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

ääBeendet 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

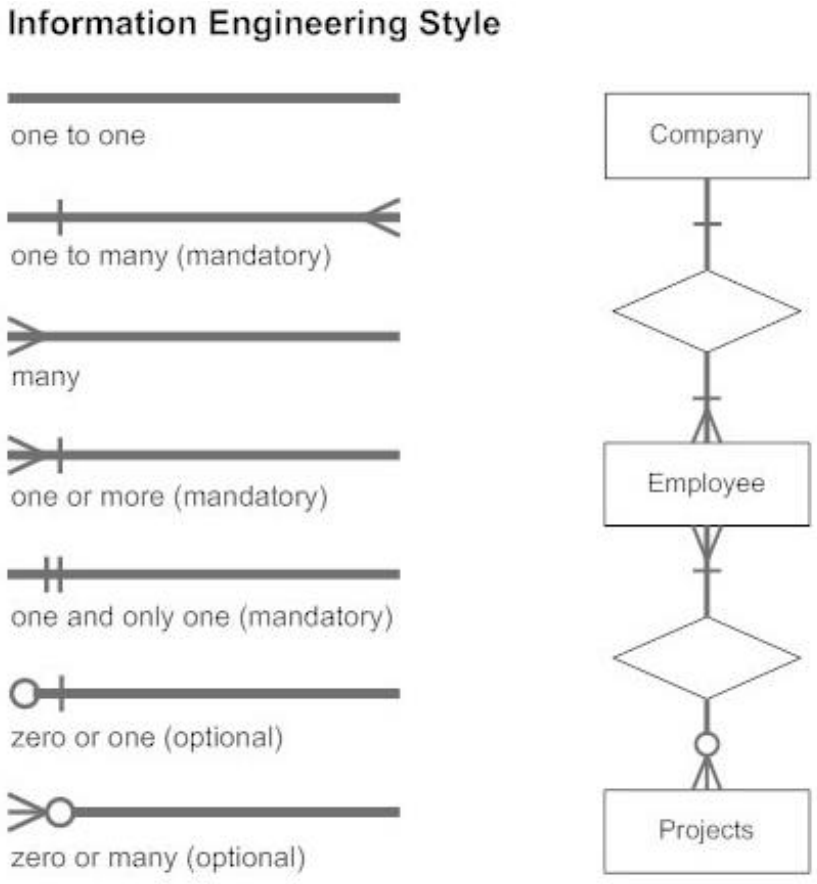
2 ERD-Erstellen

2.1 Erklärung

Ein Entity Relationship Diagram (ERD) ist eine grafische Darstellung der Tabellenstruktur in einer Datenbank. Wie der Nameschon sagt, stellt ein Entity Relationship Diagram Entitäten und Beziehungen zwischen diesen dar. Die Entitäten sind dabei die Tabellen in einer Datenbank. Das Relationship bildet die Beziehungen zwischen den Tabellen.

In relationalen Datenbanken werden Beziehungen zwischen Datenbanken mit Primärschlüssel (PK) und Fremdschlüsseln (FK) definiert. Der Primärschlüssel einer Tabelle ist ein eindeutiger Identifikator für die Tabellenzeilen. Dieser kann als Fremdschlüssel in den Zeilen einer anderen Tabelle eingefügt werden. Durch übereinstimmende Schlüsselwerte in den beiden Tabellen werden dann die Beziehungen zwischen den Zeilen der einen und den Zeilen der anderen Tabelle erkannt.

2.2 Beispiel



2.3 Beziehung, Kardinalität

Erklärung

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:

1:1/0

Jeder Primärschlüssel kann in der Fremdschlüsseltabelle maximal einmal vorkommen.

