Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2016

66116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Datenbanksysteme, Softwaretechnologie**

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

Anzahl der Druckseiten dieser Vorlage: **19**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Konzeptioneller Entwurf

Im Folgenden ist die Beschreibung einer Fotoverwaltung gegeben. Erstellen Sie zu dieser ein ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Min-Max-Notation an. Modellieren Sie keine Attribute oder Entitytypen, die nicht aus dem Text hervorgehen.
Es werden Fotos verwaltet, die über eine eindeutige ID identifiziert werden und einen Aufnahmezeitpunkt sowie eine Beschreibung besitzen. Den Fotos können Stichworte zugeordnet werden, die einen eindeutigen Namen haben. Stichworte sind hierarchisch organisiert, d.h. ein Stichwort kann maximal einem anderen Stichwort untergeordnet sein. Jedes Foto wurde von genau einem Fotografen aufgenommen und es können Models auf ihm abgebildet sein. Zu Fotografen und Models werden Name, Vorname, Postleitzahl und Wohnort gespeichert. Zu Fotografen wird zusätzlich ihre Mitgliedsnummer im Verband deutscher Fotografen abgelegt, durch die sie eindeutig identifiziert werden können. Models gehören genau einer Agentur an und haben eine Mitgliedsnummer, die innerhalb ihrer Agentur eindeutig ist. Agenturen haben einen eindeutigen Namen sowie eine E-Mail-Adresse. Ein Fotograf kann auch gleichzeitig Model sein. Models können der Veröffentlichung von bestimmten Fotos in bestimmten Medien zustimmen. Zu jedem Medium werden ein eindeutiger Name sowie eine Beschreibung abgelegt. Zu einer Zustimmung wird das Datum gespeichert, an dem sie erfolgt ist.

Fortsetzung nächste Seite!

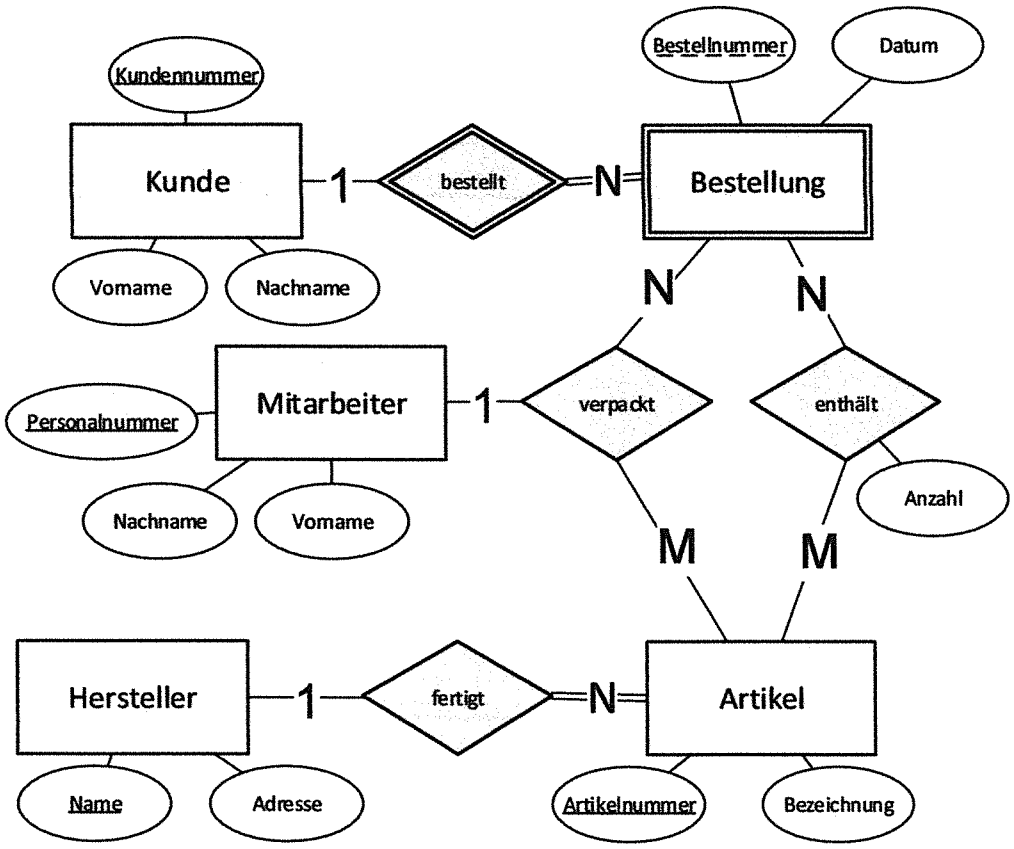
2. Relationenschema

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema in dritter Normalform (3 NF) mit möglichst wenigen Relationen.

