Sammlung aller Staatsexamensaufgaben der Prüfungsnummer

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2010

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____ Kennwort: _____ Arbeitsplatz-Nr.: _____	**Frühjahr 2010**	**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: **9**

Bitte wenden!

Thema Nr. 1

Aufgabe 1: Modellierung eines Reiseunternehmens

Es sei folgender Sachverhalt gegeben:
Ein Reiseunternehmen bietet verschiedene Reisen an. Dazu beschäftigt es eine Reihe von Reiseleitern, wobei eine Reise von mindestens einem Reiseleiter geleitet wird. Da Reiseleiter freiberuflich arbeiten, können sie bei mehreren Reiseunternehmen Reisen leiten.
An einer Reise können mehrere Teilnehmer teilnehmen, ein Teilnehmer kann auch an verschiedenen Reisen teilnehmen.

a) Modellieren Sie diesen Sachverhalt in einem UML-Klassendiagramm. Für Teilnehmer und Reiseleiter sollen Sie dabei eine abstrakte Oberklasse definieren. Achten Sie dabei auf die Multiplizitäten der Assoziationen. Sie müssen keine Attribute bzw. Methoden angeben.

b) Eine Reise kann jedoch nur mit einer begrenzten Kapazität angeboten werden, das heißt, zu einer bestimmten Reise kann nur eine begrenzte Anzahl von Teilnehmern assoziiert werden. Als Ausgleich soll pro Reise eine Warteliste verwaltet werden.
Modellieren Sie diesen erweiterten Sachverhalt in einem neuen Diagramm. Nicht veränderte Klassen brauchen nicht noch einmal angegeben werden. Beachten Sie dabei, dass die Reihenfolge bei einer Warteliste eine Rolle spielt.

c) Implementieren Sie die in Aufgabenteil b) modellierten Klassen in Java. Fügen Sie eine Methode hinzu, die einen Teilnehmer von einer Reise entfernt. Dabei soll automatisch der erste Platz der Warteliste zu einem Reiseteilnehmer werden, wenn die Warteliste nicht leer ist. Achten Sie auf die Navigierbarkeit Ihrer Assoziationen. Sie können davon ausgehen, dass die Methode nur mit Teilnehmern aufgerufen wird, die in der Tat Teilnehmer der Reise sind.

Fortsetzung nächste Seite!

Aufgabe 2:

1. Relationale Algebra

Die relationale Algebra wird aufgebaut über einer Grundmenge von mengenwertigen Operationen. Diese Grundoperationen können auf Relationen angewendet werden und erzeugen als Ergebnis wieder eine Relation.

Notation: π = Projektion; σ = Selektion; $\bowtie$ = Join; x = kartesisches Produkt; \ = Mengendifferenz; $\cap$ = Schnittmenge; $\cup$ = Vereinigungsmenge; ρ = Umbenennen; $\div$ = relationale Division

a) Beschreiben Sie ein vollständiges SQL-Statement (oder mehrere) und geben Sie an, welche Klausel mit welcher Operation aus der relationalen Algebra korrespondiert (alle Operationen müssen verwendet werden). Bedenken Sie, dass einige Operatoren an mehreren Stellen vorkommen können.

b) Definieren Sie die mengenwertige Operation „Division". Auf welche Grundoperation kann die Division zurückgeführt werden und wie? Ein Beweis ist nicht erforderlich.

2. Transaktionen und Transaktionsverarbeitung

a) Nennen und definieren Sie die vier wesentlichen Merkmale einer Datenbanktransaktion.

b) Erklären Sie steal, no-steal, force, no-force im Zusammenhang mit der Fehlerbehandlung.

c) Fassen Sie die vier Kombinationen von force/no-force, steal/no-steal hinsichtlich der Anforderungen an die Redo- und Undo-Recovery auf folgende Weise zusammen:

	Force	No-Force
Steal	• Kein Redo • Undo	• •
No-Steal	• •	• •

Übertragen Sie die Tabelle und ergänzen Sie diese.

Fortsetzung nächste Seite!

3. ER-Modellierung und Relationenmodell

Szenario: Sie sollen ein System zur Verwaltung von Firmen entwerfen.

- Eine Firma ist eindeutig bezeichnet durch ihre Handelsregisternummer. Daneben hat sie einen Namen und eine Adresse, die sich aus Straße, Hausnummer, Ort und Postleitzahl zusammensetzt.
- Firmen bestehen aus Abteilungen. Eine Firma kann dabei aus mehreren Abteilungen bestehen, eine Abteilung kann jedoch immer nur zu einer Firma gehören. Es kann Firmen geben, die nicht in Abteilungen eingeteilt sind, aber jede Abteilung gehört zu einer Firma. Abteilungen haben eine eindeutige AbteilungsID und einen Namen.
- Firmen beschäftigen Mitarbeiter. Ein Mitarbeiter kann nur für eine Firma arbeiten, eine Firma aber beliebig viele Mitarbeiter haben. Ein Mitarbeiter hat eine eindeutige Sozialversicherungsnummer (SVN), einen Namen (bestehend aus Name und Vorname), ein Geburtsdatum und ein Alter, das sich aus dem Geburtsdatum errechnet. Daneben hat ein Mitarbeiter eines oder mehrere Spezialgebiete, auf dem bzw. denen er Fachmann ist.
- Mitarbeiter können Angehörige haben. Zu diesen werden jeweils Geburtstag und Name gespeichert. Ein Mitarbeiter kann keine zwei Angehörigen mit gleichem Namen haben.
- Abteilungen bearbeiten Projekte. Ein Projekt ist gekennzeichnet durch seine ProjektID. Daneben besteht es aus einem Namen und einem Kunden. Ein Projekt kann von mehreren Abteilungen bearbeitet werden und eine Abteilung kann an mehreren Projekten arbeiten. Um nicht den Überblick zu verlieren, soll der Zeitraum gespeichert werden, in dem eine Abteilung ein Projekt bearbeitet. Dieser Zeitraum gliedert sich in die Felder „von“ und „bis“.

a) ER-Modellierung:

Erstellen Sie ein ER-Modell des oben beschriebenen Szenarios. Geben Sie Kardinalitäten von Beziehungen sowie Rollennamen an. Verwenden Sie auch abgeleitete, zusammengesetzte und mehrwertige Attribute, mehrstellige Beziehungstypen und schwache Entitätstypen, wo es sinnvoll ist.

b) Relationenmodell:

Überführen Sie das ER-Modell in ein Relationenschema.

Notation: Unterstreichen Sie Primärschlüssel; kennzeichnen Sie Fremdschlüssel durch Angabe der referenzierten Relation in eckigen Klammern.

Beispiel:
Autor(<u>AID</u>, Name)
Buch(<u>ISBN</u>, Schriftsteller[Autor])

Fortsetzung nächste Seite!

4. SQL

Ein Internetauktionshaus nutzt eine relationale Datenbank zum Speichern seiner relevanten Daten. Angenommen ist folgender Auszug aus dem Tabellenschema zu den Benutzern, Auktionen und Geboten:

User(UId, Name,...)
Auction(AId, Name, Begin_Auction, End_Auction, Seller[User], ...)
Bid(UId, AId, Bid_Time, Amount)

Das Auktionshaus will ein Bewertungssystem einführen, um die Risiken für Verkäufer eindämmen zu können.
Folgende Tabelle soll zu den existierenden Tabellen hinzugefügt werden:

Review(RId, AId[Auction], BuyerID[User], Rating, Rating_Time)

RId, AId und BuyerID sind beliebige ganzzahlige Datentypen, Rating ist ein ganzzahliger Datentyp, der die Werte 1,2,3,4,5 annehmen kann, und Rating_Time ist ein timestamp. RId ist Primärschlüssel und die beiden Fremdschlüssel dürfen nie leer sein.

a) Geben Sie das vollständige CREATE-Statement zum Anlegen von Review an, das möglichst viele Constraints erfüllt.

b) Erstellen Sie eine Sicht mit dem Namen Auction_Review mit den Attributen (AId, UId, Durchschnittsrating):
Zu jeder Auktion (gekennzeichnet mit AId) soll das Durchschnittsrating jedes Nutzers (UId) angegeben werden, der auf diese Auktion geboten hat. Die Durchschnittsratings sollen auf allen Ratings des Nutzers basieren, die zeitlich vor Ende der betrachteten Auktion vergeben wurden.

c) Erstellen Sie eine Sicht mit dem Namen Auction_Max_Amount mit den Attributen (AId, MaxAmount):
Zu jeder abgeschlossenen Auktion (aktueller Zeitpunkt mit sysdate()) soll der größte gebotene Betrag angegeben werden.

Thema Nr. 2

Aufgabe 1: Systementwurf und Programmierung: Objektorientierte Softwareentwicklung

Im Rahmen einer wissenschaftlichen Konferenz können Personen in verschiedenen Rollen auftreten, nämlich als Teilnehmer und/oder als Autor. Eine Konferenz kann von beliebig vielen Autoren und Teilnehmern besucht werden. Autoren reichen eine Veröffentlichung zu einer Konferenz ein. Jede Veröffentlichung wird von entweder zwei oder drei Gutachtern bewertet, die jeweils ein Gutachten über sie verfassen. Ein Gutachten kann positiv oder negativ sein. Jede Veröffentlichung hat mindestens einen Autor. Derselbe Autor kann allerdings mehrere Veröffentlichungen zu einer Konferenz einreichen. Jede Veröffentlichung kann nur bei einer einzigen Konferenz eingereicht werden. Die Gutachter können für mehrere Konferenzen tätig sein. Veröffentlichungen haben einen Titel. Personen haben einen Namen.

a) Geben Sie ein für diesen Sachverhalt passendes UML-Klassendiagramm an. Verwenden Sie je nach Bedarf uni- und bidirektionale Assoziationen und beschriften Sie die Assoziationsenden mit geeigneten Rollennamen und Mulitplizitäten. Die Klassen Ihres Diagramms sollen die im Text genannten Attribute mit jeweils passendem Typ enthalten. Sie können zur Modellierung entweder UML 1.*x* oder UML 2.0 wählen. Geben Sie an, welchen UML-Standard Sie Ihrer Lösung zugrunde legen.

b) Implementieren Sie die Klasse `Konferenz` mit einer Methode `findeTeilnehmendeAutoren`, die alle Personen zurückgibt, die sowohl Autor für die Konferenz sind als auch an der Konferenz teilnehmen. Implementieren Sie dazu auch alle Klassen und Assoziationen aus Ihrem Klassendiagramm, die Sie für Ihre Implementierung benötigen.

Der Rückgabetyp der Methode `findeTeilnehmendeAutoren` kann z. B. ein beliebiger Collectiontyp (in Java beispielsweise `Collection`, in C++ beispielsweise `Vector`) sein.

Geben Sie an, welche Sprache Sie gewählt haben.

c) Gegeben sei folgendes Primärszenario einer erfolgreichen Einreichung zu einer Konferenz:

Ein Autor reicht eine Veröffentlichung bei einer Konferenz ein. Anschließend wird die Veröffentlichung zwei Gutachtern zur Begutachtung zugewiesen, die daraufhin jeweils ein Gutachten erstellen. Beide Gutachten fallen positiv aus. Der Autor wird über die Annahme seiner Veröffentlichung benachrichtigt.

Geben Sie für dieses Szenario ein Sequenzdiagramm an.

Es ist nicht notwendig, dass die verwendeten Methoden im Klassendiagramm aus Teilaufgabe a) enthalten sind.

Fortsetzung nächste Seite!

Aufgabe 2:

1. Integritätsbedingungen

Eine Aufgabe von Datenbankmanagementsystemen ist die Erhaltung der Datenintegrität.

a) Welche Arten von Integritätsbedingungen gibt es?

b) Geben Sie für jede Art von Integritätsbedingung je ein Beispiel an.

2. Modellierung

Der Veranstalter der Ausstellung "Becit" möchte diese in einer relationalen Datenbank verwalten:

Zu jedem Ausstellungsraum gibt es eine eindeutige Raumnummer (RNr). Ein Raum hat eine bestimmte Fläche (in m^2), eine Höhe (in m) und zusätzliche Sonderausstattungen wie Beamer, Fernseher, Stühle, Bänke, usw. Die Ausstellungsstände werden in Räumen aufgestellt. Es hat sich eingebürgert, den Ständen innerhalb von Räumen Buchstaben zuzuordnen (SId). So ist beispielsweise Stand "3B" der 2. Stand in Raum 3. Zu Ständen ist bekannt, wieviel Fläche (in m^2) sie benötigen, die Anzahl benötigter Tische und Stühle und möglicherweise benötigte Sonderausstattungen des entsprechenden Raumes.
Die Stände werden dann den Anbietern zugeordnet. Für die Anbieter werden eindeutige Ids vergeben. Zusätzlich sind von diesen jeweils Name, Adresse und Telefonnummer bekannt.
Das Personal kümmert sich um den reibungslosen Ablauf der Ausstellung. Jedes Mitglied des Personals besitzt eine Personalnummer (PNr). Ebenfalls bekannt ist der Lohn des Personals.
Das Personal unterteilt sich in Auf-/Abbau-Personal, Sicherheitspersonal und Reinigungspersonal. Das Auf-/Abbau-Personal bekommt bestimmte Stände zugewiesen, die es auf- und abbaut. Das Sicherheitspersonal kümmert sich um die Sicherheit der Ausstellung. Die einzelnen Mitglieder des Sicherheitspersonals patroullieren hierfür in den ihnen zugewiesenen Ausstellungsräumen. Sie können über ein Funkgerät verfügen und ggf. auch bewaffnet sein. Das Reinigungspersonal kümmert sich um die Endreinigung der Ausstellungsräume. Da hier in verschiedenen Räumen unterschiedlich viel Arbeit anfallen kann, können vorab keine Räume zugeordnet werden.
Zu beachten ist insbesondere, dass sämtliche Personen des Personals auch mehrere Aufgaben übernehmen können, d.h. sie können beim Aufbau mithelfen, zusätzlich während der Ausstellung für die Sicherheit sorgen und am Ende ggf. die Räume mit reinigen.

a) Entwerfen Sie für das beschriebene Szenario ein ER-Diagramm. Bestimmen Sie hierzu

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute;
- ein passendes ER-Diagramm;
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen;
- die Kardinalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

Fortsetzung nächste Seite!

b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an.

c) Bei den Zuordnungen der Sonderausstattungen kann es zur Verletzung einer Integritätsbedingung kommen.

   i) Begründen Sie, woran dies liegt.

   ii) Wie lässt sich das Problem innerhalb einer Datenbank verhindern?

3. Normalformen

Gegeben sei das folgende Relationenschema

Kaeufe: {[*ArtikelId, ArtikelBezeichnung, EinkaufsPreis, VerkaufsPreis, Datum, TagesAktion, KundeName, KundeVorname, KundeGebdatum, Filiale, Verkaeufer*]}

und eine Menge F von funktionalen Abhängigkeiten. **Hinweis:** Die Abkürzungen innerhalb der Menge F beziehen sich auf die Großbuchstaben in den Attributnamen der Relation $Kaeufe$.

$$
\begin{aligned}
F := \{ & \\
& AI \rightarrow AB \\
& AI, AB \rightarrow EP \\
& KN, KV \rightarrow KG \\
& AI, D, KN, KV \rightarrow V \\
& AI, AB, KN, KV, D, V \rightarrow F, VP \\
& D \rightarrow TA \\
& V \rightarrow F \\
\} &
\end{aligned}
$$

a) Ermitteln Sie sämtliche Schlüsselkandidaten von $Kaeufe$.

b) Zeigen Sie, dass das Relationenschema $Kaeufe$ nicht in dritter Normalform ist.

c) Bestimmen Sie eine kanonische Überdeckung F_c zu F.

4. SQL und Relationale Algebra

Gegeben sei folgendes Schema der relationalen Datenbank eines Aktienhändlers:

$Aktie\{\underline{ANr}, Name, Firma[Firma], Ausgabedatum, Ausgabepreis\}$

$Aktienkurs\{\underline{ANr[Aktie]}, \underline{Datum}, \underline{Zeit}, Wert\}$

$Firma\{\underline{Name}, Adresse, Telefon\}$

$Kunde\{\underline{KndNr}, Name, Geburtsdatum, Telefon\}$

$Konto\{\underline{KntNr}, KndNr[Kunde], Eroeffnungsdatum, Saldierungsdatum\}$

Fortsetzung nächste Seite!

$$Buchung\{\underline{KntNr[Konto]}, \underline{BNr}, ANr[Aktie], Aktienzahl, EKdatum, EKzeit, EKpreis, VKdatum, VKzeit, VKpreis\}$$

Hinweis: Für Fremdschlüssel wird nach dem Format `<Attr> [<RefTable>]` die referenzierte Tabelle angegeben. Beispielsweise ist $KndNr[Kunde]$ in der Relation `Konto` ein Fremdschlüssel der Relation `Kunde`.

Sämtliche verfügbaren Aktien sind in der Relation `Aktie` enthalten. Die zugehörigen Kurse werden in der Relation `Aktienkurs` festgehalten. Zu jedem (im Rahmen dieser Aufgabe notwendigen) Zeitpunkt existiert ein korrespondierendes Tupel. Die Aktiengesellschaften (also Firmen, welche Aktien ausgeben) sind in der Relation `Firma` gespeichert. Die Kunden des Aktienhändlers werden in der Relation `Kunde` verwaltet. Jeder Kunde kann beliebig viele Konti eröffnen. Diese werden in der Relation `Konto` aufgeführt und dem Kunden direkt zugeordnet. Aktienkäufe und Aktienverkäufe erfolgen paketweise. Kauft ein Kunde ein Paket einer Aktie, so wird ein neues Tupel in der Relation `Buchung` erstellt. Die Buchungsnummern werden für jedes Konto jeweils in aufsteigender Reihenfolge beginnend bei 1 vergeben. Innerhalb eines neuen Tupels werden Umfang des Pakets (Aktienzahl), Datum und Zeitpunkt des Kaufs und insbesondere der Preis des Aktienpakets festgehalten. Die entsprechenden Attribute für einen Verkauf werden zu $NULL$ initialisiert. Wird später ein Verkauf des Aktienpakets durchgeführt, so werden die Werte für `VKdatum`, `VKzeit` und `VKpreis` entsprechend gesetzt. Aktuell sei der Zeitpunkt 13:00:00 Uhr und das Datum 2008/07/20. Für Datumsangaben gelte insbesondere, dass eine Verschiebung des Datums um t Tage durch folgende Addition durchgeführt werden kann: $2008/07/20 + t$. Beispielsweise ergibt $2008/07/20 + 20 = 2008/08/09$.

a) Formulieren Sie die Anfragen in den folgenden Teilaufgaben (i - ii) in der Anfragesprache SQL. Zur Vereinfachung dürfen Sie sich beliebige Sichten (Views) definieren.

   i) Gesucht sind alle Aktienanteile, welche Kunde "Müller" aktuell hält. Geben Sie für diese Kontonummer, Buchungsnummer, Aktiennummer und die zugehörigen Firmennamen aus.

   ii) Nehmen Sie an, das Ergebnis aus Teilaufgabe i) sei als View *AktuelleAktienMueller* verfügbar.
   Geben Sie nun gruppiert nach den Firmennamen die Gewinne/Verluste aus, welche Müller beim Verkauf seiner aktuellen Aktienanteile machen würde.

b) Formulieren Sie folgende Anfrage in der relationalen Algebra:

   Für welche Firmen hat sich der Aktienkurs im 1. Jahr nach der Ausgabe mindestens verdoppelt?

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2011

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2011**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **10**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

Datenmodellierung: ER–Modell, SQL

In einer *Fußballdatenbank* sollen folgende Informationen gespeichert werden:

- Zu jeder *Fußballweltmeisterschaft* werden *Jahr*, *Austragungsland* und *Weltmeisternation* gespeichert.
- Zu jeder *Nation* werden der *Sitz* des nationalen Fußballverbandes und die Anzahl der bisher errungenen *Weltmeistertitel* gespeichert.
- Zu jeder *Fußballweltmeisterschaft* werden die teilnehmenden Nationen gespeichert sowie die Spieler, die für die Nation an der Weltmeisterschaft teilgenommen haben.
- Zu jedem *Spieler* werden *Name*, *Verein* und *Geburtsdatum* gespeichert.

Nationen und Spieler sollen jeweils noch ein Attribut *Nnr* bzw. *Snr* mit einer eindeutigen Identifikationsnummer haben.

a) Erstellen Sie ein ER–Modell, welches das oben dargestellte Szenario vollständig abbildet. Verwenden Sie wann immer möglich (binäre oder auch höherstellige) Relationships.

b) Übertragen Sie Ihr ER–Modell ins relationale Modell. Erstellen Sie dazu Tabellen mit Hilfe von CREATE TABLE–Statements in SQL. Berücksichtigen Sie die Fremdschlüsselbeziehungen.

c) Erweitern Sie den Entity–Typ *Spieler* um eine *Rückennummer* und machen Sie diese – anstelle der künstlichen Identifikationsnummer *Snr* – zum „schwachen Schlüssel". Diese Rückennummer ist innerhalb einer Nation für jeden Spieler nur innerhalb einer Weltmeisterschaft eindeutig.

   Geben Sie auch das entsprechend veränderte CREATE TABLE–Statement für die Relation *Spieler* an.

d) Geben Sie geeignete INSERT-Statements an, die in alle beteiligten Tabellen jeweils mindestens ein Tupel einfügen, so dass nach Ausführung aller Einfügungen alle Integritätsbedingungen erfüllt sind.

Fortsetzung nächste Seite!

Datenbankanfragen in SQL

Gegeben seien die folgenden drei Tabellen:

- *liefert (LNr, Teil, Anzahl)*
- *Lieferant (LNr, Name, Branche)*
- *Filialen (LNr, Ort),*

wobei sich das Fremdschlüsselattribut *LNr* aus *liefert* und *Filialen* auf das Schlüsselattribut *LNr* aus *Lieferant* bezieht. Formulieren Sie auf den obigen Tabellen folgende SQL-Anfragen:

a) Bestimmen Sie die Namen aller Lieferanten aus *München.*

b) Bestimmen Sie die Namen aller Lieferanten, die das Teil T_1 liefern.

c) Bestimmen Sie, wie viele Teile vom Lieferanten L_1 geliefert werden.

d) Bestimmen Sie für jeden Lieferanten die Summe der Anzahlen der von ihm gelieferten Teile.

Funktionale Abhängigkeiten und Zerlegungen

Gegeben sei das Relationsschema $R = (U, F)$ mit der Attributmenge $U = \{A,B,C,D,E\}$ und der folgenden Menge F von funktionalen Abhängigkeiten:

$$F = \{ A \rightarrow B, AB \rightarrow C, AD \rightarrow BE, E \rightarrow D\}$$

a) Geben Sie alle Schlüssel für R an.

b) Ist R in 3. Normalform bzw. in Boyce-Codd-Normalform?

Fortsetzung nächste Seite!

Teilaufgabe 2:

1 Firmenstruktur

Eine Firma besteht aus null oder mehr Abteilungen, von denen jede null oder mehr Angestellte hat. Da sowohl die Abteilungen als auch deren Angestellten geordnet sind, sind Angestellte insgesamt geordnet. Sie haben durchgehende, ganzzahlige IDs, die bei 1 beginnen.

a) Erstellen Sie exemplarisch ein Objektdiagramm: Stellen Sie eine Firma mit dem Instanznamen `f` und den zwei Abteilungen „Produktion" (Name `p`) und „Marketing" (Name `m`) dar. Die Produktion hat zwei Angestellte, Marketing hat einen Angestellten. Die Angestellten haben die Namen `a1`, `a2`, und `a3`.

b) Implementieren Sie das Klassendiagramm in Java oder in einer anderen geeigneten objektorientierten Programmiersprache Ihrer Wahl. Beachten Sie, dass die Assoziationen bidirektional und geordnet sind. Die beiden Methoden der Klasse `Firma` sollen dabei folgendes Verhalten haben:

- Die Methode `erzeugeIDs` sorgt dafür, dass die IDs wieder korrekt zugewiesen sind. Die alten IDs können beliebig geändert werden, solange das Endergebnis wieder den obenstehenden Kriterien genügt.

c) Angestellte sollen in Manager und einfache Angestellte unterteilt werden. Zeichnen Sie ein Klassendiagramm mit der Oberklasse `Angestellter` und den zwei Unterklassen `Manager` und `EinfacherAngestellter`. Die Klasse `Angestellter` soll nicht instantiierbar sein und erzwingen, dass die Methode `getPosition()` (öffentlich, ohne Argumente, Rückgabewert `String`) von allen konkreten Unterklassen implementiert wird. `Manager` und `EinfacherAngestellter` sollen instantiierbar sein.

d) Wie lautet der Fachbegriff dafür, dass eine Methode in einer Klasse und in deren Unterklassen dieselbe Signatur hat, aber in den Unterklassen unterschiedlich implementiert ist?

Fortsetzung nächste Seite!

2 Briefe versenden

Angestellter
+ id : int
+ empfange(absenderID : int, text : String)

Briefzentrum
+ sende(absenderID : int, empfaengerID : int, text : String)

Ein Angestellter verwendet das Briefzentrum, um einem anderen Angestellten eine Nachricht zu schicken. Absender und Empfänger werden als ganzzahlige IDs angegeben, die Nachricht selbst ist eine Zeichenkette.

a) Erstellen Sie ein Sequenzdiagramm für das folgende Szenario:

- Ein Angestellter (Instanzname `abs`) mit der ID 1 verwendet die Methode `sende()` des Briefzentrums (Name `bz`), um eine Nachricht mit dem Text „Anfrage" an einen anderen Angestellten (Name `empf`) mit der ID 2 zu senden.
- `empf` sendet daraufhin eine Nachricht mit dem Text „Bestätigung" an `abs` zurück.
- Alle Methodenaufrufe sind synchron, „Bestätigung" wird also geschickt, noch während `abs` auf die Beendigung von `sende()` wartet.

3 Telefon

In jedem Büro der Angestellten steht ein Telefon. Es soll als UML-Zustandsdiagramm modelliert werden.

- Dabei gibt es folgende Ereignisse: `abheben`, `auflegen`, `ziffer`.
- Das Diagramm soll *mindestens* die folgenden Zustände enthalten: `verbunden`, `aufgelegt`, `abgehoben`.
- Der Hörer ist anfangs aufgelegt. Dann hebt man ihn ab, wählt genau drei Ziffern und ist verbunden. Der Hörer kann jederzeit aufgelegt werden.

a) Erstellen Sie ein UML-Zustandsdiagramm.

b) Nennen Sie zwei Techniken, mit denen man ein UML-Zustandsdiagramm in Java oder in einer anderen objektorientierten Programmiersprache Ihrer Wahl implementieren kann.

c) Verwenden Sie eine der beiden in Teilaufgabe (b) erwähnten Techniken, um das Zustandsdiagramm aus Teilaufgabe (a) in Java oder einer anderen geeigneten objektorientierten Programmiersprache Ihrer Wahl zu implementieren.