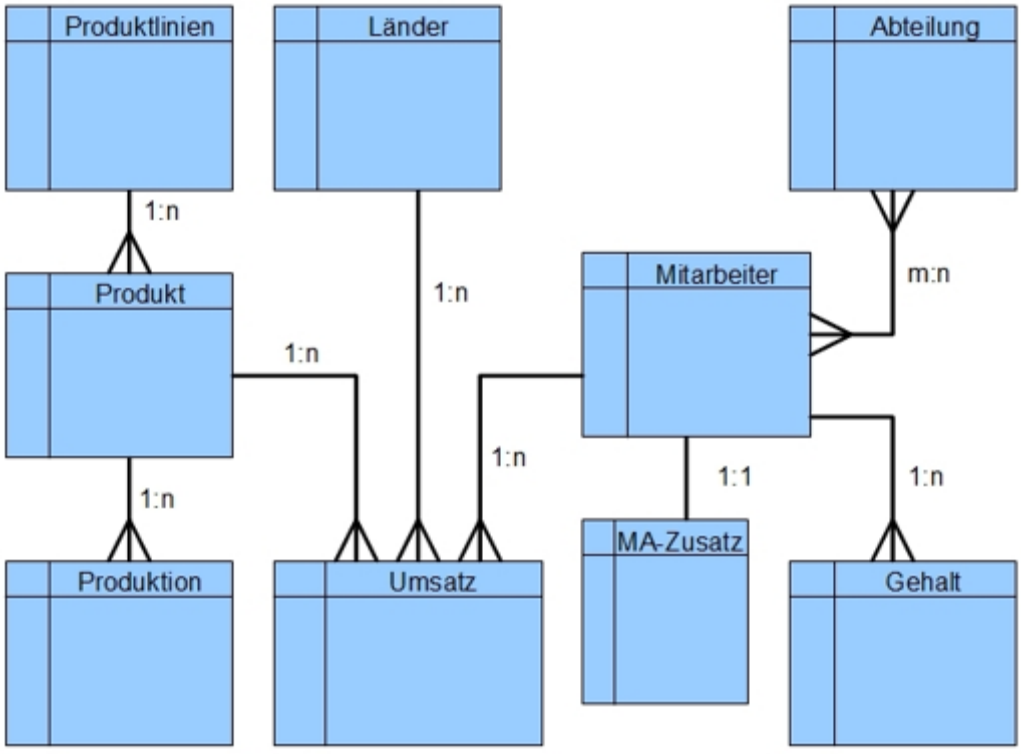
1:n

Jeder Primärschlüssel kann in der Fremdschlüsseltabelle einmal oder mehrmals vorkommen. Das ist die üblichste Form.

m:n

Beide Tabellen können mehrfach aufeinander verweisen. Diese Form ist in relationalen Datenbanken nicht direkt abbildbar und muss z.B. mit einer Mapping-Tabelle aufgelöst werden. Mit den Beziehungsarten befasst sich auch das Thema Join.

Beispiel



2.4 RM Beispiel

Erklärung

In dem relationalen Modell müssen alle Daten in Beziehungen (Tabellen) gespeichert werden, und jede Beziehung besteht aus Zeilen und Spalten. Sie können extrapolieren, dass der Übergang von einer Spalte und einer Zeile in einen eindeutigen Wert führen wird - dieser Wert wird ein Tupel genannt.

Das zweite Hauptmerkmal des relationalen Modells ist die Verwendung von Schlüsseln. Diese werden für speziell dafür vorgesehenen Spalten innerhalb einer Beziehung verwendet um Daten zu bestellen oder beziehen sich auf alle Daten von der andere Beziehung. Einer der wichtigsten Schlüssel ist der Primärschlüssel. Dieser Identifiziert jeden Datensatz.

Unterstrichen	Primärschlüssel
Strichlirt	Fremdschlüssel
Unterstrichen,Strichlirt	Week PK/FK

Beispiel

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 [4] [8]

3.1 SQL Datentypen

Erklärung

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

Numerische Datentypen

Datentyp	Speicherplatz	Optionen	Beschreibung
TINYINT	1 Byte	[(M)] [U] [Z]	Ganzzahlen von 0 bis 255 oder von -128 bis 127.
SMALLINT	2 Bytes	[(M)] [U] [Z]	Ganzzahlen von 0 bis 65.535 oder von -32.768 bis 32.767.
MEDIUMINT	3 Bytes	[(M)] [U] [Z]	Ganzzahlen von 0 bis 16.777.215 oder von -8.388.608 bis 8.388.607.
INT	4 Bytes	[(M)] [U] [Z]	Ganzzahlen von 0 bis ~4,3 Mill. oder von -2.147.483.648 bis 2.147.483.647.
INTEGER	4 Bytes	[(M)] [U] [Z]	Alias für INT.
BIGINT	8 Bytes	[(M)] [U] [Z]	Ganzzahlen von 0 bis 2^64-1 oder von -(2^63) bis (2^63)-1.

Fließkomma Datentypen

Datentyp	Speicherplatz	Optionen	Beschreibung
FLOAT	4 Bytes	[(M,D)] [U] [Z]	Fließkommazahl mit Vorzeichen. Wertebereich von $-(3{,}402823466\times10^{38})$ bis $-(1{,}175494351\times10^{-38})$, 0 und $1{,}175494351\times10^{-38}$ bis $3{,}402823466\times10^{38}$.
DOUBLE	8 Bytes	[(M,D)] [U] [Z]	Fließkommazahl mit Vorzeichen. Wertebereich von $-(1{,}79769\times10^{308})$ bis $-(2.22507\times10^{-308})$, 0 und 2.22507×10^{-308} bis $1{,}79769\times10^{308}$
REAL	8 Bytes	[(M,D)] [U] [Z]	Alias für DOUBLE.
DECIMAL	M+x Bytes	[(M,D)] [U] [Z]	Fließkommazahl mit Vorzeichen. Speicherbedarf: x=1 wenn D=0, sonst x=2. Ab MySQL 5.1 binär gespeichert, zuvor als String.
NUMERIC	M+x Bytes	[(M,D)] [U] [Z]	Alias für DECIMAL.

