Segelverein

Lukas Zainzinger

Dieses Protokoll beinhaltet die vollständige Dokumentation über das Projekt „Segelverein“.

Inhaltsverzeichnis

1. Aufgabenstellung

Beschreibung

Für Segler und Trainer sind Name (NAME) und Geburtsdatum (GEBURTSDATUM) bekannt. Sie werden beide identifiziert duch eine eindeutige Nummer (KEY). Mindestens zwei Segler, maximal jedoch vier Segler bilden eine Mannschaft. Für jede Mannschaft wird ein eindeutiger Name (NAME) und eine Altersklasse (AKLASSE) gespeichert. Jede Mannschaft wird genau von einem Trainer betreut. Ein Trainer kann jedoch mehrere Mannschaften betreuen.
Jeder Mannschaft sind Boote zugewiesen. Ein Boot kann mehreren Mannschaften zugewiesen sein. Ein Boot wird eindeutig durch eine Nummer (ID) identifiziert. Weiters sind zu jedem Boot ein Name (NAME), die Anzahl der Personen (PERSONEN) und der Tiefgang (TIEFGANG) bekannt. Es gibt Tourenboote und Sportboote. Tourenboote haben zusätzlich eine Bootsklasse (BOOTSKLASSE) und Sportboote haben zusätzlich eine Segelfläche (SEGELFLAECHE) gespeichert. Es ist außerdem bekannt welche Mannschaften mit welchen Sportbooten an welchen Regatten mit welcher Startnummer (STARTNR) teilgenommen haben.
Eine Regatta wird eindeutig identifiziert durch ihren Namen (NAME) und durch das Jahr (JAHR), in dem sie stattgefunden hat. Das Land (LAND) ist außerdem noch bekannt. Jede Regatta besteht aus mindestens drei jedoch maximal fünf Wettfahrten. Wettfahrten werden durch die zugehörige Regatta und das Datum (DATUM) identifiziert, außerdem wird die Länge (LAENGE) der Strecke gespeichert. Mannschaften können bei jeder Wettfahrent Punkte (PUNKTE) erzielen.

Java und JDBC

Schreiben Sie einen Java Client, der eine JDBC-Verbindung zur Datenbank herstellt und AUTOCOMMIT ausschaltet. Realisieren Sie eine GUI, die einfache CRUD-Befehle auf die Boote des Vereins implementiert (keine explizite SQL-Eingabe). Verwenden Sie dabei auf jeden Fall eine JTable, die auch eine grafische Veränderung der Datensätze erlauben soll.
Als Erweiterung (Bonuspunkte) soll bei der Anzeige der Boote die Möglichkeit der Sortierung und Filterung über ein neues SQL-Kommando bereitgestellt werden. Auch hier soll nicht der Benutzer die SQL-Befehle eingeben, sondern es muss die Funktionalität über entsprechende GUI-Elemente realisiert werden!
Ermöglichen Sie die gleichzeitige Verbindung von mehreren Clients auf die Datenbasis. Implementieren Sie dabei eine transaktionell, gesicherte Erstellung und Änderung von Wettfahrten. Beachten Sie dabei, dass der Punktestand der einzelnen Wettfahrten laufend und von mehreren Clients gleichzeitig aktualisiert werden könnte. Stellen Sie für die Eingabe der Wettfahrt und Mannschaft eine einfache grafische Möglichkeit zur Verfügung.

2. GIT

https://github.com/lzainzinger/Segelverein
Clone URL: https://github.com/lzainzinger/Segelverein.git

3. Aufwandsabschätzung und Aufteilung

3.1. CREATE/DROP Skript

Geschätzte Dauer: 2 h

Aus dem gegebenen RM und ERD ein Datenbank DDL schreiben, zur Erstellung und Dropen der Datenbank.

3.2. INSERT Generator

Geschätzte Dauer: 3 h

Erstellen der INSERTs für die Datenbank.
Für die Tabellen, welche nur PKs besitzen wird der Generator von „Generatedata.com“ verwendet. Für die restlichen Tabellen wird ein Generator mittels Java geschrieben. Er ließt die PKs der Vorhandenen Tabellen aus und erstellt so die INSERT Skripte.

3.3. INSERT Skript

Geschätzte Dauer: 0.25 h

Erstellen des INSERT Skripts mit Verwendung der einzelnen INSERT Skripts.

3.4. Java Programm

3.4.1. JDBC Verbindung

Geschätzte Dauer: 0.5 h

Erstellen der Verbindungsmethode mittels JDBC auf PostgreSQL.

3.4.2. GUI

Geschätzte Dauer: 2 h

Erstellen einer grafischen Oberfläche in Java, mit Berücksichtigung der in der Aufgabenstellung erstellten Vorgaben.