Verwenden Sie dabei folgende Notation:

Primärschlüssel werden durch Unterstreichen gekennzeichnet, Fremdschlüssel durch die Nennung der Relation, auf die sie verweisen, in eckigen Klammern hinter dem Fremdschlüsselattribut. Attribute zusammengesetzter Fremdschlüssel werden durch runde Klammern als zusammengehörig markiert.

Beispiel: Relation1 (Primärschlüssel, Attribut1, Attribut2, Fremdschlüssel[Relation2])



Fortsetzung nächste Seite!

3. Normalformen

a) Gegeben ist eine vollständige Extension der Relation Bestellungen mit dem zusammengesetzten Primärschlüssel {Kundennummer, Bestellnummer}:

Kundennummer	**Bestellnummer**	Kundennachname	Bestelldatum	Lieferdatum
1	1657	Frank	13.02.2015	15.02.2015
2	1306	Weizenbaum	14.01.2015	18.01.2015
3	2369	Weizenbaum	09.08.2014	10.01.2015
2	9847	Weizenbaum	21.11.2013	25.11.2013
5	2569	Schmitt	05.09.2015	05.09.2015
7	3457	Nürnberger	07.10.2012	12.12.2012
8	4468	Haberkorn	31.08.2015	01.10.2015

Geben Sie an, in welcher Normalform sich die Relation Bestellungen befindet. Begründen Sie weiterhin in zwei bis drei Sätzen, dass alle Bedingungen dieser Normalform erfüllt sind und dass nicht alle Bedingungen der nächsthöheren Normalform erfüllt sind.

b) Nennen Sie die Bezeichnungen der drei Anomalien, zu denen nicht normalisierte Relationenschemata führen können, und erläutern Sie die Anomalien in jeweils ein bis zwei Sätzen. Sie können dazu Beispiele aus der in der vorhergehenden Teilaufgabe gegebenen Relation nutzen.

Fortsetzung nächste Seite!

4. SQL

Gegeben sind folgende Relationen aus einer Personalverwaltung:

```
Mitarbeiter (MitarbeiterID, Vorname, Nachname, Vorgesetzter[Mitarbeiter],
  AbteilungsID[Abteilung], Telefonnummer, Gehalt)
Abteilung (AbteilungsID, Bezeichnung)
```

a) Schreiben Sie eine SQL-Anfrage, die Vor- und Nachnamen der Mitarbeiter aller Abteilungen mit Bezeichnung „Buchhaltung" ausgibt, absteigend sortiert nach MitarbeiterID.

b) Schreiben Sie eine SQL-Anfrage, die die Nachnamen aller Mitarbeiter zusammen mit dem Nachnamen ihres jeweiligen direkten Vorgesetzten ausgibt. Mitarbeiter ohne Vorgesetzten sollen in der Ausgabe ebenfalls enthalten sein. In diesem Fall soll der Nachname des Vorgesetzten NULL sein.

c) Schreiben Sie eine SQL-Anfrage, die die 10 Abteilungen ausgibt, deren Mitarbeiter das höchste Durchschnittsgehalt haben. Ausgegeben werden sollen der Rang (1 = höchstes Durchschnittsgehalt bis 10 = niedrigstes Durchschnittsgehalt), die Bezeichnung sowie das Durchschnittsgehalt der Abteilung.
Gehen Sie davon aus, dass es keine zwei Abteilungen mit gleichem Durchschnittsgehalt gibt. Sie können der Übersichtlichkeit halber Views oder With-Anweisungen verwenden. Verwenden Sie jedoch **keine** datenbanksystemspezifischen Erweiterungen wie limit oder rownum.

d) Schreiben Sie eine SQL-Anfrage, die das Gehalt aller Mitarbeiter aus der Abteilung mit AbteilungsID 42 um 5% erhöht.

e) Alle Abteilungen mit der Bezeichnung „Qualitätskontrolle" sollen zusammen mit den Datensätzen ihrer Mitarbeiter gelöscht werden. ON DELETE CASCADE ist für keine der Tabellen gesetzt. Schreiben Sie die zum Löschen notwendigen SQL-Anfragen.

f) Alle Mitarbeiter sollen mit SQL-Anfragen nach den Telefonnummern anderer Mitarbeiter suchen können. Sie dürfen jedoch das Gehalt der Mitarbeiter nicht sehen können. Erläutern Sie in zwei bis drei Sätzen eine Möglichkeit, wie dies in einem Datenbanksystem realisiert werden kann, ohne die gegebenen Relationen, die als Tabellen abgelegt sind, zu verändern. Sie brauchen hierzu keinen SQL-Code schreiben.