String Datentypen

Datentyp	Speicherplatz	Optionen	Beschreibung
CHAR	M Byte(s)	(M) [BINARY]	Zeichenkette fester Länge M. Wertebereich für M: 0 bis 255.
VARCHAR	L+1 Bytes	(M) [BINARY]	Zeichenkette variabler Länge, Maximum ist M. Wertebereich für M: 0 bis 255.
BINARY	M Bytes	(M)	Zum Speichern binärer Strings, unabhängig vom Zeichensatz. Wertebereich für M: 0 bis 255. Weiterer Typ: VARBINARY
BLOB	L+2 Bytes	(M)	Binäres Objekt mit variablen Daten. Weitere Typen: TINYBLOB, MEDIUMBLOB und LONGBLOB. M ist ab Version 4.1 definierbar.
TEXT	L+2 Bytes	(M)	Wie BLOB. Ignoriert beim Sortieren & Vergleichen Groß- und Kleinschreibung. Weitere Typen: TINYTEXT, MEDIUMTEXT, LONGTEXT. M ist ab Version 4.1 definierbar.
ENUM	1 oder 2 Bytes	('val1', 'val2', ...)	Liste von Werten (val1, val2, ...). 65.535 eineindeutige Elemente sind maximal möglich.
SET	x Bytes	('val1', 'val2', ...)	String-Objekt mit verschiedenen Variablen. 64 sind maximal möglich. Speicherbedarf: x ist 1, 2, 3, 4 oder 8.

Date Datentypen

Datentyp	Speicherplatz	Optionen	Beschreibung
DATE	3 Bytes	-	Datum im Format 'YYYY-MM-DD'. Wertebereich von 01.01.1000 bis 31.12.9999.
DATETIME	8 Bytes	-	Datumsangabe im Format 'YYYY-MM-DD hh:mm:ss'. Wertebereich entspricht DATE.
TIMESTAMP	4 Bytes	-	Zeitstempel. Wertebereich: 1.1.1970 bis 19.01.2038. Das Format variiert in den MySQL-Versionen.
TIME	3 Bytes	-	Zeit zwischen -838:59:59 und 839:59:59. Ausgabe: hh:mm:ss.
YEAR	1 Byte	[(2|4)]	Jahr zwischen 1901 bis 2155 bei (4) und zwischen 1970 bis 2069 bei (2).

3.2 SQL JOINS [5]

Erklärung

Abfragen über mehrere Tabellen werden JOINS genannt. Man spricht auch davon, Tabellen zu verknüpfen. Einige JOINS können mit Hilfe des Befehls SELECT gebildet werden. Viele RDBMS bieten neben dieser Möglichkeit den Befehl JOIN. In der Mengenlehre stellen JOINS die Vereinigung von Mengen dar.

Beispiel

```
select * from minister
inner join spd_politiker on minister.Name = spd_politiker.Name;
```

ID	Name	ID	Name
1	Frank-Walter Steinmeier	2	Frank-Walter Steinmeier
3	Brigitte Zypries	3	Brigitte Zypries
4	Peer Steinbrück	4	Peer Steinbrück
6	Franz Müntefering	1	Franz Müntefering
10	Ulla Schmidt	5	Ulla Schmidt
11	Wolfgang Tiefensee	6	Wolfgang Tiefensee
12	Sigmar Gabriel	7	Sigmar Gabriel

3.3 CREATE TABLE

Erklärung

Um überhaupt mit Tabellen arbeiten zu können, sprich Datensätze anzuzeigen, zu verändern oder zu bearbeiten, braucht man erste eine Tabelle, wenn sie nicht schon vorhanden ist. Tabellen legt man dabei mit dem Befehl: **CREATE TABLE** an. Möchte man nun eine Tabelle anlegen, muss man nicht nur den CREATE TABLE Befehl kennen, sondern sollte sich schon im Voraus Gedanken gemacht haben, was in der Tabelle alles stehen soll, sprich man braucht die Spalten samt ihren Datentypen. Zusätzlich kann man dann beispielsweise noch den *Primärschlüssel* und *Fremdschlüssel*definieren, kann bestimmen ob Datensätze immer befüllt werden müssen und noch einiges mehr.

