Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2014

46116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologie/Datenbanksysteme**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **16**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

Softwaretechnologie

1. Modellierung

Gegeben ist die folgende Situation: Ihre Firma hat eine Ausschreibung gewonnen, welche die Implementierung eines neuen IT-Systems zur Vereinfachung der Ausführung von Geschäftsprozessen umfasst, einschließlich der Aufnahme von Kundeninformationen im Außendienst. Es wurden bereits Anforderungen in Form von Anwendungsfällen aufgenommen.

Sie betrachten nun den folgenden Anwendungsfall im Detail:

Anwendungsfall: *Kundendaten Online erfassen*	
Kurzbeschreibung	Die Stammdaten des Kunden werden Online im Web-Frontend erfasst.
Akteur(e)	Kunde, System
Vorbedingung	Kunde ist noch nicht im System erfasst.
Nachbedingung	Kunde ist im System erfasst.

Normalverlauf

1. Der Kunde klickt auf der Webseite auf den Link „Als neuer Kunde anmelden"
2. Das System lädt die erste Webseite für die Anmeldung.
3. Der Kunde gibt seine persönlichen Daten (Name, Vorname, Titel, Anrede, *(optional) Geburtsname*, Geschlecht, Geburtsdatum, Geburtsort, Wohnanschrift, Telefonnummer, *(optional) Faxnummer*, Email-Adresse) ein. Im Anschluss klickt er auf „Weiter".
4. Das System lädt die zweite Webseite für die Anmeldung.
5. Der Kunde wählt ein Zahlungsverfahren aus und gibt danach die Zahlungs- bzw. Rechnungsdaten ein. Im Anschluss klickt er auf „Weiter".
6. Das System prüft die Zahlungs- bzw. Rechnungsdaten.
7. Sind die Daten gültig, lädt das System die dritte Webseite der Anmeldung
8. Der Kunde gibt einen Benutzernamen und ein Passwort ein. Er klickt im Anschluss auf „Anmeldung abschließen", womit der Anmeldeprozess abgeschlossen wird.

Alternativverlauf

Beginn wie Normalverlauf 1-6

7a. Sind die Daten ungültig, gibt das System eine Aufforderung zur Korrektur.
7b. Der Kunde ändert seine angegebenen Zahlungs- bzw. Rechnungsdaten und klickt anschließend auf „Korrektur übernehmen".
7c. Das System prüft die Zahlungs- bzw. Rechnungsdaten und übernimmt die Korrektur, sofern die Daten gültig sind. Andernfalls wird die Anwendung beendet.

Ende gemäß Normalverlauf 8

Aufgabe

Erstellen Sie ein UML-Sequenzdiagramm, das Normal- und Alternativverlauf für den oben beschriebenen Anwendungsfall darstellt!

Fortsetzung nächste Seite!

2. Architekturbeschreibung

Ein einfaches Studentenverwaltungssystem soll die Prüfungsbeteiligung (mit Status und Note) von Studenten (mit Name, Vorname, Matrikelnummer) an angebotenen Prüfungen (mit Bezeichnung, Ort, Zeit) verwalten. Dabei sollen folgende typische Aufgaben unterstützt werden:

(I) Anmeldung und Abmeldung eines Studenten zu einer bestimmten Prüfung,

(II) Abfrage des Status einer Prüfungsbeteiligung (angemeldet, abgemeldet, teilgenommen, bestanden, durchgefallen) zu einer bestimmten Prüfung durch einen Studenten,

(III) Abfrage der Note eines Studenten zu einer Prüfung,

(IV) Eintragung der Teilnahme eines Studenten an einer Prüfung durch die Übungsleitung,

(V) Eintragung der Note eines Studenten an einer Prüfung durch die Übungsleitung.

Nicht zu berücksichtigen sind dabei Authentifizierung und Autorisierung.