Fortsetzung nächste Seite!

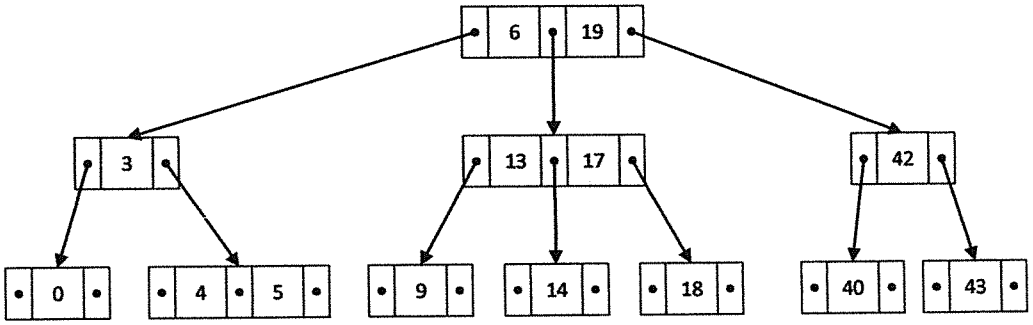
5. Physische Datenorganisation

a) Erläutern Sie die wesentliche Eigenschaft eines Tupel-Identifikators (TID) in ein bis zwei Sätzen.

b) Fügen Sie in einen anfangs leeren B-Baum mit k = 1 (maximal 2 Schlüsselwerte pro Knoten) die im Folgenden gegebenen Schlüsselwerte der Reihe nach ein. Zeichnen Sie den Endzustand des Baums nach jedem Einfügevorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie die sieben Endzustände deutlich.

3, 7, 13, 11, 9, 10, 8

c) Gegeben ist der folgende B-Baum:



Die folgenden Teilaufgaben sind voneinander unabhängig.

i. Löschen Sie aus dem gegebenen B-Baum den Schlüssel 3 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

ii. Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 17 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

iii. Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 43 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

6. Transaktionen

a) Nennen Sie in einem Satz den Unterschied zwischen dem strengen Zwei-Phasen-Sperrprotokoll und dem Zwei-Phasen-Sperrprotokoll.

b) Beim unkontrollierten Mehrbenutzerbetrieb kann es zum Problem der verlorengegangenen Änderungen (lost update) kommen. Erläutern Sie anhand eines Beispiels in drei bis vier Sätzen, wie dieses Problem auftreten kann und welche Folgen sich daraus ergeben.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1: Begriffe und Konzepte

a) Welche Kriterien werden bei der Zielbestimmung im Pflichtenheft unterschieden?

b) Nennen Sie drei Einsatzziele von Softwaremaßen.

c) Welche Arten von Softwaremaßen werden unterschieden?

d) Nennen Sie drei Merkmale evolutionärer Softwareentwicklung.

e) Nennen Sie drei Vorteile des Einsatzes von Versionsverwaltungssoftware.

f) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

g) Was verbirgt sich hinter dem Begriff „*Continuous Integration*“?

Fortsetzung nächste Seite!

Aufgabe 2: UML

a) Gegeben sei folgende natürlichsprachliche Spezifikation eines PKI-Systems:

