JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

Fußballverein

INSY Übung
4AHITM 2015/16

Kanyildiz Muhammedhizir

Version 1.0

Note:

Betreuer: Prof. Borko Michael

Begonnen am 25. März 2016

Beendet am x. April 2016

Inhaltsverzeichnis

1 Einführung

1.1 Ziele

- Sinnerfassende lesen von der Angabe.
- Eine große Menge von Inserts erstellen.
- Diese große Menge von Inserts in die Datenbank einfügen.
- Eine JDBC Verbindung erstellen und Daten von der Datenbank abrufen.

1.2 Voraussetzungen

- Aufmerksames zuhören im Unterricht.
- Verständnisvolles lesen.
- Grundkenntnisse SQL
- Java Kenntnisse
- Javafx Kenntnisse

1.3 Aufgabenstellung

ERD

- Jede Person ist identifiziert durch eine eindeutige Personalnummer (kurz "persnr"). Außerdem werden zu jeder Person folgende Informationen verwaltet: Vorname ("vname"), Nachname ("nname"), Geschlecht ("geschlecht") und Geburtsdatum ("gebdat"). Für "geschlecht" sind einzig die Eingaben "W", "M" und "N" gültig.

  Jede Person, die am Vereinsleben beteiligt ist, gehört zu genau einer der folgenden 4 Kategorien: Angestellter, Vereinsmitglied (kurz "Mitglied"), Spieler oder Trainer. Für all diese Kategorien von Personen müssen zusätzliche Informationen verwaltet werden:
  Von jedem Angestellten werden das Gehalt ("gehalt"), die Überstunden ("ueberstunden") und die E-Mail-Adresse ("e_mail") vermerkt.
  Für jedes Vereinsmitglied werden ausständige Beitrage ("beitrag") abgespeichert.
  Jeder Spieler hat eine Position ("position") (z.B. Tormann, Stürmer, etc.). Außerdem sind Gehalt ("gehalt") sowie Beginn ("von") und Ende ("bis") der Vertragsdauer abzuspeichern.
  Jeder Spieler ist zumindest einer Mannschaft zugeordnet. Die Spieler innerhalb einer Mannschaft müssen jeweils eine eindeutige Trikot-Nummer ("nummer") zwischen 1 und 99 haben.
  Jeder Trainer hat ein Gehalt ("gehalt") sowie Beginn ("von") und Ende ("bis") der Vertragsdauer. Jeder Trainer ist genau einer Mannschaft zugeordnet.

  Der Verein hat mehrere Mannschaften. Für jede Mannschaft sind folgende Informationen zu verwalten: eine eindeutige Bezeichnung ("bezeichnung") (wie 1. Mannschaft, Junioren, A-Jugend, ...), die Klasse ("klasse") sowie das Datum des nächsten Spiels ("naechstes_spiel"). Jede Mannschaft hat mindestens 11 Spieler, genau einen Chef-Trainer und genau einen Trainer-Assistenten. Beide sind als Trainer des Vereins registriert. Außerdem hat jede Mannschaft einen Kapitän (der ein Spieler des Vereins ist).

  Der Verein hat mehrere Standorte. Jeder Standort ist identifiziert durch eine eindeutige ID ("sid"). Außerdem sind zu jedem Standort folgende Informationen verfügbar: Land ("land"), Postleitzahl ("plz") und Ort ("ort").

An jedem Standort gibt es 1 oder mehrere Fan-Clubs. Jeder Fan-Club hat einen für den jeweiligen Standort eindeutigen Namen ("name"), ein Gründungsdatum ("gegruendet") und einen Obmann ("obmann"), der Vereinsmitglied sein muss. Ein Vereinsmitglied kann von höchstens einem Fan-Club der Obmann sein. Die Fan-Clubs werden von Angestellten des Vereins betreut: Und zwar kann jeder Angestellte einen Fan-Club jeweils für einen gewissen Zeitraum betreuen. Dieser Zeitraum ist durch Anfangsdatum ("anfang") und Endedatum ("ende") bestimmt. Jeder Angestellte kann 0 oder mehere Fan-Clubs betreuen, und jeder Fanclub kann von einem oder mehreren Angestellten betreut werden.

Der Verein führt Aufzeichnungen über die absolvierten Spiele. Jedes Spiel ist gekennzeichnet durch die Bezeichnung der Mannschaft ("mannschaft") und das Datum ("datum"). Für jedes Spiel werden der Gegner ("gegner") und das Ergebnis ("ergebnis") eingetragen, das einen der Werte "Sieg", "Unentschieden" oder "Niederlage" haben kann. Außerdem werden zu jedem Spiel die beteiligten Spieler abgespeichert, wobei für jeden Spieler die Dauer Einsatzes ("dauer") als Wert zwischen 1 und 90 vermerkt wird.