Aufgabe

Beschreiben Sie folgende Bestandteile einer Architektur des Systems:

a) Datenmodell (als E/R- oder Klassendiagramm)!

b) Spezifikation der Aufgaben der Komponente „Prüfungsbeteiligung“ nach „design by contract“ (als Vor- und Nachbedingungen für logische Prädikate)!

c) Integritätsbedingung zwischen Status und Note von Prüfungsbeteiligungen (als logische Zusicherungen)!

Fortsetzung nächste Seite!

3. Dynamische Software-Testverfahren

Im Folgenden ist ein Algorithmus gegeben, der für eine positive Zahl x die Summe aller Zahlen bildet, die kleiner als x und Vielfache von drei oder fünf sind. Für negative Zahlen soll 0 zurückgegeben werden.
Der Algorithmus soll also folgender Spezifikation genügen:

$$x > 0 \Rightarrow specialSums(x) = \sum\{y \mid 0 < y < x \wedge (y\%3 = 0 \vee y\%5 = 0)\}$$

$$x \leq 0 \Rightarrow specialSums(x) = 0$$

wobei % den Modulo-Operator bezeichnet.

```
1   public static long specialSums(int until)
2   {
3       long sum = 0;
4       if(until > 0)
5       {
6           for (int i = 1; i <= until; i++)
7           {
8               if (i % 3 ==0 || i % 5 == 0) {
9                sum+= i;
10              }
11          }
12      }
13      return sum;
14  }
```

Abbildung 1: Algorithmus specialSums

Fortsetzung nächste Seite!

Teilaufgabe 2:

Datenbanksysteme

Aufgabe 1: Normalformen

Gegeben sei eine Relation R mit den acht Attributen A, B, C, D, E, F, G, H. Es gelten folgende funktionale Abhängigkeiten, die alle *volle* funktionale Abhängigkeiten sein sollen:

$A \rightarrow B$
$A, C, E \rightarrow D$
$C, E \rightarrow F$
$C, E \rightarrow H$
$D \rightarrow G$

(a) Gibt es noch weitere (nicht-triviale) volle funktionale Abhängigkeiten, die sich aus den genannten ergeben? Wenn ja, welche?

(b) Geben Sie einen Schlüsselkandidaten für die Relation R an! (Mit Begründung, warum Ihre Wahl ein Schlüsselkandidat ist.)

(c) Statt der Relation R soll nun eine Menge von Relationen in dritter Normalform verwendet werden, die dieselben Informationen halten können. Geben Sie geeignete Relationen in dritter Normalform an! Für jede Relation ist ein Name sowie die Menge der Attribute (mit unterstrichenen Schlüsselattributen) anzugeben. (Eine systematische Entwicklung gemäß eines Synthesealgorithmus ist nicht notwendig.)

Fortsetzung nächste Seite!

Aufgabe 2: ER-Modellierung

Ein Schauspielhaus möchte eine relationale Datenbank für folgende Informationen anlegen:

In jedem Theaterstück werden Schauspieler mit einer bestimmten Rolle eingesetzt. Schauspieler haben eine Angestelltennummer, einen Namen und ein monatliches Gehalt. (Wir gehen davon aus, dass Schauspieler fest angestellt sind.) Sie können für verschiedene Theaterstücke eingesetzt sein. Theaterstücke haben einen eindeutigen Titel und eine bestimmte Aufführungsdauer. Sie sind von einem Autor geschrieben und werden von einem Regisseur inszeniert. Sowohl Autoren als auch Regisseure sollen durch Ihren Namen eindeutig identifizierbar sein und verschiedene Theaterstücke geschrieben bzw. inszeniert haben können. Regisseure sind unter einer bestimmten Telefonnummer zu erreichen. Für jeden Autor soll sein Geburtsland und sein Geburtsdatum bekannt sein.

(a) Erstellen Sie ein Entity-Relationship-Diagramm, das den oben beschriebenen Sachverhalt modelliert. Unterstreichen Sie die Schlüsselattribute!

(b) Geben Sie alle Kardinalitäten für die Relationships an! Die Kardinalitäten sollen entweder in der Notation von Chen (1:1, 1:m, m:1 oder m:n) oder in der UML-Notation angegeben werden.