Damit der Schüler seinem Lehrer die Hausaufgaben verschlüsselt per E-Mail übermitteln kann, bedarf es einer entsprechenden Infrastruktur. Nachdem beide Teilnehmer die notwendige Software installiert haben, erstellt der Lehrer zunächst ein sogenanntes Schlüsselpaar, bestehend aus einem „öffentlichen" (ÖS) und einem zugehörigen „privaten" Schlüssel (PS). Anschließend veröffentlicht der Lehrer seinen ÖS durch Hochladen auf einen sogenannten Keyserver (Schlüsselverzeichnisdienst). Damit steht er jedem Schüler zur Verfügung, so dass dieser den ÖS jederzeit vom Keyserver herunterladen kann. Alternativ kann der Lehrer seinen ÖS auch direkt (z.B. per E-Mail oder USB-Stick) an den Schüler übermitteln.
Der Schüler kann nun seine Nachricht (z.B. seine Lösung) mit dem ÖS des Lehrers verschlüsseln und mit einer E-Mail versenden. Empfängt der Lehrer eine solche E-Mail, so kann er (und nur er) mit seinem PS die Nachricht wieder entschlüsseln. Umgekehrt kann der Lehrer mit seinem PS beliebige Informationen (z.B. die Note) „digital signieren". Diese unterschriebenen Daten übermittelt der Lehrer dann zusammen mit der digitalen Signatur per E-Mail an den Schüler. Der Schüler kann mit dem ÖS des Lehrers prüfen, ob die übermittelte Nachricht unverändert und tatsächlich vom unterschreibenden Lehrer stammt.

Modellieren Sie die in der Spezifikation beschriebenen Anwendungsfälle zusammen mit den jeweils beteiligten Aktoren in einem Use-Case-Diagramm. Betrachten Sie den Keyserver zunächst ebenfalls als Aktor, welcher gegenüber PKI als „externer Vermittler" auftritt.

b) Erstellen Sie ein geeignetes Klassendiagramm für das obige PKI. Berücksichtigen Sie zusätzlich zur verbalen Spezifikation noch folgende Präzisierungen:

i. Die Schlüssel werden mit einer E-Mail-Adresse „benannt", damit der Schüler den richtigen Schlüssel abrufen kann.

ii. Es gibt genau einen Keyserver; dieser verwaltet aber beliebig viele Schlüssel. Er bietet die entsprechenden Dienste zum Veröffentlichen bzw. Abfragen von Schlüsseln an.

iii. Jeder Lehrer hat höchstens ein Schlüsselpaar, aber jeder Schlüssel gehört genau einem Lehrer. Der Schüler hingegen kommuniziert mit mehreren Lehrern und kennt daher mehrere E-Mail-Adressen.

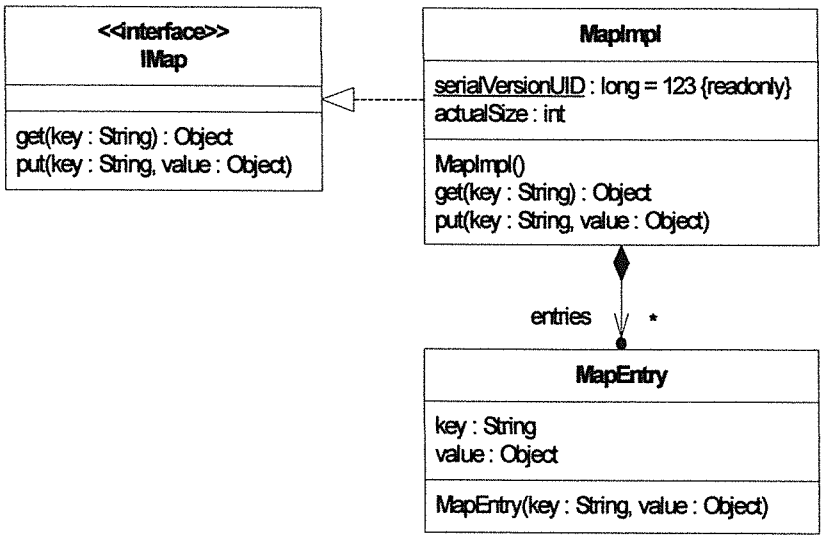
iv. Eine Nachricht kann (muss aber nicht) eine Signatur oder einen ÖS als „Anhang" zusätzlich zum eigentlichen Inhalt (zur Vereinfachung: String) mit sich führen.

v. Für das Signieren bzw. Entschlüsseln ist der PS (das zugehörige Objekt selbst) zuständig. Dafür bekommt er den Inhalt der Nachricht und gibt entsprechend eine Signatur bzw. den entschlüsselten Inhalt zurück.

vi. Für das Prüfen der Signatur und das Verschlüsseln ist der ÖS zuständig. Dazu bekommen die Methoden je nach Bedarf den Inhalt der Nachricht und die Signatur und liefern einen Wahrheitswert bzw. den verschlüsselten Inhalt zurück.