Rand-Bedingung

- Schreiben Sie die nötigen CREATE-Befehle, um die vorgestellten Relationen mittels SQL zu realisieren. Dabei sind folgende Punkte zu beachten:
  * Die Datenbank soll keine NULL-Werte enthalten.
  * Realisieren Sie folgende Attribute mit fortlaufenden Nummern mit Hilfe von Sequences: "persnr" von Personen und "sid" von Standorten. Für das "persnr"-Attribut sollen nur gerade, 5-stellige Zahlen vergeben werden (d.h.: 10000, 10002, ..., 99998). Für das "sid"-Attribut sollen beliebige positive Zahlen (d.h. 1,2, ...) vergeben werden.
  * Falls zwischen 2 Tabellen zyklische FOREIGN KEY Beziehungen herrschen, dann sind diese FOREIGN KEYs auf eine Weise zu definieren, dass es möglich ist, immer weitere Datensätze mittels INSERT in diese Tabellen einzufügen.

- Schreiben Sie INSERT-Befehle, um Testdaten für die kreierten Tabellen einzurichten. Jede Tabelle soll zumindest 100000 Zeilen enthalten. Sie dürfen die Wahl der Namen, Bezeichnungen, etc. so einfach wie möglich gestalten, d.h.: Sie müssen nicht "real existierende" Fußballer-Namen, Länder, Städte, etc. wählen. Stattdessen können Sie ruhig 'Spieler 1', 'Spieler 2', 'Land 1', 'Land 2', 'Stadt 1', 'Stadt 2', etc. verwenden. Sie können für die Erstellung der Testdaten auch entsprechende Generatoren verwenden!

- Schreiben Sie die nötigen DROP-Befehle, um alle kreierten Datenbankobjekte wieder zu löschen.

SQL-Abfragen

- (Fan-Club Betreuung) Wählen Sie "per Hand" die Personalnummer eines Angestellten aus Ihren Testdaten aus. Schreiben Sie eine SQL-Anfrage, die jene Fan-Clubs ermittelt, die dieser Angestellte im Moment nicht betreut. Geben Sie zu jedem derartigen Fan-Club die Standort-ID und den Namen des Fan-Clubs aus.
  Bemerkung: Ein Fan-Club wird von einem Angestellten im Moment nicht betreut, wenn entweder der Angestellte diesen Fan-Club überhaupt nie betreut hat oder wenn das heutige Datum (= sysdate) außerhalb des Betreuungszeitraums liegt. Vergessen Sie nicht, jene Fan-Clubs zu berücksichtigen, die von überhaupt keinem Angestellten betreut werden (dieser Fall sollte zwar laut Datenmodell nicht vorkommen. Die Einhaltung dieser Bedingung wird aber vermutlich vom Datenbanksystem nicht überprüft)!

- (Die eifrigsten Angestellten) Schreiben Sie eine SQL-Anfrage, die den Nachnamen und die Personalnummer jener Angestellten ausgibt, die im Moment sämtliche Fan-Clubs betreuen.

Ordnen Sie die Nachnamen alphabetisch.
Bemerkung: Passen Sie die Testdaten so an, dass diese Anfrage zumindest zwei Angestellte liefert.

- (Spielereinsätze) Geben Sie für alle Spiele des Jahres 2015 jeweils alle Spieler und die Dauer ihres Einsatzes aus, d.h.: Gesucht sind alle Tupel (mannschaft, datum, vorname, nachname, dauer), mit folgender Eigenschaft:
  "mannschaft" ist die Bezeichnung der Mannschaft, die gespielt hat.
  "datum" ist das Datum, an dem das Spiel stattfand.
  "vorname" und "nachname" beziehen sich auf einen Spieler, der bei diesem Spiel zum Einsatz kam.
  "dauer" gibt die Dauer des Einsatzes (in Minuten) dieses Spielers bei diesem Spiel an.

- (Spieler-Ranking) Geben Sie für jeden Spieler den Vornamen und Nachnamen sowie die Gesamtdauer ("gesamtdauer") der von ihm bei Spielen im Jahr 2015 geleisteten Einsätze aus. Vergessen Sie nicht, jene Spieler des Vereins zu berücksichtigen, die im Jahr 2015 bei keinem einzigen Spiel mitgespielt haben (d.h. gesamtdauer = 0). Ordnen Sie die Ausgabe in absteigender Gesamtdauer. Bei Gleichheit der Gesamtdauer sollen die Spieler in alphabetischer Reihenfolge (zuerst des Nachnamen, dann des Vornamen) sortiert werden.

- (Der fleißigste Spieler) Geben Sie den Vornamen und Nachnamen jenes Spielers aus, von dem die unter b) berechnete Gesamtdauer am größten ist, d.h.: dieser Spieler ist bei Spielen im Jahr 2015 insgesamt am längsten im Einsatz gewesen. Falls sich mehrere Spieler den ersten Platz teilen (d.h. sie kommen auf die gleiche Gesamtdauer), dann sollen diese in alphabetischer Reihenfolge (zuerst des Nachnamen, dann des Vornamen) geordnet werden. Der Fall, dass im Jahr 2015 überhaupt kein Spiel stattfand, darf ignoriert werden.
  Bemerkung: Berücksichtigen Sie bei Ihren Testdaten die Situation, dass sich zumindest 2 Spieler den ersten Platz teilen.