3.4.3. Funktion

Geschätzte Dauer: 5 h

Erstellen der benötigten Java-Methoden und Klassen um den in der Aufgabenstellung erstellten Funktionsumfang zu erfüllen.

3.5. Dokumentation

Geschätzte Dauer: 2 h

3.6. Gesamt

Gesamtdauer: 13,75 h

4. Zeitaufzeichnung

02.03.2015: 0.5 h INSERT Skripts für Tabellen die nur PKs haben.
08.03.2015: 0.25 h Erstellen des Protokolls + 0.5h Insert Generator
09.03.2015: 1 h JDBC Verbindung + 0.25 h Erstellen der Datenbank
10.03.2015: 1 h CREATE Skript
11.03.2015: 1 h CREATE Skript (Fehler bei FK! NOT UNIQUE)
12.03.2015: 1 h GUI Boote generiert + 0.5 h CREATE und DROP Skript (FERTIG)
13.03.2015: 1 h GUI Login + 1 h INSERT Methode
16.03.2015: 2 h DELETE und INSERT Methode
17.03.2015: 2 h UPDATE Methode + Testing
18.03.2015: 5 h ActionListener + Testing + Funktion bearbeiten + JTable
19.03.2015: 6 h GUI für Mannschaft/Wettfahrt + ActionListener + 1 h DataFiller (NICHT FERTIG)
22.03.2015: 1 h Protokoll + 2 h SQL-Abfragen

Gesamt: 26,75 h

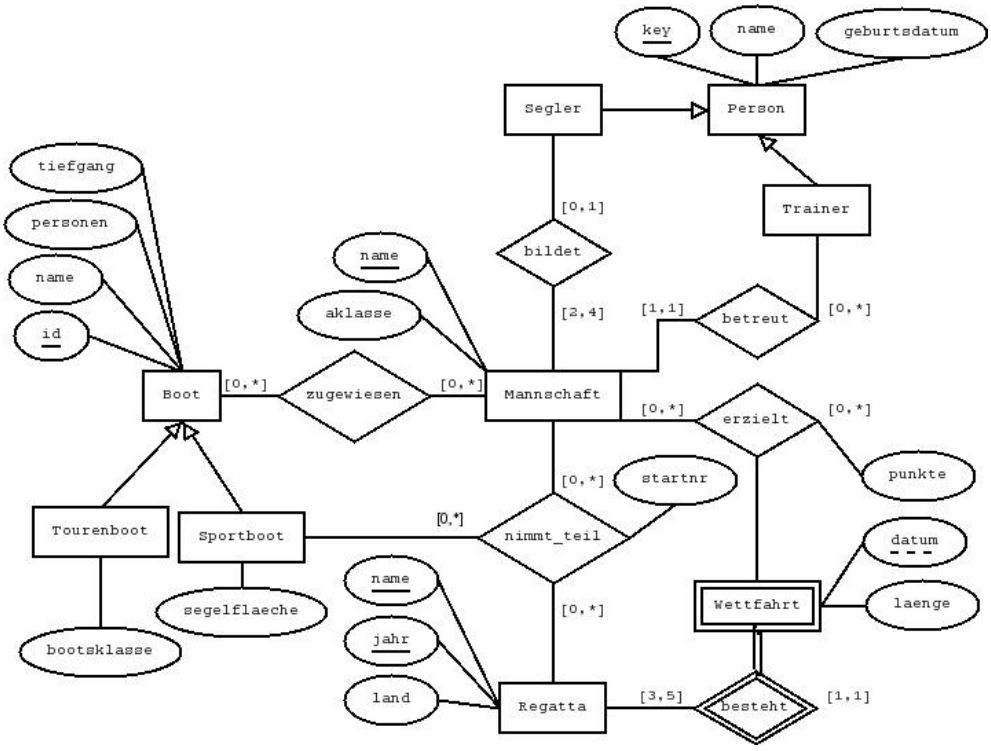
5. Durchführung

5.1. Datenbank und User erstellt

CREATE USER lukaszainzinger WITH PASSWORD ‚1234';
CREATE DATABASE segelverein OWNER lukaszainzinger;
GRANT ALL PRIVILEGES ON DATABASE segelverein TO lukaszainzinger;

5.2. CREATE.sql

Aus dem gegebenen RM und ERD das CREATE.sql Skript erstellt.



Besipiel:

```
CREATE TABLE wettfahrt ( -- Tabellenname
  wname varchar(255), -- Attribute
  wjahr varchar(4),
  datum date NOT NULL,
  laenge int NOT NULL,
  FOREIGN KEY (wname, wjahr) REFERENCES regatta(name, jahr), -- Angabe der FKs
  PRIMARY KEY (wname, wjahr, datum) – Angabe der PKs
);
```

5.3. DROP.sql

Erstellen des DROP.sql.