- Das Telefon soll sich die gewählten Ziffern merken, aber nur bis zum Auflegen. Dann werden sie wieder gelöscht.
- Bei einem illegalen Ereignis soll die Ausnahme `IllegalStateException` (in Java eine Unterklasse von `RuntimeException`) geworfen werden.

Thema Nr. 2

Teilaufgabe 1:

Aufgabe 1) Relationales Modell, SQL Data Definition Language (DDL)

Eine Krankenversicherung will Daten über die ärztlichen Behandlungen ihrer Versicherten in einer relationalen Datenbank speichern. Zusätzlich zu den Patientendaten der Versicherten sollen die Daten jeder Behandlung, welche durch einen bestimmten Arzt in einem bestimmten Krankenhaus an einem bestimmten Tag durchgeführt wurde, gespeichert werden. Es gibt also Ärzte, die an einem bestimmten Tag einen oder mehrere Patienten behandelt haben. Ein Arzt arbeitet in genau einem Krankenhaus.

a) Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank.

b) Setzen Sie das gegebene E/R-Diagramm in ein entsprechendes relationales Datenbankschema um. Identifizieren Sie dazu zunächst weitere Attribute, die in obigem Diagramm noch nicht enthalten sind und die gegebenen Entities in sinnvoller Weise beschreiben. Es sollen mindestens zwei Attribute pro Entity angegeben werden. 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 Relationships.

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

Fortsetzung nächste Seite!

Aufgabe 2) SQL Data Manipulation Language (DML)

Gegeben sei das folgende Datenbank-Schema mit den zugehörigen Ausprägungen. Die Primärschlüssel-Attribute sind dabei jeweils unterstrichen.

Lieferant

LNr	LName	LStadt
L1	Huber	München
L2	Müller	Köln
L3	Meier	München
L4	Weiß	Hamburg
L5	Schwarz	Köln

Ware

WNr	Bezeichnung	Preis
W1	Milch	12
W2	Mehl	15
W3	Brot	32
W4	Zucker	58

Kunde

KNr	KName	KStadt
K1	Fuchs	Hamburg
K2	Wolf	München
K3	Vogel	Köln
K4	Wurm	München

Auftrag

LNr	WNr	KNr	Menge
L1	W1	K1	4
L1	W2	K3	21
L2	W2	K3	3
L2	W4	K3	9
L3	W2	K4	18
L4	W2	K1	27
L4	W4	K4	54
L5	W1	K3	7
L5	W2	K3	4

Fortsetzung nächste Seite!

Formulieren Sie die folgenden Anfragen in SQL:

a) Finden Sie die Namen aller Lieferanten, für welche ein Auftrag zur Milchlieferung gespeichert ist.
b) Erstellen Sie eine Liste mit den Namen von Kunden, welche mindestens zwei Aufträge erteilt haben.
c) Finden Sie zu jeder Ware (WNr) die Anzahl der Aufträge, bei denen diese Ware auftaucht. Sortieren Sie das Ergebnis absteigend nach dieser Anzahl.
d) Errechnen Sie für jeden Lieferanten den Gesamtpreis aller in Auftrag gegebenen Waren. Das Ergebnis soll die Lieferantennummer, den Lieferantennamen und die errechnete Summe enthalten.

Wie sieht die Ergebnisrelation zu folgenden Anfragen aus?

e) SELECT LNr, KNr FROM Lieferant, Kunde WHERE LStadt = KStadt;
f) SELECT * FROM Kunde WHERE KNr NOT IN (SELECT KNr FROM AUFTRAG NATURAL JOIN LIEFERANT WHERE LName = 'Schwarz');

Formulieren Sie die folgenden Anfragen in relationaler Algebra:

g) Finden Sie die Namen aller Lieferanten aus Köln.
h) Geben Sie Nummern und Bezeichnungen aller Waren aus, die nach Hamburg geliefert werden.

i) Geben Sie das Ergebnis des folgenden Ausdrucks der relationalen Algebra als Tabelle an:

$$\pi_{WNr}(\text{Auftrag}) \bowtie \text{Ware}$$

Aufgabe 3) Entwurfstheorie

Gegeben sei die nachfolgende relationale Datenbank mit unterstrichenen Schlüsselattributen. Sie enthält die Daten der ausgeliehenen Filme einer Videothek. Dabei kann ein Film in mehreren Kopien vorhanden sein. Ein Kunde kann aber nur eine Kopie eines Filmes ausleihen:

Relation „Ausleihe":

FilmNr	KdNr	Name	Adresse	Titel	Leihgebühr
1	1	Müller	München	Hangover	3
2	1	Müller	München	Transformers 2	6
3	2	Huber	Nürnberg	Avatar	4
2	2	Huber	Nürnberg	Transformers 2	6
5	3	Meier	Hamburg	Zweiohrküken	4
6	4	Meier	München	13 Semester	2

a) Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können.
b) Geben Sie für das obige Datenbankschema alle funktionalen Abhängigkeiten (inkl. der transitiven) an.
c) Überführen Sie das obige Relationenschema in die dritte Normalform.

Fortsetzung nächste Seite!

Teilaufgabe 2:

1. Lebenszyklus und Vorgehensmodelle

Welche der folgenden Aussagen sind richtig? Geben Sie jeweils eine kurze, stichpunktartige Begründung an!

A	Das Wasserfallmodell sollte nur für große Projekte eingesetzt werden, da der Einarbeitungsaufwand sehr groß ist.
B	Das oberste Ziel des Spiralmodells ist die Minimierung des Risikos durch wiederkehrendes Durchlaufen der einzelnen Phasen.
C	Bei der inkrementellen Entwicklung werden Kundenwünsche explizit berücksichtigt.
D	Die Anforderungsbeschreibung legt fest, was ein Produkt „können soll" und „wie es realisiert ist", aber nicht: wie es aussieht.
E	Die Anforderungen an das zu entwickelnde System werden vom Auftraggeber klassifiziert und hierarchisiert.
F	Eine gute Anforderungsspezifikation muss vor allem für Ingenieure verständlich sein, da die Anforderungsspezifikation die Grundlage der Systementwicklung bildet.
G	Der häufigste Fehler bei der Softwareentwicklung besteht darin, das zu entwickeln, was der Kunde braucht, und nicht das, was er will.
H	Verifikation ist der Prozess der Beurteilung eines Systems mit dem Ziel festzustellen, ob die spezifizierten Anforderungen erfüllt sind.
I	Durch Validierung kann überprüft werden, ob das Produkt den Erwartungen des Kunden entspricht.
J	Das Ziel von Black-Box-Tests ist die umfassende Überprüfung der spezifizierten Funktionalität.
K	Die Definition des Softwareproduktes in der Definitionsphase ist ein iterativer Prozess.
L	Nicht-funktionale Anforderungen sind sowohl Teil des Pflichtenhefts als auch des Lastenhefts.

2. Funktionale Programmierung

Schreiben Sie in einer funktionalen Programmiersprache (oder in entsprechendem Pseudocode) folgende Funktionen:

a) Die Funktion *leneo* soll prüfen, ob die Länge eines Strings gerade oder ungerade ist! Verwenden Sie beim Rückgabeparameter einen möglichst sinnvollen Datentyp.

b) Die Funktion *codetext* soll einen String mit gerader Länge umkehren und einen String ungerader Länge zweimal aneinander hängen.

z. B.: *codetext*(„ADAM") ergibt „MADA", *codetext*(„EVA") ergibt „EVAEVA"

Fortsetzung nächste Seite!

3. Objektorientierter Entwurf

Beim Spiel „TicTacToe" belegen zwei Spieler auf einem Spielfeld von 3x3 Kästchen abwechselnd ein Feld mit ihrem „Symbol". Gewonnen hat derjenige, dem es als erstes gelingt, drei seiner Symbole senkrecht, waagerecht oder diagonal zusammenhängend zu setzen. Das Spiel verfügt über eine Methode zum Spielen. Zu jedem Spieler sollen der Name und sein Symbol gespeichert werden. Nach jedem Setzen eines Symbols prüft das Spiel, ob bereits einer der Spieler gewonnen hat, und gibt ggf. eine entsprechende Meldung zum Spielausgang aus.

a) Erstellen Sie für dieses Szenario ein Klassendiagramm, das mindestens die Klassen Spiel und Spieler enthält. Verwenden Sie nur die nötigen Attribute und Methoden.

b) Erstellen Sie ein Objektdiagramm für ein Spiel zwischen Peter und Paula, bei dem folgender Spielstand entstanden ist:

c) Erstellen Sie ein Sequenzdiagramm für die Erzeugung eines Spiels sowie die erste Spielrunde, bei der jeder der Spieler einmal an die Reihe kommt.

d) Implementieren Sie die Klasse „Spiel" in einer gängigen objektorientierten Programmiersprache (oder in entsprechendem Pseudocode).

4. Entwurfsmuster

Welches Entwurfsmuster ist in der obigen Abbildung dargestellt? Erläutern Sie dessen Funktionsweise und geben Sie ein Beispiel für seinen Einsatz an.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2012

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2012**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **10**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

1. ER-Modellierung

Im Folgenden wird eine vereinfachte Kinoverwaltung beschrieben.

Ein Kino hat einen Namen und eine Anschrift (bestehend aus Straße, Postleitzahl und Ort), durch die es eindeutig beschrieben wird. Jedes Kino besitzt eine bestimmte Anzahl an Sälen, in welchen Filme vorgeführt werden. Jeder Saal hat eine eigene Nummer und eine bestimmte Anzahl an Sitzplätzen. In den Sälen werden regelmäßig Filme gezeigt. Ein Film zeichnet sich durch seinen Titel sowie seine Kopie-Nummer aus und hat eine bestimme Spieldauer. Jede Filmvorführung hat einen bestimmten Preis und findet an einem bestimmten Datum zu einer bestimmten Uhrzeit statt. In jedem Film spielen Schauspieler mit. Ein Schauspieler hat einen Namen und kommt aus einem Land. Es können Schauspieler mit demselben Namen im selben Land beheimatet sein.

Erstellen Sie zum obigen Szenario ein ER-Diagramm. Geben Sie dabei alle notwendigen Entitäten und Beziehungen sowie sämtliche Kardinalitäten an. Kennzeichnen Sie jeweils die Schlüssel. Sollte es nötig sein, führen Sie Surrogatschlüssel (künstliche Schlüssel) ein.

2. Relationale Datenbankmodellierung

Das folgende Diagramm zeigt die Modellierung der Tischreservierung in einem Lokal.

Geben Sie zu diesem ER-Modell ein Relationenschema an und vereinfachen Sie es anschließend soweit wie möglich.

Fortsetzung nächste Seite!

3. SQL

Gegeben ist folgendes einfache Datenbankschema zur Personalverwaltung (Schlüsselattribute unterstrichen, Fremdschlüssel überstrichen):

Angestellter (PersNr, Name, Gehalt, Beruf, $\overline{\text{AbtNr}}$, Ort)

Abteilung (AbtNr, Name, Ort)

Formulieren Sie folgende Datenbankoperationen in SQL:

a) Welche Angestellten sind von Beruf Koch und verdienen mehr als 5000 €?

b) Welche Angestellten der Abteilung B17 sind aus München?

c) Hans Meier aus der Abteilung C4 zieht von München nach Erlangen um.

d) Abteilung C4 wird aufgelöst.

Formulieren Sie folgende SQL-Anfrage umgangssprachlich und so exakt wie möglich.

e)
```
SELECT AbtNr
FROM Abteilung a, (SELECT PersNr, Name, AbtNr
                   FROM Angestellter
                   WHERE Gehalt < 2000)
                   WHERE a.AbtNr = t.AbtNr
                   AND a.Ort = 'Nuernberg';
```

4. Normalformenlehre

a) Nennen und erläutern Sie kurz die Ziele der Normalisierung einer Datenbank.

b) Gegeben ist das Datenbankschema in 3. Normalform aus der vorhergehenden Teilaufgabe:

Angestellter (PersNr, Name, Gehalt, Beruf, $\overline{\text{AbtNr}}$, Ort)

Abteilung (AbtNr, Name, Ort)

Zu jedem Ort sollen nun auch die Postleitzahl und die Straße gespeichert werden. Dabei soll die dritte Normalform erhalten bleiben. Ein Firmenmitarbeiter schlägt folgende zwei Lösungen vor:

I. Speicherung der Postleitzahl und Straße zusammen mit dem Ort (z. B. „91058 Erlangen Martensstraße 3“ als Eintrag unter Ort)

II. Folgende Erweiterung des Datenbankschemas:

Angestellter (PersNr, Name, Gehalt, Beruf, $\overline{\text{AbtNr}}$, Ort, Straße, Postleitzahl)

Abteilung (AbtNr, Name, Ort, Straße, Postleitzahl)

Begründen Sie, warum beide Lösungen die 3. Normalform verletzen.

c) Geben Sie einen eigenen Vorschlag zur Lösung des Problems an.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1

Erläutern Sie zwei verschiedene Vorgehensmodelle der Softwareentwicklung und diskutieren Sie jeweils Vor- und Nachteile.

Aufgabe 2

Erläutern Sie die Drei-Schichtenarchitektur für betriebliche Informationssysteme (jeweils mit den Zuständigkeiten der einzelnen Schichten).

Aufgabe 3

Es seien folgende Anforderungen gegeben: Eine Bank hat Kunden und sie führt Konten. Ein Konto gehört genau einem Kunden, aber nicht jeder Kunde muss ein Konto besitzen. Es gibt Spar- und Girokonten.

Zu einem Zeitpunkt soll die folgende Situation bestehen: Die Bank mit Namen *XY* hat als Kunden Frau *Sandra Huber* mit der Kundennummer 1001 und Herrn *Peter Meier* mit der Kundennummer 2002. Frau Huber besitzt ein Girokonto mit Kontonummer 304 und einem Kontostand von -2500,30 Euro sowie ein Sparkonto mit Kontonummer 507 und einem Kontostand von 4500,00 Euro. Herr Meier besitzt ein Girokonto mit Kontonummer 811 und einem Kontostand von 3700,00 Euro. Für jedes Girokonto wird monatlich eine Gebühr von 4,09 Euro erhoben, für das Sparkonto gibt es 2,75% Zinsen im Jahr.

(a) Erstellen Sie ein UML-Klassendiagramm, das die beschriebene Situation zulässt. Insbesondere sind Multiplizitäten und die Typen von Attributen anzugeben. Assoziationen sollen bidirektional sein. Operationen sind (noch) nicht zu berücksichtigen.

(b) Erstellen Sie ein UML-Objektdiagramm (Instanzendiagramm), das die oben beschriebene Situation darstellt.

(c) Ein Bankangestellter soll einen Monatsabschluss bei einer Bank durchführen können. Wird ein Monatsabschluss veranlasst, dann führt die Bank bei jedem ihrer Konten eine Monatsabrechnung durch, wobei bei einem Sparkonto die monatlichen Zinsen gutgeschrieben werden und bei einem Girokonto die monatliche Gebühr abgezogen wird.

Konstruieren Sie ein Sequenzdiagramm, das die Abfolge der Nachrichten bei der Durchführung eines Monatsabschlusses für die oben genannten Objekte zeigt.

(d) Ergänzen Sie die betreffenden Klassen um die Operationen, die sich aus dem Sequenzdiagramm ergeben. Welche Operationen sind abstrakt? Für nicht abstrakte Operationen soll die Wirkung mit Hilfe eines Programmcodes einer objektorientierten Programmiersprache Ihrer Wahl (Sprache angeben) oder mittels (intuitiv klarem) Pseudo-Code formuliert werden.

Fortsetzung nächste Seite!

Aufgabe 4

Das folgende Klassendiagramm ist in einer objektorientierten Programmiersprache Ihrer Wahl (Sprache angeben) zu realisieren. Der Rumpf der beiden konkreten move-Methoden braucht nicht angegeben zu werden. Die Operation `add(g:GrafikObjekt)` soll ein Grafikobjekt `g` zum Inhalt eines Blattes hinzufügen.

Thema Nr. 2

Teilaufgabe 1:

1 ER-Modellierung und Relationenmodell

1.1

Sie sollen die Verwaltung einer Fluggesellschaft modellieren. Erstellen Sie unter Berücksichtigung der unten beschriebenen Kriterien ein ER-Diagramm bestehend aus Entitäts- und Beziehungstypen sowie Attributen. Geben Sie auch die Kardinalitäten mit an.

Verwenden Sie bei Entitäten und Attributen ausschließlich die beschriebenen, soweit dies möglich ist. Es ist ebenfalls nicht erlaubt, künstliche Schlüssel zu erfinden. Die Kardinalitätseinschränkungen können Sie entweder in der (min,max)-Notation oder der einfachen Notation nach Chen (1:1, 1:N, M:N) angeben.

Ein Flugzeug hat eine eindeutige Kennzeichnung sowie eine Beschreibung.

Ein Flugzeugtyp hat eine eindeutige ID, einen Namen, Kapazität und Reichweite.

Jedes Flugzeug ist von genau einem Flugzeugtyp.

Flugzeuge werden auf Routen eingesetzt. Natürlich können mehrere Flugzeuge dieselbe Route befliegen. Ebenso kann ein Flugzeug auf verschiedenen Routen eingesetzt werden. Eine Route hat einen eindeutigen Namen sowie einen Start- und Zielflughafen. Zu einer Route wird außerdem die Länge in Meilen gespeichert.

Passagiere werden durch ihre Ausweisnummer unterschieden. Sie haben einen Vor- und einen Nachnamen sowie eine Kreditkartennummer.

Passagiere können an bestimmten Daten mit einem Flugzeug auf einer Route mitfliegen. Ein Datum besteht aus Jahr, Monat und Tag. Natürlich muss es möglich sein, dass ein Passagier eine Route mit einem Flugzeug an unterschiedlichen Daten nutzt.

1.2

Erstellen Sie zu dem ER-Diagramm aus Aufgabe 1.1 ein Relationenschema.
Berücksichtigen Sie dabei totale Partizipationen und vermeiden Sie unnötiges Ausprägen von Relationen bei allen Beziehungen.

Beispiel für die Notation:

```
Relationenname (Primärschlüssel, Attribut1, Attribut2, …,
    Fremdschlüssel[AndereRelation],
    (FKAttrA, FKAttrB)[DritteRelation])

    Attribut2 NOT NULL
```

Fortsetzung nächste Seite!

2 Normalformen

2.1

Die Normalisierung von Relationenschemata dient der Vermeidung von Redundanzen und dadurch bedingter Anomalien. Geben Sie ein Beispiel für eine nicht-normalisierte Relation an und erläutern Sie zwei mögliche Anomalien an diesem Beispiel.

2.2

Normalisierung beruht auf dem Erkennen und Eliminieren von funktionalen Abhängigkeiten. Erläutern Sie in diesem Zusammenhang kurz die folgenden Begriffe:

a) Funktionale Abhängigkeit

b) Voll-funktionale Abhängigkeit

c) Transitive funktionale Abhängigkeit

d) Superschlüssel

e) Determinante

2.3

In welcher Normalform ist das folgende Beispiel? Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind! Welche Bedingung der nächsthöheren Normalform ist verletzt? Berücksichtigen Sie, dass eine Relation nur dann in n-ter Normalform ist, wenn sie die Bedingungen aller m-ten Normalformen mit $m \leq n$ erfüllt.

PersNr	Name	Strasse	Ort	PLZ	Vorwahl
1	Meier	Abbestr.	München	80999	089
2	Meier	Allacher Str.	München	80997	089
3	Müller	Zieglhof	Regensburg	93055	0941
4	Schmidt	Bucher Hauptstr.	Nürnberg	90427	0911
5	Schmid	Osserweg	Passau	94034	0851
6	Meier	Alfred-Nobel-Str.	Würzburg	97080	0931

Fortsetzung nächste Seite!

3 Anfrageverarbeitung

3.1

Nennen und charakterisieren Sie die wesentlichen Schritte, die eine SQL-Anfrage bei der Verarbeitung durchläuft.

3.2

Zeichnen Sie für folgendes SQL-Statement einen nicht-optimierten Anfragegraphen.

```
SELECT R1.id AS r1, R2.id AS r2, R3.id AS r3
  FROM R1, R2, R3
 WHERE R1.R2_id = R2.id AND R3.R2_id = R2.id
```

Fortsetzung nächste Seite!

Teilaufgabe 2:

1. UML - Klassendiagramm

Erstellen Sie zu der folgenden Beschreibung eines Systems zur Organisation eines Hotels ein Klassendiagramm, das Attribute und Assoziationen mit Kardinalitäten sowie Methoden enthält. Setzen Sie dabei das Konzept der Vererbung sinnvoll ein.

- Ein Hotel besteht aus genau einer Küche und mehreren Gästezimmern.
- Es hat einen Namen, eine Adresse und kann Werbeaktionen durchführen.
- In der Küche arbeitet eine bestimmte Anzahl von Mitarbeitern. Die Küche kann geöffnet oder geschlossen sein.
- Gästezimmer haben eine Nummer und eine bestimmte Anzahl an Betten, die ausgegeben werden kann.
- In diesen Gästezimmern wohnen Gäste, die von einer oder mehreren Servicekräften umsorgt werden.
- Servicekräfte sind Angestellte des Hotels und sind nur für die ihnen zugeordneten Gästezimmer verantwortlich.
- Jede Servicekraft hat einen Namen und eine Personalnummer sowie einen Vorgesetzten, der auch eine Servicekraft ist.
- In der Küche arbeiten Küchenmitarbeiter, die einen Namen haben und ein Gehalt bekommen.
- Küchenmitarbeiter sind entweder Köche oder Aushilfen. Köche können zudem Sterneköche sein, also mit einem oder mehreren Sternen dekoriert sein.
- Aushilfen bauen die Büffets für die Mahlzeiten auf.
- Gäste können sich unterhalten. Jeder Gast hat einen Namen und eine Adresse und ist seinem Zimmer zugeordnet.

Fortsetzung nächste Seite!

2. Objektorientierte Programmierung

Ein Versicherungsunternehmen bietet KFZ-Versicherungen mit verschiedenen Jahresbeiträgen und Selbstbeteiligungen im Schadensfall an. Lösen Sie die folgenden Aufgaben in einer objektorientierten Programmiersprache Ihrer Wahl (Sprache vorher angeben):

a) Alle Versicherungsverträge des Unternehmens sollen fortlaufende Vertragsnummern erhalten. Erstellen Sie eine Klasse Vertrag. Die eindeutigen Vertragsnummern sollen über ein Objektfeld, das im Standardkonstruktor die nächste freie Vertragsnummer zugewiesen bekommt, an ihre Unterklassen vererbt werden.

b) Erstellen Sie nun für KFZ-Versicherungsverträge eine weitere Klasse Vollkasko als Unterklasse von Vertrag mit Objektfeldern für

- den Jahresbeitrag (für 100%)
- die Schadensklasse (in Prozent)
- die Höhe der Selbstbeteiligung

Geben Sie einen Konstruktor an, der alle Objektfelder auf die übergebenen Werte setzt und implementieren Sie folgende öffentliche Methoden:

- `double berechneBeitrag()`
  gibt den zu zahlenden Jahresbeitrag entsprechend der Schadensklasse zurück.
- `double schadensregulierung(double schaden)`
  berechnet den Betrag, den die Versicherung bei einem Schadensfall abzüglich Selbstbeteiligung übernehmen muss. Falls der Schadenswert die Selbstbeteiligung überschreitet, soll die Schadensklasse um 15 Prozentpunkte erhöht werden.

c) Implementieren Sie ein kurzes Testprogramm: Deklarieren und erzeugen Sie ein Objekt der Klasse Vollkasko mit Beispielwerten, berechnen Sie dessen Jahresbeitrag und geben Sie diesen auf der Standardausgabe aus.

d) Das Unternehmen führt einen weiteren Vollkaskovertrag ein, dessen Schadensklasse erst bei jedem zweiten Schadensfall steigt. Implementieren Sie dafür eine Klasse VollkaskoPlus als Unterklasse von Vollkasko mit einem zusätzlichen Objektfeld, das als Wahrheitswert angibt, ob bereits ein Schaden gemeldet worden ist, der die Selbstbeteiligung übersteigt.
Die Klasse soll neben einem Konstruktor, der alle benötigten Objektfelder sinnvoll setzt, auch die folgende Methode enthalten:
`double schadensregulierung(double schaden)`
soll die Schadensklasse nur dann erhöhen, falls bereits ein Schaden vorlag. Kam es zu einer Erhöhung, muss der Wahrheitswert wieder zurückgesetzt werden, da der nächste Schadensfall wieder „frei" ist.

e) Erklären Sie kurz die Begriffe *Redefinieren* und *dynamisches Binden* von Methoden in einer objektorientierten Programmiersprache. Was muss beim Redefinieren von Methoden in Hinblick auf die Sichtbarkeiten beachtet werden?

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2013

RS ✓

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2013**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **11**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

1. Sie sollen für einen Online-Shop für Musikinstrumente eine Software zur Verwaltung des Inventars implementieren. Zu jedem Artikel gehört eine Nummer, eine Art (Instrument, Noten, Zubehör, ...), ein Hersteller, ein Lieferant, ein Einzelpreis, sein Lagerbestand (Anzahl der Einheiten). Dazu kommt eine Zusatzinformation, etwa ob und in welchem Umfang ein Artikel vorbestellt wurde.
   Es soll möglich sein, den Lagerbestand aufzufüllen und zu reduzieren (zum Beispiel beim Verkauf). Außerdem soll es möglich sein, Aufträge an Lieferanten zu erstellen.
   Die beteiligten Personen können Kunden, Verkäufer oder Administratoren sein; der Einfachheit halber können Verkäufer und Administratoren nicht Kunden sein.
   Beteiligte Personen können sich beim System an- und wieder abmelden.
   Kunden können Artikel kaufen und bezahlen. Außerdem können Kunden den Bestand durchsuchen und erhalten dann für jeden geführten Artikel die Mitteilung, ob er verfügbar ist oder nicht. In letzterem Falle können sie ihn vorbestellen. Kunden können aber nicht Stückzahlen oder Lieferanten einsehen.
   Verkäufer können Artikel verkaufen, den Bestand durchsuchen, verändern und Aufträge an Lieferanten tätigen. Außerdem können sie Kunden neu aufnehmen oder löschen.
   Administratoren haben alle Befugnisse von Verkäufern und können außerdem Verkäufer und Administratoren aufnehmen oder löschen.

   a) Entwickeln Sie aus der gegebenen Anwendungsbeschreibung ein UML-Klassendiagramm!