- Schreiben Sie CREATE und DROP Befehle für eine View, die alle Informationen über Trainer aus der Personen- und Trainer-Tabelle zusammenfügt, d.h.: sowohl die allgemeinen Personendaten (Personalnummer, Vorname, Nachname, Geschlecht und Geburtsdatum) als auch die Trainer-spezifischen Informationen (Gehalt sowie Beginn und Ende der Vertragsdauer). In Summe ist also folgende View erforderlich:
  Trainer_view (persnr, vname, nname, geschlecht, gebdat, gehalt, von, bis).

Datenbankclient

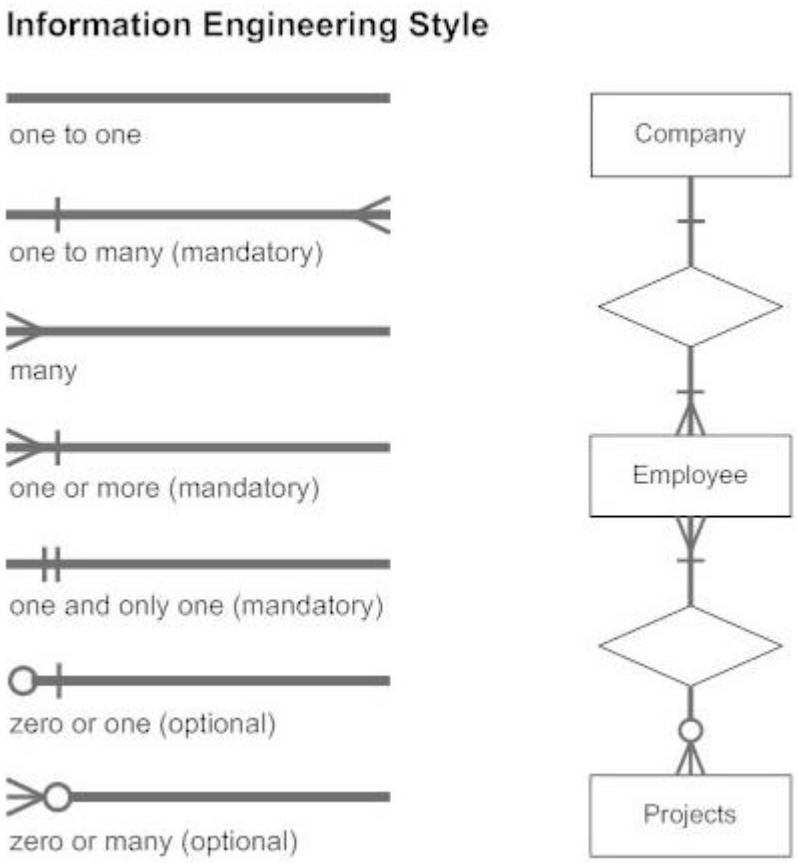
- Schreiben Sie einen Client, der eine Datenbank-Verbindung herstellt. Realisieren Sie eine GUI (JavaFX/Qt), die das einfache Ändern (CRUD) der Spieler des Vereins erlaubt. Verwenden Sie dabei auf jeden Fall eine Tabelle (TableView, QTableView), die auch eine grafische Veränderung der Datensätze erlauben soll.

2 ERD-Erstellen

2.1 Grundkenntnisse

Das Entity-Relationship-Modell dient dazu, im Rahmen der semantischen Datenmodellierung einen in einem gegebenen Kontext, z. B. einem Projekt zur Erstellung eines Informationssystems, relevanten Ausschnitt der realen Welt zu beschreiben. Das ER-Modell besteht aus einer Grafik (ER-Diagramm) und einer Beschreibung der darin verwendeten Elemente, wobei deren Bedeutung (Semantik) und ihre Struktur dargestellt wird.

- Entität (Entity): individuell identifizierbares Objekt der Wirklichkeit
- Beziehung (Relationship): Verknüpfung / Zusammenhang zwischen zwei oder mehreren Entitäten
- Eigenschaft (engl. attribute): Was über eine Entität (im Kontext) von Interesse ist



2.2 Beziehung, Kardinalität

Die Kardinalität einer Beziehung beschreibt, wie viele Objekte eines Objekttyps der einen Seite mit wie vielen Objekten eines Objekttyps der anderen Seite in Beziehung stehen. Es gibt folgende Möglichkeiten:

- 1 : 1
- 1 : N
- M : N
- 1 : N weak

2.3 RM zum ERD

Person (persnr, vname, nname, geschlecht, gebdat)
Angestellter (Person.persnr, gehalt, ueberstunden, email)
Mitglied (Person.persnr, beitrag)
Spieler (Person.persnr, position, gehalt, vertragvon, vertragbis)
Trainer (Person.persnr, gehalt, vertragsbeginn, vertragsende)
Mannschaft (bezeichnung, klasse, neachstesspiel)
Spiel (datum, Mannschaft.bezeichnung, gegner, ergebnis)
FanClub (name, *Mitglied.persnr*, Standort.sid, gegruendet, istobman)
Standort (sid, land, plz, ort)
SpielerEigenschaft (Mannschaft.bezeichnung, Spieler.persnr, istkapitean, nummer)
Betreut(FanClub.name, FanClub.sid, Angestellter.persnr, ende, anfang)
Spielt (Spieler.persnr, Spiel.datum, dauer)