(c) Überführen Sie Ihr ER-Diagramm in eine Menge von Relationen. Für jede Relation sollen deren Name und deren Attribute angegeben werden. Schlüsselattribute *und* Fremdschlüssel sollen gekennzeichnet werden (durch unter- bzw. überstreichen).

Fortsetzung nächste Seite!

Aufgabe 3: SQL

Ein Wettbüro für Pferderennen benützt eine relationale Datenbank mit den drei Relationen *Rennen, NimmtTeil* und *Siegwette*. Schlüsselattribute sind unterstrichen.

In der Relation *Rennen* wird für jedes Rennen die Rennnummer, die Anzahl der Runden, über die das Rennen geht, und die Prämie für den Sieger des Rennens gespeichert.

Relation *Rennen*:

Renn-Nr	Runden	Siegprämie
1	2	20000
2	4	35000
3	3	25000

In der Relation *NimmtTeil* wird gespeichert, welche Pferde mit welchen Jockeys an welchen Rennen teilnehmen und wie die Gewinnquote bei einem Sieg des jeweiligen Pferdes ist. Es wird davon ausgegangen, dass jedes Pferd nur bei einem Rennen teilnimmt. Allerdings kann derselbe Jockey verschiedene Pferde reiten.

Relation *NimmtTeil*:

Renn-Nr	Pferd	Gewinnquote	Jockey
1	Fortuna	1.5	Karsten Jung
1	My-Love	3.0	Chris Berger
1	Maximal	4.0	Tom Hauser
2	Fedor	1.5	Andy Leis
2	Felicita	2.5	Vera Raaf
3	Victoria	3.0	Gerd Steger
3	Pan	1.5	Chris Berger

In der Relation *Siegwette* werden die abgeschlossenen Wetten gespeichert. In der Spalte *Kunde* wird der Name der wettenden Person angegeben, wobei jeder Kunde pro Rennen höchstens ein Pferd auf Sieg wetten kann. Falls das gewettete Pferd gewinnt, dann berechnet sich der Wettgewinn des Kunden durch das Produkt seines Einsatzes mit der Gewinnquote des getippten Pferdes.

Relation *Siegwette*:

Kunde	Renn-Nr	Pferd	Einsatz
Ulf Jobst	1	Fortuna	100
Gerda Hase	1	My-Love	200
Kai Haper	1	Fortuna	180
Kai Haper	2	Felicita	100
Jan Huber	2	Fedor	170
Gerda Hase	2	Fedor	250
Adam Riese	3	Victoria	200
Ulf Jobst	3	Pan	400

Fortsetzung nächste Seite!

(a) Welche Ergebnistabellen liefern die beiden folgenden SQL-Anfragen bei dem oben angegebenen Datenbestand?

(i) SELECT Renn-Nr FROM NimmtTeil WHERE Jockey = 'Tom Hauser'

(ii) SELECT MAX(Siegprämie)
FROM Rennen
WHERE Renn-Nr IN
(SELECT Renn-Nr FROM NimmtTeil WHERE Jockey = 'Tom Hauser')

(b) Der Datenbankinhalt werde nun durch folgende SQL-Anweisung verändert:

INSERT INTO NimmtTeil
VALUES (2, 'Girlanda', 2.5, 'Tom Hauser')

Welche Ergebnistabellen liefern jetzt die beiden Anfragen (i) und (ii) aus Teil (a)?

Formulieren Sie folgende Anfragen in SQL:

(c) Finde für den Jockey 'Chris Berger' alle Pferde, die von ihm geritten werden, und deren Gewinnquoten!

(d) Finde alle Pferde, die an einem Rennen teilnehmen, das über eine Distanz von mindstens drei Runden geht!

(e) Finde alle Kunden, die in einer einzelnen Siegwette mehr als 400 Euro gewinnen können!

(f) Finde alle Jockeys, die an mehr als einem Rennen teilnehmen!

(g) Finde alle Jockeys, die insgesamt mindestens 50000 Euro Siegprämie gewinnen können!