Fortsetzung nächste Seite!

c) Übertragen Sie folgendes UML-Klassendiagramm in Programm-Code einer gängigen und geeigneten objektorientierten Sprache Ihrer Wahl. Die Methodenrümpfe dürfen Sie dabei leer lassen (auch wenn das Programm dann nicht übersetzbar bzw. ausführbar wäre).



Fortsetzung nächste Seite!

Aufgabe 3: Testen

Gegeben sei folgende Java-Methode sort zum Sortieren eines Feldes ganzer Zahlen:

```
public static int[] sort(int[] array) {
  boolean swapped;
  int swapTmp;
  int[] newArray = (int[])array.clone();
  do {
    swapped = false;
    for (int index = 0; index < newArray.length - 1; index++) {
      if (newArray[index] > newArray[index + 1]) {
        swapTmp = newArray[index];
        newArray[index] = newArray[index + 1];
        newArray[index + 1] = swapTmp;
        swapped = true;
      }
    }
  } while(swapped);
  return newArray;
}
```

a) Konstruieren Sie den Kontrollflussgraphen des obigen Code-Fragments und annotieren Sie an den Knoten und Kanten die zugehörigen Datenflussinformationen (Definitionen bzw. berechnende oder prädikative Verwendungen von Variablen).

b) Nennen Sie die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

c) Geben Sie einen möglichst kleinen Testdatensatz an, der eine 100%-ige Verzweigungsüberdeckung dieses Moduls erzielt.

d) Beschreiben Sie kurz, welche Eigenschaften eine Testfallmenge allgemein haben muss, damit sie das datenflussorientierte Überdeckungskriterium „all-uses“ erfüllt.

e) Skizzieren Sie kurz die drei klassischen Integrationstest-Strategien „bottom-up“, „top-down“ und „sandwich“ und nennen Sie je einen typischen Vor- bzw. Nachteil.

Fortsetzung nächste Seite!

Aufgabe 4: Formale Verifikation

Gegeben seien die beiden Java-Methoden zur Berechnung der Chebyshev-Reihe:

$$C_n := (n+2)\cdot 2^{n-1}$$

```
long cpRec(long n) {
      if (n == 0) {
            return 1;
      } else if (n == 1) {
            return 3;
      } else {
            return 4 * cpRec(n-1)
                  - 4 * cpRec(n-2);
      }
}
```

```
long cpIter(long n) {
      long a = 1, k = n-1, p = 1;
      if (n > 0) {
            while (k > 0) {
                  p *= 2;
                  k--;
            }
            a = (n+2)*p;
      }
      return a;
}
```

a) Beweisen Sie formal mittels vollständiger Induktion:

$$\forall n \geq 0: \mathit{cpRec}(n) = C_n$$

Geben Sie dabei jeweils beide Induktionsanfänge (Basisfälle) und Induktionsvoraussetzungen (Induktionsannahmen) explizit an.

b) Welches der folgenden Prädikate stellt eine gültige Schleifeninvariante für die Schleife in **`cpIter(n)`** dar?

a. $a = (n+2)\cdot 2^{n-1}$

b. $p = 2^{n-k}$

c. $p = 2^{n-k-1} \wedge k \geq 0$

d. $C_k = (k+2)\cdot p$

c) Weisen Sie *formal* (z.B. mit der Methode der schwächsten Vorbedingung) nach, dass die von Ihnen gewählte Invariante unmittelbar vor dem ersten Betreten des Schleifenrumpfs in **`cpIter(n)`** gilt.

d) Weisen Sie nun *formal* nach, dass die von Ihnen gewählte Invariante nach jedem Durchlaufen des Schleifenrumpfs in **`cpIter(n)`** gilt.

e) Beweisen Sie schließlich *formal*, dass folgende Nachbedingung erfüllt ist:

$\forall n \geq 0$: **`cpIter(n)`** $= C_n$

f) Geben Sie eine geeignete Schleifenvariante (Terminierungsfunktion) für die Schleife in der Methode **`cpIter(n)`** an und skizzieren Sie kurz, warum damit die Terminierung bewiesen werden kann.