3 SQL

3.1 SQL Datentypen

Die einzelnen DBMS-Hersteller haben diese Liste jedoch um eine Unzahl weiterer Datentypen erweitert. Die wichtigsten Standarddatentypen sind:

`integer`

Ganze Zahl (positiv oder negativ), wobei je nach Zahl der verwendeten Bits Bezeichnungen wie smallint, tinyintoder bigint verwendet werden. Die jeweiligen Grenzen und die verwendete Terminologie sind vom Datenbanksystem definiert.

`numeric (n, m) || decimal (n, m)`

Festkommazahl (positiv oder negativ) mit insgesamt maximal n Stellen, davon m Nachkommastellen. Wegen der hier erfolgenden Speicherung als Dezimalzahl ist eine besonders für Geldbeträge notwendige Genauigkeit gegeben.

`float (m)`

Gleitkommazahl (positiv oder negativ) mit maximal `m` Nachkommastellen.

`real`

Gleitkommazahl (positiv oder negativ). Die Genauigkeit für diesen Datentyp ist jeweils vom Datenbanksystem definiert.

`double || double precision`

Gleitkommazahl (positiv oder negativ). Die Genauigkeit für diesen Datentyp ist jeweils vom Datenbanksystem definiert.

`float || double`

sind für technisch-wissenschaftliche Werte geeignet und umfassen auch die Exponentialdarstellung. Wegen der Speicherung im Binärformat sind sie aber für Geldbeträge nicht geeignet, weil sich beispielsweise der Wert 0,10 € (entspricht 10 Cent) nicht exakt abbilden lässt.

`character (n) || char (n)`

Zeichenkette Text mit `n` Zeichen.

`varchar (n) || character varying (n)`

Zeichenkette (also Text) von variabler Länge, aber maximal n druckbaren und/oder nicht druckbaren Zeichen. Die Variante `varchar2` ist
für Oracle spezifisch, ohne dass sie sich tatsächlich unterscheidet.

`text`

Zeichenkette (zumindest theoretisch) beliebiger Länge. In manchen Systemen synonym zu `clob`.

`date`

Datum (ohne Zeitangabe)

`time`

Zeitangabe (evtl. inklusive Zeitzone)

`timestamp`

Zeitstempel (umfasst Datum und Uhrzeit; evtl. inklusive Zeitzone), meistens mit Millisekundenauflösung, teilweise auch mikrosekundengenau

`boolean`

Boolesche Variable (kann die Werte `true`(wahr) oder `false` (falsch) annehmen). Dieser Datentyp ist laut SQL:2003 optional und nicht alle DBMS stellen diesen Datentyp bereit.

`blob (n) || binary large object (n)`

Binärdaten von maximal n Bytes Länge.

`clob (n)` || `character large object (n)`

Zeichenketten mit maximal `n` Zeichen Länge.

Wenn es die Tabellendefinition erlaubt, können Attribute auch den Wert `NULL` annehmen, wenn kein Wert bekannt ist oder aus anderen Gründen kein Wert gespeichert werden soll. Der `NULL`-Wert ist von allen anderen möglichen Werten des Datentyps verschieden.

3.2 SQL JOINS

Join als Kreuzprodukt

In der allereinfachsten Form des Joins werden sämtliche Datensätze der ersten Tabelle mit sämtlichen Datensätzen der zweiten Tabelle zusammengeführt, indem man das sogenannteKreuzprodukt der Tabellen bildet.

X

XA	XB
1	2
2	3
3	4

Y

YA	YB	YC
1	2	3
2	3	4
3	4	5

X, Y

XA	XB	YA	YB	YC
1	2	1	2	3
1	2	2	3	4
1	2	3	4	5
2	3	1	2	3
2	3	2	3	4
2	3	3	4	5
3	4	1	2	3
3	4	2	3	4
3	4	3	4	5

Für den praktischen Datenbankeinsatz ist diese Reinform in der Regel unbrauchbar, da keinerlei Beziehungen zwischen den Daten beachtet werden.

EQUI-JOIN

Während beim Kreuzprodukt keinerlei Anforderungen an die Kombination der Datensätze gestellt werden, führt der Equi-Join eine solche ein: Die Gleichheit von zwei Spalten. Im Gegensatz zur ersten Variante sind hier also nur noch die Datensätze in der Ergebnismenge, die das Kriterium der Gleichheit erfüllt haben.

NATURAL-JOIN

Ein Natural Join ist eine Kombination von zwei Tabellen, in denen Spalten gleichen Namens existieren. Die Werte in diesen Spalten werden sodann auf Übereinstimmungen geprüft (analog Equi-Join), Das vorliegende Beispiel ist genau so gewählt, dass in beiden Relationen eine Spalte Aund eine Spalte B existiert. Genau wie beim Equi-Join werden A=A und B=B geprüft. Im Anschluss an die Zusammenführung werden die Spalten paarweise zu einer einzigen A- bzw. B-Spalte zusammengefasst.