   Alle Angaben aus dem Text sollen sich in Ihrem Modell wiederfinden; weitere sollten nur dann hinzugenommen werden, wenn es aufgrund der von Ihnen gewählten Modellierung erforderlich erscheint!
   Verwenden Sie Vererbung und Interfaces, wo es sinnvoll möglich ist und achten Sie auf eine angemessene Darstellung von Assoziationen und Aggregationen, wo erforderlich. Es geht hier nur um das UML-Klassendiagramm; insbesondere ist eine Modellierung als relationale Datenbank *nicht* verlangt und auch nicht im Sinne der Aufgabenstellung!

   b) Modellieren Sie den gesamten Anwendungsfall des Kaufens eines Artikels (Anmelden, Bestand anzeigen, Aussuchen, Kaufen, Abmelden) zunächst als UML-Anwendungsfall und dann als UML-Aktivitätsdiagramm. Achten Sie auch auf die Möglichkeit des Fehlschlagens einer Teilaktivität!

2. Erläutern Sie in jeweils ca. 3 Sätzen die folgenden Begriffe aus der modernen Softwaretechnologie: Pair Programming, Refactoring, agile Softwareentwicklung.

Fortsetzung nächste Seite!

Teilaufgabe 2:

I. Datenbanksysteme

1. Modellierung

Sie sollen ein System zur Verwaltung von Pferderennen entwerfen.
Gehen Sie dabei von folgendem Szenario aus:

- Unternehmen werden über ihre eindeutige Unternehmens-Id identifiziert. Sie haben eine Adresse und besitzen Rennställe.
- Der Name eines Rennstalls ist nur innerhalb eines Unternehmens eindeutig. Für jeden Rennstall wird das Gründungsdatum gespeichert.
- Pferde gehören immer zu einem Rennstall. Pferdenamen werden in einem Rennstall nur jeweils maximal einmal vergeben.
- Jockeys sind in einem Rennstall beschäftigt. Jeder Rennstall vergibt seine eigenen Personalnummern. Für jeden Jockey werden sein Vorname und Name gespeichert.
- Rennen haben ein Datum, ein Preisgeld und einen Namen, über den sie identifiziert werden.
- Unternehmen unterstützen Rennen finanziell mit einem bestimmten Betrag.
- Jockeys nehmen mit Pferden an Rennen teil. Im Rennen erreichen sie einen bestimmten Platz. Die Kombination aus Jockey und Pferd ist nicht fest, bei unterschiedlichen Rennen können Jockeys verschiedene Pferde reiten. Jockeys können auch mit Rennpferden von fremden Rennställen, die anderen Unternehmen gehören können, an Rennen teilnehmen.

a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell.
Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- ein passendes ER-Diagramm,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Fortsetzung nächste Seite!

2. Anfragen

Zu einer Website, auf der Besucher im Browser Online-Spiele spielen können, liegt das folgende relationale Schema einer Datenbank vor:

$$\begin{aligned} \text{Team} &: \{[\underline{\text{TNr}}, \text{Name}, \text{Teamfarbe}]\} \\ \text{Spieler} &: \{[\underline{\text{SNr}}, \text{Name}, \text{Icon}, \text{TNr}, \text{EMail}]\} \\ \text{Minispiel} &: \{[\underline{\text{MNr}}, \text{Name}, \text{Kategorie}, \text{Schwierigkeit}]\} \\ \text{Wettkampf} &: \{[\underline{\text{WNr}}, \text{Sieger}, \text{Geschlagener}, \text{MNr}, \text{Dauer}]\} \end{aligned}$$

Auf der Website treten jeweils zwei Spieler gegeneinander in Minispielen an. In diesen ist es das Ziel, den gegnerischen Spieler in möglichst kurzer Zeit zu besiegen. Minispiele gibt es dabei in verschiedenen Schwierigkeitsstufen ('leicht', 'mittel', 'schwer', 'sehr schwer') und verschiedenen Kategorien ('Denkspiel', 'Geschicklichkeitsspiel', usw.). Die Attribute *Sieger* und *Geschlagener* sind jeweils Fremdschlüsselattribute, die auf das Attribut *SNr* der Relation *Spieler* verweisen. Beachten Sie, dass das *Dauer*-Attribut der Wettkampf-Relation die Dauer eines Wettkampfes in der Einheit Sekunden speichert.

a) Formulieren Sie geeignete Anfragen in relationaler Algebra für die folgenden Teilaufgaben:

   i. Geben Sie die Namen der Minispiele zurück, die zur Kategorie 'Denkspiele' zählen.

   ii. Geben Sie die Namen und E-Mail-Adressen aller Spieler zurück, die in einem Minispiel des Typs 'Geschicklichkeitsspiel' gewonnen haben.

b) Formulieren Sie geeignete SQL-Anfragen für die folgenden Teilaufgaben. Beachten Sie dabei, dass Ihre Ergebnisrelationen keine Duplikate enthalten sollen.

   i. Geben Sie jede Spielekategorie aus, für die ein Minispiel der Schwierigkeitsstufe 'sehr schwer' vorhanden ist.

   ii. Geben Sie die Wettkämpfe aus, deren Dauer unter der durchschnittlichen Dauer der Wettkämpfe liegt.

   iii. Geben Sie für jeden Spieler seine *SNr*, seinen Namen, die Anzahl seiner Siege, die durchschnittliche Dauer seiner siegreichen Wettkämpfe und die Anzahl der Teams, aus denen er bereits mindestens einen Spieler besiegt hat, zurück.

c) Verwenden Sie den relationalen Tupelkalkül, um die folgenden Anfragen zu formulieren:

   i. Finden Sie die Namen der Spieler des Teams 'Dream Team'.

   ii. Geben Sie die Namen der Minispiele zurück, bei denen bereits Spieler gegeneinander angetreten sind, deren Teams dieselbe Teamfarbe haben.

Hinweis: Die Anfragen im relationalen Tupelkalkül dürfen auch nicht sicher sein.

Fortsetzung nächste Seite!

3. Entwurfstheorie

Gegeben ist der folgende Ausschnitt einer relationalen Datenbank:

Vereine							
UserId	ClubId	Vorname	Name	Login	ClubName	Funktion	Vorstand
1	1	Max	Muster	maxm	IT Kicker	Trainer	false
1	2	Max	Muster	maxm	CC Nerds	2. Vorsitzende(r)	true
2	2	Marta	Maier	maier1	CC Nerds	1. Vorsitzende(r)	true
3	1	Tim	Peters	timpeters	IT Kicker	1. Vorsitzende(r)	true
4	1	Peter	Huber	timpeters	IT Kicker	Spieler(in)	false
5	3	Tina	Müller	tinam	IT Club	Kassenprüfer(in)	true
...	...	...	...	...	...	...	...

In der Datenbank sollen folgende funktionale Abhängigkeiten gelten:

$UserId \rightarrow Vorname, Name, Login$
$Login \rightarrow Vorname, Name, UserId$
$UserId, ClubId, Name \rightarrow Funktion, ClubName$
$Login, ClubId, Name \rightarrow Funktion, ClubName$
$ClubId \rightarrow ClubName$
$Funktion \rightarrow Vorstand$

a) Geben Sie sämtliche Schlüsselkandidaten an.

b) Welcher Normalform genügt die Relation *Vereine*? Begründen Sie Ihre Antwort ausführlich.

c) Verwenden Sie den Synthesealgorithmus, um die Relation in die dritte Normalform zu überführen.

d) Befinden sich die neuen Relationen in Boyce-Codd-Normalform? Begründen Sie Ihre Antwort.

Thema Nr. 2

Teilaufgabe 1:

Aufgabe 1:

Gegeben sei folgendes Klassendiagramm:

a) Implementieren Sie die Klassen „Order", „Payment", „Check", „OrderDetail" und „Item" in einer geeigneten objektorientierten Sprache Ihrer Wahl. Beachten Sie dabei insbesondere Sichtbarkeiten, Klassen- vs. Instanzzugehörigkeiten und Schnittstellen bzw. abstrakte Klassen.

b) Erstellen Sie ein Sequenzdiagramm (mit konkreten Methodenaufrufen und Nummerierung der Nachrichten) für folgendes Szenario:
   i. „Erika Mustermann" aus „Rathausstraße 1, 10178 Berlin" wird als neue Kundin angelegt.
   ii. Frau Mustermann bestellt heute 1kg Gurken und 2kg Kartoffeln.
   iii. Sie bezahlt mit ihrer Visa-Karte, die im August 2014 abläuft und die Nummer „1234 567891 23456" hat – die Karte erweist sich bei der Prüfung als gültig.
   iv. Am Schluss möchte sie noch wissen, wie viel ihre Bestellung kostet – dabei wird der Anteil der Mehrwertsteuer extra ausgewiesen.

Fortsetzung nächste Seite!

Aufgabe 2:

Rekonstruieren Sie aus dem folgenden Java-Code das zugehörige UML-Klassendiagramm. Dabei sind folgende Randbedingungen zu beachten:

- Sichtbarkeiten von Feldern (Attributen u. Klassenvariablen), Methoden und Assoziationsrollen sind entspr. anzugeben – die der Klassen sind hingegen irrelevant.
- Methodenrümpfe und Initialwerte von Feldern können, müssen aber nicht angegeben werden.
- Abstrakte Klassen und Methoden sind mit einem ★ hinter dem Namen deutlich zu kennzeichnen.
- Assoziationen zwischen den gegebenen Klassen sind zusammen mit Rollennamen, Richtung der Navigierbarkeit und Multiplizitäten (graphisch im Diagramm) anzugeben – eine Unterscheidung in Aggregation und Komposition ist hingegen nicht erforderlich.

```
public interface A {
      public static int aa = 4711;
      public double ab = 0.815;

      public void ac();
}

abstract class B implements A {
      private static void ba() { }

      public abstract String bb(int x);
}

class C extends B {
      public A ca;
      private D cb;

      @Override
      public void ac() { }

      @Override
      public String bb(int x) {bb(""); return "";}

      private boolean bb(String y) {cb.ac(); return true;}
}

class D extends B {
      private C[] da;

      public void ac() { }

      public String bb(int x) {db(0, 0); return da.toString();}

      private void db(long a, int b) {}
}
```

Fortsetzung nächste Seite!

Teilaufgabe 2:

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

Für die Verwaltung eines Zoos soll folgendes System entworfen werden:

- Die zentrale Entität eines Zoos ist ein Gehege. Ein Gehege hat eine eindeutige Nummer und einen Namen und kann eingeteilt werden in die Unterkategorien offenes Gehege, Streichelgehege oder Aquarium.
- Zusätzliche Attribute für spezielle Gehege sind Größe in Quadratmetern (offenes Gehege), Anzahl der Futterautomaten und maximal eingelassene Anzahl von Besuchern (Streichelgehege) sowie die Anzahl von Räumen und Becken (Aquarium). Für offene Gehege soll zusätzlich modelliert werden können, ob sich darin ein Gebäude für die Tiere befindet.
- In jedem Gehege des Zoos lebt mindestens eine Tierart. Eine Tierart ist eindeutig durch eine Bezeichnung beschrieben und hat zudem eine lateinische Bezeichnung. Eine Tierart lebt in genau einem Gehege des Zoos.
- Jede Tierart gehört genau einer bestimmten Klasse von Lebewesen an (z.B. Säugetiere). Diese Klasse besitzt ebenso eine eindeutige Bezeichnung sowie einen lateinischen Namen.
- Für eine später mögliche Neugestaltung der Gehege werden Informationen darüber benötigt, welche Tierarten sich untereinander vertragen.
- Für jede im Zoo lebende Tierart gibt es mindestens ein Tier, welches eine eindeutige Nummer hat. Zusätzlich sollen für jedes Tier der Name und das Geburtsdatum aufgezeichnet werden.
- Jede Tierart im Zoo wird durch mindestens einen Tierpfleger betreut. Ein Tierpfleger hat eine eindeutige Mitarbeiternummer sowie einen Namen und einen Vornamen. Jeder Tierpfleger betreut nur genau eine Tierart.
- Im Zoo werden wöchentlich für einige Tierarten öffentliche Fütterungen angeboten. Dies erfolgt für mindestens eine Tierart durch einen oder mehrere Tierpfleger. Zusätzliche Infos sind hier der Wochentag und die Uhrzeit.

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). Nummerische Attribute sollen geeigneten Integritätsbedingungen unterliegen. Geben Sie außerdem weitere sinnvolle Constraints an.
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 Musikalben. Die Primärschlüssel-Attribute sind jeweils unterstrichen, Fremdschlüssel sind überstrichen.

"Interpret":

IID	Name	Vorname	Künstlername	Geburtsstadt
I1	Germanotta	Stefani	Lady Gaga	New York
I2	Vetter	Jan	Farin Urlaub	Berlin
I3	Würdig	Paul	Sido	Berlin
I4	Köhler	Horst	Guildo Horn	Trier

"Komponist":

KID	Name	Vorname
K1	Raab	Stefan
K2	Khayat	Nadir
K3	Beardie	Roe
K4	Vetter	Jan

"Album":

AID	Albumtitel	Erscheinungsdatum
A1	The Fame	19.07.2008
A2	Maske	10.07.2004
A3	Endlich Urlaub!	01.07.2001

"Lied":

TID	Komponist	Interpret	Album	Titel	BesteChartplazierung
T1	K1	I4	null	Guildo hat euch lieb	5
T2	K2	I1	A1	Pokerface	1
T3	K2	I1	A1	Paparazzi	2
T4	K3	I3	A2	Fuffies im Club	18
T5	K4	I2	A3	Jeden Tag Sonntag	58

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

1. Finden Sie die Namen aller Interpreten, welche in Berlin geboren sind.
2. Suchen Sie alle Komponisten, die Lieder für "Lady Gaga" geschrieben haben.
3. Finden Sie die Titel aller Alben mit mindestens einem Lied, dessen Interpret und Komponist den gleichen Vornamen und Nachnamen haben.
4. Finden Sie alle Komponisten, welche einen Titel komponiert haben, der vor dem 01.01.2000 auf einem Album erschienen ist.

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

5. $\pi_{KID,AID}(Komponist \times Album)$
6. $\pi_{Album,Titel,BesteChartplazierung}(Lied) \bowtie_{Lied.Album=Album.AID} \pi_{AID,Erscheinungsdatum}(Album)$

Fortsetzung nächste Seite!

Aufgabe 3: SQL

Gegeben sei das relationale Datenbank-Schema aus Aufgabe 2.

"Interpret":

IID	Name	Vorname	Künstlername	Geburtsstadt
I1	Germanotta	Stefani	Lady Gaga	New York
I2	Vetter	Jan	Farin Urlaub	Berlin
I3	Würdig	Paul	Sido	Berlin
I4	Köhler	Horst	Guildo Horn	Trier

"Komponist":

KID	Name	Vorname
K1	Raab	Stefan
K2	Khayat	Nadir
K3	Beardie	Roe
K4	Vetter	Jan

"Album":

AID	Albumtitel	Erscheinungsdatum
A1	The Fame	19.07.2008
A2	Maske	10.07.2004
A3	Endlich Urlaub!	01.07.2001

"Lied":

TID	Komponist	Interpret	Album	Titel	BesteChartplazierung
T1	K1	I4	null	Guildo hat euch lieb	5
T2	K2	I1	A1	Pokerface	1
T3	K2	I1	A1	Paparazzi	2
T4	K3	I3	A2	Fuffies im Club	18
T5	K4	I2	A3	Jeden Tag Sonntag	58

Formulieren Sie die folgenden Anfragen in SQL:

1. Finden Sie die Titel aller Lieder, welche irgendwann unter den Top 10 der Charts waren.
2. Geben Sie die Titel aller Lieder und, falls sie auf einem Album erschienen sind, den entsprechenden Albumtitel aus.
3. Finden Sie die Künstlernamen aller Interpreten, welche mehr als 10 Titel in den Top 10 der Charts hatten.
4. Finden Sie die ID des Komponisten, der für die meisten verschiedenen Künstler Lieder geschrieben hat.

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

5. `select Komponist, Min(BesteChartplazierung) from Lied group by Komponist;`
6. `Select * from Interpret where IID not in (select Interpret from Lied, Album where Album = AID and Erscheinungsdatum > '01.06.2002');`

Fortsetzung nächste Seite!

Aufgabe 4: Entwurfstheorie

Gegeben sei die nachfolgende relationale Datenbank mit unterstrichenen Schlüsselattributen. Sie enthält Informationen über Zugverbindungen. Relation „Zugverbindungen“:

Zugtyp	Start	Start Gleis	Ziel	ZielGleis	Bordbistro	Halt	HaltGleis
ICE	Nürnberg	9	München	22	ja	Ingolstadt	4
IC	Hamburg	7	Berlin	1	ja	Berlin-Spandau	5
RE	Nürnberg	14	München	26	nein	Ingolstadt	2
IC	Hamburg	7	Berlin	1	ja	Wittenberge	3
ICE	München	22	Essen	4	ja	Frankfurt	7
RE	Nürnberg	14	München	26	nein	Pfaffenhofen	1
ICE	Hamburg	14	München	15	ja	Hannover	4
ICE	Hamburg	8	Berlin	11	ja	Berlin-Spandau	5
TGV	Stuttgart	9	Paris	3	ja	Karlsruhe	6

Sowie die folgenden funktionalen Abhängigkeiten:

FD1: Zugtyp —> Bordbistro
FD2: Zugtyp, Start, Ziel —> StartGleis, ZielGleis
FD3: Zugtyp, Start, Ziel, Halt —> HaltGleis

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. Welcher Normalform genügt das Relationenschema dieser Datenbank? Begründen Sie Ihre Entscheidung!
3. Überführen Sie das Relationenschema der Datenbank in die dritte Normalform und geben Sie die mit obigen Daten gefüllten Relationen an.
4. Erläutern Sie kurz, welchen Nachteil Normalisierung allgemein für die Anfragebearbeitung haben kann.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2014

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: *Kunden 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!

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Herbst 2014

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2014**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **11**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

Aufgabe 1: „OOA/OOD"

Betrachten Sie folgende Spezifikation einer Stammbaum-App:

Ein Stammbaum besteht aus Personen, die Verbindungen miteinander haben. Man kann einen Stammbaum laden oder speichern, wofür man jeweils auch den Pfad als Zeichenkette angibt - die Methode „speichern" gibt zurück, ob der Stammbaum erfolgreich gespeichert werden konnte. Dem Stammbaum kann man beliebig viele Personen oder Verbindungen hinzufügen; ebenso kann man eingetragene Personen bzw. Verbindungen auch wieder löschen. Eine Person bzw. eine Verbindung kann nur in genau einem einzigen Stammbaum vorkommen.
Beim Anlegen einer Person können Vorname, Nachname, Geburtsdatum und optional (sofern überhaupt schon bekannt) auch das Sterbedatum angegeben werden. Das Sterbedatum kann nachträglich gesetzt werden, wobei ein evtl. schon eingetragenes Datum zurückgegeben wird. Einer Person wird zunächst genau ein vorgegebenes Bild zugeordnet, das aber jederzeit geändert werden kann. Die Angaben zur Person sind privat.
Zwischen genau zwei Personen kann eine der drei Beziehungen bestehen: Verliebt, Verlobt, Verheiratet. Eine abstrakte Beziehung kann es im Stammbaum zwar nicht geben, aber zu jeder der drei genannten Beziehungen kann man jeweils das Datum angeben, von wann und bis wann sie gehalten hat (muss man aber nicht eintragen!) und welches Personen-Paar darin verwickelt war. Selbstverständlich kann jede Person beliebig viele Beziehungen gleichzeitig haben. Bei Verliebten kann man noch angeben, ob beide glücklich sind (könnte ja sein, dass Romeo zwar Julia liebt, aber bei Julia bereits ein Sterbedatum eingetragen werden konnte...). Verliebte können sich trennen, eine Verlobung kann man auflösen und ist man erst verheiratet, dann lässt man sich ggf. auch mal scheiden - wir sagen dazu: eine Beziehung kann man kaputt machen, wozu dann aber das Datum angegeben werden muss, bis zu dem die Beziehung gehalten hat.
Eine ganz andere Verbindung stellt ein Kindschaftsverhältnis dar. Dazu gehören stets eine Mutter und ein Kind, allerdings kann der tatsächliche Vater nicht immer zugeordnet werden (der Kerl ist dann ja wohl „eine ziemliche Null"). Auch werden Zeugungsdatum und Zeugungsort möglicherweise erst später bekannt, so dass man sie nachträglich gemeinsam setzen kann. Eine Person kann offensichtlich beliebig viele öffentlich bekannte Kindschaftsverhältnisse haben und auch die Mutter-/Vater-/Kind-Rolle wird publik gemacht.

Erstellen Sie aus der obigen „Spezifikation" ein UML-Klassendiagramm mit allen Klassen, Vererbungsrelationen, Attributen, Methoden (auch Konstruktoren) und Assoziationen. Tragen Sie bei Assoziationen stets auch die Multiplizitäten und Navigationspfeile ein. Falls Rollennamen bekannt sind, so müssen diese ebenfalls angegeben werden. Achten Sie ferner auf die Sichtbarkeiten von Methoden und Attributen.
Als UML-Klassen explizit (graphisch) modellierte Typen sollen nicht als Attribute im Klassenkästchen erscheinen. Nehmen Sie hierbei aber an, dass `Datum` und `Pixel` sowie `String` vorgegebene („primitive") Datentypen sind, die nicht mehr als explizite UML-Klasse eingezeichnet werden müssen. Verwenden Sie *möglichst restriktive,* aber sinnvolle Sichtbarkeiten, sofern der obige Text nichts Genaueres spezifiziert.

Fortsetzung nächste Seite!

Aufgabe 2: „Abstrakte Datentypen (ADT)“

Gegeben sei folgender Ansatz eines abstrakten Datentyps (ADT) *SortedSet*, der eine Menge von Objekten eines generischen Typs `T` verwaltet. Auf `T` ist außerdem eine Ordnung „$\leq$ “ sowie die Relation „= “ mit der üblichen Semantik definiert. Das spezielle Element `nil` ist (ähnlich der `null`-Referenz in Java) ein Hilfskonstrukt und soll als Element vom Typ `T` verstanden werden – berücksichtigen Sie diesen Sonderfall in Ihren Axiomen, sodass `nil` nicht zur Menge hinzugefügt werden kann.

adt *SortedSet*
sorts *SortedSet, T, int, boolean*
ops

create:		$\rightarrow$ *SortedSet*	// Konstruktor
add:	*SortedSet* $\times$ *T*	$\rightarrow$ *SortedSet*	
...			

axs
...
end *SortedSet*

a) Ergänzen Sie den vorgegebenen Ausschnitt der algebraischen Spezifikation des ADTs `SortedSet` um die notwendigen Signaturen und Axiome folgender Operationen:

   i) `remove`: entfernt ein als Argument übergebenes Element aus der Menge, sofern darin vorhanden – andernfalls bleibt die Menge unverändert

   ii) `contains`: prüft, ob ein als Argument übergebenes Element in der Menge enthalten ist und gibt einen Wahrheitswert zurück

   iii) `size`: gibt die Anzahl der in der Menge enthaltenen Elemente zurück

Vereinfachen Sie folgenden Ausdruck schrittweise soweit wie möglich (also bringen Sie ihn in die kanonische Normalform) und geben Sie jeweils an, welches Ihrer Axiome Sie zur Transformation angewandt haben:

$$contains(add(add(remove(add(add(create, 42), 42), 42), 666), 42), 42)$$

b) Ergänzen Sie Signatur und Axiome, um folgende Operationen zu modellieren:

   i) `front`: gibt das kleinste Element der Menge zurück; ist die Menge leer, soll `nil` zurückgegeben werden

   ii) **pop**: entfernt das kleinste Element der Menge, sofern eines vorhanden ist

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1: Modellierung

Zur digitalen Verwaltung deutscher Handballvereine soll eine zweiteilige Datenbank eingerichtet werden. Der erste Teil soll die Mannschaften, Spieler und Trainer beinhalten und speichert dafür folgende Informationen:

- Für eine Mannschaft wird der Name und die Stadt, zu der diese gehört, gespeichert.
- Logos werden durch ihre Form beschrieben. Jede Mannschaft besitzt genau ein Logo.
- Ein Spieler besitzt Informationen über seine Spieler-ID, seinen Vor- und Nachnamen. Eine Mannschaft hat eine beliebige Anzahl an Spielern. Umgekehrt kann ein Spieler aber nur in einer Mannschaft spielen.
- Trainer besitzen eine Funktion (z.B. Trainer, Co-Trainer, Konditionstrainer, ...), sowie Vor- und Nachnamen. Ein Trainer trainiert genau eine Mannschaft und soll in seiner Rolle nur in Zusammenhang mit der zugehörigen Mannschaft bestehen und eindeutig identifiziert werden können. Eine Mannschaft kann mehrere Trainer besitzen.

Im zweiten Teil der Datenbank werden Trainingsdaten erfasst. Diese soll folgende Daten umfassen:

- Ein Trainingsblock wird durch seinen zugehörigen Namen bestimmt. Dieser Block hat mehrere einzelne Trainingseinheiten als Teile. Eine Trainingseinheit wird durch eine Einheits-ID identifiziert.
- Trainingseinheiten unterteilen sich in Konditions- und Spielzugtraining. Spielzugtraining kann auf einer oder mehreren anderen Spielzugtrainingseinheiten aufbauen.
- Zu einer Halle wird deren Name und die Stadt, in der sie sich befindet, gespeichert.
- Eine Trainingseinheit wird von einem Trainer in einer Halle gehalten.

a) Entwerfen Sie für das Szenario des **ersten** Teils der Datenbank ein ER-Diagramm in Chen-Notation. Bestimmen Sie hierzu:
   - die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
   - die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
   - die Funktionalitäten der Relationship-Typen.

b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

c) Übernehmen Sie den Entity-Typen für einen Trainer aus Aufgabenteil a) (Speichern Sie hier als Primärschlüssel eine ID) und entwerfen Sie ausgehend davon ein weiteres ER-Modell für den **zweiten** Teil der Datenbank. Bestimmen Sie dazu ebenfalls dieselben Kriterien wie in Aufgabenteil a).

Fortsetzung nächste Seite!

Aufgabe 2: SQL-Anfragen

Gegeben sei folgendes ER-Modell, welches eine Datenbank für die App *TravelGuide 2.0* beschreibt. Darin sind Hotels, Sehenswürdigkeiten, Restaurants sowie deren Adressen enthalten. Der Einfachheit halber können Sie davon ausgehen, dass ein Hotel, eine Sehenswürdigkeit oder ein Restaurant exakt eine Adresse besitzt. Des Weiteren bieten Hotels Fahrten zu verschiedenen Sehenswürdigkeiten an.