Thema Nr. 2

Teilaufgabe 1

1. ER-Modellierung

Gegeben seien folgende Informationen:

- Krankenhäuser bestehen aus Stationen. Zu jedem Krankenhaus ist dessen Adresse (wodurch es identifiziert werden kann) und die Anzahl an Betten bekannt. Stationen besitzen einen Namen, der nur innerhalb eines Krankenhauses eindeutig ist.
- Stationen bestehen wiederum aus Zimmern, welche nummeriert sind. Eine solche Nummer ist innerhalb einer Station eindeutig.
- Jedes Krankenhaus beschäftigt Personal. Dabei wird festgehalten, welches Gehalt bezahlt wird. Personal kann in verschiedenen Krankenhäusern beschäftigt sein.
- Personal ist durch eine Personal-Nummer gekennzeichnet und kann unter anderem in Ärzte und Krankenpfleger unterteilt werden. Zu einem Arzt ist sein Fachbereich bekannt und von welchen Krankenpflegern er Vorgesetzter ist. Kein Krankenpfleger kann zugleich Arzt sein, aber durchaus mehrere vorgesetzte Ärzte haben.
- Eine Station kann von mehreren Ärzten gleitet werden. Ein Arzt kann ebenso mehrere Stationen leiten. Außerdem ist bekannt, ob und in welchem Zimmer ein Arzt sein Büro hat. Kein Arzt muss sich sein Büro mit einem anderen Arzt teilen.
- Personal arbeitet auf Stationen in Schichten. Eine Schicht kann über Datum und Zeitraum eindeutig identifiziert werden. Eine Person kann in einer Schicht auf nur einer Station arbeiten.

a) Erstellen Sie für das oben gegebene Szenario ein geeignetes ER-Diagramm.
Verwenden Sie dabei – wenn angebracht – das Prinzip der Spezialisierung. Kennzeichnen Sie die Primärschlüssel der Entity-Typen, totale Teilnahmen und schwache Entity-Typen.
Zeichnen Sie die Funktionalitäten der Relationship-Typen in das Diagramm ein.

b) Überführen Sie Ihr in Aufgabe a) erstelltes Modell in ein verfeinertes relationales Schema.
Kennzeichnen Sie die Schlüssel durch Unterstreichen. Datentypen müssen nicht angegeben werden.

Fortsetzung nächste Seite!

2. SQL und relationale Algebra

Gegeben sei der folgende Ausschnitt aus dem Schema einer Schulverwaltung:

Person : {[
  ID : INTEGER,
  Name : VARCHAR(255),
  Wohnort : VARCHAR(255),
  Typ : CHAR(1)
]}

Klasse : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Klassenlehrer : INTEGER
]}

Unterricht : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Lehrer : INTEGER,
  Fach : VARCHAR(100)
]}

Klassenverband : {[
  Schüler : INTEGER,
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER
]}

Hierbei enthält die Tabelle *Person* Informationen über Lehrer (Typ 'L') und Schüler (Typ 'S'); andere Werte für Typ sind nicht zulässig. *Klasse* beschreibt die Klassen, die in jedem Schuljahr gebildet wurden, zusammen mit ihrem Klassenlehrer. In *Unterricht* wird abgelegt, welcher Lehrer welches Fach in welcher Klasse unterrichtet; es ist möglich, dass derselbe Lehrer mehr als ein Fach in einer Klasse unterrichtet. *Klassenverband* beschreibt die Zuordnung der Schüler zu den Klassen.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Unterricht* mit allen ihren Constraints (einschließlich Fremdschlüsselconstraints) anlegt.

b) Definieren Sie ein geeignetes Constraint, das sicherstellt, dass nur zulässige Werte im Attribut Typ der (bereits angelegten) Tabelle *Person* eingefügt werden können.

c) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung der Klassen bestimmt, die im Schuljahr 2015 die meisten Schüler haben.

d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Lehrer bestimmt, die nur Schüler aus ihrem Wohnort unterrichtet haben.

e) Schreiben Sie eine SQL-Anweisung, die die Namen aller Schüler bestimmt, die immer den gleichen Klassenlehrer hatten.

f) Schreiben Sie eine SQL-Anweisung, die alle Paare von Schülern bestimmt, die mindestens einmal in der gleichen Klasse waren. Es genügt dabei, wenn Sie die ID der Schüler bestimmen.

g) Formulieren Sie eine Anfrage in der relationalen Algebra, die die ID aller Schüler bestimmt, die mindestens einmal von 'Ludwig Lehrer' unterrichtet wurden.

h) Formulieren Sie eine Anfrage in der relationalen Algebra, die Namen und ID der Schüler bestimmt, die von allen Lehrern unterrichtet wurden.

Beachten Sie bei der Formulierung der SQL-Anfragen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

Fortsetzung nächste Seite!