Dazu kommen wir aber weiter unten, hier erst einmal die minimalen Voraussetzungen für eine Tabelle:

```
CREATE TABLE tabellennamen
(
   Spaltenname1 datentyp1 [feldeinschränkung]
   Spaltenname2 datentyp2 [feldeinschränkung]
   [...]
);
```

Beispiel

```
CREATE TABLE Filme
(
   FilmNr INTEGER NOT NULL,
   Titel VARCHAR (100),
   Genre VARCHAR (30),
   USK INTEGER,
   Regisseur VARCHAR (30),
   Laenge INTEGER,
   Veroeffentlichung DATE,
    [CONSTRAINT filmnr]
   PRIMARY KEY(FilmNr)
);
```

3.4 DROP TABLE

Erklärung

DROP TABLE Wie wir gelernt haben, kann man mit dem Befehl CREATE TABLE eine Tabelle mit all ihren Spalten erstellen. Gelöscht werden kann eine Tabelle mit dem Schlüsselwort DROP, anschließend muss das Objekt folgen, das gelöscht werden soll, in diesem Fall die Tabelle, sowie der Name der Tabelle. Die Allgemeine Syntax sieht also folgendermaßen aus:

```
DROP TABLE tabellenname;
```

Nach dem Ausführen des DROP Befehls wird keine Sicherheitsabfrage mehr erscheinen. Die Tabelle wird sofort gelöscht, sodass man sich im Klaren sein sollte ob man wirklich die richtige Tabelle angewählt hat.

Beim DROP TABLE Befehl ist das Einzigste was man beachten muss, die Tatsache, dass die bestehenden Integritätsregeln beim Löschen nicht verletzt werden. Beinhaltet z.B. die zu löschende Tabelle einen Fremdschlüssel, wird das Datenbanksystem das Löschen nicht zulassen, da dadurch Verweise auf die Tochtertabelle gelöscht werden.

Beispiel

```
DROP TABLE Filme;
```

3.5 UPDATE TABLE

Erklärung

In bestimmten Fällen kann es erforderlich sein, bereits vorhandene Daten zu aktualisieren. Dies können wir mit dem Befehl UPDATE tun. Die entsprechende Syntax lautet:

Beispiel

```
UPDATE kunden SET Postleitzahl = '47110', Ort = 'Musterdorf'
WHERE Kundennummer = 1;
```

3.6 CREATE VIEW

Erklärung

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.

Der Datenbankbenutzer kann eine Sicht wie eine normale Tabelle abfragen. Wann immer eine Abfrage diese Sicht benutzt, wird diese zuvor durch das Datenbankmanagementsystem berechnet. Eine Sicht stellt im Wesentlichen einen Alias für eine Abfrage dar.

```
CREATE VIEW "SICHT_NAME" AS "SQL-Anweisung"
```

Beispiel

```
CREATE VIEW mitglieder_und_dateien
AS
SELECT pro.proname, p.forename, p.surname, f.fileid
FROM person p
INNER JOIN member_person mp ON p.pid = mp.mppid
INNER JOIN project pro ON mp.mpproid = pro.proid
INNER JOIN file f ON pro.proid = f.is_part_of
ORDER BY pro.proname
;
```

3.7 INSERT-ANWEISUNG [1]

Erklärung

Im in einer Datenbank, Datensätze abrufen zu können muss man die Datenbank irgendwie fühlen.
Die einfachste Lösung ist mittels INSERT-Befehlen, aber der Nachteil ist, dass es alle Inserts nacheinander Einfügen muss, dadurch dauert das füllen der Tabelle bei großen Mengen von Daten sehr lange.
Die Syntax schaut so aus:

```
insert into <tabelle> (<spalte_1>,..,<spalte_n>)
values (<wert_1>,..,<wert_n>);
```

Beispiel

```
insert into einheit (einheit_kurz, bezeichnung)
values ('ml', 'Milliliter');

insert into einheit values ('ml', 'Milliliter');


insert into einheit (einheit_kurz, bezeichnung)
select einheit_ref, 'Bezeichnung von ' || einheit_ref
from artikel;
```

3.8 COPY-ANWEISUNG

Erklärung

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.

Beispiel

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;
```

3.9 SELECT-ANWEISUNG

Erklärung

Der SELECT-Befehl ist der elementare Bestandteil für die Datenabfrage. Mit dem SELECT-Befehl können Inhalte von einer oder mehreren Tabellen ausgewählt und angezeigt werden. So benötigt man für das Auswählen der Daten mit dem SELECT-Befehl, den auszuwählenden Spaltennamen sowie die dazugehörige Tabelle. Die Grundsyntax des SELECT-Befehls lautet folgendermaßen:

```
SELECT spaltenname, spaltenname, [..]
FROM tabellenname;
```

Nach dem Schlüsselwort **SELECT** folgt der oder die *Spaltennamen* die man auswählen möchte. Nach den Spaltennamen kommt das Schlüsselwort **FROM** und anschließend die *Tabelle* in der die Spalten existieren müssen, die ausgewählt werden. Sind die über SELECT ausgewählten Spalten nicht in der nach FROM angegebenen Tabelle, wird ein leeres Ergebnis zurückgegeben.

Generell wird durch den SELECT-Befehl eine virtuelle Tabelle erzeugt, die anschließend ausgegeben wird. Diese virtuelle Tabelle wird auch Recordset genannt, da sie eine Menge (=Set) aus einzelnen Records (= Datenzeilen) ist. Solch ein Recordset besteht immer nur temporär im Arbeitsspeicher und wird nach Beendigung der Ausführung des SELECT-Befehls auch gleich wieder verworfen.

select [**distinct**] <ausdruck_1> [alias_1], <ausdruck_2> [alias_2]...
from <quelle_1>, <quelle_2> ...
where <bedingung>
group by <gruppierungs_ausdruck_1>, <gruppierungs_ausdruck_2> ..
having <bedingung>
order by <sortier_ausdruck_1>, <sortier_ausdruck_2> ...
for update of <spalte_1>, <spalte2> ... [nowait];

<ausdruck>	beliebiger Ausdruck, i.d.R. mit Spalten gebildet
distinct	doppelte Zeilen werden nur einmal geliefert
alias	Ausdruck erhält Namen in der Ergebnismenge
quelle	Tabelle, View, Inline-View
talias	Aliasnamen für die Quelle in diesem Befehl
<bedingung>	beliebig verschachtelte Ausdrücke möglich