Gehen Sie von folgendem relationalen Schema aus:

Hotel {[Name, Adresse_ID, Sterne, Beschreibung]}
Sehenswürdigkeit {[ID, Name, Beschreibung, Eintrittspreis]}
Adresse {[ID, Land, Stadt, Strasse, HausNr]}
Restaurant {[Name, Adresse_ID, Typ, Öffnungszeit, Gault-Millau-Pkt]}
Bietet Fahrt An {[Hotel Name, Adresse ID, Sehenswürdigkeit ID]}

Formulieren Sie folgende Anfragen in SQL. Achten Sie falls nötig auf Duplikat-freie Ausgaben:

a) Geben Sie die Beschreibung und die Sterne-Anzahl aller Hotels, die den Namen „Hilton" tragen, aus.

b) Geben Sie die Namen aller italienischen Restaurants der Stadt München aus.

c) Geben Sie die Namen aller 4-Sterne Hotels in Berlin aus, die eine Fahrt zur Sehenswürdigkeit „Schloss Sans Souci" anbieten.

d) Geben Sie den Namen der Stadt mit den besten Restaurants, gemessen am Durchschnitt ihrer Gault-Millau-Punkte, aus.

e) Geben Sie die Straßennamen und die dazugehörigen Städte aus, an denen mehr als ein Hotel angesiedelt sind.

Fortsetzung nächste Seite!

Aufgabe 3: Normalformen

Eine IT-Beratungsfirma stellt zwei Unternehmen vorübergehend Experten zur Verfügung, die an gemeinsamen IT-Projekten arbeiten. Die folgende Tabelle (Relation S) listet die Unternehmen sowie die Zeit, die jeder Experte bei ihnen verbringt, auf. Außerdem ist jeder Experte durch eine **eindeutige** Nummer (Attribut *Experte ID*) identifiziert:

Experte ID	**Vertrag Nr**	**Stunden**	**Expertenname**	**Unternehmen ID**	**Unternehmensort**
E12345	V100	36	M. Schneider	U810	München
E6789	V100	60	K. Schmitt	U810	München
E2468	V500	40	W. Mayer	U920	Berlin
E12345	V500	40	M. Schneider	U920	Berlin

a) Bestimmen Sie alle funktionalen Abhängigkeiten von S.

b) Bestimmen Sie alle Kandidatenschlüssel von S.

c) Welcher Normalform genügt die Relation S (1NF, 2NF, 3NF, BCNF)? Begründen Sie Ihre Aussagen!

d) Bestimmen Sie eine Zerlegung von S in 3NF. Verwenden Sie hierfür den Synthesealgorithmus.

e) Erstellen Sie aus dem 3NF der Relation S das entsprechende Relationenschema. Bestimmen Sie für jede Tabelle den **Primärschlüssel** und, wenn nötig, **Fremdschlüssel** mit den entsprechenden Verknüpfungen (markieren Sie Fremdschlüssel-Beziehungen mit einem *).

Aufgabe 4: Transaktionen

Gegeben seien die beiden folgenden, aus elementaren Operationen der Transaktionen T_2, T_2 und T_3 bestehenden Schedules:

$S_1 = r_1(a)\ r_2(b)\ r_3(a)\ w_3(a)\ r_1(b)\ w_1(b)\ r_2(c)\ r_1(c)\ r_2(d)\ r_1(d)\ w_1(d)\ c_1\ w_3(c)\ c_2\ c_3$

$S_2 = r_1(c)\ r_2(b)\ w_1(a)\ w_2(b)\ r_1(b)\ r_2(c)\ w_1(b)\ w_2(c)\ c_2\ c_1$

a) Geben Sie den Konfliktgraphen für S_1 und S_2 an. Wie kommen die Knoten und Kanten zustande?

b) Sind die beiden Schedules S_1 und S_2 serialisierbar? Begründen Sie Ihre Antwort kurz. Falls S_1 und/oder S_2 serialisierbar sind, geben Sie jeweils einen äquivalenten seriellen Plan an.

c) Benennen und erläutern Sie die drei klassischen Synchronisationsprobleme.

Thema Nr. 2

Teilaufgabe 1:

Aufgabe 1: ***Allgemeine SWT, Vorgehensmodelle und Requirements Engineering***

Kreuzen Sie für die folgenden Multiple-Choice-Fragen genau die richtigen Antworten deutlich an. Es kann mehr als eine Antwort richtig sein.
Jedes korrekt gesetzte oder korrekt nicht gesetzte Kreuz wird mit 1 Punkt gewertet. Jedes falsch gesetzte oder falsch nicht gesetzte Kreuz wird mit -1 Punkt gewertet. Eine Frage kann entwertet werden, dann wird sie nicht in der Korrektur berücksichtigt. Einzelne Antworten können nicht entwertet werden. Entwerten Sie eine Frage wie folgt:

	Fragen
1.	Frage 1 ☐ Antwort 1 ☐ Antwort 2

Die gesamte Aufgabe wird nicht mit weniger als 0 Punkten gewertet.

	Fragen
1.	Welche Aussage ist wahr? ☐ Je früher ein Fehler entdeckt wird, umso teurer ist seine Korrektur. ☐ Je später ein Fehler entdeckt wird, umso teurer ist seine Korrektur. ☐ Der Zeitpunkt der Entdeckung hat keinen Einfluss auf die Kosten.
2.	Mit welcher Methodik können Funktionen spezifiziert werden? ☐ Als Funktionsvereinbarung in einer Programmiersprache ☐ Mit den Vor- und Nachbedingungen von Kontrakten ☐ Als Zustandsautomaten
3.	Welche Vorgehensmodelle sind für Projekte mit häufigen Änderungen geeignet? ☐ Extreme Programming (XP) ☐ Das V-Modell 97 ☐ Scrum
4.	Welche der folgenden Aussagen ist korrekt? ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren. ☐ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse. ☐ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.
5.	Welche der folgenden Aussagen ist korrekt? ☐ Bei der Architektur sollten funktionale und nicht-funktionale Anforderungen beachtet werden. ☐ Bei der Architektur sollten nur funktionale Anforderungen beachtet werden. ☐ Bei der Architektur sollten nur nicht-funktionale Anforderungen beachtet werden. ☐ Bei der Architektur sollte auf die mögliche Änderungen von Komponenten geachtet werden.

Fortsetzung nächste Seite!

Aufgabe 2: Vorgehensmodelle

Software wird oft in definierten Prozessen entwickelt. Diese nennt man Vorgehensmodelle.

Allgemein

a) Was sind die Aufgaben eines Vorgehensmodells im Allgemeinen?

b) Was sind die wesentlichen Bestandteile eines Vorgehensmodells und in welcher Beziehung stehen diese zueinander?

Ein frühes Vorgehensmodell ist das von Dr. Winston Royce 1970 formalisierte Wasserfallmodell.

c) Geben Sie eine schematische Darstellung des Wasserfallmodells an.

d) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

e) In welchen Situationen lässt sich das Wasserfallmodell gut einsetzen?

Barry Boehm erweiterte das Wasserfallmodell 1979 zum so genannten V-Modell.

f) Geben Sie eine schematische Darstellung des V-Modells an.

g) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

h) Welchen Vorteil hat das V-Modell gegenüber dem Wasserfallmodell?

In neuerer Zeit finden immer häufiger iterative und inkrementelle Vorgehensweisen Anwendung.

i) Erklären Sie den Begriff iterative Softwareentwicklung.

j) Erklären Sie den Begriff inkrementelle Softwareentwicklung und grenzen Sie ihn von iterativer Softwareentwicklung ab.

k) Nennen Sie jeweils zwei Vor- und Nachteile eines iterativen und inkrementellen Vorgehens im Vergleich zum Wasserfallmodell.

Fortsetzung nächste Seite!

Aufgabe 3: UML Diagramme in der Anwendung

Gegeben sei folgender Sachverhalt:

> Für eine Verwaltungssoftware einer Behörde soll ein Bestellsystem entwickelt werden. Dabei sollen die Nutzer ihre Raummaße eingeben können. Anschließend können die Nutzer über ein Web-Interface das Büro gestalten und Möbel (wie zum Beispiel Wandschränke) und andere Einrichtungsgegenstände in einem virtuellen Büro platzieren. Aus dem Web-Interface kann die Einrichtung dann direkt bestellt werden. Dazu müssen die Nutzer ihre Büro-Nummer und den Namen und die Adresse der Behörde eingeben und die Bestellung bestätigen.
>
> Weiterhin können Nutzer auch Büromaterialien über das Web-Interface bestellen. Dazu ist anstatt der Eingabe der Raummaße nur das Eingeben von Büro-Nummer und des Namens und der Adresse der Behörde erforderlich.
>
> Zusätzlich zum Standard-Nutzer können sich auch Administratoren im System anmelden und Möbel zur Kollektion hinzufügen und aus der Kollektion entfernen. Die Möbel können eindeutig durch ihre Inventurnummer identifiziert werden.
>
> Um jegliche Veränderungen im System protokollieren zu können müssen Nutzer und Administratoren zur Bestätigung eingeloggt sein.

a) Erfassen Sie die drei Systemfunktionen *Möbel bestellen*, *Login* und *Kollektion verwalten* in einem UML-konformen Use Case Diagramm.

b) Erstellen Sie ein UML-Klassendiagramm, welches die Beziehungen und sinnvolle Attribute der Klassen „Nutzer, Büro, Möbelstück und Wandschrank" darstellt.

c) Instanziieren Sie das Klassendiagramm in einem Objektdiagramm mit den zwei Nutzern mit Namen *Ernie* und *Bernd*, einem Büro mit der Nummer *CAPITOL2* und zwei Schränken mit den Inventurnummern *S1.88* und *S1.77*.

d) Geben Sie für ein Büromöbelstück ein Zustandsdiagramm an. Überlegen Sie dazu, welche möglichen Zustände ein Möbelstück während des Bestellvorgangs haben kann und finden Sie geeignete Zustandsübergänge.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

a) Erstellen Sie das konzeptuelle Modell eines (einfachen) Bibliotheksverwaltungssystems (z. B. für Ihre Schule) als Entity-Relationship-Schema. Es sollen insbesondere folgende Aspekte modelliert werden:

- Bücher können in mehreren Exemplaren vorhanden sein, die über die Signaturen eindeutig identifiziert werden.
- Schüler/innen können Bücher reservieren und ausleihen.
- Lehrer unterrichten Klassen in bestimmten Fächern und empfehlen dafür Bücher.
- Lehrer haben die Möglichkeit Bücher solange auszuleihen, bis eine Reservierung erfolgt. Daraufhin wird für sie und das entliehene Buch ein Mahneintrag erstellt.
- Die Bibliothek wird von einem Lehrer geleitet.

Erstellen Sie Ihr ER-Schema so, dass jedem Gegenstandstyp mindestens 3 Attribute zugeordnet sind. Spezifizieren Sie die Funktionalität / Kardinalität der Beziehungstypen.

b) Setzen Sie Ihr ER-Schema systematisch in ein relationales Schema ein. Wenn möglich sollten Relationen zusammengefasst werden, wenn dadurch keine Redundanzen / Anomalien auftreten. Markieren Sie die Schlüssel.

c) Für Ihr Bibliotheksverwaltungssystem formulieren Sie folgende SQL-Anfragen:

- Welche Lehrer haben wie viele Bücher ausgeliehen?
- Für welches Buch liegen die meisten Reservierungen vor?
- Welche/r Lehrer/in empfehlt für welche Klasse und welches Fach ein Buch, dessen Autor denselben Nachnamen hat wie der Lehrer.

Aufgabe 2:

Normalisieren Sie folgendes relationale Schema:

Allerlei: {[A, F, V, T, P, M]}

mit den funktionalen Abhängigkeiten

$A \rightarrow F$
$V \rightarrow T$
$P \rightarrow M$
$V \rightarrow P$
$A \rightarrow P$

Fortsetzung nächste Seite!

Aufgabe 3:

Zeigen Sie schrittweise (aber nicht jeden Einfügevorgang – nur vor und nach dem Überlauf) den Aufbau eines B*-Baums mit der maximalen Knotenkapazität von 4 Einträgen (die sowohl für innere Knoten als auch für die Blätter gilt). Die Einträge sind in folgender Reihenfolge einzufügen.

130, 120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10

Wie sieht eine (statische) Hashtabelle für die obigen Einträge aus?

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2015

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr**	**46116**
Kennwort: _____	**2015**	
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologhie/Datenbanksysteme**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 11

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1: Softwaretechnologie

1. **Objektorientierung**

   a) Eine Volkshochschule bietet verschiedene Kurse an. Implementieren Sie in einer objektorientierten Programmiersprache Ihrer Wahl eine Klasse `Kurs`, wobei eine Instanz der Klasse folgende Eigenschaften besitzen soll:

   - eine eindeutige Kursnummer
   - das Thema des Kurses (z.B. Spanisch, Yoga)
   - die Höhe der Kursgebühr
   - die maximale Teilnehmerzahl
   - die Anzahl der gemeldeten Teilnehmer

   Geben Sie auch einen geeigneten Konstruktor an, der auch die eindeutige Kursnummer festlegt. Verwenden Sie dazu einen geeigneten Zähler für die nächste zu vergebende Nummer.

   b) Schreiben Sie eine Methode, die die Auslastung des Kurses, d.h. das Verhältnis zwischen der gemeldeten und möglichen Teilnehmerzahl, berechnet.

   c) Implementieren Sie für die Klasse `Kurs` eine Methode zur Kursbuchung, die als Übergabeparameter die Anzahl der zu buchenden Kursplätze übergeben bekommen soll. Die Methode soll vor der Buchung zunächst prüfen, ob noch genügend Plätze frei sind. Der Rückgabewert ist der zu zahlende Gesamtbetrag an Kursgebühren, falls die Buchung erfolgreich war.

   d) Schreiben Sie ein Testprogramm, in dem Sie ein Objekt vom Typ Kurs erzeugen, Kursplätze buchen und den Rechnungsbetrag z. B. auf der Standardausgabe ausgeben.

   e) Die Kurse sollen mit Hilfe einer dynamischen Liste verwaltet werden. Ergänzen Sie dazu in der Klasse `Kurs` ein Attribut, das auf den nachfolgenden Kurs in der Liste verweist. Schreiben Sie eine Klasse `KursSystem`, die den Anfang der Liste speichert.

   f) Implementieren Sie in der Klasse `KursSystem` eine Suchmethode, die einen Kurs anhand einer Kursnummer aus der Liste sucht und - falls er vorhanden ist - als Rückgabewert zurückgibt.

Fortsetzung nächste Seite!

2. **UML**
Ein Tennis-Turnier lässt sich folgendermaßen beschreiben:

- Zu einem Turnier gehören Spiele, Spieler und Personal. Überlegen Sie hierbei, welche Elemente einzigartige Teile des Turniers sind und welche in mehreren Turnieren vorkommen können.
- Ein Turnier findet in genau einem Land statt.
- Das Personal besteht aus Schiedsrichtern und Hilfsrichtern.
- Jeder Schieds- bzw. Hilfsrichter besitzt einen Namen und stammt aus genau einem Land.
- Jeder Spieler tritt für genau ein Land an.
- Jedes Land hat einen Namen.
- In einem Spiel treten zwei Spieler gegeneinander an. Es wird von einem Schiedsrichter und bis zu 9 Hilfsrichtern begleitet.
- Ein Schiedsrichter ist der Oberschiedsrichter des Turniers.
- Jedes Spiel wird gespielt und dabei von einem Spieler gewonnen.

Erstellen Sie ein UML-Klassendiagramm, das den angegebenen Anforderungen entspricht. Wählen Sie dabei geeignete Datentypen für die Attribute und passende Signaturen für Methoden (üblicherweise parameterlos und leerer Rückgabetyp).

Hinweis: Geben Sie auf jeden Fall **Rollennamen** an! Die Assoziationen müssen selbst **keine Namen** haben. Tragen Sie bitte auch **keine Konstruktoren**, bzw. get- und set-Methoden für Attribute ein.

3. **Projektmanagement**
Betrachten Sie folgendes CPM-Netzwerk:

a) Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet.

b) Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten.

c) Berechnen Sie nun für jedes Ereignis die Pufferzeiten.

d) Bestimmen Sie den kritischen Pfad.

e) Was ist ein Gantt-Diagramm? Worin unterscheidet es sich vom CPM-Netzwerk?

Fortsetzung nächste Seite!

Teilaufgabe 2: Datenbanksystem

1. **Relationale Anfragesprachen**
   Gegeben sei folgendes relationales Schema, dessen Attribute nur atomare Attributwerte besitzen.

   Computer {[*IP*, *Name*, *Hersteller*, *Modell*, *Standort*]}

   - Geben Sie für die folgenden Anfragen einen relationalen Ausdruck an:
     a) Geben Sie die IP-Adresse des Computers mit Namen "Chiemsee" aus.
     b) Geben Sie 2er-Tupel von IP-Adressen der Computer am selben Standort aus.
   - Formulieren Sie die folgenden Anfragen in SQL:
     a) Geben Sie die IP-Adressen der Rechner am Standort "Büro2" aus.
     b) Geben Sie alle Computer-Namen in aufsteigender Ordnung mit ihren IP-Adressen aus.
     c) Geben Sie für jeden Hersteller die Anzahl der unterschiedlichen Modelle aus.
     d) Geben Sie für jeden Hersteller, welcher mindestens 2 unterschiedliche Modelle hat, die Anzahl der unterschiedlichen Modelle aus.

2. **Normalformen**
   Gegeben sei folgendes relationales Schema, dessen Attribute nur atomare Attributwerte besitzen.

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

   Es gelte die folgende Menge FD von funkionalen Abhängigkeiten:
   $FD = \{$

   $$ABC \rightarrow DE,$$
   $$E \rightarrow BCD$$

   $\}$

   a) Bestimmen Sie alle Superschlüssel und Kandidatenschlüssel von R.
   b) Welcher Normalform genügt R?

   Begründen Sie Ihre Aussagen in beiden Teilaufgaben.

Fortsetzung nächste Seite!

3. **Modellierung**
Es sind folgende Informationen zu einer Datenbank für Konsulate gegeben:

- Jedes Konsulat hat einen Sitz in einer Stadt.
- Zu einem Konsulat sollen ein eindeutiger Name (z.B. Konsulat Bayern), die Adresse, und der Vor- bzw. Nachname des Konsuls gespeichert werden.
- Für jede Stadt sollen der Name, die Anzahl der Einwohner, sowie das Land, in dem es liegt, festgehalten werden. Gehen Sie davon aus, dass eine Stadt nur in Zusammenhang mit dem dazugehörigen Land identifizierbar ist.
- Für ein Land sollen der Name in Landessprache, der Name des Staatspräsidenten und eine eindeutige ID gespeichert werden.

a) Entwerfen Sie für das obige Szenario ein ER-Diagramm in Chen-Notation. Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Funktionalitäten der Relationship-Typen.

Hinweis: Achten Sie darauf, alle Totalitäten einzutragen.

b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1: Softwaretechnologie

Aufgabe 1:
Grundlagen

Übertragen Sie die Buchstaben a) - q) auf Ihr Blatt und notieren Sie dahinter, ob die entsprechende Aussage wahr oder falsch ist.

Allgemein:

a) Im Software Engineering geht es vor allem darum, qualitativ hochwertige Software zu entwickeln.

b) Software Engineering ist gleichbedeutend mit Programmieren.

Vorgehensmodelle:

c) Das Spiralmodell ist ein Vorläufer sog. Agiler Methoden.

d) Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.

e) Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.

Anforderungserhebung:

f) Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z. B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zum Vorschein kommen könnten.

g) Ein Szenario beinhaltet eine Menge von Anwendungsfällen.

h) Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.

Architekturmuster:

i) Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.

UML:

j) Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.

k) Komponentendiagramme beschreiben die Struktur eines Systems.

l) Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.

Fortsetzung nächste Seite!

Enwurfsmuster:

m) Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.

n) Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.

Testen:

o) Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.

p) Validation dient der Überprüfung von Laufzeitfehlern.

q) Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

Fortsetzung nächste Seite!

Aufgabe 2:
Zeichnen Sie den Ablauf des V-Modells und beschreiben Sie die einzelnen Phasen in kurzen Sätzen.

Aufgabe 3:
Modellierung

Sie sollen ein Car-Sharing System modellieren. Das System soll die Verwaltung von Buchungen ermöglichen. Beachten Sie dabei folgendes:

1. Eine Buchung hat eine BuchungsNR, ein startDatum und ein endDatum. Außerdem beinhaltet sie zugehörige Fahrzeuge und einen Kunden. Zusätzlich soll es möglich sein, eine potentielle Buchung zu prüfen (ob die Fahrzeuge für besagten Zeitraum noch verfügbar sind) und evtl. zu bestätigen.
2. Ein Kunde hat eine ID, einen Namen und eine Adresse und beliebig viele Buchungen.
3. Ein Fahrzeug hat eine ID, einen Hersteller, ein Baujahr und eine Farbe. Außerdem gehört es einer Kategorie an.
4. Eine Kategorie hat einen Namen und einen Preis pro Tag.

Aufgabe a)
Erstellen Sie ein Datenmodell in Form eines UML-Klassendiagramms.

Aufgabe b)
Erstellen Sie ein UML-Sequenzdiagramm, welches die Kommunikation zwischen einem Kunden, dem System und einem Objekt Buchung (welches die Details zur Buchung enthält) modelliert. Dabei soll ein Anwendungsfall modelliert werden, der aus den nachfolgend genannten Schritten besteht.

1. Ein Kunde sucht nach Fahrzeugen einer bestimmten Kategorie für einen bestimmten Zeitraum.
2. Der Kunde kann dann eine neue Buchung tätigen.
3. Das System erstellt dabei eine neue (potentielle) Buchung.
4. Danach wird geprüft, ob die Buchung möglich ist (die Prüfung erfolgt über eine Methode des Objektes Buchung).
5. a) Wenn die Buchung möglich ist, wird sie bestätigt und der Kunde darüber informiert.
   b) Ist die Buchung nicht möglich, wird die Buchung gelöscht und der Kunde darüber informiert.

Aufgabe c)
Implementieren Sie in einer gängigen objektorientierten Programmiersprache eine Methode "kosten" für das Objekt Buchung, welche die gesamten Kosten für eine Buchung berechnet.

Fortsetzung nächste Seite!

Teilaufgabe 2: Datenbanksysteme

Aufgabe 1:

Datenbankentwurf

Versicherung

Für die fiktive Versicherung "Safe" soll ein Informationssystem aufgebaut werden. Fertigen Sie, um die dafür benötigten Daten speichern zu können, ein ER-Schema an. Geben Sie dabei Attribute von Entitäten und Beziehungen an; kennzeichnen Sie Schlüsselattribute durch Unterstreichen. Die Kardinalitäten von Beziehungen sollen in der (min, max)-Notation in das Modell aufgenommen werden. Führen Sie Surrogatschlüssel (künstliche Schlüsse) nur ein, falls es nötig ist.

Die zu modellierende Versicherung betreibt mehrere Filialen, die eine eindeutige Adresse besitzen und von denen der jährliche Umsatz erfasst werden soll. In einer Filiale arbeiten verschiedene Mitarbeiter der Versicherung, die einer Tarifgruppe zugeordnet werden. Jeder Mitarbeiter ist genau einer Filiale zugewiesen.

Eine weitere wichtige Personengruppe stellen Kunden dar, die über eine Kundennummer verfügen und von denen die Berufsgruppe bedeutsam ist. Für jede Person, die in der Versicherung von Bedeutung ist, sollen der Name, das Geburtsdatum, die Adresse (Straße, Hausnummer, PLZ) und die eindeutige Identifikationsnummer abgespeichert werden.

Für die Berechnung bestimmter tariflicher Optionen ist es wichtig, dass für jeden Kunden Angehörige in der Datenbank hinterlegt werden können. Diese besitzen einen Namen und ein Geburtsdatum.

Die Versicherung bietet verschiedene Produkte an, die eindeutig über eine Nummer identifiziert werden und über einen bestimmten Zeitraum angeboten werden. Die wichtigsten (aber nicht alle) Produktgruppen sind: KFZ-, Hausrat- und Lebensversicherungen. Bei KFZ-Versicherungen ist die Höhe des versicherten Personenschadens von Bedeutung. Außerdem sollen bestimmte (potentiell mehrere) Zusatzleistungen abgespeichert werden. Hausratversicherungen besitzen einen gewissen Umfang und Lebensversicherungen eine Auszahlungsfrist. Produkte werden in der Versicherung von einem Mitarbeiter an einem gewissen Datum zu einem gewissen Preis an einen Kunden vermittelt.

Fortsetzung nächste Seite!

Aufgabe 2:

Relationen

Transformieren Sie das unten angegebene ER-Schema in ein Relationenschema. Kennzeichnen Sie Primärschlüssel durch Unterstreichen. Geben Sie bei Fremdschlüsseln deren Herkunft an.

Hinweis: Sie dürfen für FOREIGN KEY (Attributmenge) IS KEY OF Relation die Abkürzung FK (Attributmenge) IKO Relation verwenden.

Aufgabe 3:

SQL

Gegeben ist das Schema eines Turn- und Sportvereins (TSV). Teilnehmer besuchen Kurse (gegen monatliche Gebühr), welche von einem Kursleiter gehalten werden. Ein Teilnehmer kann einen Kurs noch aktiv besuchen (aktiv="ja") oder bereits ausgetreten sein (aktiv="nein"). Die entrichtete monatliche Kursgebühr kommt zu 80 % dem Kursleiter zu.

Kursleiter (PersNr, Name, Adresse, Ausbildung)
Teilnehmer (MitgliedsNr, Name, Adresse, Geburtsjahr)
Kurs (KursID, Name, Gebühr, Beschreibung, Ort, Leiter)

- Leiter ist Fremdschlüssel und bezieht sich auf den Primärschlüssel von Kursleiter

besucht (Kurs, Mitglied, aktiv, Eintrittsjahr, Austrittsjahr)

- Kurs ist Fremdschlüssel und bezieht sich auf den Primärschlüssel von Kurs
- Mitglied ist Fremdschlüssel und bezieht sich auf den Primärschlüssel von Teilnehmer

Fortsetzung nächste Seite!

a) Gesucht werden alle Teilnehmer, die dem Karatekurs mit einem Alter von maximal 12 Jahren (bezogen auf die Jahreszahl) beigetreten sind. Auszugeben sind dabei deren Name, Adresse, Eintrittsalter, sowie ob sie noch aktiv an diesem Kurs teilnehmen. Geben Sie das entsprechende SQL-Statement an.

b) Die Gebühren des Tenniskurses sind um 10 % erhöht worden. Der entsprechende Eintrag ist in der Datenbank auszubessern. Geben Sie das entsprechende SQL-Statement an.

c) Gesucht werden alle Mitglieder, die momentan mindestens zwei verschiedene Kurse besuchen, wobei alle Attribute des jeweiligen Mitglieds inklusive der Anzahl der besuchten Kurse ausgegeben und die Mitglieder mit den meisten Kursen zuerst aufgeführt werden sollen. Geben Sie das entsprechende SQL-Statement an.

d) Formulieren Sie umgangssprachlich, wonach mit folgendem SQL-Statement gesucht wird.