Thema Nr. 2

Teilaufgabe 1:

Softwaretechnologie

Aufgabe 1: „Formale Verifikation"

Gegeben sei folgende Methode zur Berechnung der Anzahl der notwendigen Züge beim Spiel „Die Türme von Hanoi":

```
int hanoi(int nr, char from, char to) {
        char free = (char) ('A' + 'B' + 'C' - from - to);
        if (nr > 0) {
                int moves = 1;
                moves += hanoi(nr - 1, from, free);
                System.out.println("Move piece nr. " + nr + " from " + from + " to " + to);
                moves += hanoi(nr - 1, free, to);
                return moves;
        } else {
                return 0;
        }
}
```

a) Beweisen Sie formal mittels vollständiger Induktion, dass zum Umlegen von k Scheiben (z.B. vom Turm A zum Turm C) insgesamt $2^k - 1$ Schritte notwendig sind, also dass für $k \geq 0$ folgender Zusammenhang gilt:

$$hanoi(k,'A','C') = 2^k - 1$$

b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie kurz Ihre Wahl!

Fortsetzung nächste Seite!

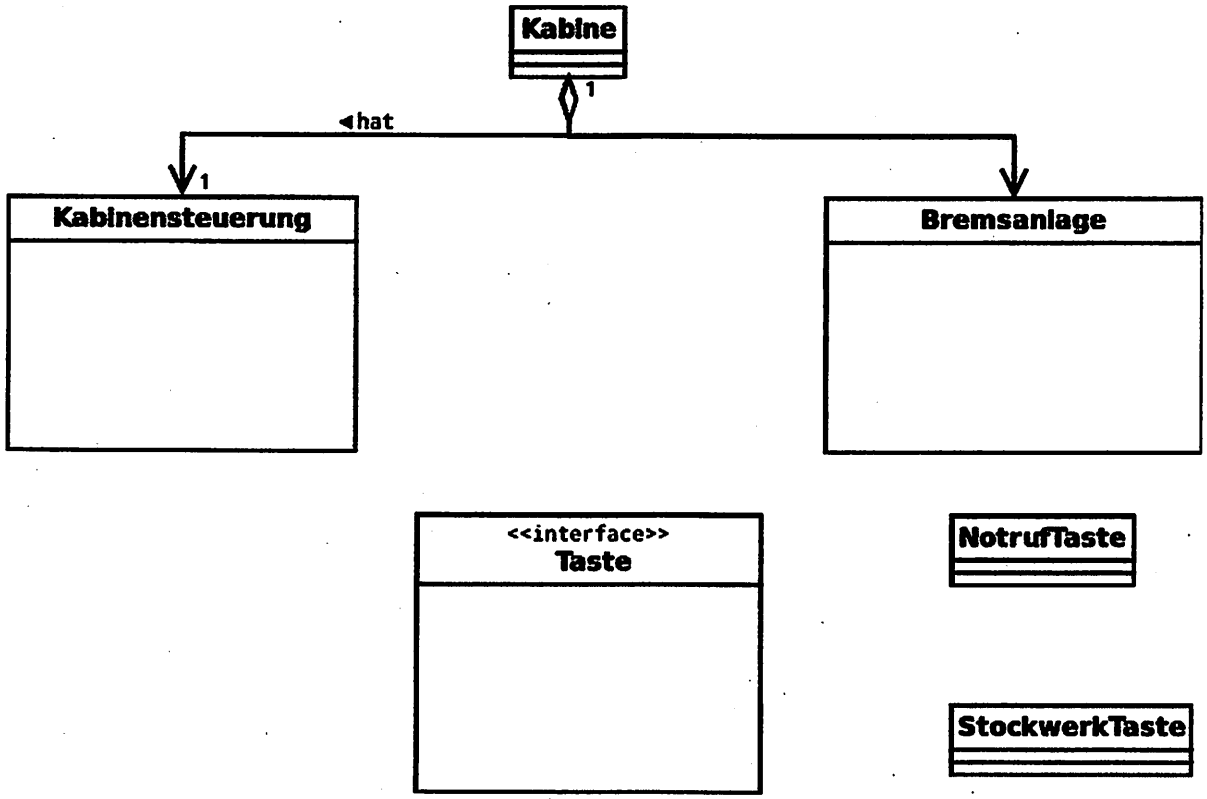
Aufgabe 2: „UML"