group by	Zusammenfassen von Zeilen
having	Bedingung für zusammengefasste Zeilen
order by	Sortierkriterien
for update	Lesen mit gleichzeitigem Sperren der Ergebnismenge
nowait	Zusatz zu "for update": Ist der Satz bereits gesperrt, wird nicht gewartet, sondern ein Fehler produziert.

Beispiel

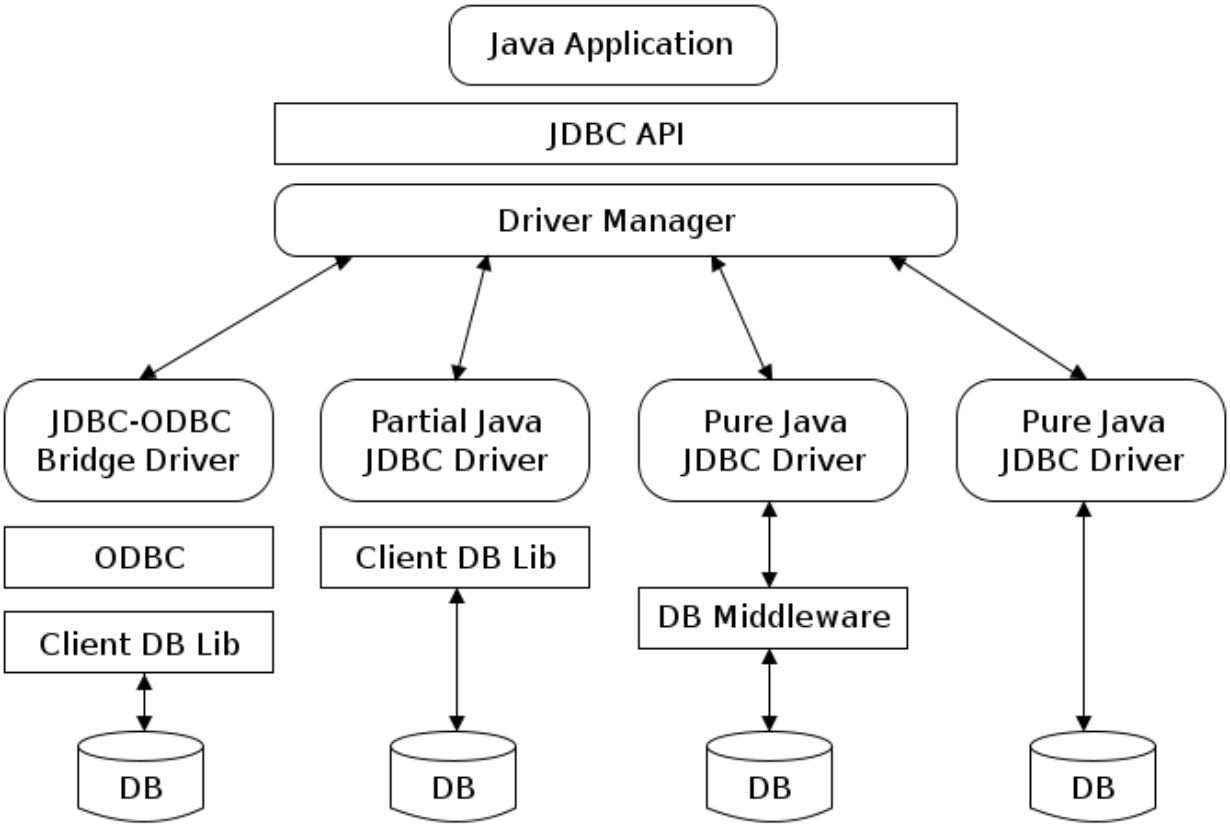
```
SELECT  bezeichnung as mannschaft, s1.datum, vname as vorname, nname as nachname, dauer
FROM Spiel s1
JOIN Spielt s2 ON s2.datum = s1.datum
JOIN Person p1 ON p1.persnr = s2.persnr
WHERE s1.datum between '2015-01-01' and '2015-12-31';
```

4 JDBC (Java Database Connectivity) [6] [8]

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.


Java Application
JDBC API
Driver Manager
JDBC-ODBC Bridge Driver
Partial Java JDBC Driver
Pure Java JDBC Driver
Pure Java JDBC Driver
ODBC
Client DB Lib
Client DB Lib
DB Middleware
DB
DB
DB
DB

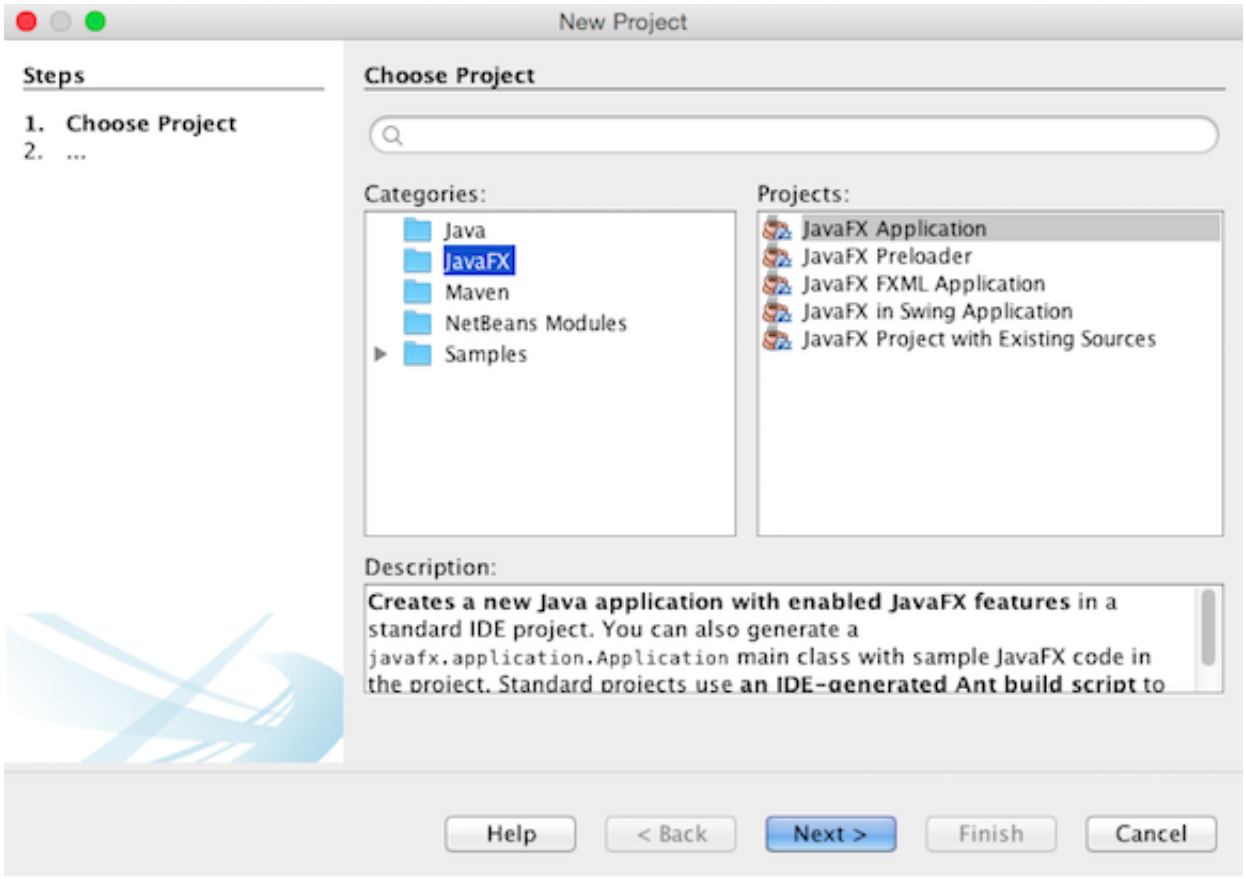
.

5 JAVAFX [1] [6] [7]

Erklärung

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.

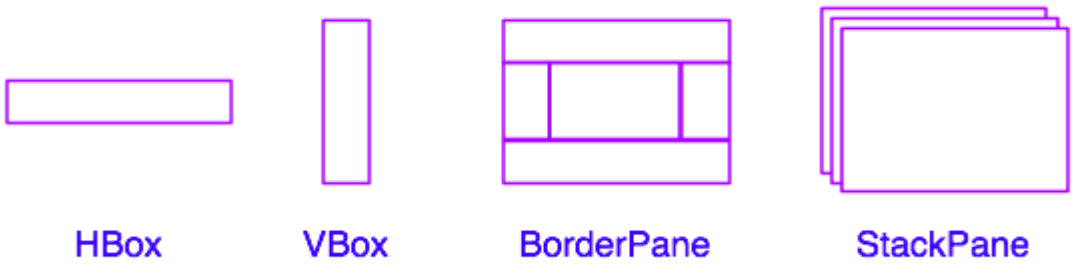
Wenn wir in NetBeans eine JavaFX-Application erstellen, spendiert uns NetBeans ein Stück Beispielcode. Dazu müssen Sie folgende Auswahl treffen, wenn Sie ein neues Projekt erstellen:

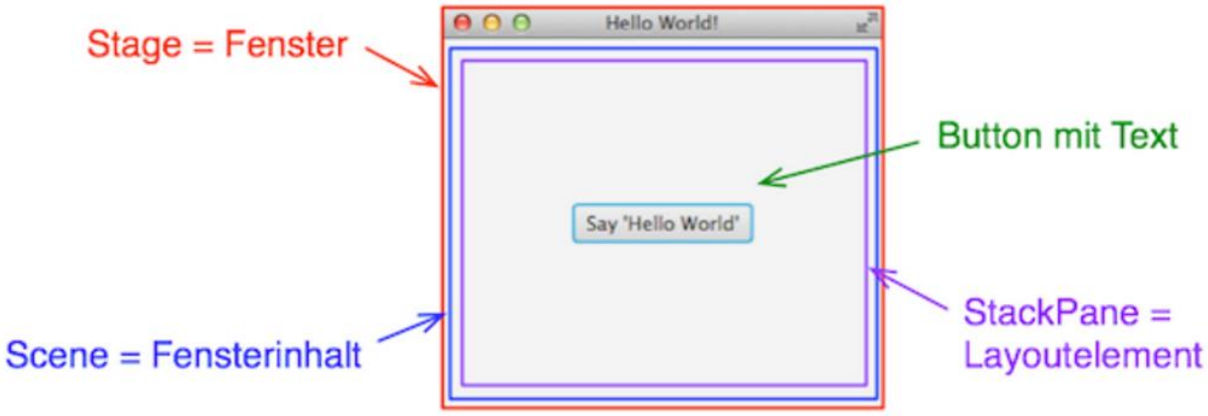


Ein Fenster in JavaFX besteht aus Objekten, die ineinander verschachtelt sind. Auf der obersten Ebene steht ein Objekt vom TypStage (engl. für Bühne), welches das Fenster als Ganzes repräsentiert.