SELECT Leiter, **SUM**(0.8*Gebühr) **AS** Gesamtsumme
FROM Kurs
GROUP BY Leiter
ORDER BY Gesamtsumme **ASC**

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Herbst 2015

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2015**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **13**

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1:

1. Klassendiagramm

Erstellen Sie zu der folgenden Beschreibung ein UML-Klassendiagramm. Setzen Sie insbesondere Sichtbarkeiten von Elementen, abstrakten Klassen, Vererbungsbeziehungen, Multiplizitäten, Assoziationen, Rollennamen, Assoziationsnamen, Kompositionsbeziehungen, Leserichtungen und Eigenschaften ein, um die Beschreibung möglichst vollständig umzusetzen. Wenn Sichtbarkeiten nicht explizit erwähnt sind, gehen Sie von öffentlicher Sichtbarkeit aus. Wählen Sie geeignete Datentypen für Attribute.

Eine Organisationseinheit hat einen Namen und enthält beliebig viele Organisationsobjekte. Jedes Organisationsobjekt ist in höchstens einer Organisationseinheit enthalten. Ein Organisationsobjekt ist entweder eine Organisationseinheit oder eine Person. Eine Person hat genau einen Nachnamen, beliebig viele Vornamen, einen Geburtsort und ein Geburtsdatum. Ein Datum wird durch einen Datentyp modelliert, der Tag, Monat und Jahr gruppiert. Zu einer Person kann das Alter in Jahren mittels einer Operation bestimmt werden, die das aktuelle Datum als Parameter erhält.

Jede Person ist entweder ein Student oder ein Dozent. Ein Dozent hat ein Personalkennzeichen (eine beliebige Zeichenkette). Einem Studenten ist eine (ganzzahlige) Matrikelnummer zugeordnet, die nur gelesen werden darf. Ferner soll ein Zähler für die Matrikelnummer verwaltet werden, für den als Anfangswert 1 gewählt wird.

Eine Lehrveranstaltung hat einen Namen. Lehrveranstaltungen können voneinander abhängen (uneingeschränkte Multiplizitäten). Jeder Dozent hält mindestens eine Lehrveranstaltung, die ihrerseits von mindestens einem Dozenten angeboten wird. Jeder Student hört mindestens eine Lehrveranstaltung, die ihrerseits von mindestens einem Studenten gehört wird. Jede Lehrveranstaltung ist entweder eine Vorlesung oder eine Übung. Eine Übung kann höchstens eine Vorlesung unterstützen, und zu jeder Vorlesung gibt es höchstens eine Übung.

2. Zustandsdiagramm

Erstellen Sie ein UML-Zustandsdiagramm, das die möglichen Zustände und Transitionen von Benutzern eines Stock-Photo-Systems (Vorrat-Foto-System) beschreibt. Vor- und Nachbedingungen sowie Aktionen dürfen Sie in Java-ähnlicher Syntax angeben.

Orientieren Sie sich an der folgenden Verhaltensbeschreibung:

- Nach der Registrierung kann ein Benutzer entscheiden, ob er zunächst aktiv (als Fotograf) oder passiv (nicht als Fotograf) teilnimmt.
- Ein (aktiver oder passiver) Benutzer kann sich jederzeit vom System abmelden. Abgemeldete Benutzer werden automatisch aus dem System gelöscht, wenn sie ihre Entscheidung nicht innerhalb von 100 Tagen revidieren.
- Ein passiver Benutzer kann jederzeit zu einem aktiven Benutzer werden und seine Karriere als Fotograf starten.

Fortsetzung nächste Seite!

- Ein aktiver Benutzer wird überprüft, sobald eines seiner Fotos von einem anderen Benutzer als unangemessen gemeldet wird. Die Überprüfung erfolgt durch einen Administrator. Dieser kann den betroffenen Benutzer jederzeit sperren.
- Ein Administrator kann einen gesperrten Fotografen nach einiger Zeit wieder entsperren.
- Ein Administrator kann einen beliebigen aktiven (nicht gesperrten) Benutzer jederzeit zum Administrator ernennen.
- Ein Administrator kann sein Amt niederlegen und somit zum aktiven Benutzer werden. Dies ist jedoch nur möglich, falls davor noch mindestens drei Administratoren im Sytem registriert sind.
- Administratoren und gesperrte Benutzer sowie Benutzer, die sich in Prüfung befinden, können sich nicht vom System abmelden.

Hinweis: Überlegen Sie sich auch, welche Parameter Sie Ihren auf den Transitionen stehenden Ereignissen mitgeben.

3. Projektmanagement

Betrachten Sie die folgende Tabelle zum Projektmanagement:

Arbeitspaket	Dauer (Tage)	abhängig von
A1	10	
A2	5	A1
A3	15	A1
A4	10	A2, A3
A5	15	A1, A3
A6	10	
A7	5	A2, A4
A8	10	A4, A5, A6

Tabelle 1: Übersicht Arbeitspakete

a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.
b) Wie lange dauert das Projekt mindestens?
c) Geben Sie den oder die kritischen Pfad(e) an.
d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

Fortsetzung nächste Seite!

4. Überladung und Überschreibung

Im folgenden ist eine Reihe von Klassen aufgelistet, links in Java und rechts in C++. Wählen Sie eine Spalte entsprechend Ihrer Kenntnisse aus.

(a) Nennen Sie die Zeilennummern der Funktionssignaturen, bei denen Überladung auftritt.

Zeile ____ überlädt Zeile ____.
Zeile ____ überlädt Zeile ____.

(b) Nennen Sie die Zeilennummern der Funktionssignaturen, bei denen Überschreibung auftritt.

Zeile ____ überschreibt Zeile ____.
Zeile ____ überschreibt Zeile ____.

(c) Geben Sie die ausgegebenen Zahlen in der richtigen Reihenfolge in den folgenden zwei Zeilen an.

1. Ausgabezeile:
2. Ausgabezeile:

```
1  class Item {
2
3    public void druck(Item x) {
4      System.out.println("1");
5    }
6
7    public void druck(Subitem x) {
8      System.out.println("2");
9    }
10 }
11
12 class Subitem extends Item {
13
14   public void druck(Item x) {
15     System.out.println("3");
16   }
17
18   public void druck(Subitem x) {
19     System.out.println("4");
20   }
21 }
22
23 public class Main {
24   public static void main(String[] args) {
25     Item a;
26     Subitem b;
27
28     a = new Subitem();
29     b = new Subitem();
30
31     a.druck(b);
32     b.druck(a);
33   }
34 }
```

```
1  class Item {
2    public:
3     virtual void druck(Item* x) {
4        cout << "1"<< endl;
5     }
6
7     virtual void druck(Subitem* x) {
8        cout << "2" << endl;
9     }
10 };
11
12 class Subitem : public Item {
13   public:
14    virtual void druck(Item* x) {
15      cout << "3"<< endl;
16    }
17
18    virtual void druck(Subitem* x) {
19      cout << "4" << endl;
20  }
21 };
22
23
24 int main() {
25   Item *a;
26   Subitem *b;
27
28   a = new Subitem();
29   b = new Subitem();
30
31   a->druck(b);
32   b->druck(a);
33 }
```

Fortsetzung nächste Seite!

Teilaufgabe 2:

1. Grundlagen (Begriffe definieren)

Erläutern Sie die folgenden Begriffe in knappen Worten:

a) Datenunabhängigkeit

b) Superschlüssel

c) Referentielle Integrität

2. Datenbankentwurf

2.1 Modellierung

Krankenhaus

Für das Krankenhaus "Healthy Heart" soll ein Informationssystem aufgebaut werden. Um die dafür benötigten Daten zu speichern, soll zunächst ein E/R-Schema angefertigt werden. Attribute von Entitäten und Beziehungen sind anzugeben; Schlüsselattribute werden durch Unterstreichen gekennzeichnet. Die Kardinalitäten von Beziehungen sollen in der Chen-Notation in das Modell aufgenommen werden. Führen Sie Surrogatschlüssel nur ein, falls es nötig ist:

In dem zu modellierenden Krankenhaus sind Patienten stationär in Zimmern untergebracht, wobei grundsätzlich mehrere Patienten auf einem Zimmer, je nach Anzahl der Betten, liegen können. Es wird festgehalten, von wann bis wann ein Patient einem Zimmer zugewiesen wurde. Patienten haben eine Patientennummer, einen Namen und eine Krankheit. Wird ein Patient zu einem späteren Zeitpunkt wegen einer anderen oder der gleichen Krankheit erneut im Krankenhaus behandelt, erhält er eine neue Patientennummer und zählt somit als neuer Patient.

Zimmer sind Stationen zugeordnet, wobei die Zimmernummern nur für jede Station eindeutig sind. Stationen haben Namen wie Chirurgie, Intensivstation oder Palliativstation. Ebenso wie die Zimmer ist auch das Stationspersonal einer Station zugeordnet. Jeder Angestellte des Stationspersonals besitzt eine Personalnummer und einen Namen. Ärzte, die einen Teil des Personals darstellen, haben zusätzlich noch ein Fachgebiet und einen Rang. Der andere Teil, das Pflegepersonal, hat zusätzlich eine oder mehrere Qualifikationen.

Jedem Patienten sind behandelnde Ärzte zugeordnet. Das Pflegepersonal kann nur indirekt über die Station jedem Patienten zugeordnet werden.

2.2
Erläutern Sie anhand eines selbstgewählten Beispiels, wann und warum es notwendig ist, bei rekursiven Beziehungen Rollennamen zu vergeben.

2.3
Erklären Sie, warum eine binäre Beziehung zwischen einem schwachen und einem starken Entitätstyp keine N:M-Beziehung sein kann.

Fortsetzung nächste Seite!

3. Relationen

Transformieren Sie das unten angegebene ER-Schema in ein Relationenschema. Kennzeichnen Sie Primärschlüssel durch Unterstreichen. Geben Sie bei Fremdschlüsseln deren Herkunft an. Geben Sie bei der Auflösung von Generalisierungen/Spezialisierungen eine kurze Begründung für Ihre Wahl an.

Hinweis: Sie dürfen für FOREIGN KEY (*Attributmenge*) IS KEY OF *Relation* die Abkürzung FK (*Attributmenge*) IKO *Relation* verwenden.

Fortsetzung nächste Seite!

4. Relationale Algebra & SQL

Das Datenbankschema der Versicherungsgesellschaft „Insurance Pro" enthält sowohl Kunden- als auch Mitarbeiterdaten, sowie Informationen über Schadensfälle. Das Schema ist von folgender Gestalt, wobei FS für Fremschlüssel und PS für Primärschlüssel steht:

Versicherungsnehmer (KNr, Name, Anschrift, Geburtsdatum)

Fahrzeug (Kennzeichen, Farbe, Fahrzeugtyp)

Mitarbeiter (MNr, Name, Kontaktdaten, Abteilung, Leiter)

- Abteilung ist FS und bezieht sich auf den PS von Abteilung.

Abteilung (ANr, Bezeichnung, Ort)

Versicherungsvertrag (VNr, Abschlussdatum, Art, Versicherungsnehmer, Fahrzeug, Mitarbeiter)

- Versicherungsnehmer ist FS und bezieht sich auf den PS von Versicherungsnehmer.
- Fahrzeug ist FS und bezieht sich auf den PS von Fahrzeug.
- Mitarbeiter ist FS und bezieht sich auf den PS von Mitarbeiter.
- Attribut Art kann die Werte HP (Haftpflicht), TK (Teilkasko) und VK (Vollkasko) annehmen.

Schadensfall (SNr, Datum, Ort, Gesamtschaden, Beschreibung, Mitarbeiter)

- Mitarbeiter ist FS und bezieht sich auf den PS von Mitarbeiter.

Beteiligung (Schadensfall, Fahrzeug, Fahrzeugschaden)

- Schadensfall ist FS und bezieht sich auf den PS von Schadensfall.
- Fahrzeug ist FS und bezieht sich auf den PS von Fahrzeug.

Fortsetzung nächste Seite!

4.1
Herr Meier schließt eine Teilkaskoversicherung bei Insurance Pro am 11.11.2011 für seinen neuen Wagen, einen roten VW Golf VII mit amtlichem Kennzeichen BT-BT 2011, ab. Der Vertrag erhält die laufende Nummer 1631 und wird von Frau Schmied mit der Personalnummer 27 bearbeitet. Herr Meier erhält die Kundennummer 588, da er bisher noch kein Kunde war.
Geben Sie die SQL-Befehle an, um die neue Versicherung in die Datenbank von Insurance Pro einzutragen. Werte, die hierbei nicht bekannt sind, sollen weggelassen werden, wobei angenommen werden darf, dass die entsprechenden Attribute nicht mit der Bedingung NOT NULL versehen sind.

4.2
Beschreiben Sie umgangssprachlich, wonach mit folgendem Ausdruck gesucht wird:

$$\Pi_{Versicherungsnehmer}\left(\sigma_{Fahrzeugtyp='Fiat500'}\left(FZ \bowtie_{\rho_{Kennzeichen \leftarrow Fahrzeug}} (VV)\right)\right)$$

Anmerkung: Die Abkürzung FZ steht für die Tabelle Fahrzeug und VV für die Tabelle Versicherungsvertrag.

4.3
Die Angaben der Mitarbeiter (Nr., Name und Kontakt), deren Abteilung ihren Sitz in München oder Stuttgart hat, sollen explizit alphabetisch nach Namen sortiert ausgegeben werden. Geben Sie einen SQL-Befehl hierfür an.

4.4
Es soll ausgegeben werden, wie oft jeder Versicherungsnehmer (KNr, Name, Anschrift) an einem Schadensfall beteiligt war. Hierbei sollen nur die Kunden, die an mindestens drei Schadensfällen beteiligt waren, in absteigender Reihenfolge aufgelistet werden.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1:

Aufgabe 1: „OOD"

Betrachten Sie folgende Spezifikation eines vereinfachten XML-artigen Dokuments:

Ein Dokument besteht aus baumartig geschachtelten Knoten. Es verwaltet genau eine Wurzel, die man lediglich mit einer entsprechenden Methode erfragen (aber nicht mehr ändern) kann, und stellt fabrikartige Methoden zur Verfügung, um konkrete Knoten zu erzeugen. Derzeit werden die konkreten Knoten-Arten „Element" und „TextKnoten" unterstützt. Die Dokument-Wurzel ist zwingend ein Element-Knoten und auch das Dokument selbst ist ein Knoten.
Knoten können beliebig viele Kinder haben, die ihrerseits Knoten sind, wobei jedes Kind zu genau einem Vater-Knoten gehört und zusätzlich zum Vater auch das umgebende Dokument (also den „Eigentümer") kennt. Der abstrakte Knoten stellt Methoden zur Verfügung, um Kind-Knoten zu ergänzen bzw. zu löschen oder alle Kinder als List<Knoten> zurückzugeben. Knoten haben außerdem eine AttributListe, die eine HashMap<String, String> ist und benannte Attribute enthält – Knoten haben eine Methode, die die ganze AttributListe bereitstellen.
Ein Element ist ein Knoten und hat einen elementNamen (vom Typ String), der beim Erzeugen eines Elements (im Dokument!) festgelegt wird und mit einer entsprechenden Methode erfragt werden kann. Zusätzlich bietet das Element Methoden an, um einzelne Attribute mit ihrem Namen (als String) zu erfragen oder samt Wert (auch String) zu setzen.
TextKnoten sind ebenfalls Knoten, allerdings kapseln sie lediglich einen Text (String), den man mit entsprechenden Methoden erfragen oder ersetzen kann (die ersetzende Methode gibt dabei den alten Text zurück). Sie werden ebenfalls im Dokument erzeugt.

Erstellen Sie aus der obigen „Spezifikation" ein UML-Klassendiagramm mit allen Klassen, Vererbungsrelationen, Attributen, Methoden (auch Konstruktoren) und Assoziationen. Tragen Sie bei Assoziationen stets auch die Multiplizitäten und Navigationspfeile ein. Falls Rollennamen bekannt sind, so müssen diese ebenfalls angegeben werden. Achten Sie ferner auf die Sichtbarkeiten von Methoden und Attributen.
Als UML-Klassen explizit (graphisch) modellierte Typen sollen nicht als Attribute im Klassenkästchen erscheinen. Nehmen Sie hierbei vereinfachend an, dass **String** und **List<String>** vorgegebene („primitive") Datentypen sind, die nicht mehr als explizite UML-Klasse eingezeichnet werden müssen; die ebenfalls vorgegebene Klasse **HashMap<K,V>** dürfen Sie ohne Attribute und Methoden einzeichnen, um die AttributListe entsprechend modellieren zu können. Verwenden Sie möglichst restriktive aber sinnvolle Sichtbarkeiten, sofern der obige Text nichts Genaueres spezifiziert.

Beispiel-Dokument:
```
<klausur>
        <titel>Staatsexamen</titel>
        <aufgabe name="UML" punkte="42"/>
        <aufgabe name="Testen" punkte="47">
                <teilaufgabe name="a)" punkte="11">Geben Sie    </teilaufgabe>
        </aufgabe>
</klausur>
```

Fortsetzung nächste Seite!

Aufgabe 2: „Testen"

Gegeben sei folgende Methode `isPalindrom` und ihr Kontrollflussgraph:

```
boolean isPalindrom(String s) {
    boolean yesItIs = true;
    if (s != null && s.length() > 1) {
        do {
            if (s.charAt(0) != s.charAt(s.length() - 1)) {
                yesItIs = false;
            }
            s = s.substring(1, s.length() - 1);
        } while (yesItIs && s.length() > 1);
    }
    return yesItIs;
}
```

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 und **möglichst kurzen** Pfaden 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 den zugehörigen Testfall 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) Übernehmen Sie den vorgegebenen Kontrollflussgraphen und annotieren Sie ihn mit allen relevanten Datenflussereignissen. Geben Sie jeweils an, ob die Verwendungen berechnend (c-use) oder prädikativ (p-use) sind.

Fortsetzung nächste Seite!

Aufgabe 3: „Formale Verifikation"

Sei *wp(A, Q)* die schwächste Vorbedingung (*weakest precondition*) eines Programmfragments *A* bei gegebener Nachbedingung *Q* so, dass *A* alle Eingaben, die *wp(A,Q)* erfüllen, auf gültige Ausgaben abbildet, die *Q* erfüllen.
Bestimmen Sie schrittweise und *formal* (mittels Floyd-Hoare-Kalkül) jeweils *wp(A, Q)* für folgende Code-Fragmente *A* und Nachbedingungen *Q* und vereinfachen Sie dabei den jeweils ermittelten Ausdruck so weit wie möglich.
Die Variablen **x**, **y** und **z** in folgenden Pseudo-Codes seien ganzzahlig (vom Typ **int**). Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

a) Sequenz:

```
x = -2 * (x + 2 * y);
y += 2 * x + y + z;
z -= x - y - z;
```

$Q :\equiv x = y + z$

b) Verzweigung:

```
if (x < y) {
      x = y + z;
} else if (y > 0) {
      z = y - 1;
} else {
      x -= y -= z;
}
```

$Q :\equiv x > z$

c) Mehrfachauswahl:

```
switch (z) {
case 'x':
      y = 'x';
case 'y':
      y = --z;
      break;
default:
      y = 0x39 + '?';
}
```

$Q :\equiv 'x' = y$

Hinweis zu den ASCII-Codes:

- ➢ '**x**' = $120_{(10)}$
- ➢ '**y**' = $121_{(10)}$
- ➢ **0x39**$_{(16)}$ = $57_{(10)}$
- ➢ '**?**' = $63_{(10)}$

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1: Datenmodellierung: Entity-Relationship-Modell und relationales Datenmodell

Entwerfen Sie eine Datenbank zur Verwaltung eines Online-Multiplayerspiels.

- Das Spiel besteht aus mehreren Servern. Jeder der Server besitzt eine eindeutige IP-Adresse und einen Namen.
- Für die Nutzer des Spiels werden eine eindeutige Nutzer-ID, ein Benutzername und das Passwort gespeichert.
- Sollte ein Nutzer eine unerlaubte Aktion auf einem Server durchführen, kann dieser bis zu einem festgelegten Datum auf diesem Server gesperrt werden.
- Jeder Nutzer kann Charaktere besitzen, mit denen er auf dem Server spielen kann. Jeder Charakter ist genau einem Nutzer zugeordnet.
- Auf jedem Server können Charaktere der Nutzer spielen. Die Spieldauer eines Charakters auf dem Server wird festgehalten. Ein Charakter besteht aus seinem Namen, dem Charaktertyp als Integer und seiner Stufe. Jeder Charakter wird durch den Nutzer, den Typ und seinen Namen eindeutig identiziert.
- Jeder Charakter ist dauerhaft an einen Server gebunden.

a) Modellieren Sie das oben dargestellte Szenario möglichst vollständig in einem ER-Modell. Verwenden Sie wann immer möglich (binäre oder auch höherstellige) Relationships.
b) Wann treten schwache Entitys auf? Ist der Schlüssel einer schwachen Entity global gültig? Wenn nein, wie entsteht ein global eindeutiger Schlüssel für eine solche Entity?
c) Übertragen Sie Ihr ER-Modell in das relationale Datenmodell. Erstellen Sie dazu Tabellen mit Hilfe von CREATE TABLE-Statements aus SQL. Berücksichtigen Sie die Fremdschlüsselbeziehungen.
d) Beschreiben Sie, was eine View in SQL ist und erzeugen Sie eine solche mit dem Namen „OftenPlayed". Diese soll den Charakternamen, den Typ und die Stufe für Charaktere und den dazugehörigen Nutzernamen enthalten, die mehr als 7 Stunden Spielzeit aufweisen.

Aufgabe 2: Datenbankanfragen und -änderungen in SQL

Formulieren Sie in SQL die folgenden Anfragen, Löschungen, Einfügungen und Änderungen an die Datenbank aus Aufgabe 1. Achten Sie darauf, dass keine Duplikate in der Ergebnisliste vorkommen.

Nutzer (NutzerID, Nutzername, Passwort)
Server (Adresse, Servername)
Charakter (NutzerID, Name, Typ, Stufe, Spielzeit, Serveradresse)
gebannt (Serveradresse, NutzerID, Datum)

a) Bestimmen Sie alle Nutzernamen, bei denen mindestens ein Bann vor dem 01.04.2015 ausläuft.

Fortsetzung nächste Seite!

b) Bestimmen Sie für die Charaktere mit der höchsten Spieldauer den Charakternamen, seine Spielzeit und den Servernamen, dem er zugeordnet ist.
c) Bestimmen Sie für jeden Server den Namen und die dazugehörige Anzahl an Charakteren, die auf diesem Server aktiv ist und deren Gesamtspielzeit.
d) Löschen Sie alle Charaktere, die den Server mit dem Namen „S1" oder „S2" benutzen.
e) Verändern Sie den Nutzernamen und das Passwort des Nutzers „Griffin" auf den Namen „Simpson" und das Passwort auf „abcdef".
f) Fügen Sie einen neuen Server Aram mit der Adresse 123456789 in die Datenbank ein.

Aufgabe 3: Funktionale Abhängigkeiten und Zerlegungen

Gegeben sei das Relationenschema $R=(U,F)$ mit der Attributmenge U und der Menge F von funktionalen Abhängigkeiten:

$$U=\{Dozent, Student, Fach, Fakultät, Adresse\},$$
$$F=\{\{Student, Fach\}\rightarrow\{Adresse, Dozent, Fakultät\},$$
$$\{Student\}\rightarrow\{Adresse\}, \{Dozent\}\rightarrow\{Fach, Fakultät\}\}.$$

a) Geben Sie eine erlaubte Instanz r zu R mit mindestens 5 Tupeln an, so dass die funktionalen Abhängigkeiten $\{Dozent\}\rightarrow\{Student\}$ sowie $\{Student\}\rightarrow\{Dozent\}$ nicht gelten.
b) Geben Sie alle Schlüssel für das Relationenschema R (jeweils mit Begründung) sowie die Nichtschlüsselattribute an.
c) Ist R in 3NF, ist R in BCNF (jeweils mit Begründung)?
d) Geben Sie eine Basis G von F an und führen Sie damit die 3NF-Synthese aus.
e) Sei D das in Teil d) erhaltene Datenbankschema. Berechnen Sie die D entsprechende Zerlegung Ihrer Relation r aus Teil a).

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2016

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2016**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **12**

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1: Softwaretechnologie

Aufgabe 1:

a) Erklären Sie die Begriffe *abstrakte Klasse* und *Interface*, wie sie in UML (und Java) gebräuchlich sind.

b) Was versteht man unter einer *abstrakten Methode*?

c) Erläutern Sie kurz den Begriff einer *Schablonenmethode* (engl. template method).

d) Erläutern Sie kurz, inwiefern die objektorientierte Softwareentwicklung die Wiederverwendung von Software unterstützt.

Aufgabe 2:

Gegeben sei ein Softwaresystem mit drei Klassen *MainClass*, *FirstClass*, *SecondClass*, so dass die Klasse *MainClass* die *main*-Methode enthält. Zur Laufzeit sollen die im Sequenzdiagramm in Abb. 1 dargestellten Interaktionen stattfinden.

Abbildung 1: Interaktionen

Fortsetzung nächste Seite!

a) Geben Sie ein Klassendiagramm an, das die statische Struktur des Systems modelliert. Das Klassendiagramm soll alle für die obigen Interaktionen benötigten Modellelemente zeigen, einschließlich Assoziationen (mit Rollennamen und Multiplizitäten an gerichteten Assoziationsenden), sowie Konstruktoren und Operationen (jeweils ggf. mit formalen Parametern und deren Typen).

b) Implementieren Sie das System in einer objektorientierten Programmiersprache Ihrer Wahl. Für benutzerdefinierte Konstruktoren und Operationen (Methoden) sind die Rümpfe anzugeben, so dass zur Laufzeit die in Abb. 1 dargestellten Interaktionen stattfinden. Die Methode *callback* braucht nicht implementiert zu werden.

Aufgabe 3:

Es wird das Verhalten einer automatischen Straßenbahntüre betrachtet, die durch einen Verriegelungsschalter des Fahrers und durch eine Öffnungstaste des Fahrgasts gesteuert werden kann. Der Verriegelungsschalter kann vor- und zurückgestellt werden.

Zu Beginn ist die Straßenbahntüre verriegelt und geschlossen. Wird der Verriegelungsschalter vom Fahrer vorgestellt, dann ist die Türe entriegelt (bleibt aber geschlossen). Nach dem Zurückstellen des Schalters ist sie verriegelt. Das Drücken der Taste durch einen Fahrgast bewirkt das Folgende: Ist die Türe entriegelt, dann öffnet sie sich. Ist die Türe verriegelt, dann öffnet sie sich erst, wenn der Fahrer den Verriegelungsschalter vorstellt. Eine offene Türe schließt sich, wenn der Fahrer den Verriegelungsschalter zurückstellt.

Modellieren Sie das oben beschriebene Verhalten einer Straßenbahntüre durch ein UML-Zustandsdiagramm. Das Öffnen und Schließen der Türe kann als ununterbrechbare Aktion aufgefasst werden. Stellen Sie klar, welche Ereignisse Zustandsänderungen der Straßenbahntüre bewirken und welche Aktionen ggf. von einem Ereignis ausgelöst werden.

