das zum komplexen Teil gehört
anzahl	Anzahl der Einzelteile einzelteilnr, die zum komplexen Teil gehören
einheit	Einheit (z.B. ST für Stück, damit gehören anzahl Einzelteil zum komplexen Teil, bei Einheit = CM gehören anzahl Zentimeter des Einzelteils zum komplexen Teil)

5.3 Auftrag

Für das Auftragswesen werden alle Aufträge in der Entität Auftrag hinterlegt. Diese hat Beziehungen zu Kunden (Entität Kunde) als auch Personal. Einzelheiten zu den Aufträgen findet man in Auftragsposten. Zu jedem Auftrag werden hier also nur das Datum, die Kundennummer und die Personalnummer des Vertreters (Persnr) gespeichert. Einzelheiten dazu befinden sich in der Relation Auftragsposten.

Spaltenname	Bedeutung
auftrnr	Nummer des Auftrags
datum	Datum, an dem der Auftrag erteilt wurde
kundnr	Nummer des Kunden, der den Auftrag erteilt hat
persnr	Nummer des Mitarbeiters, der den Auftrag bearbeitet

5.4 Auftragsposten

Diese Relation nimmt mit 2 Beziehungen zum Teilestamm und einer Beziehung zum Auftrag eine zentrale Rolle ein. Eine der Beziehungen zum Teilestamm wird über die Beziehungsrelation

Teilereservierung realisiert. Über die direkte Beziehung wird das in Auftrag gegebene Teil gemerkt, über die Beziehungsrelation Teilereservierung die dafür zu reservierenden Einzelteile.

Spaltenname	Bedeutung
posnr	Nummer der Auftragsposition
auftrnr	Nummer des Auftrags, zu dem diese Position gehört
teilnr	Nummer des bestellten Teils
anzahl	Anzahl der bestellten Teile mit der Nummer teilnr
gesamtpreis	Gesamtpreis dieser Auftragsposition

5.5 Lager

Neben dem aktuellen Bestand finden sich hier auch der Lagerort, der Mindestbestand, die Anzahl der reservierten als auch die Anzahl der bestellten Teile. Dazu muss ein Teil aus dem Teilestamm hier nicht aufgeführt sein, wenn es nicht auf Lager, reserviert oder bestellt ist.

Spaltenname	Bedeutung
teilnr	Nummer des Teils im Lager
lagerort	Ort der Teile im Lager
bestand	Lagerbestand
mindbest	Mindestbestand, der im Lager immer aufrechterhalten werden muss
reserviert	für den Verkauf reservierte Teile
bestellt	bereits bestellte Teile

5.6 Lieferung

Die Relation gibt an, welche Teile von welchem Lieferanten in welcher Zeit geliefert werden. Außerdem werden der Preis und die aktuellen Bestellungen vermerkt.

Spaltenname	Bedeutung
teilnr	Nummer des Teils
liefnr	Nummer des Lieferanten
lieferzeit	Lieferzeit des Teils in Anzahl an Tagen
nettopreis	Preis des Teils ohne Mehrwertsteuer
bestellt	Anzahl bestellter Exemplare des Teils

5.7 Lieferant

Die Relation enthält alle wichtigen Attribute der Lieferanten. Sperre gibt an, ob z.B. aufgrund schlechter Erfahrungen hier keine Bestellungen mehr getätigt werden sollen. In der Praxis werden oft noch weitere Attribute für Statistiken oder zusätzliche Informationen gespeichert.

Spaltenname	Bedeutung
nr	Nummer des Lieferanten
name	Name des Lieferanten
strasse	Straße und Hausnummer
plz	Postleitzahl
ort	Ort des Lieferanten
sperre	1 = Der Lieferant ist gesperrt, 0 = bei dem Lieferanten darf bestellt werden

5.8 Teilereservierung

Die Relation gibt Auskunft, welche Teile für welchen Auftragsposten reserviert wurden. Es handelt sich um eine Beziehungsrelation, in der die beiden Fremdschlüssel zusammen den Primärschlüssel bilden. Weiterhin gibt es noch die Anzahl der reservierten Teile.

Spaltenname	Bedeutung
posnr	Nummer des Auftragspostens, für den dieses Teil reserviert wurde
teilnr	Nummer des reservierten Teils
anzahl	Anzahl reservierte Exemplare

5.9 Kunde

In Kunde werden alle wichtigen Attribute für Kunden gespeichert. Das Attribut Sperre gibt an, ob von einem Kunden z.B. aufgrund mangelnder Liquidität kein Auftrag mehr angenommen werden darf.

Spaltenname	Bedeutung
nr	Kundennummer
name	Name des Kunden
strasse	Straße und Hausnummer
plz	Postleitzahl
ort	Ort des Kunden
sperre	1 = Der Kunde ist gesperrt, 0 = der Kunde darf beliefert werden

5.10 Personal

Die Relation ist ähnlich wie die Relation Kunde aufgebaut. Neben den Standardattributen befinden sich hier aber noch das Geburtsdatum, der Familienstand, die/der Vorgesetzte (rekursiv!), das Gehalt, eine persönliche Beurteilung und die Aufgabe in der Firma. In der Praxis werden häufig noch viele mehr Daten über Mitarbeiter gespeichert.

Spaltenname	Bedeutung
`persnr`	Personalnummer
`name`	Name des Mitarbeiters
`strasse`	Straße und Hausnummer
`plz`	Postleitzahl
`ort`	Ort des Mitarbeiters
`gebdatum`	Geburtsdatum
`stand`	`verh` = verheiratet, `led` = ledig
`vorgesetzt`	Personalnummer des Vorgesetzten, ist `null`, wenn die Person keinen Vorgesetzten hat
`gehalt`	Gehalt
`beurteilung`	Beurteilung der Leistung
`aufgabe`	textuelle Beschreibung der Aufgabe in der Firma

Literaturverzeichnis

[DBeaver] https://dbeaver.jkiss.org/

[DbVis] https://www.dbvis.com/download/

[MobaXterm] https://mobaxterm.mobatek.net/

[Putty] http://www.putty.org/

[Sch14] Schicker E.: Datenbanken und SQL, Springer-Vieweg, ISBN-13: 978-3834817327, 2014 (als e-book für Angehörige der Hochschule kostenlos unter http://dx.doi.org/10.1007/978-3-658-16129-3)

[SSH-Key] http://sshkeychain.sourceforge.net/mirrors/SSH-with-Keys-HOWTO/SSH-with-Keys-HOWTO-4.html

[SquirreL] https://sourceforge.net/projects/squirrel-sql/

[Uni] https://db.in.tum.de/teaching/bookDBMSeinf/daten/index.shtml