5.4. START.sql

Erstellen des START.sql.
Um ein File aufzurufen muss \i file.sql verwendet werden!

5.5. JDBC-Verbindung

Schreiben einer Klasse zur Verbindung auf einen PostgreSQL Server und Methoden zum EINTRAGEN, LÖSCHEN und UPDATEN von Datensätzen.

Code-example Verbindung:

```
/**
 * Konstruktor von Controller
 * @param server der Server der verwendet wird
 * @param benutzer der benutzer mit den amgemeldet wird
 * @param passwort das passwort des benutzer (leer wenn es keines gibt)
 * @param datenbank die datenbank die verwendet wird
 */
public JDBC_Controller_PSQL(String server,String benutzer,String passwort,String datenbank){
    this.server = server;
    this.benutzer = benutzer;
    this.passwort = passwort;
    this.datenbank = datenbank;
}
public void connect() throws SQLException {
    ds = new PGPoolingDataSource();
    ds.setServerName(server);
    ds.setUser(name);
    ds.setPassword(passwort);
    ds.setDatabaseName(datenbank);

    con = ds.getConnection();
    con.setAutoCommit(false);
    st = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
}
```

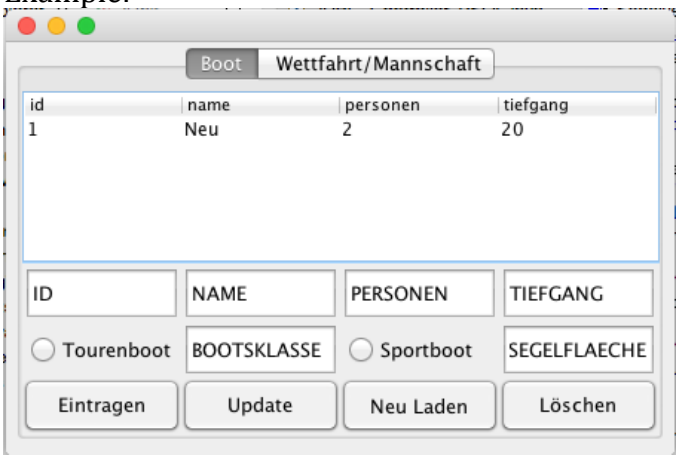
Code-example UPDATE:

```
/**
 * UPDATE Methode
 * @param tabname Name der Tabelle
 * @param param Die Parameter der Tabelle nach der Verglichen wird (WHERE param)
 * @param values Der benötigte Wert, (= values)
 * @param id Identification by
 * @param expr Expression to Identify
 * @throws SQLException
 */
public void update(String tabname, String param, String values, String id, String expr) throws SQLException{
    String sql = "UPDATE " + tabname + " SET " + param +" = " + values +" WHERE " + id + " = " + expr + " ;";
    System.out.println(sql);
    st.executeUpdate(sql);
    con.commit();
}
```

5.6. GUI

GUI der Applikation mit Hilfe von WindowBuilder.

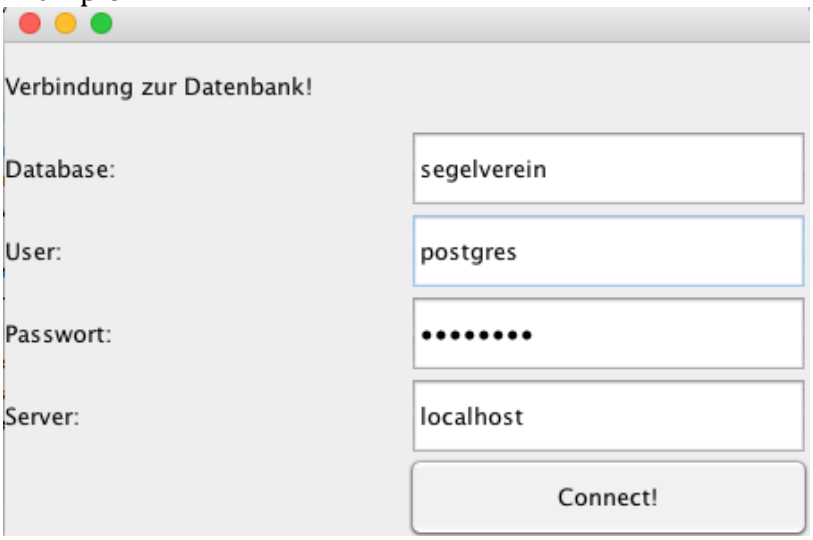
Example:



5.7. Login GUI

Login GUI für eine grafische Eintragung für den Verbindungsaufbau.

Example:



6. Quellen

PostgreSQL Manual:
http://www.postgresql.org/docs/9.4/static/