<u>Teilaufgabe 2:</u> Datenbanksysteme

Aufgabe 1 (Konzeptioneller Entwurf)

Im Folgenden ist die Beschreibung einer Bibliotheksverwaltung 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.

In der Bibliotheksverwaltung werden Medien verwaltet. Jedes Medium wird über eine eindeutige Signatur identifiziert und hat einen Titel. Medien sind entweder Bücher oder Zeitschriften. Bücher haben ein Erscheinungsdatum. Zeitschriften haben einen Jahrgang und eine Ausgabenummer. Von jedem Medium kann es kein, ein oder mehrere Exemplare geben, die eine für das jeweilige Medium eindeutige Exemplarnummer haben. Ein Medium stammt von einem oder mehreren Autoren und ist von einem Verlag verlegt. Zu einem Autor werden Vorname und Nachname abgelegt, sowie das

Fortsetzung nächste Seite!

Geburtsdatum und der Geburtsort. Ein Autor wird über diese vier Eigenschaften eindeutig identifiziert. Verlage haben einen eindeutigen Namen. Autoren und Verlage sollen auch schon angelegt werden können, wenn noch kein zugehöriges Medium existiert. Nutzer haben einen Nachnamen, einen Vornamen und eine eindeutige Nutzernummer. Nutzer können Medienexemplare (auch mehrmals hintereinander) ausleihen. Zu jeder Ausleihe werden dauerhaft Entleihdatum und Rückgabedatum gespeichert. Nutzer können Bücher in verschiedenen Bewertungskategorien mit einer Anzahl Sterne bewerten. Eine Bewertungskategorie hat einen eindeutigen Namen und eine Beschreibung.

Aufgabe 2 (Normalformen)

Gegeben ist die Relation R(a, b, c, d), deren Attributwertebereiche alle atomar sind, mit den funktionalen Abhängigkeiten abc → d und d → b. Geben Sie an, in welcher höchsten Normalform R ist. Begründen Sie Ihre Antwort.

Aufgabe 3 (SQL)

Gegeben sind die folgenden Relationen mit den in Klammern angegebenen Attributen. Primärschlüssel sind unterstrichen, bei Fremdschlüsseln ist der Name der Relation, auf die sie sich beziehen, hinter dem Attributnamen in eckigen Klammern angegeben.

```
Laender (ID, Landesname)
Kunden (Kundennummer, Nachname, Vorname, Geburtsdatum, Land[Laender])
Hat_Geworben (Werber[Kunden], Geworbener[Kunden])
```

Keines der Attribute kann NULL-Werte enthalten.

Verwenden Sie in dieser Aufgabe nur standardkonformes SQL. Produktspezifische Erweiterungen (z.B. `limit`, `rownum`) werden als Fehler gewertet.

a) Schreiben Sie eine SQL-Anweisung, die Kundennummer, Vorname und Nachname aller Kunden ausgibt, aufsteigend sortiert nach Alter.

b) Schreiben Sie eine SQL-Anweisung, die eine Liste mit den Kundennummern der Kunden ausgibt, deren Geburtsdatum nicht einmalig unter den Kunden ist. Das heißt, ein Kunde muss ausgegeben werden, wenn es zu ihm einen anderen Kunden gibt, der dasselbe Geburtsdatum hat, sonst nicht. Jede Kundennummer soll maximal einmal ausgegeben werden.

c) Schreiben Sie eine SQL-Anweisung, die die Anzahl der Kunden pro Land ausgibt. Die Ausgabe soll den Namen des Landes und die Kundenanzahl enthalten. Sollten zwei Länder mit unterschiedlicher ID den gleichen Namen besitzen, sind beide getrennt voneinander auszugeben.

d) Schreiben Sie eine SQL-Anweisung, die pro Land den Kunden ermittelt, der am meisten andere Kunden (unabhängig von deren Land) geworben hat. Ausgegeben werden sollen pro Land der Name des Landes zusammen mit dem Vor- und Nachnamen des Kunden.

Fortsetzung nächste Seite!

Wenn es in einem Land mehrere Kunden mit gleicher höchster Geworbenenanzahl gibt, sollen diese alle ausgegeben werden. Wenn es in einem Land keine Kunden gibt, die andere Kunden geworben haben, soll das Land nicht ausgegeben werden. Sollten zwei Länder mit unterschiedlicher ID den gleichen Namen besitzen, sind beide getrennt voneinander auszugeben. Es wird empfohlen, eine View als Zwischenschritt zu verwenden.

e) Schreiben Sie eine SQL-Anweisung, die bei allen Kunden aus *'Deutschland'* Vorname und Nachname vertauscht. Die Änderung soll auch dann korrekt stattfinden, falls *'Deutschland'* (mit verschiedenen IDs) mehrmals in der Tabelle Laender gespeichert ist.

Aufgabe 4 (Physische Datenorganisation)

a) Nennen Sie, aus welchen zwei Teilen ein TID (= Tupel-Identifikator) besteht.

b) Geben Sie an, wie viele Seiten maximal gelesen werden müssen, um das zu einem TID gehörende (unfragmentierte) Tupel zu lesen. Begründen Sie Ihre Antwort in zwei bis drei Sätzen.

c) Nennen Sie zwei Gründe, warum der Baum in der folgenden Skizze kein korrekter B-Baum ist. Eingetragen sind nur die Schlüsselwerte.

d) Nennen Sie den wesentlichen Unterschied eines B+-Baums (auch als B*-Baum bezeichnet) zu einem B-Baum.

e) Geben Sie an, welche Seite bei der Seitenersetzung im Datenbankpuffer idealerweise ersetzt werden sollte.

f) Gegeben sind die zwei Indexstrukturen B+-Baum (auch als B*-Baum bezeichnet) und Hash-Index, basierend auf erweiterbarem Hashing.
Geben Sie an, welche der beiden Indexstrukturen für Bereichsanfragen (z.B. Personen mit Alter zwischen 20 und 40) besser geeignet ist. Erklären Sie in ein bis drei Sätzen, wie eine solche Anfrage mit Hilfe des besser geeigneten Index ausgewertet wird.

Fortsetzung nächste Seite!

Aufgabe 5 (Transaktionen)

a) Nennen Sie die vier wesentlichen Eigenschaften einer Transaktion und erläutern Sie jede Eigenschaft kurz (ein Satz pro Eigenschaft).

b) Gegeben ist die folgende Historie (Schedule) dreier Transaktionen:
$r_1(B) \rightarrow w_1(C) \rightarrow r_3(C) \rightarrow r_1(A) \rightarrow c_1 \rightarrow r_2(C) \rightarrow r_3(C) \rightarrow r_2(C) \rightarrow w_2(B) \rightarrow c_2 \rightarrow c_3$

Zeichnen Sie den Serialisierbarkeitsgraphen zu dieser Historie und begründen Sie, warum die Historie serialisierbar ist oder nicht.

c) Geben Sie an, wodurch die erste und die zweite Phase des Zwei-Phasen-Sperrprotokolls jeweils charakterisiert sind (ein Satz pro Phase).

Thema Nr. 2

(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1: Softwaretechnologie

Aufgabe 1: „Formale Spezifikation Abstrakter Datentypen (ADT)"

WICHTIG: In dieser Aufgabe stehen **keine** anderen Datentypen (also **kein int**, **String** usw.), **keine** anderen Konstanten (auch **nicht** 0, 1 usw.) und **keine** anderen Operationen (also **kein** <, +, =, ≠ usw.) zur Verfügung: Sie dürfen ausschließlich die vorgegebenen oder zu ergänzenden **adt**s mit deren **ops** verwenden!
Gegeben seien die folgenden Gerüste der ADTs *Nat* (Repräsentation natürlicher Zahlen $\mathbb{N}_0$) und *NatStack* (Stapel mit *Nat*-Elementen):

```
adt   Nat
sorts Nat
ops
      NaN:                  →Nat  // „Not a Nat" – vergleichbar mit Double.NaN
      zero:                 →Nat  // entspricht der natürlichen Zahl „0"
      succ:   Nat           →Nat  // Nachfolger „(x + 1)" der Nat-Zahl x
      add:    Nat × Nat →Nat  // Addition im Nat-Raum
      sub:    Nat × Nat →Nat  // Subtraktion im Nat-Raum (z.B. „42 − 666 = 0")
      mul:    Nat × Nat →Nat  // Multiplikation im Nat-Raum
      ...
axs
      succ(NaN) = NaN           // alle Operationen mit NaN ergeben stets wieder
                                // NaN
      ...                       // weitere Axiome aus Platzgründen weggelassen
end   Nat
```

```
adt   NatStack
sorts NatStack Nat
      ,
ops
      empty:                  →NatStack // erzeugt einen neuen leeren Stapel
      push:   NatStack × Nat  →Nat      // legt ein Nat auf den Stapel
      peek:   NatStack        →Nat      // gibt bei leerem Stapel NaN und bei
                                        // nicht-leerem Stapel das „oberste" Nat zurück
      pop:    NatStack        →NatStack // gibt bei leerem Stapel empty und bei nicht -
                                        // leerem
      ...                               // Stapel den Stapel ohne das „oberste" Nat
                                        // zurück
axs
      ...                               // Axiome aus Platzgründen weggelassen
end   NatStack
```

Fortsetzung nächste Seite!

a) Ergänzen Sie *NatStack* um diejenigen Axiome, die das Zusammenspiel von *peek* bzw. *pop* mit den Primärkonstruktoren *empty* und *push* spezifizieren.

b) Ergänzen Sie den ADT *NatStack* um Signaturen und Axiome für die Operation *size*, die die Anzahl der Elemente im Stapel als *Nat*-Zahl berechnet.

c) Ergänzen Sie den ADT *NatStack* um Signaturen und Axiome, die das Verhalten der folgenden Methode **nat2bin** spezifizieren, die für eine *Nat*-Zahl ihre Binärdarstellung (Zweierkomplement) als *NatStack* erzeugt:

```
public static NatStack nat2bin(Nat n) {
if (n == zero)
            return push(empty, zero);
      else
return push(nat2bin(div(n, succ(succ(zero)))),
                        mod(n, succ(succ(zero))));
}
```

$$Beispiel: 42_{(10)} \xrightarrow{nat2bin} \left\{\begin{matrix} \mathbf{zero} \\ succ(zero) \\ zero \\ succ(zero) \\ zero \\ succ(zero) \\ zero \end{matrix}\right\} = 0101010_{(2)}$$

d) Ergänzen Sie den ADT *NatStack* um Signaturen und Axiome für die Operation **bin2nat**, die aus einer Binärdarstellung im Stapel die zugehörige *Nat*-Zahl berechnet. Es wird garantiert, dass nur *zero* oder *succ(zero)* auf dem Stapel liegen und dass das *niederwertigste* Bit das „**oberste**" ist.

e) Ergänzen Sie den ADT *NatStack* um die Axiome für die Operation *revHelper* so, dass *reverse* die Reihenfolge der Elemente im Stapel „umdreht", d.h. u.a. dass das oberste Element zum untersten wird:

Fortsetzung nächste Seite!

ops

revHelper: NatStack × NatStack → NatStack
reverse: NatStack → NatStack

axs

reverse(s) = revHelper(s, empty)
revHelper(...) = ...

Aufgabe 2: „Petri-Netze"

Gegeben sei das folgende Petri-Netz:

a) Erstellen Sie den zum Petri-Netz gehörenden Erreichbarkeitsgraphen. Die Belegungen sind jeweils in der Form [A, B, C] anzugeben. Beschriften Sie auch jede Kante mit der zugehörigen Transition. Beachten Sie die auf 1 beschränkte Kapazität von Stelle A oder alternativ die Inhibitor-Kante von A zu t3 (beides ist hier semantisch äquivalent).
b) Wie kann man mit Hilfe des Erreichbarkeitsgraphen feststellen, ob ein Petri-Netz lebendig ist?
c) Aufgrund von Transition t4 ist das gegebene Petri-Netz nicht stark lebendig. Wie müssten die Pfeilgewichte der Transition t4 verändert werden, damit das Petri-Netz mit der gegebenen Startmarkierung **beschränkt bleibt** und **lebendig wird**?

Fortsetzung nächste Seite!

Aufgabe 3: „W-Modell“

Das sogenannte W-Modell für die Softwareentwicklung erweitert das klassische V-Modell um ein zweites (leicht nach rechts versetztes) V, das die Testaktivitäten explizit und jeweils parallel zu den entsprechenden Entwicklungsaktivitäten aufführt:

a) Übernehmen Sie das skizzierte W-Modell und ergänzen Sie die jeweils fehlenden Phasen, Aktivitäten und Produkte („???“ im Bild).
b) Beschreiben Sie kurz die Tätigkeiten bzw. Produkte, die in jeder Phase des klassischen V-Modells (der linke absteigende und der linke aufsteigende Ast im W-Modell) ausgeführt respektive erzeugt werden.
c) Erklären Sie kurz den Zweck und das zugehörige Vorgehen der vier Aktivitäten im zweiten absteigenden Ast des W-Modells – oder mit anderen Worten: Welche Bedeutung haben die Doppelpfeile?

Fortsetzung nächste Seite!

Teilaufgabe 2: Datenbanksysteme

1. Modellieren Sie eine Agentur, die Konzerte veranstaltet. Auf diesen Konzerten treten mehrere Bands auf. Die Eintrittskarten sollen eindeutig nummeriert sein (Ticket#) und sollen Block, Reihe und Platz# enthalten. Die Preise sollen pro Block gleich sein. Die Reihenfolge des Auftritts der Bands ist relevant.

   a. Erstellen Sie den konzeptionellen Entwurf mittels ER- oder UML-Diagramm.
   b. Markieren Sie die Funktionalitäten der Beziehungen.

2. Setzen Sie Ihren konzeptuellen Entwurf systematisch in ein relationales Schema um. Markieren Sie Schlüssel und Fremdschlüssel. Für eine der Relationen geben Sie auch die SQL-Definition mitsamt Primär- und Fremdschlüsselspezifikationen an.

3. Ihr relationales Schema könnte (muss aber nicht) beispielsweise folgende Relation enthalten:

   KonzertBandsTickets: {[KonzertID, Ticket#, BandID]}

   a. Welche funktionalen Abhängigkeiten gibt es?
   b. Markieren Sie die Kandidatenschlüssel.
   c. Normalisieren Sie diese Relation systematisch.
   d. Diskutieren Sie die Qualität der resultierenden Relationen kritisch.

4. Auf der Basis der Relation <u>KonzertBandsTickets</u> sollen folgende Anfragen in SQL formuliert werden:

   a. Welches Konzert hatte die meisten Besucher?
   b. Welche Band-Paare wurden häufig zusammen gebucht (sind also mindestens fünf Mal gemeinsam bei Konzerten aufgetreten)?
   c. Welche Dreier-Kombinationen von Bands wurden mehrmals (also mindestens zweimal) bei Konzerten gebucht?

Fortsetzung nächste Seite!

5. Betrachten Sie folgenden B-Baum mit maximaler Knotenkapazität 4.

Zeigen Sie nach jeder nachfolgenden Operation den resultierenden Baum:

a. Fügen Sie 9 ein.
b. Fügen Sie 13 ein.
c. Löschen Sie 7.

6. Vergleich von Indexstrukturen

a. Welche Höhe hat ein B-Baum, der als Schlüssel alle 10 Milliarden Telefonnummern weltweit indexiert? Die maximale Knotenkapazität sei 200; mimimal also 100 (bis auf die Wurzel).

b. Welche Vorteile bzw. Nachteile bietet ein Hash-basierter Index im Vergleich zum B-Baum?

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Herbst 2016

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2016**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **14**

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1

Aufgabe 1: Begriffe und Konzepte

a) Was ist der Unterschied zwischen einem Software-Fault und einem Software-Failure?

b) Was ist der Unterschied zwischen Testing und Model-Checking? Welchen Vorteil hat Testing gegenüber Model-Checking bzw. umgekehrt?

c) Eine Unternehmensberatung stellt fest: „Die Wartungskosten Ihrer Projekte machen bis zu 70% der gesamten Projektkosten aus!" Diskutieren Sie, ob dieser hohe Wert grundsätzlich negativ zu bewerten ist.

d) Welche Phasen werden in jedem Prozess-Modell der Softwareentwicklung durchlaufen?

e) Nennen Sie zwei Vorteile und zwei Nachteile des Wasserfallmodells, jeweils mit Begründung.

f) Nennen Sie zwei Vorteile und zwei Nachteile der agilen Softwareentwicklung, jeweils mit Begründung.

g) Nennen Sie zwei Maßnahmen, mit denen agile Methoden versuchen, eine frühe Validierung zu erreichen. Erläutern Sie zusätzlich kurz, warum eine frühe Validierung von Vorteil ist.

h) Bewerten Sie folgende Aussage eines Projektmanagers: „Falls wir in Verzug geraten, dann können wir jederzeit neue Programmierer hinzuholen, und können die Deadline doch noch einhalten."

Fortsetzung nächste Seite!

Aufgabe 2: Entwurfsmuster

a) **Singleton:** Im Folgenden sehen Sie eine unvollständige Klassendefinition der Klasse Singleton des gleichnamigen Entwurfsmusters. Unter der Klasse befindet sich ein Aufruf, der der Variablen s die Instanz der Klasse zuweist.

Geben Sie den vervollständigten Quelltext der Klassendefinition und des Aufrufs an (nicht auf dem Aufgabenblatt vervollständigen).

```
public class Singleton {
  [      ]static final Singleton INSTANCE = new[      ];

  [      ]Singleton() {}

  [      ][      ]Singleton getInstance() {
      return[      ];
    }
}

Singleton s =[                ];
```

b) **Fehlersuche und Identifikation:** Im Folgenden sehen Sie ein UML-Diagramm eines Entwurfsmusters mit Fehlern.

Geben Sie ein korrektes UML-Diagramm für das gemeinte Entwurfsmuster an und nennen Sie den Namen des nun korrekten Entwurfsmusters (nicht auf dem Aufgabenblatt berichtigen).

Fortsetzung nächste Seite!

Aufgabe 3: „OOA/OOD - UML"

Gegeben sei folgende Beschreibung eines Informationssystems für Parkhäuser:

„Ein Parkhaus kann einen oder mehrere Ein- und Ausgänge haben. Jeder Ein- bzw. Ausgang hat einen entsprechenden Ticketautomaten und eine Schranke. Das Parkhaus hat mehrere Stockwerke. Jedes Stockwerk und das Parkhaus selbst haben eine bestimmte Kapazität sowie je eine Auslastung, die sich ändert, wenn ein Auto in das Parkhaus bzw. in das Stockwerk einfährt oder es wieder verlässt. Um ein solches Ereignis zu erkennen, sind an jedem Eingang, jedem Ausgang und in jedem Stockwerk entsprechende Sensoren angebracht, die dem Parkhaus bzw. dem Stockwerk einzelne ein- bzw. ausfahrende Autos signalisieren. An jedem Stockwerk und über jedem Eingang sind darüber hinaus Displays angebracht, die die Anzahl der freien Parkplätze im jeweiligen Stockwerk bzw. im gesamten Parkhaus anzeigen."

a) Identifizieren Sie in der obigen Beschreibung alle möglichen Kandidaten für Klassen, Attribute und Methoden. Geben Sie jeweils an, welche davon Sie für geeignet halten und begründen Sie bei den anderen kurz, weshalb Sie sie eher verwerfen würden. Ordnen Sie die geeigneten Attribute und Methoden den jeweiligen Klassenkandidaten zu.

b) Erstellen Sie aus den von Ihnen vorangehend als geeignet identifizierten Kandidaten ein UML-Klassendiagramm einschließlich etwaiger Assoziationen zwischen den Klassen.

c) Implementieren Sie die im folgenden UML-Diagramm dargestellten Klassen in einer geeigneten objektorientierten Programmiersprache Ihrer Wahl:

Der Konstruktor von **`GedeckterTisch`** soll **`gedecke`** mit einem neuen Array der Größe **`maxAnzahlGedecke`** initialisieren. Der Konstruktor von **`BesteckKasten`** soll die beiden Array-Attribute mit entsprechenden Arrays der Größe **`groesse`** belegen.
Die **`get`**...-Methoden im **`BesteckKasten`** sollen den jeweiligen Array-Eintrag zurückgeben und stattdessen **`null`** eintragen. Die **`set`**...-Methoden im **`BesteckKasten`** sollen genau dann das übergebene Objekt an der genannten Stelle im Array eintragen und **`true`** zurückgeben, falls dort kein Objekt eingetragen ist - andernfalls sollen sie nichts tun und **`false`** zurückgeben.

Fortsetzung nächste Seite!

Aufgabe 4: „Zustandsautomaten"

Modellieren Sie das Einstellen einer Digitaluhr mit zwei Druckknöpfen mittels eines Zustandsautomaten, so dass Folgendes gilt:

- Nach Einlegen der Batterie befindet sich die Digitaluhr im Startzustand. Sie zeigt dabei zunächst die Uhrzeit „00:00:00" und läuft an (ein „interner Mechanismus" sendet dazu im Sekundentakt ein „tick"-Ereignis).
- Durch Betätigen des Druckknopfes A werden die drei Modi (Stunden stellen, Minuten stellen, Uhrzeitanzeige) zyklisch durchgeschaltet.
- Durch Betätigen des Druckknopfes B in einem der ersten beiden Modi wird die entsprechende Einheit (Stunden, Minuten) zyklisch erhöht – z.B. springt der Stundenzähler nach 21, 22, 23 wieder auf 0, 1, 2, ... Während der Einstellung einer Einheit steht die Uhr (d.h. „tick"s werden ignoriert). Die Betätigung des Druckknopfs B im Modus „Uhrzeitanzeige" ist wirkungslos.

Beschränken Sie sich dabei auf die folgenden drei Zustandsvariablen und stellen Sie die restlichen Zustände explizit graphisch dar:
Stunde: $h \in \{0, 1, 2, \ldots, 23\}$; Minute: $m \in \{0, 1, 2, \ldots, 59\}$; Sekunde: $s \in \{0, 1, 2, \ldots, 59\}$

Fortsetzung nächste Seite!

Teilaufgabe 2

1. ER-Modellierung

Gegeben seien folgende Informationen:

- Zirkusse geben Vorführungen. Zu jedem Zirkus ist dessen eindeutiger Name bekannt. Eine Vorführung wird durch ihr Datum gekennzeichnet, welches jedoch nur innerhalb eines Zirkus eindeutig ist.
- Es gibt Tickets für Vorführungen. Ein Ticket besitzt eine Kategorie und eine innerhalb einer Vorführung eindeutige Nummer.
- Für jeden Zirkus arbeiten Künstler. Für ein solches Beschäftigungsverhältnis ist die jeweilige Gage bekannt. Kein Künstler kann für mehrere Zirkusse arbeiten.
- Künstlern wird eine Personal-Nummer zugeordnet. Außerdem können Künstler in genau drei Kategorien eingeteilt werden: Akrobaten, Dresseure und Clowns. Manche Clowns sind auch Akrobaten.
- Es sind die Verantwortlichkeiten von Darbietungen innerhalb einer Vorführung folgendermaßen bekannt: Für Darbietungen in einer Vorführung ist immer nur ein Künstler verantwortlich. Für die gleiche Darbietung in verschiedenen Vorführungen können auch verschiedene Künstler verantwortlich sein. Zudem ist eine Darbietung über Ihren Namen eindeutig gekennzeichnet und besitzt zudem eine Dauer.
- Jeder Künstler erhält kostenlose Tickets.
- Manche Zirkusse drucken Werbeplakate. Ein Plakat besitzt eine ID und bewirbt höchstens einen Zirkus. Auf machen Plakaten sind ein oder mehrere Clowns abgebildet.

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

Gegeben sei der folgende Ausschnitt des Schemas eines Flugverwaltungssystems:

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

Airline : {[
ID : CHAR(2),
Name : VARCHAR(100),
Sitz : VRACHAR(100)
]}

Flug : {[
Airline : CHAR(2),
Nummer : INTEGER,
Datum : DATE,
Start : VARCHAR(100),
Ziel : VARCHAR(100),
Entfernung: INTEGER,
Flugzeit: REAL
]}

Ticket : {[
PassagierID : INTEGER,
Airline : CHAR(2),
Nummer : INTEGER,
Datum : DATE,
Preis : REAL,
Sitz : CHAR(3)
]}

Pilot : {[
PilotID : INTEGER,
Airline : CHAR(2),
Nummer : INTEGER,
Datum : DATE
]}

Die Tabelle *Person* enthält Informationen über Passagiere und Piloten. *Airline* beschreibt Airlines, die Flüge anbieten, mit ihrer Kurzbezeichnung, ihrem Namen und ihrem (Firmen-)Sitz. In *Flug* wird abgelegt, welche Flüge von welcher Airline an welchem Tag durchgeführt werden. *Ticket* beschreibt, welche Passagiere auf welchem Flug gebucht sind. *Pilot* beinhaltet die Piloten der Flüge.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Ticket* mit allen Constraints (einschließlich Fremdschlüsselconstraints) anlegt. Stellen Sie dabei sicher, dass negative Preise ausgeschlossen sind. Erklären Sie kurz, warum *Sitz* Teil des Primärschlüssels ist.

b) Schreiben Sie eine SQL-Anweisung, die für jede Airline die Flugnummer bestimmt, mit der im Jahr 2014 die größten Einnahmen erzielt wurden. *Hinweis: Verwenden Sie die Funktion EXTRACT(YEAR FROM Datum), um das Jahr aus Attribut Datum zu extrahieren.*

c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Passagiere bestimmt, die nur mit Airlines geflogen sind, deren Sitz an ihrem Wohnort ist.

d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Passagiere bestimmt, die immer mit dem gleichen Kapitän geflogen sind.

e) Schreiben Sie eine SQL-Anweisung, die alle Städte bestimmt, die mit maximal einem Zwischenstopp von 'München' aus erreichbar sind.

Beachten Sie bei der Formulierung der SQL-Anweisungen, 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:

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

und die Zerlegung $\rho = \{R_1, R_2\}$ von R mit $R_1 = \{A, D\}$ und $R_2 = \{B, C, D\}$.
Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$\begin{aligned} AB &\rightarrow CD, \\ B &\rightarrow C, \\ D &\rightarrow A, \end{aligned}$$

$\}$

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) Zeigen oder widerlegen Sie, dass die Zerlegung ρ von R abhängigkeitserhaltend bzgl. FD *ist.*

d) Zeigen oder widerlegen Sie, dass die Zerlegung ρ von R verlustfrei bzgl. FD ist.

e) Bestimmen Sie eine verlustfreie Zerlegung von R in BCNF-Relationen.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1

Aufgabe 1: Modellierung von statischen Strukturen durch Klassendiagramme

Entwerfen Sie ein UML-Klassendiagramm mit den Klassen

Schule, Lehrerzimmer, Direktor, Klassenzimmer, Schüler, Lehrer,
Bibliothek, Raum, Mitarbeiter, Buch, Person

und mit geeigneten Assoziationen und Vererbungsbeziehungen. Assoziationen sind mit einem Namen zu versehen und an allen Assoziationsenden sind Multiplizitäten anzubringen. Verwenden Sie, wo möglich, abstrakte Klassen und kennzeichnen Sie diese. Es brauchen keine Attribute und Operationen angegeben zu werden.