Einige Datenbanksysteme erkennen das Schlüsselwort NATURAL und eliminieren entsprechend automatisch doppelte Spalten.

Die in Datenbanken weiterhin gebräuchlichen Joins werden in nachfolgenden Abschnitten separat behandelt:

- INNER JOIN
- LEFT/RIGHT JOIN
- FULL OUTER JOIN

3.3 CREATE SCRIPT

Die CREATE TABLE-Anweisung dient zur Anlage von Tabellen im DBS. Sie gehört zum Sprachumfang der SQL-DDL. Hier ist nur der Auszug aus dem Befehl zu finden, der für das Anlegen von Spalten
und Integritätsbedingungen relevant ist. Die vielen Optionen zur Spezifikation einer optimierten Speicherung der Daten sind hier nicht zu finden, zum einen, weil sie äußerst umfangreich sind und zudem trotz aller Standardisierungen doch recht herstellerspezfisch ausfallen. Für weitere Recherchen sei auf die Online-Hilfen der verschiedenen Hersteller verwiesen.

3.4 CREATE VIEW

Ansichten können als virtuelle Tabellen angesehen werden. Allgemein ließe sich sagen, dass eine Tabelle über einen Satz von Definitionen verfügt und Daten physikalisch abspeichert. Eine Ansicht verfügt ebenfalls über einen Satz von Definitionen, die auf einer oder mehreren Tabellen oder anderen Ansichten aufbauen, speichert die Daten aber nicht physikalisch ab.

3.5 INSERT-ANWEISUNG

Die INSERT-Anweisung fügt mindestens eine neue Zeile zu einer Tabelle hinzu. In einem vereinfachten Fall weist INSERT die folgende Form auf:

INSERT [INTO] table_or_view [(column_list)] data_values

Die INSERT-Anweisung fügt die mit data_values angegebenen Datenwerte als eine oder mehrere Zeilen in die angegebene Tabelle oder Sicht
ein. column_list ist eine Liste von durch Trennzeichen getrennten Spaltennamen, die zum Angeben der Spalten verwendet werden kann, für die Daten bereitgestellt werden. Falls column_list nicht angegeben wird, erhalten alle Spalten in der Tabelle oder Sicht Daten.

Wenn column_list nicht alle Spalten in einer Tabelle oder Sicht angibt, wird entweder der Standardwert (wenn ein Standard für die Spalte definiert wurde) oder NULL in jede Spalte eingefügt, die nicht in der Liste angegeben ist. Alle nicht in der Spaltenliste angegebenen Spalten müssen entweder NULL-Werte zulassen, oder es muss ihnen ein Standardwert zugewiesen sein.

3.6 COPY-ANWEISUNG

COPY Befehle sind dazu da um Datensätze schnell und einfach von einer Datenbank/Tabelle in einem File zu speichern. Man kann auch Datensätze von einem File in die Datenbank einsetzten/kopieren.
Der Unterschiede von einem COPY Befehl und einem Insert ist es, dass ein COPY Befehl mehrere Datensätze auf einmal in die Datenbank/Tabelle kopieren kann.

Extracting all employees to a tab delimited file:

```
\copy (SELECT * FROM employees) TO '~/employees.tsv';
```

Extracting all employees to a csv delimited file:

```
\copy (SELECT * FROM employees) TO '~/employees.csv' WITH (FORMAT CSV);
```

Extracting all employees to a binary file (note the quotes around the word Binary):

```
\copy (SELECT * FROM employees) TO '~/employees.dat' WITH (FORMAT "Binary");
```

And for loading data into a table the equivalent for each of the above:

```
\copy employees FROM '~/employees.tsv';
\copy employees FROM '~/employees.csv' WITH CSV;
\copy employees FROM '~/employees.dat' WITH BINARY;
```

3.7 SELECT-ANWEISUNG

Die SQL-Abfrage erfolgt mit dem Befehl SELECT unter Angabe von bis zu sechs Komponenten. Die allgemeine Syntax hat die Gestalt:

SELECT [ALL | DISTINCT] {spalten | *}
FROM tabelle [alias] [tabelle [alias]] ...
[WHERE {bedingung | unterabfrage}]
[GROUP BY spalten [HAVING {bedingung | unterabfrage}]]
[ORDER BY spalten [ASC | DESC] ...];

Die schwierige Syntax lässt sich wie folgt verstehen:

Klausel	Erläuterung
SELECT [DISTINCT]	**Wähle** die Werte aus der/den Spalte(n) **[mehrfache Datensätze nur einmal]**...
FROM	... **aus** der Tabelle bzw. den Tabellen ...
WHERE	... **wobei** die Bedingung(en) erfüllt sein soll(en) ...
GROUP BY	... und **gruppiere** die Ausgabe von allen Zeilen mit gleichem Attributwert zu einer einzigen ...
HAVING	... **wobei darin** folgende zusätzliche Bedingung(en) gelten müssen/muss ...
ORDER BY [ASC/DESC]	... und sortiere nach den Spalten **[auf- bzw. absteigend]**.