Für alles Inhaltliche verwendet JavaFX das Konzept eines**Szenengraphs** (engl. scene graph), das in der 3D-Grafik häufig verwendet wird. Daher ist die Basis der inhaltlichen Repräsentation ein Objekt vom Typ Scene (engl. für Szene).

Innerhalb des Scene-Objekts wird so etwas wie ein Setzkasten aus verschiedenen strukturellen Objekten hergestellt. Hier einige der **Layout-Klassen**, die JavaFX dafür anbietet (sie haben häufig "pane" im Namen, engl. für Fensterleiste):





JavaFX hat sich eine wichtige Methode aus der Webseiten- Technologie abgeschaut: die Trennung von Layout und Stil. Layout meint hier die Wahl und Anordnung der Elemente (z.B. alle Buttons unten, rechts ausgerichtet). Stil bezieht sich auf Farbe, Form, Schrift etc.

Während das Layout in Java programmiert wird, indem man Layout-Objekte ineinander verschachtelt, wird der Stil in einem separaten Dokument, dem *Stylesheet* angegeben.

Die Schematik orientiert sich an **CSS (cascading style sheets)**, welches in der Webtechnologie entwickelt wurde und mit HTML verwendet wird.

In der CSS-Datei haben Sie bislang **Klassen** wie ".button" gesehen. Eine Klasse betrifft mehrere Einzelelemente (z.B. die zwei Buttons) und wird mit einem Punkt gekennzeichnet. Will man **ein einziges Element**speziell stylen, so kann man eine ID verwenden. Vielleicht möchte man den OK-Button hervorheben, dann gibt man ihm im Code eine ID:

```
Button bOk = new Button("OK");
bOk.setId("special");
```

6 User Erstellen und Rechte vergeben

create user user01 with password 'user01';

alter user user01 with superuser;

create database fussballverein with owner user01;

grant all privileges on database fussballverein to user01;

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

7 GitHub [2]

7.1 Lokales Code-Verzeichnis Git-fähig machen

Auf eurem Rechner öffnet ihr nun ein Terminal und navigiert in euer Code Quellverzeichnis. Falls ihr Git noch nicht installiert habt, solltet ihr das an dieser Stelle nachholen. Pakete dafür gibt es für jede Linux-Distribution. Das Quellverzeichnis wird mit folgendem Kommando „Git-fähig" gemacht:

```
git init
```

Als nächstes wird das lokale Repository mit dem Remote-Repository auf GitHub verbunden, sodass zwischen den beiden Repos ein Austausch stattfinden kann. Dabei bekommt das Repo auf den GitHub-Servern den Namen „origin". Die Adresse, die am Ende des Kommandos steht, ist die SSH-Adresse zum GitHub Repo. Diese kann in GitHub in der rechten Spalte entnommen werden (Vorher Klick auf „SSH"!).

```
git remote add origin git@github.com:ThomasLeister/tl-test.git
```

7.2 Repos abgleichen

Als nächstes müssen die beiden Repos abgeglichen / synchronisiert werden. Während auf GitHub bisher nur die beiden Dateien „LICENSE" und „README" zu sehen sind, befindet sich der Code des Projekts noch im lokalen Repository auf dem Rechner. Im ersten Schritt werden die Dateien „LICENSE" und „README" in das lokale Repo kopiert, damit sie auch dort verfügbar sind und ggf. geändert werden können. Die Änderungen des Master-Branches im Origin-Repo auf den Rechner „ziehen":

```
git pull origin master
```

8 Probleme

8.1 Entity Relation Diagramm

Erklärung

Es war schwer die Angabe zu verstehen, da die Angabe etwas offen gestaltet wurde.

Lösung

ERD Grundkenntnisse anschauen
Skizzieren und Besprechen

8.2 SQL Inserts [1]

Erklärung

Um die Inserts zu erstellen wurde ein Java-Projekt entwickelt.
Dieses Java-Projekt ist sehr aufwendig aber unnötig gemacht worden, da es einen DataFiller gibt mit dem man über Copy-Befehlen alles in die Datenbank Inserten kann (Dies habe ich zu spät bemerkt).

Lösung

In der Java – API nachschauen und auf Stack Overflow.

8.3 PostgreSQL [8]

Erklärung

Zuweisung der Rechte auf die gewünschte Datenbank.
Datensätze in die Datenbank Inserten.

Lösung

Stack Overflow

8.4 GitHub [2]

Erklärung

Da die Insert Files sehr groß sind, konnten nicht alle auf die Datenbank hochgeladen werden.

Lösung

GitHub hat eine extra Funktion, die heißt git lfs.
Womit man Dateien mit der Maximalen Größe von 1.5 GB hochladen kann.

8.5 JavaFX [1] [6] [7]

Erklärung

Wenn man in JavaFx ein 2. Fenster oder mehrere Fenster öffnen will, erstellt JavaFx ein neues Objekt vom Kontroller. Das ist schlecht, denn es die Gespeicherten Sachen werden sozusagen gelöscht.

Lösung

Ein Tab Fenster.
(Es wäre eigentlich kein Problem mehrere Fenster zu erstellen oder in einem Fenster mittels Material Design Effekte zu erschaffen, aber meine Zeiteinteilung war leider sehr schlecht.)

9 Fußballverein

Die Eigentliche Aufgabe ist auf GitHub.

10 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

11 Literatur Verzeichnisse

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

12 Quellen

INDEX	Quelltitel + Information
[1]	Online: https://docs.oracle.com/javase/7/docs/api/java/io/BufferedWriter.html Kapitel: 5, 3.7, 8.2, 8.5 zuletzt abgerufen am [15.4.2016]
[2]	Online: https://git-scm.com/download/gui/linux Kapitel: 7, 8.4 zuletzt abgerufen am [15.4.2016]
[3]	Online: http://mobaxterm.mobatek.net/ Kapitel: zuletzt abgerufen am [15.4.2016]
[4]	Online: http://www.w3schools.com/sql/sql_dates.asp Kapitel: 3 zuletzt abgerufen am [15.4.2016]
[5]	Online: http://www.sqldocu.com/four/join.htm#nonEquiJoin Kapitel: 3.2 zuletzt abgerufen am [15.4.2016]
[6]	Online: https://docs.oracle.com/javase/7/docs/api/ Kapitel: 4, 5, 8.5 zuletzt abgerufen am [15.4.2016]
[7]	Online: https://docs.oracle.com/javafx/2/api/ Kapitel: 5, 8.5 zuletzt abgerufen am [15.4.2016]
[8]	Online: http://www.postgresql.org/docs/9.5/interactive/sql-syntax.html Kapitel: 3,4,8.3 zuletzt abgerufen am [15.4.2016]