3. Entwurfstheorie

Gegeben sei folgendes relationale Schema R in erster Normalform:

$$R : \{[A,B,C,D,E,F]\}$$

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$\begin{aligned} AC &\rightarrow B, \\ DEF &\rightarrow BC, \\ F &\rightarrow AB, \\ D &\rightarrow F, \\ BC &\rightarrow E \end{aligned}$$

$\}$

a) Bestimmen Sie alle Kandidatenschlüssel von R mit FD.
Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.

b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung FD_C von FD:
   i. Führen Sie eine Linksreduktion von FD durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an (FD_L).
   ii. Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion (FD_L) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an (FD_R).
   iii. Bestimmen Sie eine kanonische Überdeckung FD_C von FD auf Basis des Ergebnisses der Rechtsreduktion (FD_R).

d) Zerlegen Sie R mit FD_C mithilfe des *Synthesealgorithmus* in 3NF.

e) Prüfen Sie *für alle Relationen* der Zerlegung aus d), ob sie jeweils in BCNF sind.

Fortsetzung nächste Seite!

4. Transaktionen

a) Erläutern Sie kurz die Konzepte Atomarität und Isolation.
b) Betrachten Sie den folgenden Schedule S:

T_1	T_2	T_3
$r_1(x)$		
	$r_2(x)$	
	$r_2(y)$	
	$w_2(x)$	
	c_2	
$r_1(y)$		
		$r_3(y)$
		$w_3(y)$
		c_3
$w_1(y)$		
c_1		

Geben Sie den Ausgabeschedule (einschließlich der Operationen zur Sperranforderung und -freigabe) im strikten Zweiphasen-Sperrprotokoll für den obigen Eingabeschedule S an.
Hinweis: Sollte während der Ausführung ein Deadlock auftreten, führen Sie ein Rollback einer der beteiligten Transaktionen durch.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 (Wissen)

a) Ordnen Sie die unten stehenden Aussagen entsprechend ihres Wahrheitsgehalts in einer Tabelle der folgenden Form ein:

Kategorie	*WAHR*	*FALSCH*
X	X1, X3	X2
Y	Y2	Y1
...	...	...

A-	Allgemein
A1	Im Software Engineering geht es vor allem darum qualitativ hochwertige Software zu entwickeln.
A2	Software Engineering ist gleichbedeutend mit Programmieren.
B-	Vorgehensmodelle
B1	Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.
B2	Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.
B3	Das Spiralmodell ist ein Vorläufer sogenannter Agiler Methoden.
C-	Anforderungserhebung
C1	Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z.B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zum Vorschein kommen könnten.
C2	Ein Szenario beinhaltet eine Menge von Anwendungsfällen.
C3	Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.
D-	Architekturmuster
D1	Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
D2	Das Blackboard Muster ist besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
D3	„Dependency Injection" bezeichnet das Konzept, welches Abhängigkeiten zur Laufzeit reglementiert.
E-	UML
E1	Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.
E2	Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.
E3	Komponentendiagramme beschreiben die Struktur eines Systems.
F-	Entwurfsmuster
F1	Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.
F2	Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.
F3	Im Kommando Entwurfsmuster (eng.: „Command Pattern") werden Befehle in einem sog. Kommando-Objekt gekapselt, um sie bei Bedarf rückgängig zu machen.
G-	Testen
G1	Validation dient der Überprüfung von Laufzeitfehlern.
G2	Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.
G3	Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

b) Nennen Sie jeweils zwei unterschiedliche Verfahren mit welchen in der Praxis die Prozesse der „Validierung" bzw. der „Verifikation" durchgeführt werden.

c) Worin besteht der Unterschied zwischen „White-Box-Testing" und „Black-Box-Testing"?

d) Nennen Sie fünf Herausstellungsmerkmale des „eXtreme Programming" Ansatzes.

e) Begründen Sie, warum bei der Entwicklung nach der Methode des „eXtreme Programming" langfristig gesehen Refactorings zwingend notwendig werden.

f) Wie wird in der Praxis während und nach erfolgtem Refactoring sichergestellt, dass keine neuen Defekte eingeführt werden bzw. wurden?

Fortsetzung nächste Seite!

Aufgabe 2 (UML: Klassendiagramme)
Betrachten Sie folgende natürlich-sprachliche Spezifikation eines Dateisystems:

Ein Directory hat einen Namen und kann
- *Dateien,*
- *harte Verweise („hard links") auf eine Datei,*
- *symbolische Verweise („soft links") auf eine Datei oder ein Directory,*
- *sowie selbst wieder Directories*

enthalten. Die Wurzel dieses Directory-Baums wird mit „/" bezeichnet.