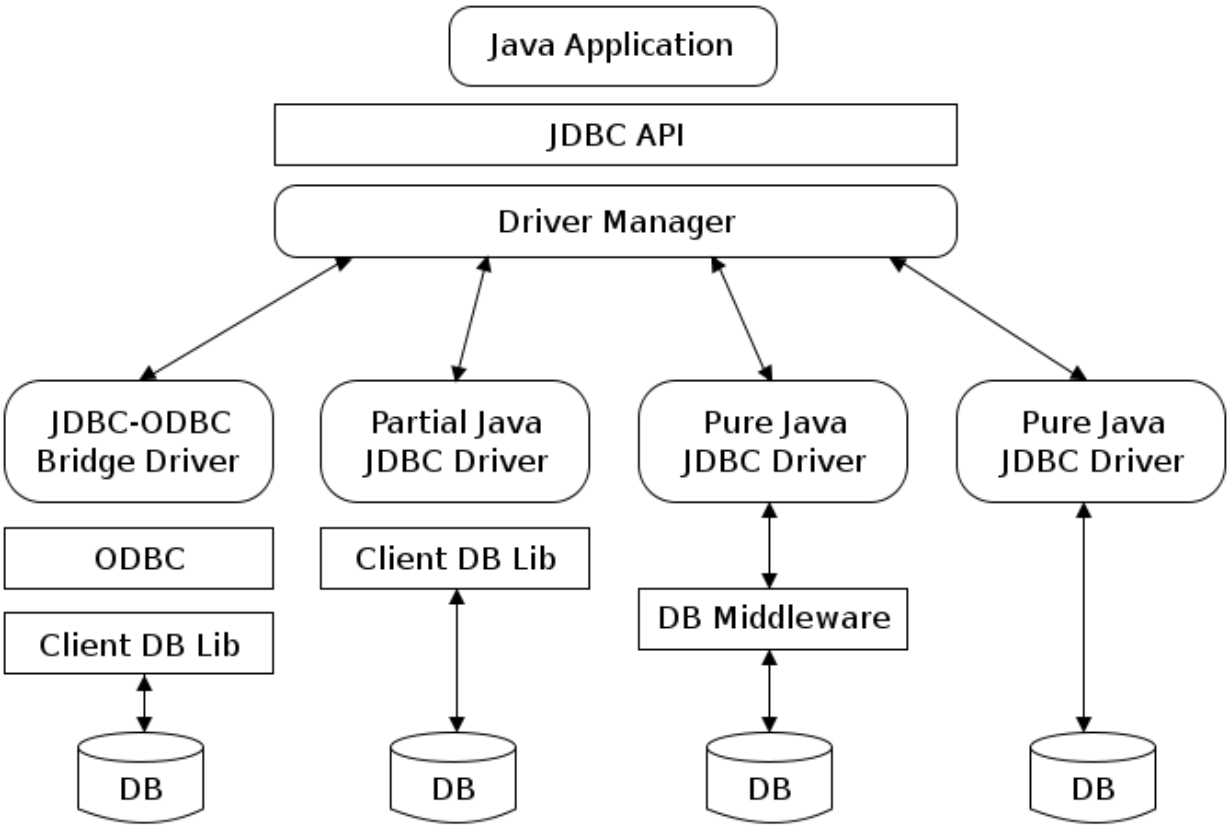
Die SQL-Befehle sind maximal 256-Zeichen lang und müssen mit Semikolon abgeschlossen werden. Attributbezeichner, die Leerzeichen oder Satzzeichen enthalten müssen in eckigen Klammen gesetzt werden.

4 JDBC (Java Database Connectivity)

Java Database Connectivity ist eine Datenbankschnittstelle derJava-Plattform, die eine einheitliche Schnittstelle zu Datenbankenverschiedener Hersteller bietet und speziell auf relationale Datenbanken ausgerichtet ist.

Zu den Aufgaben von JDBC gehört es, Datenbankverbindungen aufzubauen und zu verwalten, SQL-Anfragen an die Datenbank weiterzuleiten und die Ergebnisse in eine für Java nutzbare Form umzuwandeln und dem Programm zur Verfügung zu stellen.

Für jede spezifische Datenbank sind eigene Treiber erforderlich, die die JDBC-Spezifikation implementieren. Diese Treiber werden meist vom Hersteller des Datenbank-Systems geliefert.



5 JAVAFX

JavaFX ist ein Framework für plattformübergreifende Rich Internet Applikation. Es ist eine Java-Spezifikation von Oracle. JavaFX-Anwendungen in der Version 1.x werden mithilfe von JavaFX Scrip programmiert. Ab der Version 2.x werden JavaFX-Anwendungen in Standard-Java geschrieben. Altternativ kann man Oberflächen mit einer XML-Sprache names FXML beschreiben und damit eine Trennung nach dem MVC-Konzept (Model-View-Controller) umsetzen (der Controller ist typischerweise in Java geschrieben). Die in den Java-Klassenbibliotheken enthaltenen Funktionen und Schnittstellen sind bei der Entwicklung von JavaFX-Applikationen verfügbar.

