Prof. Dr. V. Linnemann
Stefan Werner
Universität zu Lübeck
Institut für Informationssysteme

Lübeck, den 28. Januar 2013

Nonstandard Datenbanken

Wintersemester 2012/2013

Probe-Klausur

Hinweise:

- Bitte versuchen Sie, die Aufgaben zum Test in der für die Klausur vorgegebenen Zeit (90 Minuten) zu lösen.
- Die Lösungen dieses Aufgabenblattes können bis Montag, den 4. Februar, 14 Uhr, abgegeben werden (ohne Wertung).

Aufgabe 1: Java Server Pages und JDBC

In dieser Aufgabe soll ein Login-Formular mittels JSP realisiert werden. Die gültigen Login-Daten sind in einer Datenbank mit der Tabelle *Accounts(username, password, realname)* hinterlegt.

a) Welche Vorteile haben JSP's gegenüber Java Servlets? Welches Konstrukt ist dabei besonders hilfreich?

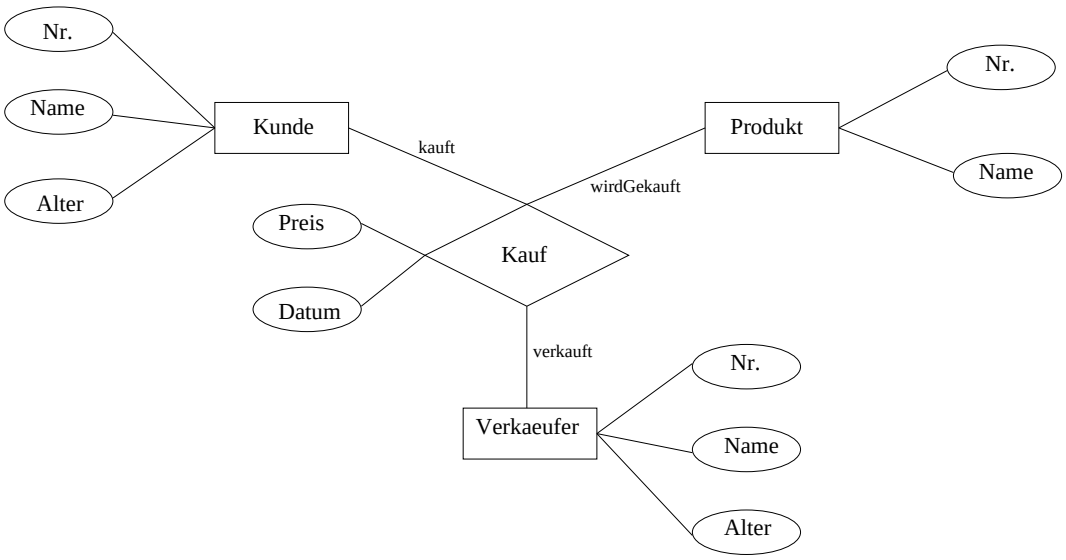
b) Vervollständigen Sie die folgende Java-Klasse, so dass diese erfolgreich mittels JDBC eine Anfrage an die Tabelle *Accounts* stellen kann. Im Falle einer gültigen Kombination von *username* und *password* soll der reale Name (*realname*) des Benutzers zurückgegeben werden, andernfalls `null`.

```
1  package myPackages;
2  import java.sql.*;
3  import java.util.*;
4
5  public class myJDBCcon {
6    public myJDBCcon(){}
7    public String checkLogin(String username, String password){
8      String result = null;
9
10     if (username == null || username.length() == 0 ||
11       password == null || password.length() == 0){
12         return null;
13     }
```

```
15      try {
16        Class.forName("org.hsqldb.jdbcDriver");
17        Connection con = DriverManager.getConnection(
18                    "jdbc:hsqldb:hsql://localhost", "sa", "");
19        Statement st = (1)__________________;
20        ResultSet rs = (2)______________________;
21        if ((3)________){
22          result = (4)___________;
23        }
24        con.close();
25      } catch (Exception e) {System.out.println(e);}
26      return result;
27    }
28  }
```

c) Implementieren Sie nun das noch fehlende Login-Formular mittels JSP unter Verwendung der zuvor erstellten Java-Klasse. Im Falle des erstmaligen Aufrufs der Seite oder der fehlerhaften Eingabe von Login-Daten, soll dem Benutzer ein Formular zur Eingabe von *username* und *password* präsentiert werden. Nach erfolgreicher Verifizierung der Eingabe, soll der Benutzer mit seinem realen Namen begrüßt werden.