Gegeben sind die im Folgenden aufgeführten Auszüge aus der Spezifikation einer Aufzuganlage und dem zugehörigen Java-Source-Code:

... Eine Kabine hat genau eine Kabinensteuerung und verfügt über genau eine Bremsanlage. Eine Kabinensteuerung beobachtet die vorhandene Bremsanlage. Eine Kabinensteuerung verwaltet mindestens zwei Tasten, die sie als Array-Parameter im Konstruktor übergeben bekommt. „NotrufTaste" und „StockwerkTaste" sind Spezialisierungen einer Taste. Eine Bremsanlage hat einen sichtbaren Zustand (`true`/`false` für ausgelöst/nicht ausgelöst). ...

```
// ...
public class Kabinensteuerung {
    private Bremsanlage bremsanlage;
    protected Taste[] tasten;
    // ...
    public boolean istBereit() {
      boolean result = true;
      // ...
      result &= bremsanlage.zustand;
      // ...
      return result;
    }
    // ...
}
// ...
```

Übernehmen und ergänzen Sie das unvollständige UML-Klassendiagramm mittels aller für Sie aus der Spezifikation und dem Source-Code ersichtlichen Informationen! Achten Sie bei Assoziationen insbesondere auf Assoziationsnamen, deren Leserichtung, Multiplizitäten, Navigationspfeile und Rollennamen (inkl. Sichtbarkeiten)!

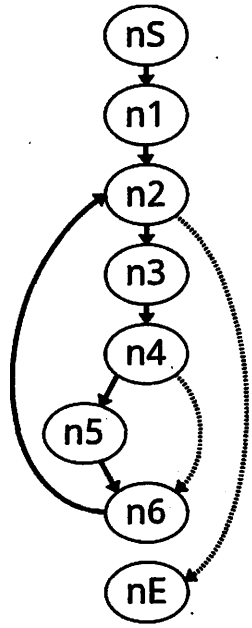


Fortsetzung nächste Seite!

Aufgabe 3: „Testen"

Gegeben sei folgende Methode **`searchFirstZero`** und ihr Kontrollflussgraph:

```
int[] searchFirstZero(int[][] m) {
      int rs = m.length, cs = m[0].length;
      int r = 0, c = 0;
      boolean d = false;
      for (int p = 0; p < rs * cs && !d; p++) {
            c = p % cs;
            r = p / cs;
            if (m[r][c] == 0) {
                  d = true;
            }
      }
      return new int[] { r, c };
}
```



a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen

   i) Verzweigungsüberdeckung (Branch-Coverage, C_1)

   ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

   mit **minimaler** Testfallanzahl genügen würden.

b) Welche der vorangehend ermittelten Pfade für die $C_{\infty,2}$-Überdeckung sind mittels Testfällen tatsächlich überdeckbar („feasible")? Falls der Pfad ausführbar ist, geben Sie bitte den zugehörigen logischen Testfall (also die Eigenschaften der Matrix `m` beim Aufruf der Methode) an – andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe!

d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort!

e) Geben Sie zu jedem Knoten die jeweilige Datenflussannotation (defs bzw. uses, sofern überhaupt vorhanden) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an!

Fortsetzung nächste Seite!

Teilaufgabe 2:

Datenbanksysteme

Aufgabe 1: E/R-Modellierung, SQL-DDL

Zur Verwaltung von Schulen soll folgendes System entworfen werden:

- Eine Schule hat einen eindeutigen Namen und eine Adresse. Eine Schule hat mindestens eine Schulklasse.
- Eine Klasse gehört zu genau einer Schule und kann eindeutig durch den Namen der Schule, ihre Stufe und einen zusätzlichen Buchstaben beschrieben werden. Eine Schulklasse besteht aus mehreren Schülern.
- Ein Schüler wird eindeutig durch seinen Namen und Vornamen beschrieben und besucht genau eine Schulklasse. Ein Schüler besitzt zusätzlich ein Geburtsdatum und eine Adresse.
- Ein Lehrer wird eindeutig durch eine ID bestimmt und hat einen Namen und einen Vornamen sowie eine Adresse. Außerdem kann ein Lehrer die Leitung einer Klasse übernehmen, wobei es pro Klasse genau einen Klassenleiter gibt. Als Rektor kann ein Lehrer zusätzlich die Schule leiten, in der er arbeitet. Jede Schule hat genau einen Rektor.
- Die Adressen von Schulen, Schülern und Lehrern sollen als eigene Entität repräsentiert werden und enthalten Informationen über Straße, Hausnummer, Postleitzahl und Ort.
- Lehrer unterrichten in mehreren Klassen verschiedene Unterrichtsfächer mit einer bestimmten Stundenzahl. Ein Unterrichtsfach wird durch seinen Namen eindeutig gekennzeichnet.

1. Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank!

2. Setzen Sie das gegebene E/R-Diagramm in ein entsprechendes relationales Datenbankschema um! Geben Sie die resultierenden Relationenschemata in folgender Schreibweise an:

   ```
   Relation (Attribut1, Attribut2, ..., AttributN)
   ```

   Identifizieren Sie für jede Relation einen Primärschlüssel und unterstreichen Sie diesen. Achten Sie auf eine geeignete Modellierung der Beziehungen (Relationships).

3. Geben Sie die Anweisungen in SQL-DDL an, die notwendig sind, um die Relationen aus Teilaufgabe (2) in einer relationalen Datenbank zu erzeugen! Kennzeichnen Sie dabei die Primär- und Fremdschlüssel der Relationen!

Fortsetzung nächste Seite!

Aufgabe 2: Relationale Algebra

Gegeben sei das folgende relationale Schema mitsamt Beispieldaten für eine Datenbank von Mitfahrgelegenheiten. Die Primärschlüssel-Attribute sind jeweils unterstrichen, Fremdschlüssel sind überstrichen.

„Kunde“:

KID	Name	Vorname	Stadt
K1	Meier	Stefan	S3
K2	Müller	Petra	S3
K3	Schmidt	Christine	S2
K4	Schulz	Michael	S4

„Stadt“:

SID	SName	Bundesland
S1	Berlin	Berlin
S2	Nürnberg	Bayern
S3	Köln	Nordrhein-Westfalen
S4	Stuttgart	Baden-Württemberg
S5	München	Bayern

„Angebot“:

KID	Start	Ziel	Datum	Plätze
K4	S4	S5	08.07.2011	3
K4	S5	S4	10.07.2011	3
K1	S1	S5	08.07.2011	3
K3	S2	S3	15.07.2011	1
K4	S4	S1	15.07.2011	3
K1	S5	S4	09.07.2011	2

„Anfrage“:

KID	Start	Ziel	Datum
K2	S4	S5	08.07.2011
K2	S5	S4	10.07.2011
K3	S2	S3	08.07.2011
K3	S3	S2	10.07.2011
K2	S4	S5	05.07.2011
K2	S5	S4	17.07.2011

Formulieren Sie die folgenden Anfragen auf das gegebene Schema in relationaler Algebra:

1. Finden Sie die Namen aller Städte in Bayern!
2. Finden Sie die SIDs aller Städte, für die weder als Start noch als Ziel eine Anfrage vorliegt!
3. Finden Sie alle IDs von Kunden, welche eine Fahrt in ihrer Heimatstadt starten!
4. Geben Sie das Datum aller angebotenen Fahrten von München nach Stuttgart aus!

Geben Sie das Ergebnis (bezüglich der Beispieldaten) der folgenden Ausdrücke der relationalen Algebra als Tabellen an:

5. $\pi_{KID}(Angebot) \bowtie Kunde$
6. $\pi_{KID,Stadt}(Kunde) \underset{Kunde.Stadt=Angebot.Ziel}{\bowtie} \pi_{\text{Plätze}}(Angebot)$

Fortsetzung nächste Seite!