Fortsetzung nächste Seite!

Aufgabe 2: Modellierung von Interaktionen durch Sequenzdiagramme

Die Fahrtrichtung der ersten elektrischen Spielzeugeisenbahnen wurde häufig durch Stromunterbrechung gesteuert. Dazu betrachten wir das Anwendungsfall-Diagramm in Abb. 1.

Abbildung 1: Anwendungsfall-Diagramm

An dem Anwendungsfall „Steuere Lokomotive" sind ein Spieler, als Aktor außerhalb des Systems, und jeweils ein Objekt der Klassen Stromschalter, Lokomotive, Scheinwerfer und Räder, als Objekte innerhalb des Systems, beteiligt. Zur Vereinfachung werden alle vier Räder durch ein einziges Objekt der Klasse Räder modelliert. Der Anwendungsfall zum Steuern einer Lokomotive wird durch folgendes Szenario beschrieben.

1. Der Spieler schaltet den Stromschalter ein, woraufhin der Schalter der Lokomotive *Strom zuführt*.
2. Die Lokomotive schickt nun den Rädern ein Signal um *vorwärts* zu fahren.
3. Dann schaltet der Spieler den Stromschalter aus, woraufhin der Schalter die Stromzufuhr bei der Lokomotive *abstellt*.
4. Daraufhin schickt die Lokomotive das Signal *stop* an die Räder.
5. Der Spieler schaltet jetzt den Stromschalter wieder ein, woraufhin der Schalter der Lokomotive *Strom zuführt*.
6. Die Lokomotive schickt den Rädern ein Signal um *rückwärts* zu fahren.
7. Nun schaltet der Spieler den Stromschalter wieder aus, woraufhin der Schalter die Stromzufuhr bei die Lokomotive *abstellt*.
8. Daraufhin schickt die Lokomotive wieder das Signal *stop* an die Räder.

Geben Sie ein Sequenzdiagramm an, das die oben beschriebenen Interaktionen zwischen Spieler, Stromschalter, Lokomotive und Rädern beschreibt.

Fortsetzung nächste Seite!

Aufgabe 3: Implementierung

Gegeben sei das folgende UML Klassendiagramm. Der Konstruktor der Klasse `Firma` soll den Namen der Firma initialisieren. Ein neu erzeugtes Firmenobjekt soll keine Mitarbeiter haben. Die Operation namens `add` der Klasse `Firma` soll einen Mitarbeiter zu dem Personal einer Firma hinzufügen. Die Konstruktoren der Klassen `FestAngestellter` und `FreierMitarbeiter` sollen die für die Objekte der jeweiligen Klasse gültigen Attribute initialisieren. Die Operationen namens `getGehalt` und `setGehalt` sollen das Gehalt eines fest angestellten Mitarbeiters liefern bzw. neu setzen.

a) Implementieren Sie das angegebene Klassendiagramm in einer objektorientierten Programmiersprache Ihrer Wahl. Die verwendete Sprache ist zu nennen. Beachten Sie dabei auch die im Klassendiagramm angegebenen Sichtbarkeiten. Insbesondere spezifiziert das Symbol # eine „protected" Sichtbarkeit. Für die Realisierung des Rollennamens `personal` und der Operation `add` der Klasse `Firma` kann ein geeigneter Kollektionstyp namens `HashSet` vorausgesetzt werden, der eine Methode namens `add` anbietet, die ein Objekt zu einer Kollektion hinzufügt, und einen Standardkonstruktor `HashSet()` besitzt, der ein Objekt erzeugt, das eine leere Menge repräsentiert. Alternativ dazu können Sie auch partielle Arrays zur Implementierung verwenden.

b) Schreiben Sie eine main-Methode mit Kopf `public static void main(String [ ] args)`, so dass nach Ausführung der Methode eine Firma namens „Magnifico" existiert, die eine fest Angestellte namens „Martha Huber" mit Nummer 1 und einem Gehalt von 4000 Euro sowie einen freien Mitarbeiter namens „Hugo Frost" mit Nummer 2 und einem Stundensatz von 100 Euro hat.

c) Erläutern Sie den Begriff *Polymorphismus* anhand eines Beispiels aus der obigen Anwendung.

Fortsetzung nächste Seite!

Aufgabe 4: Formale Verifikation mit vollständiger Induktion

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `cn` nur für Werte `n` ≥ 0 aufgerufen wird.

```
int cn(int n) {
if (n == 0)
return 1;
else
return (4*(n-1) + 2) * cn(n-1) / (n+1);
}
```

Sie können im Folgenden vereinfachend annehmen, dass es keinen Überlauf in der Berechnung gibt, d.h. dass der Datentyp `int` für die Berechnung des Ergebnisses stets ausreicht.

a) Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `cn(n)` für jedes `n` ≥ 0 die n-te Catalan-Zahl C_n berechnet, wobei

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

Beim Induktionsschritt können Sie die beiden folgenden Gleichungen verwenden:

(i) $(2(n+1))! = (4n+2) \cdot (n+1) \cdot (2n)!$
(ii) $(n+2)! \cdot (n+1)! = (n+2) \cdot (n+1) \cdot (n+1)! \cdot n!$

b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie, warum der Methodenaufruf `cn(n)` für jedes `n` ≥ 0 terminiert.

Fortsetzung nächste Seite!

Teilaufgabe 2

1. Sie sollen für die Fluggesellschaft „FanHansa“ ein Fluginformationssystem erstellen. Als Grundlage des Entwurfs dient das folgende rudimentäre Entity-Relationship-Diagramm:

Zeichnen Sie das ER-Diagramm ab und vervollständigen Sie es mit

- Funktionalitäten (1:1, 1:N, N:1, N:M) und
- (min, max) – Angaben.

2. Setzen Sie das ER-Diagramm in ein relationales Schema um. Beachten Sie dabei:

- Fassen Sie, wenn möglich, Relationen zusammen.
- Markieren Sie Primärschlüssel.
- Markieren Sie die Fremdschlüssel.

3. Geben Sie die Statements in SQL an, welche die Relationen aus 2. (inklusive Primär- und Fremdschlüsseldefinitionen) in eine relationale Datenbank implementieren.

4. Formulieren Sie in SQL folgende Anfragen auf Basis Ihrer oben erstellten Fluginformationssystem-Datenbank.
   a. Welche Städte kann man von München aus direkt erreichen?
   b. Welche Städte kann man von München aus mit maximal einmaligem Umsteigen erreichen?

Fortsetzung nächste Seite!

5. Gegeben sei folgendes ER-Diagramm:

Ein nicht-versierter Datenbank-Laie hat daraus das folgende relationale Schema generiert:

AllesZusammen: {[SNR, SN, VNR, VN, PNR, PN]}

a. Bestimmen Sie alle nicht-trivialen funktionalen Abhängigkeiten der Relation *AllesZusammen.*

b. Bestimmen Sie alle Kandidatenschlüssel der Relation *AllesZusammen.*

c. In welcher höchsten Normalform befindet sich das Schema?

d. Nomalisieren Sie die Relation. Sie dürfen wahlweise den Synthese- oder den Dekompositionsalgorithmus anwenden.

e. In welcher höchsten Normalform befinden sich Ihre generierten Schemata? Begründen Sie Ihre Antwort mithilfe der zugeordneten funktionalen Abhängigkeiten.

6. Wofür stehen die folgenden Abkürzungen?

- ACID
- WAL-Prinzip
- LSN

Erläutern Sie jeweils die Bedeutung kurz. Erläutern Sie auch kurz das Konzept der Serialisierbarkeit.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2017

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	**Frühjahr 2017**	**46116**
Arbeitsplatz-Nr.: _____		

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: **15**

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1: Begriffe und Konzepte

a) Erläutern Sie kurz den Unterschied zwischen starker und schwacher Kohäsion einer Softwarekomponenten.

b) Erläutern Sie kurz den Unterschied zwischen starker und schwacher Kopplung zwischen Softwarekomponenten.

c) Wie wirkt sich der Einsatz von public-Instanzvariablen in einer Klasse auf die Kopplung im System aus und warum?

d) Wie wirkt sich der Einsatz des Entwurfsmusters MVC auf Kopplung und Kohäsion des Systems aus und warum?

e) Nennen Sie drei Merkmale des Extreme Programming.

f) Was versteht man unter dem Liskovschen Substitutionsprinzip?

g) Welche Arten von Softwarewartung werden unterschieden?

Fortsetzung nächste Seite!

Aufgabe 2: Use-Case-Diagramm

Im Folgenden sehen Sie ein fehlerhaftes Use-Case-Diagramm für das System „Geldautomat". Geben Sie ein korrektes Use-Case-Diagramm (Beziehungen und Beschriftungen) entsprechend der folgenden Beschreibung an (nicht auf dem Aufgabenblatt berichtigen). Sollte es mehrere Möglichkeiten geben, begründen Sie Ihre Entscheidung kurz.

Kunden können am Geldautomaten verschiedene Services nutzen, es kann Geld abgehoben und eingezahlt sowie der Kontostand abgefragt werden. Für jeden dieser Services ist eine Authentifizierung notwendig. Diese besteht aus der Prüfung der Bankkarte und des eingegebenen PINs. Manche Bankautomaten können Karten bei zu vielen Fehlversuchen (3) bei der Anmeldung sperren, andere geben Fehlermeldungen aus, falls die Karte gesperrt wurde oder nicht mehr genügend Geld auf dem Konto oder im Bankautomaten vorhanden ist. Mitarbeiter bzw. Mitarbeiterinnen der Bank können sich ebenfalls über PIN und Karte authentifizieren, um dann den Bankautomaten neu zu starten oder mit Geld aufzufüllen. Bevor Geld abgehoben werden kann, ist sicherzustellen, dass auf dem Konto und im Bankautomaten genügend Geld vorhanden ist.

Fortsetzung nächste Seite!

Aufgabe 3: UML Diagramme

Betrachten Sie die folgenden UML-Diagramme. Sind diese korrekt? Falls nein, begründen Sie warum nicht. Geben Sie in diesem Fall außerdem eine korrigierte Version an.

a)

b)

Fortsetzung nächste Seite!

Aufgabe 4: Entwurfsmuster

In einem Abenteuerspiel soll ein Held Monster bekämpfen. Zu Beginn ist die Spielfigur schwach und kann als Attacke nur mit den Fäusten boxen. Im Laufe des Spiels erhält der Charakter diverse Items und wird dadurch zu einer stärkeren Figur. Bei jeder Attacke auf ein Monster werden alle Items, die er erworben hat, der Reihe nach eingesetzt.
Der folgende Code zeigt einen Ausschnitt aus der Implementierung.
Welches Designpattern wurde hier eingesetzt? Zeichnen Sie das zugehörige UML-Diagramm, das genau die im Code beschriebene Situation wiedergibt.

```
interface Spielfigur {
    public void attackieren();
}
```

```
class EinfacheFigur implements Spielfigur {
    public void attackieren() {
        // boxen();
    }
}
```

```
abstract class BewaffneteFigur implements Spielfigur {
    protected Spielfigur figur;

    public BewaffneteFigur (Spielfigur figur) {
        this.figur = figur;
    }
    public void attackieren() {
        figur.attackieren();
    }
}
```

```
class FigurMitHammer extends BewaffneteFigur {
    public FigurMitHammer (Spielfigur figur) {
        super(figur);
    }
    public void attackieren() {
        // mitHammerZuschlagen();
        super.attackieren();
    }
}
```

```
class FigurMitTraenengas extends BewaffneteFigur {
    public FigurMitTraenengas (Spielfigur figur) {
        super(figur);
    }
    public void attackieren() {
        // spruehen();
        super.attackieren();
    }
}
```

```
class FigurMitWasserpistole extends BewaffneteFigur {
    public FigurMitWasserpistole (Spielfigur figur) {
        super(figur);
    }
    public void attackieren() {
        // nasspritzen();
        super.attackieren();
    }
}
```

Fortsetzung nächste Seite!

Aufgabe 5: Projektmanagement

Betrachten Sie die folgende Tabelle zum Projektmanagement:

Arbeitspaket	Dauer (Tage)	Abhängig von
A1	5	
A2	10	
A3	5	A1
A4	15	A2, A3
A5	10	A1
A6	10	A1, A2
A7	10	A2, A4
A8	15	A4, A5

Tabelle 1: Übersicht Arbeitspakete

a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.

b) Wie lange dauert das Projekt mindestens?

c) Geben Sie den oder die kritischen Pfad(e) an.

d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

Aufgabe 6: Entwurfsmuster Abstract Factory

Zeichnen Sie das Strukturdiagramm des Entwurfsmusters Abstract Factory. Erklären Sie die Wirkungsweise und den Vorteil der Verwendung dieses Entwurfsmusters.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanksysteme

Aufgabe 1: Grundlagen

a) Grenzen Sie die Begriffe Datenbanksystem, Datenbankverwaltungssystem und Datenbank in ein bis zwei Sätzen voneinander ab.

b) Erläutern Sie in ein bis zwei Sätzen, was man unter physischer Datenunabhängigkeit versteht.

c) Erläutern Sie in ein bis zwei Sätzen, was ein Trigger ist. Nennen Sie weiterhin einen beispielhaften Einsatzzweck eines Triggers.

Aufgabe 2: Konzeptioneller Entwurf

Der Metadatenkatalog eines Datenbankverwaltungssystems wird meist ebenfalls in Relationen abgelegt. Im Folgenden finden Sie die Beschreibung eines Ausschnitts des Metadatenkatalogs eines Datenbankverwaltungssystems. Erstellen Sie zu dieser Beschreibung ein ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Modellieren Sie keine Attribute oder Entitytypen, die nicht aus dem Text hervorgehen.

Die Datenbank enthält Datenobjekte, die genau einem Benutzer gehören und durch den Benutzer zusammen mit einem für diesen Benutzer eindeutigen Namen identifiziert werden. Ein Benutzer hat ein Passwort und einen eindeutigen Benutzernamen. Benutzer können anderen Benutzern Zugriffsrechte auf Datenobjekte geben. Dabei wird auch die Berechtigungsart gespeichert. Datenobjekte sind entweder Tabellen oder Indexe. Eine Tabelle kann verschiedene Spalten enthalten. Jede Spalte gehört zu genau einer Tabelle und hat einen Datentyp sowie einen innerhalb ihrer Tabelle eindeutigen Namen. Indexe haben eine Gültigkeit. Es gibt B-Baum- und Hash-Indexe. Zu einem B-Baum-Index wird die aktuelle Höhe abgelegt, zu einem Hash-Index der aktuelle Belegungsfaktor. Jeder Index und jede Tabelle ist in genau einem eigenen Segment abgelegt. Segmente werden durch eine eindeutige Nummer identifiziert und haben eine Größe. Jedes Segment ist in einer Datei abgelegt. Eine Datei kann mehrere Segmente enthalten. Dateien haben einen eindeutigen Pfad.

Aufgabe 3: SQL

Gegeben sind folgende Relationen aus der Verwaltungssoftware eines Fahrradverleihs:

```
Fahrrad (FahrradID, Marke, Modell, Tagesgebuehr)
Kunde (KundenID, Vorname, Nachname)
Ausleihe (KundenID[Kunde], FahrradID[Fahrrad], Entleihdatum, Rueckgabedatum)
```

a) Schreiben Sie eine SQL-Abfrage, die zu allen Ausleihen das Entleihdatum zusammen mit dem Nachnamen des Kunden ausgibt, der die Ausleihe getätigt hat. Die Ausgabe soll absteigend nach Nachnamen sortiert sein.

Fortsetzung nächste Seite!

b) Schreiben Sie eine SQL-Abfrage, die für jeden Kunden die Ausleihe mit der längsten Dauer (= Rueckgabedatum – Entleihdatum) ermittelt und deren Dauer zusammen mit der KundenID ausgibt. Die Ausgabe soll nur Ausleihdauern größer 7 enthalten.

c) Schreiben Sie eine SQL-Abfrage, die FahrradID und Entleihdatum aller Ausleihen zusammen mit der für die jeweilige Ausleihe insgesamt anfallenden Ausleihgebühr ausgibt. Für den Entleihtag wird keine Gebühr berechnet, für den Rückgabetag und alle anderen Tage wird jeweils die Tagesgebuehr berechnet.

d) Es soll ermittelt werden, ob die Tabelle Ausleihe Datensätze mit unmöglichen Ausleihvorgängen enthält.

Hierzu sollen FahrradID und Entleihdatum aller Datensätze der Tabelle Ausleihe ausgegeben werden, bei denen sich die Entleihzeit mit der Entleihzeit anderer Datensätze für die gleiche FahrradID überschneidet. Das Entleihen eines Fahrrads am gleichen Tag seiner Rückgabe aus einem anderen Ausleihvorgang gilt nicht als Überschneidung.

Geben Sie eine entsprechende SQL-Abfrage an.

e) Schreiben Sie eine SQL-Abfrage, die für alle Fahrräder der Marke „Schnelles Rad" die Tagesgebuehr um 3% erhöht.

Aufgabe 4: Physische Datenorganisation

a) Erläutern Sie in ein bis zwei Sätzen, aus welchen zwei Teilen sich ein TID (Tupel-Identifikator) zusammensetzt.

b) Erläutern Sie in ein bis zwei Sätzen das Vorgehen, wenn ein durch einen TID adressierter Satz innerhalb einer Seite verschoben werden muss.

c) Erläutern Sie in ein bis zwei Sätzen das Vorgehen, wenn ein durch einen TID adressierter Satz erstmalig in eine andere Seite verschoben werden muss.

d) Erläutern Sie in zwei bis drei Sätzen das Vorgehen, wenn ein durch einen TID adressierter und bereits einmal über Seitengrenzen hinweg verschobener Satz erneut in eine andere Seite verschoben werden muss.

e) Nennen Sie zwei Gründe, warum der folgende Baum kein gültiger B-Baum ist. Beziehen Sie sich dabei auf konkrete Stellen im gegebenen Baum.

Fortsetzung nächste Seite!

Aufgabe 5: Transaktionen

a) Nennen Sie die Bezeichnungen der vier wesentlichen Eigenschaften des Transaktionskonzepts und erläutern Sie jede Eigenschaft in ein bis zwei Sätzen.

b) Erläutern Sie in ein bis zwei Sätzen, ob innerhalb einer Transaktion ein inkonsistenter Zustand vorliegen darf.

c) Gegeben ist die folgende Historie (Schedule) dreier Transaktionen:
$w_1(O) \rightarrow w_1(O) \rightarrow r_3(P) \rightarrow r_1(P) \rightarrow c_1 \rightarrow r_2(T) \rightarrow w_3(T) \rightarrow w_2(P) \rightarrow w_2(O) \rightarrow c_2 \rightarrow c_3$

Zeichnen Sie den Serialisierbarkeitsgraphen zu dieser Historie und begründen Sie, warum die Historie serialisierbar ist oder nicht.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1: „Begriffe und Konzepte"

a) Nennen Sie vier Qualitätsmerkmale von Software.

b) Was versteht man unter dem Begriff „Refactoring"?

c) Wie stehen „Refactoring" und „Regression-Testing" in Beziehung?

d) Nennen Sie drei Vorteile für den Einsatz von Software zur Versionsverwaltung.

Fortsetzung nächste Seite!

Aufgabe 2: „Testen"

Gegeben sei folgende Java-Methode:

```
static int test(int x) {
        int y = 0;
        if (x < 0) {
                x = -x;
        }
        do {
                if (x == 0) {
                        x++;
                } else if (x % 2 == 1) {
                        x--;
                } else {
                        x = x/2;
                        y++;
                }
        } while(x > 0);
        return y;
}
```

a) Konstruieren Sie den Kontrollflussgraphen des obigen Code-Fragments.

b) Nennen Sie die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe, und geben Sie an, wie Sie den Wert ermittelt haben.

c) Geben Sie einen Testdatensatz mit möglichst wenigen Testfällen (d.h. jeweils den Aufrufparameter der Methode) an, der eine 100%-ige Verzweigungsüberdeckung der Methode **test(x)** erzielt. Welche Pfade werden dabei jeweils durchlaufen?

d) Welche Pfade sind **im folgenden** Kontrollflussgraphen einer anderen Methode zu überdecken, um eine vollständige „Schleife-Inneres-Überdeckung" („boundary interior coverage") zu erreichen?

Fortsetzung nächste Seite!

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

Aufgabe 3: „Entwurfsmuster"

Beschreiben Sie das Entwurfsmuster „Zustandsmuster", das zur Kapselung unterschiedlicher, zustandsabhängiger Verhaltensweisen eines Objektes eingesetzt wird. Geben Sie dazu ein Klassendiagramm an (mindestens drei verschiedene Zustände). Zeichnen Sie in diesem auch die Interaktion mit der das Muster nutzenden Klasse ein. Was ist der Hauptvorteil im Vergleich zu einer Umsetzung von zustandsabhängigen Operationen mittels Bedingungen?

Aufgabe 4: „Zustandsdiagramm und Klassendiagramm"

a) Beschreiben Sie einen Protokollzustandsautomat für die Klasse Buch mit den Operationen ausleihen(Leser), zurückgeben(), vorbestellen(Leser) und den Buchzuständen BuchPräsent, BuchAusgeliehen, BuchVorbestellt, BuchReserviert (d.h. ein vorbestelltes Buch wird für den Leser, der es als erstes vorbestellt hat, zur Abholung bereitgestellt). Es kann beliebig viele Vorbestellungen geben. Der Fall, dass ein reserviertes Buch nicht abgeholt wird, ist nicht zu betrachten. In Ihrem Diagramm sollen die Zustandswechsel bei den Operationen vollständig dargestellt werden. Geben Sie auch an, welche Attribute und Assoziationen für die Buchzustände erforderlich sind, um die Zuordnung zu den Lesern zu verwalten.

b) Geben Sie zu dem Protokollzustandsautomat das zugehörige Klassendiagramm gemäß Anwendung des Zustandsmusters aus der ersten Teilaufgabe an. Ein Buch hat dabei Titel und Autor. Der Leser ist als eigener Typ zu modellieren.

c) Geben Sie weiterhin für eine Operation ausleihen(Leser) im Zustand BuchPräsent den zugehörigen Code in einer Notation Ihrer Wahl an (z.B. PseudoCode, Java, Aktivitätsdiagramm).

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanksysteme

1. Aufgabe

Gegeben sei eine Relation GRUPPEN(GruppenId, MatrNr, Datum) in der vermerkt wird, welche Studierende an welchem Termin an welcher Übungsgruppe teilgenommen haben. Folgende Bedingungen sollen gelten:

- Dieselbe Gruppe trifft sich nie zwei Mal am gleichen Tag.
- Eine Gruppe wird dann abgehalten, wenn mindestens ein Teilnehmer anwesend ist.
- Studierende dürfen die Gruppe nicht wechseln.

Beispielausprägung:

GRUPPEN

GruppenID	MatrNr	Datum
1	123	2014-10-01
1	456	2014-10-01
1	456	2014-10-08
2	147	2014-10-02
2	258	2014-10-09
2	369	2014-10-09
3	159	2014-10-10

- Schreiben Sie eine SQL-Abfrage, die überprüft, dass die letztgenannte Bedingung (Studierende dürfen die Gruppe nicht wechseln) auch tatsächlich eingehalten wurde. Falls nicht, sollen die Studenten ausgegeben werden, die die Bedingung verletzt haben; ansonsten ein leeres Ergebnis.
- Schreiben Sie eine SQL-Abfrage, die für alle Studierenden (MatrNr) ausgibt, wie häufig sie an einer Übungsgruppe teilgenommen haben.
- Schreiben Sie eine SQL-Abfrage, die für alle Studierenden ausgibt, wie häufig sie in ihrer Übungsgruppe gefehlt haben.
- Geben Sie eine SQL-Abfrage an, welche die Gruppen ausgibt, bei denen nie ein Studierender gefehlt hat, wenn die Gruppe abgehalten wurde.

Fortsetzung nächste Seite!

2. Aufgabe

Folgendes ER-Diagramm sei gegeben:

a) Übernehmen Sie das ER-Diagramm in Ihre Aufgabenbearbeitung und vervollständigen Sie es dort um Funktionalitätsangaben.

Ein „Datenbank-Laie" hat das obige ER-Diagramm in folgendes Relationenschema übersetzt:

UniWelt: {[MatrNr, SName, FreundMatrNr, VorlNr, Titel, PersNr, PName]}

b) Bestimmen Sie alle funktionellen Abhängigkeiten, geben Sie alle Schlüsselkandidaten der Relation UniWelt an und normalisieren Sie das Relationsschema UniWelt.
Dokumentieren Sie alle Schritte Ihres Vorgehens nachvollziehbar und geben Sie an, welche funktionalen Abhängigkeiten (FDs) in den generierten Schemata gelten.

c) Formulieren Sie auf der Basis der normalisierten Relationen folgende Abfragen in SQL:
   - Welche befreundeten Studenten-Paare hören dieselbe Vorlesung?
   - Wer hört mindestens zwei unterschiedliche Vorlesungen, die der Professor namens „James Bond" anbietet?

Fortsetzung nächste Seite!

3. Aufgabe

Fügen Sie in einen anfangs leeren B-Baum mit der Knotenkapazität 4 folgende Daten ein:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

Geben Sie auch die relevanten Zwischenergebnis-Bäume an.

Löschen Sie anschließend zuerst die 12 und danach die 4 aus dem Baum und geben Sie den Baum nach dem jeweiligen Löschvorgang an.

Beantworten Sie zusätzlich folgende Fragen:

- Welche Vorteile bietet ein B-Baum gegenüber einer Hash-basierten Indexierung?
- Welchen Vorteil bieten Hash-basierte Indexe im Vergleich zum B-Baum?

4. Aufgabe

Erklären Sie in kurzen Texten folgende Begriffe:

- WAL-Prinzip
- ACID
- Serialisierbarkeit
- physische Datenunabhängigkeit
- Sichten-Konsolidierung

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Herbst 2017

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2017**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **14**

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1: Grundwissen Softwaretechnik

1. Welche zwei Eigenschaften unterscheiden die Komposition von der Aggregation?
2. Nennen Sie drei Merkmale, die Schnittstellen (*interface*) und abstrakte Klassen (*abstract class*) voneinander unterscheiden!
3. Geben Sie die Laufzeit für die lineare Suche an. Nennen Sie ein effizienteres Verfahren unter der Annahme, dass die Eingabedaten sortiert sind. Was ist dessen Laufzeit?
4. Nennen Sie je einen Vorteil für verkettete Listen und für Arrays!