5.1 JavaFX Scene Builder

Der JavaFX Scene Builder ist ein grafisches Tool, das die Erstellung von FXML-Dateien vereinfacht. Mit dem Tool können GUI-Elemente ohne Programmierkenntnisse entworfen werden. Oracle hat die Entwicklung des Tools eingestellt.

6 PostgreSql

6.1 User erstellen

create user user01 with password 'user01';

6.2 Rechte Vergeben

alter user user01 with superuser;

create database fussballverein with owner user01;

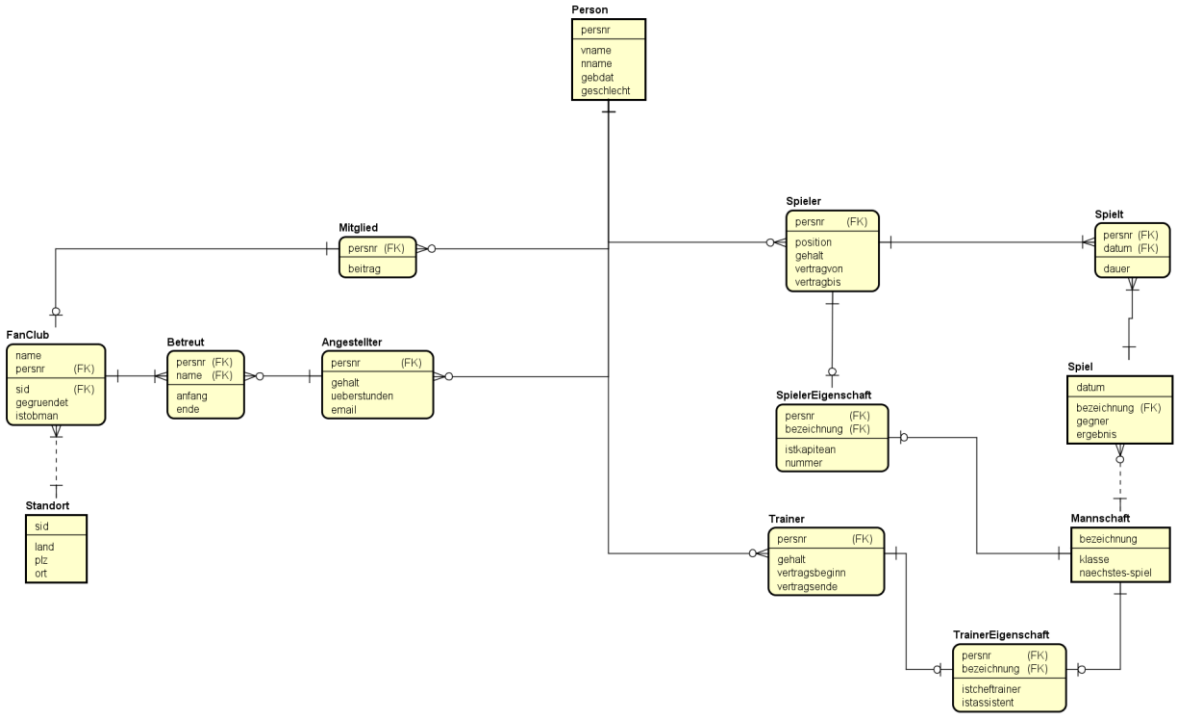
grant all privileges on database fussballverein to user01;

6.3 Einloggen mit dem User

psql -U user01 -d fussballverein -h 127.0.0.1 -W

7 Fußballverein

7.1 ERD



7.2 RM

Person (persnr, vname, nname, geschlecht, gebdat)
Angestellter (Person.persnr, gehalt, ueberstunden, email)
Mitglied (Person.persnr, beitrag)
Spieler (Person.persnr, position, gehalt, vertragvon, vertragbis)
Trainer (Person.persnr, gehalt, vertragsbeginn, vertragsende)
Mannschaft (bezeichnung, klasse, neachstesspiel)
Spiel (datum, Mannschaft.bezeichnung, gegner, ergebnis)
FanClub (name, *Mitglied.persnr*, Standort.sid, gegruendet, istobman)
Standort (sid, land, plz, ort)
SpielerEigenschaft (Mannschaft.bezeichnung, Spieler.persnr, istkapitean, nummer)
Betreut(FanClub.name, FanClub.sid, Angestellter.persnr, ende, anfang)
Spielt (Spieler.persnr, Spiel.datum, dauer)

7.3 Create Script

Astah erstellt das Create Script automatisch solange man das ERD richtiggemacht hat.

Ist im Literatur Verzeichnis: GIT HUB

7.4 INSERT

Durch ein selbst geschriebenes Java Programm sind die Inserts erstellt worden. Eigentlich war es sehr leicht, aber ein bisschen aufwendig.