Aufgabe 3: SQL

Gegeben sei das relationale Datenbank-Schema aus Aufgabe 2.

„Kunde“:

KID	Name	Vorname	Stadt
K1	Meier	Stefan	S3
K2	Müller	Petra	S3
K3	Schmidt	Christine	S2
K4	Schulz	Michael	S4

„Stadt“:

SID	SName	Bundesland
S1	Berlin	Berlin
S2	Nürnberg	Bayern
S3	Köln	Nordrhein-Westfalen
S4	Stuttgart	Baden-Württemberg
S5	München	Bayern

„Angebot“:

KID	Start	Ziel	Datum	Plätze
K4	S4	S5	08.07.2011	3
K4	S5	S4	10.07.2011	3
K1	S1	S5	08.07.2011	3
K3	S2	S3	15.07.2011	1
K4	S4	S1	15.07.2011	3
K1	S5	S4	09.07.2011	2

„Anfrage“:

KID	Start	Ziel	Datum
K2	S4	S5	08.07.2011
K2	S5	S4	10.07.2011
K3	S2	S3	08.07.2011
K3	S3	S2	10.07.2011
K2	S4	S5	05.07.2011
K2	S5	S4	17.07.2011

Formulieren Sie die folgenden Anfragen in SQL:

1. Geben Sie alle Attribute aller Anfragen aus, für die passende Angebote existieren! Ein Angebot ist passend zu einer Anfrage, wenn Start, Ziel und Datum identisch sind.

2. Finden Sie Nachnamen und Vornamen aller Kunden, für die kein Angebot existiert!

3. Geben Sie das Datum aller angebotenen Fahrten von München nach Stuttgart aus und sortieren Sie das Ergebnis aufsteigend!

4. Geben Sie für jeden Startort einer Anfrage den Namen der Stadt und die Anzahl der Anfragen aus.

Wie sieht die Ergebnisrelation zu folgenden Anfragen auf den Beispieldaten aus?

5. ```
   select * from Stadt where not exists (select * from Anfrage
   where Start=SID or Ziel=SID);
   ```

6. ```
   select KID, sum(Plätze) from Angebot where Plätze > 2 group by
   KID having sum(Plätze) > 4;
   ```

Fortsetzung nächste Seite!

Aufgabe 4: Entwurfstheorie

Gegeben sei die nachfolgende relationale Datenbank mit unterstrichenen Schlüsselattributen. Sie enthält die Daten einer Universität in erster Normalform. Relation „Universität“:

Veranstaltung	Semester	Dozent	Lehrstuhl	Matrikelnummer	Note	SWS
Datenbanksysteme 2	SS11	Kröger	DBS	12345	1,7	3
Formale Sprachen	SS10	Hofmann	TCS	73987	1,0	3
Mobilkommunikation	WS10	Linnhoff-Popien	MNM	46421	2,3	2
Datenbanksysteme 1	WS10	Schubert	DBS	12345	3,0	3
Datenbanksysteme 2	SS11	Kröger	DBS	25432	5,0	3
Programmieren im Grid	SS11	Kranzlmüller	MNM	25432	3,0	4
Programmieren im Grid	SS10	Kranzlmüller	MNM	45621	1,0	4
Datenbanksysteme 2	SS10	Schubert	DBS	73987	2,0	3

Sowie die folgenden funktionalen Abhängigkeiten:

FD1: Veranstaltung, Semester, Matrikelnummer → Dozent, Lehrstuhl, Note, SWS
FD2: Veranstaltung, Semester → Dozent
FD3: Dozent → Lehrstuhl
FD4: Veranstaltung → SWS

1. Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können! Geben Sie ein Beispiel für jede Anomalie an!

2. Überführen Sie das obige Relationenschema zunächst in die zweite und danach in die dritte Normalform! Geben Sie die mit obigen Daten gefüllten Relationen in dritter Normalform an!

3. Erläutern Sie kurz, welchen Nachteil Normalisierung allgemein für die Anfragebearbeitung haben kann!