Aufgabe 2: Relationale und objektorientierte Modellierung



a) Formulieren Sie zu dem ER-Modell in der Abbildung ein relationales Modell. Geben Sie dabei Primär- und Fremdschlüssel an. Benutzen Sie dabei die Schreibweise: *NameDerRelation(Attribut1, Attribut 2, Attribut3->NameEinerAnderenRelation)*, wobei unterstrichene Attribute den Primärschlüssel kennzeichnen. In diesem Fall ist Attribut 3 ein Fremdschlüssel.

b) Geben Sie in SQL die Namen der Produkte an, die gekauft wurden.

c) Geben Sie eine Anfrage an, die die Namen der Kunden eines Verkäufers 'Meier' ausgibt.

d) Geben Sie die Namen der Produkte an, die ein Verkäufer namens 'Meier' einem Kunden namens 'Schulze' verkauft hat.

e) Modellieren Sie den dargestellten Sachverhalt objektorientiert mit Hilfe von Java-Klassen. Dabei soll es zusätzlich eine Klasse '*Person*' geben, von der *Verkäufer* und *Kunde* erben. Die gemeinsamen Attribute sollen nur in der Klasse *Person* vorkommen. Dabei sollen auftretende Referenzen zwischen den Klassen jeweils in beide Richtungen realisiert werden.

f) Formulieren Sie die 3 Anfragen in b), c) und d) jeweils in der OQL.

g) Schätzen Sie ganz grob die Komplexität der Anfrage aus c) für die SQL und die OQL-Version ab.

Aufgabe 3: Oracle

a) Definieren Sie einen eigenen Typ für Fahrzeuge mit den Attributen *Id, Modellbezeichnung, Leistung* und legen Sie eine entsprechende Tabelle an!

b) Ändern Sie die erzeugte Tabelle derart, dass das Attribut Id ein Primärschlüssel ist!

c) Fügen Sie eine Member Function `ist_langsam` hinzu, die 1 (=true) liefert, falls die Leistung eines Fahrzeugs geringer als 50 PS ist!

Aufgabe 4: XML und XPath

Die folgende DTD sowie das folgende XML-Dokument liegen dieser Aufgabe zugrunde:

DTD:

```
<!ELEMENT newspaper (article+)>
<!ELEMENT article (headline, body, reference*)>
<!ELEMENT headline (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!ELEMENT reference (#PCDATA)>
<!ATTLIST article author CDATA #REQUIRED>
<!ATTLIST article id CDATA #REQUIRED>
<!ATTLIST article date CDATA #REQUIRED>
```

Dokument:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE newspaper SYSTEM "newspaper.dtd">
<newspaper>
 <article author="Autor 1" id="01" date="damals">
  <headline>Überschrift 1</headline>
  <body>Hier steht der Text 1.. </body>
  <reference>03</reference>
  <reference>01</reference>
 </article>
 <article author="Autor 2" id="02" date="früher">
  <headline>Überschrift 2</headline>
  <body>Hier steht der Text 2.. </body>
 </article>
 <article author="Autor 3" id="03" date="damals">
  <headline>Überschrift 3</headline>
  <body>Hier steht der Text 3.. </body>
 </article>
</newspaper>
```

a) Stellen Sie folgende Anfragen in XPath:

   (i) Geben Sie die Namen aller Autoren aus!

   (ii) Welche Überschrift hat der Artikel mit der ID „01“?

   (iii) Wieviele Referenzen sind insgesamt in allen Artikeln vorhanden?

   (iv) Wie lautet das Datum des 3. Artikels?

b) Entwerfen Sie ein relationales Schema für die gegebene DTD.

c) Wandeln Sie das gegebene Beispieldokument in ein generisches relationales Schema um und geben Sie die Tabelle an.

d) Stellen Sie die XPath Anfragen in SQL. Die Anfragen sollen sich auf die Tabelle t aus der vorherigen Teilaufgabe beziehen!

   (i) Geben Sie die Namen aller Autoren aus!

   (ii) Welche Überschrift hat der Artikel mit der ID „01“?

   (iii) Wieviele Referenzen sind insgesamt in allen Artikeln vorhanden?

   (iv) Wie lautet das Datum des 3. Artikels?