Modellieren Sie obigen Sachverhalt als UML-Klassendiagramm und berücksichtigen Sie dabei folgende Anforderungen daran:

- Ordnen Sie die Operationen `rename` (umbenennen), `delete` (löschen) und `dir` (Inhaltsverzeichnis zurückgeben) den richtigen Klassen zu und führen Sie sie mitsamt ihren Signaturen an. Dabei gibt `delete` zurück, ob das Löschen erfolgreich war oder nicht (z.B. weil der zu löschenden Ordner nicht leer war).
- Verwenden Sie für Ihr Klassendiagramm die in diesem Zusammenhang sinnvollen Abstraktionen der Generalisierung/Spezialisierung sowie der Aggregation/Komposition.
- Geben Sie sinnvolle Multiplizitäten bei Assoziationen an und erläutern Ihre Wahl kurz.

Fortsetzung nächste Seite!

Aufgabe 3 (Testen)

Beim strukturellen Testen wird versucht, möglichst viele Pfade im Kontrollflussgraphen eines Programms abzudecken. Die Adäquatheit der Tests misst man dabei häufig mittels Überdeckungskriterien wie etwa Anweisungsüberdeckung, Zweigüberdeckung, Bedingungsüberdeckung und Pfadüberdeckung.

Gegeben sei folgendes Problem (Spezifikation):
Für eine Zahl $x > 0$ soll die Summe aller Zahlen gebildet werden, die kleiner als x und Vielfache > 0 von 3 oder 5 sind. Für $x \leq 0$ soll 0 zurückgegeben werden.

Beispiel: Angenommen x wäre 7, dann sind 3, 5 und 6 alle Vielfachen von 3 und 5, die gleichzeitig kleiner als 7 sind. Die Summe dieser Zahlen ist 14.

Gegeben sei folgendes Programmstück zur Ermittlung dieser speziellen Summe:

```
public static long specialSums(int until) {
    long sum = 0;
    if (until > 0) {
        for (int i = 1; i <= until; i++) {
            if (i % 3 == 0 || i % 5 == 0) {
                sum += i;
            }
        }
    }
    return sum;
}
```

a) Zeichnen Sie den zum Programm gehörigen Kontrollflussgraphen.

b) Skizzieren Sie anhand der Bedingung (`i % 3 == 0 || i % 5 == 0`) den Unterschied zwischen einfacher und mehrfacher Bedingungsüberdeckung.

c) Nennen Sie Testfälle, die mindestens benötigt werden, um vollständige Anweisungsüberdeckung, Zweigüberdeckung und mehrfache Bedingungsüberdeckung bei dem oben beschriebenen Code zu erreichen. Geben Sie für jeden Testfall die laut Spezifikation erwartete Ausgabe an.

d) Welcher Fehler hat sich in obigem Code eingeschlichen und welches ist der Testfall mit dem kleinsten Eingabewert, der diesen Fehler findet?

e) Warum kann man für ein realistisches Projekt im Regelfall keine vollständige Pfadüberdeckung herstellen?

Fortsetzung nächste Seite!

Aufgabe 4 (UML: Anwendungsfall- und Sequenzdiagramme)

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 zeigt 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*, E-Mail-Adresse) ein. 4. Das System zeigt die zweite Webseite für die Anmeldung. 5. Der Kunde wählt ein Zahlungsverfahren aus und gibt danach die Zahlungs- bzw. Rechnungsdaten ein. 6. Das System prüft die Zahlungs- bzw. Rechnungsdaten. 7. Sind die Daten gültig, zeigt das System die dritte Webseite der Anmeldung. 8. Der Kunde gibt einen Benutzernamen und ein Passwort ein, womit der Anmeldeprozess abgeschlossen wird.	
Alternativverlauf	
Beginn wie Normalverlauf 1-6 7a. Sind die Daten ungültig, zeigt das System eine Aufforderung zur Korrektur. 7b. Der Kunde ändert seine angegebenen Zahlungs- bzw. Rechnungsdaten. 7c. Das System prüft die Zahlungs- bzw. Rechnungsdaten. 7d. Sind die Daten gültig, übernimmt das System die Korrektur und zeigt die dritte Webseite der Anmeldung. Andernfalls wird die Anwendung beendet. Ende gemäß Normalverlauf 8	

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