Wichtige Java Elemente:

- BufferedReader
- BufferedWritter
- Array
- Random Date

BufferedReader

BufferedReader ermöglicht eine Zeilenweise Ein-lesung aus Textdateien.

```
BufferedReader bufferedReader = null;
//Der Pfad zur Textdatei
String filePath ="C:/beispiel.txt";
File file = new File(filePath);
try {
    //Der BufferedReader erwartet einen FileReader.
    //Diesen kann man im Konstruktoraufruf erzeugen.
    bufferedReader = new BufferedReader(new FileReader(file));
  String line;
  //null wird bei EOF oder Fehler zurueckgegeben
  while (null != (line = bufferedReader.readLine())) {
    //Zeile auf der Konsole ausgeben
    System.out.println(line);
    //Hier kann Ihr Code stehen ...
  }
} catch (IOException e) {
    e.printStackTrace();
} finally {
  if (null != bufferedReader) {
    try {
      bufferedReader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

BufferedWritter

Um Text in eine Datei zu schreiben, benötigt man einen FileWriter. Man kann auch gleich mit dem FileWriterschreiben, aber es empfiehlt sich einen BufferedWriter darüber zu legen, das ist effizienter.

Einen Writer oder OutputStream sollte man immer mit close() schließen. Wenn man einen BufferedWriter schließt, wird auch der darunterliegende Stream geschlossen.

Wenn man große Datenmengen in ein File reinschreiben möchte sollte man ein flush() machen, denn es könnte sein, dass kein Byte mehr im BufferedWritter passen könnte.

```
import java.io.*;

class WriteFile
{
  public static void main(String[] args) throws IOException
  {
    FileWriter fw = new FileWriter("ausgabe.txt");
    BufferedWriter bw = new BufferedWriter(fw);

    bw.write("test test test");
    bw.write("tset tset tset");

    bw.close();
  }
}
```

So sollte übrigens kein File ausschauen, denn es fehlt ein Try Catch rundherum. (Sehr wichtig bei Files)

Random Date

```
public class RandomDateOfBirth {

    public static void main(String[] args) {

        GregorianCalendar gc = new GregorianCalendar();

        int year = randBetween(1900, 2010);

        gc.set(gc.YEAR, year);

        int dayOfYear = randBetween(1, gc.getActualMaximum(gc.DAY_OF_YEAR));

        gc.set(gc.DAY_OF_YEAR, dayOfYear);

        System.out.println(gc.get(gc.YEAR) + "-" + (gc.get(gc.MONTH) + 1) + "-" + gc.get(gc.

    }

    public static int randBetween(int start, int end) {
        return start + (int)Math.round(Math.random() * (end - start));
    }
```

```
int year = // generate a year between 1900 and 2010;
int dayOfYear = // generate a number between 1 and 365 (or 366 if you need to handle leap ye
Calendar calendar = Calendar.getInstance();
calendar.set(Calendar.YEAR, randomYear);
calendar.set(Calendar.DAY_OF_YEAR, dayOfYear);
Date randomDoB = calendar.getTime();
```

Ist im Literatur Verzeichnis: GIT HUB

8 Probleme

Einer der großen Probleme war es, die Angabe zu verstehen und das ERD zu erstellen. Da die Angabe auch ein bisschen „offen" formuliert ist. Die Inserts zu erstellen war eigentlich kein Problem, aber das Inserten der Inserts in die Datenbank war etwas problematisch (ohne Copy Befehle). Da es mindestens 100 000 Datensätze pro Tabelle existieren müssen. Das Programmieren mit JavaFx war auch etwas kompliziert, da ich keine Erfahrung mit JavaFx hatte. Einer der großen Probleme war, dass ich mit 2 Fenstern immer ein neues Objekt vom Controller lade/erstelle. Daher habe ich dies einfach mit einem Tab Fenster gemacht, da dies nur einmal ein Objekt vom Controller erstellt.

9 Aufwandsabschätzung

Arbeit	Stundenabschätzung
ERD + Create Script	1:50 – 2:00 h
INSERTS mittels Java	5:00 – 6:00 h
INSERTS in die DB	1:30 – 2:00 h
JavaFx Applikation	3:00 – 4:00 h
SQL Abfragen	1:30 – 2:00 h

10 Literatur Verzeichnisse

Information
GitHub: https://github.com/mkanyildizo1/TGM-SEW-INSY/tree/master/INSY/Fussballverein

11 Quellen

INDEX	Quelltitel + Information
[1]	Wikipedia: https://docs.oracle.com/javase/7/docs/api/java/io/BufferedWriter.html Kapitel: 2 zuletzt abgerufen am [2.3.2016]
[2]	Online: https://git-scm.com/download/gui/linux Kapitel: 3,4 zuletzt abgerufen am [2.3.2016]
[3]	Online: http://mobaxterm.mobatek.net/ Kapitel: 5 zuletzt abgerufen am [2.3.2016]
[4]	Online: http://www.w3schools.com/sql/sql_dates.asp Kapitel: 3,4 zuletzt abgerufen am [2.3.2016]