Aufgabe 2: UML

Betrachten Sie das folgende UML-Diagramm:

1. Geben Sie zu jeder Beziehung die Multiplizitäten der Beziehungen an. (Nicht ins Diagramm eintragen!)
2. Setzen Sie Person, Kreditkarte und Bank mit Attributen und Beziehungen in einer objektorientierten Programmiersprache Ihrer Wahl um. Sie können Typen aus der jeweiligen Standard-Bibliothek verwenden (zum Beispiel: Set). Methoden sind nicht verlangt.

Fortsetzung nächste Seite!

Aufgabe 3: Systementwicklung

Sie wurden beauftragt, ein neues Bibliotheksverwaltungssystem zu entwickeln. Dieses System soll den Bücherbestand verwalten sowie Ausleih- und Vormerkungsfunktionalität bieten.
Rahmendaten zum Projekt: Von Seiten des Auftraggebers wird großer Wert auf besonders hohe Sicherheitseigenschaften des Systems gelegt. Im Rahmen des Projekts sollen von Anfang an Prototypen den Projektfortschritt dokumentieren. Zur Entwicklung dieses Systems steht Ihnen ein Team von sechs Entwicklern zur Verfügung. Das System muss innerhalb von 8 Monaten fertig gestellt sein.

1. Nennen Sie drei Stakeholder des Bibliotheksverwaltungssystems und jeweils zwei Anwendungsfälle!
2. Welches der Vorgehensmodelle V-Modell XT, Spiralmodell und Extreme Programming würden Sie unter den geschilderten Umständen für die Entwicklung dieses Systems anwenden? Treffen Sie eine eindeutige Entscheidung und begründen Sie diese mit drei schlüssigen Argumenten!
3. Das zu entwickelnde Softwaresystem sollte die Qualitätseigenschaften Benutzbarkeit, Zuverlässigkeit und Wartbarkeit erfüllen. Geben Sie zu jeder dieser Qualitätseigenschaften je zwei Qualitätssicherungsmaßnahmen an.
4. Das von Ihnen zu entwickelnde System soll ein derzeit in Betrieb befindliches System ablösen. Das bisherige System verwendet ein Chipkartensystem zur Identifikation der Bibliotheksbenutzer. Im Zuge der Systemumstellung soll ein neues Kartensystem eingeführt werden. Um nicht sofort alle alten Karten ersetzen zu müssen, soll das neue Bibliotheksverwaltungssystem die alten Karten weiterhin unterstützen. Zeichnen Sie ein Klassendiagramm, um unter Verwendung des Patterns TemplateMethod (s.u.) das konkrete Einlesen einer Karte mit beiden Systemen zu ermöglichen.

   Intention des TemplateMethod-Patterns: Definiere das Gerüst eines Algorithmus in einer Methode wobei die Implementierung einiger Schritte in Unterklassen ausgelagert wird. Template-Method lässt die Unterklassen bestimmte Schritte in einem Algorithmus redefinieren, ohne die Struktur des Algorithmus zu verändern.

Fortsetzung nächste Seite!

Aufgabe 4: Qualitätssicherung

Ein gängiger Ansatz zur Messung der Qualität von Software ist das automatisierte Testen von Programmen. Im Folgenden werden praktische Testmethoden anhand des nachstehend angegebenen Sortieralgorithmus diskutiert.

Algorithmus 1 Bubble Sort

```
   void bubblesort(int[] array, int len) {
1      for (int i = 0 ; i < len - 1 ; i++) {
2         for (int j = 0 ; j < len - 1 ; j++) {
3            if ( array[j] > array[j + 1] ) {
4               int temp = array[j];
5               array[j] = array[j + 1];
6               array[j + 1] = temp;
            }
         }
      }
   }
```

a) Nennen Sie eine Art des Black-Box-Testens und beschreiben Sie deren Durchführung anhand des vorgegebenen Algorithmus.

b) Zeichnen Sie ein mit Zeilennummern beschriftetes Kontrollflussdiagramm für den oben angegebenen Sortieralgorithmus.

c) Erklären Sie, ob eine vollständige Pfadüberdeckung für die gegebene Funktion möglich und sinnvoll ist.

Teilaufgabe II: Datenbanksysteme

Aufgabe 1: Grundwissen Datenbanksysteme

1. Nennen Sie den Namen **des Algorithmus**, der ein Relationenschema in einer Normalform in eine höhere Normalform überführt und dabei **abhängigkeitsbewahrend** ist.
2. Nennen Sie drei Gründe für den Einsatz eines Datenbanksystems. (3 Stichworte)

   a) _____

   b) _____

   c) _____
3. Geben Sie außerdem drei Gründe an für den Einsatz eines Datenbanksystem **für die Datenverwaltung**.
4. Geben Sie die Laufzeit von Einfüge- und Löschoperationen in erweiterbaren Hashtabellen an. (Stichwort)
5. Nennen Sie eine Indexstruktur, die sich anbietet, wenn viele Bereichsanfragen auf der zu indizierenden Spalte (Range Queries) erwartet werden. (Stichwort)

Fortsetzung nächste Seite!

Aufgabe 2: Relationale Modellierung

Gegeben das folgende ER-Diagramm:

Nehmen Sie an, es handle sich bei

X(A;B)

Y(D;C;A)

um die einzige gültige vollständig vereinfachte Überführung des Modells in das relationale Modell.

1. Geben Sie Funktionalitätsangaben für X und Y (siehe Boxen) entsprechend der gegebenen relationalen Modellierung an **(Nicht direkt in das Diagramm eintragen!).**
2. Geben Sie *SQL*-92 Statements an, die die hier modellierten Relationen erzeugen. Nehmen Sie an, alle Attribute seien vom Typ *integer*.
3. Im Folgenden wird ein Bibliotheksverwaltungssystem beschrieben:
   - Es gibt Bücher, welche über ihre ISBN und ExemplarNummer identifiziert werden können. Jedes Buchexemplar hat einen Titel.
   - Es gibt Bibliotheken, welche über ihre Straße und Hausnummer identifiziert werden können. Eine Bibliothek verleiht Bücher.
   - Es gibt Personen. Personen werden über ihren Namen und ihr Geburtsdatum identifiziert. Eine Person kann eine beliebige Anzahl an Bibliotheken besuchen und Bücher ausleihen. Es soll möglich sein, dass sich Personen untereinander kennen.
   - Ein Buch kann zu einem gegebenen Zeitpunkt maximal von einer Person ausgeliehen sein.

   Erstellen Sie für das oben beschriebene System ein ER-Diagramm. Tragen Sie die Funktionalitäten in Ihr Diagramm ein und markieren Sie die Primärschlüssel durch Unterstreichen.

Fortsetzung nächste Seite!

Aufgabe 3: Relationale Anfragesprachen

Gegeben ist folgendes Universitätsschema (eine beispielhafte Ausprägung hängt der Klausur an):

Studenten : {[MatrNr : integer, Name : string, Semester : integer]}
Vorlesungen : {[VorlNr : integer, Titel : string, SWS : integer, gelesenVon : integer]}
Professoren : {[PersNr : integer, Name : string, Rang : string, Raum : integer]}
Assistenten : {[PersNr : integer, Name : string, Fachgebiet : string, Boss : integer]}
hören : {[MatrNr : integer, VorlNr : integer]}
voraussetzen : {[Vorgänger : integer, Nachfolger : integer]}
prüfen : {[MatrNr : integer, VorlNr : integer, PersNr : integer, Note : integer]}

1. Finden Sie alle Studenten (**nur** Name ausgeben), die in keinem Fach geprüft wurden. Formulieren Sie die Anfrage in **relationaler Algebra**.
2. Finden Sie alle Studenten (**nur** Name zurückgeben), die in mindestens zwei verschiedenen Fächern bestanden (Note $\leq$ 4.0) haben! Formulieren Sie die Anfrage in **Tupelkalkül**.

Aufgabe 4: SQL

Gegeben sei das folgende Universitätsschema (eine beispielhafte Ausprägung hängt der Klausur an):

Studenten : {[MatrNr : integer, Name : string, Semester : integer]}
Vorlesungen : {[VorlNr : integer, Titel : string, SWS : integer, gelesenVon : integer]}
Professoren : {[PersNr : integer, Name : string, Rang : string, Raum : integer]}
Assistenten : {[PersNr : integer, Name : string, Fachgebiet : string, Boss : integer]}
hören : {[MatrNr : integer, VorlNr : integer]}
voraussetzen : {[Vorgänger : integer, Nachfolger : integer]}
prüfen : {[MatrNr : integer, VorlNr : integer, PersNr : integer, Note : integer]}

Geben Sie jeweils ein *SQL*-92 Statement an, welches die gestellte Frage beantwortet.

1. Wie viele Vorlesungen hört der Student mit dem Namen Fichte? (Ausgabe der Anfrage soll die Anzahl der Vorlesungen sein.)
2. Welche Studenten außer Fichte haben eine Vorlesung gehört, die auch Fichte gehört hat? Geben Sie die Namen dieser Studenten (und nicht Fichte) duplikatfrei aus. (Im Beispiel sind dies Feuerbach, Theophrastos und Schopenhauer.)
3. Finden Sie alle Studenten **(MatrNr und Name duplikatfrei ausgeben)**, die in jeder Prüfung, die sie geschrieben haben, eine 1 erhalten haben. Geben Sie hierbei nur Studenten aus, die mindestens in einem Fach geprüft wurden.

Fortsetzung nächste Seite!

4. Geben Sie eine absteigend sortierte Liste aus allen Vorlesungen und der Anzahl pro Vorlesung abgehaltener Prüfungen aus, etwa so:

VorlNr	Anzahl
4630	2
5001	1
5041	1
5022	0
5043	0
⋮	⋮

Formulieren Sie Ihre Lösung auf zwei verschiedene Varianten: Einmal mit korrelierter Unterabfrage und einmal ohne.

a) Variante 1: mit korrelierter Unterabfrage

b) Variante 2: ohne korrelierter Unterabfrage

Aufgabe 5: Physische Datenorganisation

1. Wie heißt die gezeigte Datenstruktur?
2. Geben Sie den Wert von k (minimale Belegung von inneren Knoten) für das Beispiel an.
3. Geben Sie den Wert von $k*$ (minimale Belegung von Blättern) für das Beispiel an.
4. Fügen Sie die 5 in die Datenstruktur ein. Zeichnen Sie die vollständige Datenstruktur nach dem Einfügen und allen möglicherweise notwendigen Ausgleichsoperationen.

Fortsetzung nächste Seite!

Beispielausprägung

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2
25403	4630	2137	5

Abbildung 1: Beispielausprägung für eine Universitäts-Datenbank

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1: Projektmanagement

Gegeben ist die folgende **Tabelle** zur Grobplanung eines hypothetischen Softwareprojekts:

Aktivität	**Dauer**	**Einschränkungen**
Anforderungs-analyse	4 Monate	Endet frühestens zwei Monate nach dem Start der Entwurfsphase.
Entwurf	3 Monate	Startet frühestens einen Monat nach dem Start der Anforderungsanalyse.
Implemen-tierung	7 Monate	Endet frühestens drei Monate nach dem Ende der Entwurfsphase. Darf erst starten, nachdem die Anforderungsanalyse abgeschlossen ist.

a) Geben Sie ein **CPM-Netzwerk** an, das die Aktivitäten und Abhängigkeiten des obigen Projektplans beschreibt. Gehen Sie von der Zeiteinheit „Monate" aus.
- Das Projekt hat einen Start- und einen Endknoten.
- Jede Aktivität wird auf einen Start- und einen Endknoten abgebildet.
- Die Dauer der Aktivitäten sowie Abhängigkeiten sollen durch Kanten dargestellt werden.
- Der Start jeder Aktivität hängt vom Projektstart ab, das Projektende hängt vom Ende aller Aktivitäten ab. Modellieren Sie diese Abhängigkeiten durch Pseudoaktivitäten mit Dauer null.

b) Berechnen Sie für jedes Ereignis (d.h. für jeden Knoten) die **früheste Zeit**, die **späteste Zeit** sowie die **Pufferzeit.** Beachten Sie, dass die Berechnungsreihenfolge einer topologischen Sortierung des Netzwerks entsprechen sollte.

c) Geben Sie einen **kritischen Pfad** durch das CPM-Netzwerk an.
Welche **Aktivität** darf sich demnach wie lange verzögern?

Fortsetzung nächste Seite!

Aufgabe 2: Objektorientierte Modellierung mit UML

Gegeben sei das folgende **UML-Klassendiagramm**, welches die statische Struktur von einfachen **Zustandsautomaten** beschreibt:

a) Geben Sie ein **Glossar** an, welches die im Klassendiagramm definierten Konzepte und Beziehungen natürlicher Sprache beschreibt, ohne auf die grafische Notation einzugehen. Ergänzen Sie die Definitionen gegebenenfalls durch Angaben zur Semantik von Zustandsautomaten. Verwenden Sie ein bis zwei Sätze pro Klasse.

b) Nachfolgend ist ein Beispiel eines **Zustandsautomaten** in der gängigen grafischen Notation abgebildet. Stellen Sie den Zustandsautomaten als **UML-Objektdiagramm** *konform zum obigen UML-Klassendiagramm* dar. Referenzieren Sie die dort definierten Klassen und Assoziationen; auf Objektbezeichner und Rollennamen dürfen Sie verzichten.

Fortsetzung nächste Seite!

Aufgabe 3: Entwurfsmuster

Verwenden Sie geeignete **Entwurfsmuster**, um die folgenden Sachverhalte mit Hilfe von **UML-Klassendiagrammen** zu beschreiben. Nennen Sie das zu verwendende Entwurfsmuster namentlich, wenden Sie es zur Lösung der jeweiligen *Fragestellung* an und erstellen Sie damit das problemspezifische UML-Klassendiagramm. Beschränken Sie sich dabei auf die statische Sicht, d.h. definieren Sie keinerlei Verhalten mit Ausnahme der Definition geeigneter Operationen.

a) Es gibt unterschiedliche Arten von Bankkonten: Girokonto, Bausparkonto und Kreditkarte. Bei allen Konten ist der Name des Inhabers hinterlegt. Girokonten haben eine IBAN. Kreditkarten sind immer mit einem Girokonto verknüpft. Bei Bausparkonten werden ein Sparzins sowie ein Darlehenszins festgelegt. Es gibt eine *zentrale* Klasse, die die *Erzeugung* unterschiedlicher Typen von Bankkonten steuert.

b) Beim Ticker für ein Hockeyspiel können sich verschiedene Geräte registrieren und wieder abmelden, um auf *Veränderungen* des Spielstands *zu reagieren*. Hierzu werden im Ticker die Tore der Heim- und Gastmannschaft sowie die aktuelle Spielminute vermerkt.
Als konkrete Geräte sind eine Smartphone-App sowie eine Stadionuhr bekannt.

c) Dateisysteme sind *baumartig* strukturiert. Verzeichnisse können wiederum selbst Verzeichnisse und/oder Dateien beinhalten. Sowohl Dateien als auch Verzeichnisse haben einen Namen. Das jeweilige Elternverzeichnis ist eindeutig. Bei Dateien wird die Art (Binär, Text oder andere) sowie die Größe in Byte, bei Verzeichnissen die Anzahl enthaltener Dateien hinterlegt.

<u>Teilaufgabe II: Datenbanksysteme</u>

Aufgabe 1: ER-Modellierung

Für nachfolgend gegebene Miniwelt soll ein ER-Modell erstellt werden. Geben Sie Kardinalitäten in Chen-Notation an. Vergessen Sie nicht, dass zur Chen-Notation nicht nur Funktionalitäten, sondern auch Partizipitäten, also die Angabe von Existenzabhängigkeit bzw. totaler Teilnahme, zählen.

Der örtliche Blasmusikverein möchte seine Verwaltung mithilfe einer Datenbank regeln. Zum einen möchte er seine Mitglieder mit Namen, Adresse - bestehend aus Straße, Hausnummer, PLZ und Ort, und Geburtsdatum festhalten. Jedes Mitglied bekommt bei Vereinseintritt eine laufende Nummer zugewiesen. Mitglieder teilen sich komplett auf in aktive Musiker und passive Vereinsmitglieder. Die aktiven Musiker spielen alle mindestens eine Art von Instrument. Ein Instrument hat einen eindeutigen Namen (z. B. Trompete oder Flöte) und einen Typ.
Ein paar der Mitglieder sind außerdem im Vorstand des Vereins. Im Vorstand können sowohl aktive Musiker als auch passive Mitglieder sein. Für die Vorstandsmitglieder wird der jeweilige Posten notiert.

Fortsetzung nächste Seite!

Um die Aktivität der Musiker zu kontrollieren, wird festgehalten, welcher Musiker bei welchem Auftritt anwesend war. Ein Auftritt wird identifiziert über Datum und Uhrzeit, außerdem hat er einen Namen und einen Ort. Während eines Auftritts werden verschiedene Stücke gespielt. Um Wiederholungen in zukünftigen Auftritten am selben Ort zu vermeiden, werden für jeden Auftritt die Stücke, die eine eindeutige Nummer im Musikverein haben, notiert. Musikstücke haben außerdem einen Titel sowie einen oder mehrere Komponisten. Jedes Stück ist für gewisse Instrumente geschrieben. Da sich die Anzahl der Stimmen für Stück und Instrument unterscheiden kann, wird für jede Zuordnung von Musikstück zu Instrument die Anzahl der Stimmen festgehalten.

Aufgabe 2: Relationenmodell

Überführen Sie das folgende ER-Modell in ein Relationenschema, das die Bedingungen der dritten Normalform erfüllt. Geben Sie bei jedem Fremdschlüssel an, worauf er verweist. Überlegen Sie sich bei der Transformation von Generalisierung bzw. Spezialisierung, welche der zur Verfügung stehenden Möglichkeiten die beste ist. Beantworten Sie anschließend noch kurz die Frage, warum Sie sich für die von Ihnen gewählte Methode entschieden haben.

Fortsetzung nächste Seite!

Aufgabe 3: Normalisierung

Gegeben ist die Relation
`Reparatur(KID, Kundenname, Adresse, Geburtsdatum, Autokennzeichen, Modell, Baujahr, MitarbeiterID, Reparaturdatum, Reparaturbeschreibung, Reparaturkosten).`
Sie beschreibt den Sachverhalt, dass Autos, die einem bestimmten Kunden gehören, von einem Mitarbeiter zu einem bestimmten Zeitpunkt repariert werden. Zum Kunden wird eine eindeutige ID (KID), ein Name, eine Adresse und sein Geburtsdatum gespeichert. Das Auto wird über sein Kennzeichen identifiziert und kann eindeutig einem Kunden zugeordnet werden. Zusätzlich wird das Modell und das Baujahr aufgenommen. Für die Identifikation eines Reparaturvorgangs muss das Autokennzeichen und das Reparaturdatum bekannt sein. Dazu können dann die Beschreibung, die Kosten und der Mitarbeiter, der die Reparatur durchgeführt hat, bestimmt werden.

3.1 Funktionale Abhängigkeiten

Bestimmen Sie für die obige Beschreibung die funktionalen Abhängigkeiten. Verwenden Sie hierfür nur die Informationen aus der Beschreibung, wobei Sie Zusammenhänge, die sich aus Transitivität ergeben, nicht angeben müssen.

3.2 Schlüsselkandidaten

Bestimmen Sie die Schlüsselkandidaten für die gegebene Relation mit Hilfe der funktionalen Abhängigkeiten aus Aufgabe 3.1. Begründen Sie Ihre Ergebnisse stichhaltig und nennen Sie die verwendeten Methoden.

3.3 Normalformen

Welche Normalform wird von der gegebenen Relation maximal erfüllt? Begründen Sie Ihre Entscheidung in kurzen Sätzen.

Aufgabe 4: SQL

Für die bayerische Meisterschaft im Turmspringen ist folgendes Datenbankschema angelegt:

Springer (<u>Startnummer</u>, Nachname, Vorname, Geburtsdatum, Körpergröße)

Sprung (<u>SID</u>, Beschreibung, Schwierigkeit)

springt (<u>SID, Startnummer, Durchgang</u>)

FK (SID) referenziert Sprung (SID)
FK (Startnummer) referenziert Springer (Startnummer)

Fortsetzung nächste Seite!

Das Attribut Schwierigkeit kann die Werte 1 bis 10 annehmen, das Attribut Durchgang ist positiv und ganzzahlig. Die Körpergröße der Springer ist in Zentimeter angegeben.

4.1 Welche Springer sind größer als 1,80 m? Schreiben Sie eine *SQL*-Anweisung, welche in der Ausgabe mit dem größten Springer beginnt.

4.2 Welche Springer haben im ersten Durchgang einen Sprung mit einer Schwierigkeit von unter 6 gezeigt? Schreiben Sie eine *SQL*-Anweisung, welche Startnummer und Nachname dieser Springer ausgibt.

4.3 Formulieren Sie in Umgangssprache, aber trotzdem möglichst präzise, wonach mit folgender Abfrage gesucht wird:

```
SELECT springt.Startnummer, s.Nachname, s.Vorname, MAX(springt.Durchgang)
FROM springt, Springer s
WHERE springt.Startnummer=s.Startnummer
GROUP BY springt.Startnummer, s.Nachname, s.Vorname;
```

4.4 Gesucht ist die „durchschnittliche Körpergröße" all der Springer, die vor dem 01.01.2000 geboren wurden. Formulieren Sie eine *SQL*-Anweisung, welche die entsprechenden Springer ausgibt, wobei die Spalte mit der durchschnittlichen Körpergröße genau diesen Namen „durchschnittliche Körpergröße" haben soll.

Aufgabe 5: Grundlagen

5.1 Durch Anwendung der Drei-Schichten-Architektur nach ANSI/SPARC erreicht man Datenunabhängigkeit. Erklären Sie diesen Begriff und die zwei Ausprägungen in der Drei-Schichten-Architektur nach ANSI/SPARC.

5.2 Erläutern Sie die ACID-Eigenschaft einer Transaktion. Nennen Sie hierfür, welche Eigenschaften mit dem Begriff „ACID" zusammengefasst sind und erklären Sie diese Eigenschaften jeweils in ein bis zwei Sätzen!

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2018

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	**Frühjahr 2018**	**46116**
Arbeitsplatz-Nr.: _____		

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: **14**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

Aufgabe 1 (Anwendungsfälle und Sequenzdiagramme)

Wir betrachten ein Netzwerk von Geldautomaten (engl. ATM = Automated Teller Machine) und Banken. Alle Geldautomaten und Banken sind mit einer Zentrale verbunden, über die die Kommunikation zwischen ATMs und Banken geregelt wird. Abb. 1 zeigt die Struktur des Netzwerks als Objektdiagramm für den Fall von drei ATMs und zwei Banken.

Abbildung 1: Netzwerk von Geldautomaten und Banken mit Zentrale

Wir betrachten den Anwendungsfall "Geld abheben am ATM". Für diesen Anwendungsfall gibt es einen Hauptablauf, der wie unten angegeben textuell beschrieben ist. Die Substantive Kreditkarte und Karte werden hier und im Folgenden synonym verwendet.

Hauptablauf:

1. Das ATM fordert den Kunden auf, eine Kreditkarte einzugeben.
2. Der Kunde gibt seine Kreditkarte ein.
3. Das ATM liest die Kreditkarte und erfragt daraufhin die Geheimzahl.
4. Der Kunde gibt die Geheimzahl ein.
5. Das ATM überprüft die Geheimzahl und lässt dann die Karte bei der Zentrale überprüfen.
6. Die Zentrale überprüft die BIC (Bank Identifier Code) und lässt dann die Karte von der Bank überprüfen.
7. Die Bank überprüft, ob die Karte gesperrt ist und benachrichtigt dann die Zentrale, dass alles in Ordnung ist.
8. Daraufhin gibt auch die Zentrale dem ATM ihr Okay.
9. Das ATM fragt den Kunden nach dem abzuhebenden Betrag.

Fortsetzung nächste Seite!

10. Der Kunde gibt den gewünschten Betrag ein.
11. Das ATM überprüft, ob der Betrag nicht höher als 1000 Euro ist und fordert dann die Zentrale auf, die Abhebung zu veranlassen.
12. Die Zentrale leitet die Anforderung an die Bank weiter, die daraufhin das Konto auffordert, den Betrag abzuheben.
13. Das Konto überprüft den Abhebungsbetrag (der nicht den Kreditrahmen des Kunden überschreiten darf) und führt dann die Abhebung durch.
14. Das Konto benachrichtigt die Bank, dass die Abhebung erfolgreich war.
15. Das Bank gibt diese Nachricht an die Zentrale weiter, die dem ATM die erfolgreiche Abhebung mitteilt.
16. Das ATM gibt den gewünschten Betrag in Bargeld aus.
17. Der Kunde entnimmt das Geld, woraufhin das ATM die Karte ausgibt.
18. Der Kunde entnimmt die Karte.

a) Geben Sie ein Sequenzdiagramm an, das die im obigen Hauptablauf beschriebenen Interaktionen zwischen dem Kunden (als Aktor), dem ATM, der Zentrale, der Bank und dem Konto zeigt. Für die Kreditkarte braucht kein Objekt im System eingeführt zu werden. Beachten Sie, dass bestimmte Aktionen auch nur ein Objekt betreffen können und dann als Selbstaufrufe darzustellen sind, wie z.B. die Aktion `Karte lesen` in Abb. 2.

b) Das Sequenzdiagramm in Abb. 2 spezifiziert einen Ausnahmeablauf, der durchgeführt wird, wenn die Kreditkarte nicht lesbar ist.

Abbildung 2: Ausnahmeablauf, falls Karte nicht lesbar

Fortsetzung nächste Seite!

Zeichnen Sie ein Sequenzdiagramm mit einer Alternative, so dass sowohl der Hauptablauf als auch der oben spezifizierte Ausnahmeablauf mögliche Abläufe des Sequenzdiagramms sind. Sie müssen dazu nicht alle Interaktionen des Hauptablaufs nochmal einzeichnen, sondern es genügt, wenn deutlich erkennbar ist, wo welche Interaktionen des Hauptablaufs vorkommen sollen.

c) Schildern Sie kurz in Worten vier weitere Situationen, in denen es **nicht** zu einem erfolgreichen Abschluss der Abhebung kommt. Geben Sie an, in welchem Schritt des Hauptablaufs die Ausnahmesituation auftreten kann. (Alternativabläufe brauchen hierzu nicht angegeben zu werden.)

Aufgabe 2 (Objektdiagramme, Klassendiagramme, Vererbung)

Gegeben sei das Objektdiagramm in Abb. 3.

Abbildung 3: Objektdiagramm

Konstruieren Sie ein Klassendiagramm, das genau fünf Klassen `A`, `B`, `C`, `D` und `E` enthält und folgende Bedingungen erfüllt:

1. Das Objektdiagramm in Abb. 3 ist eine korrekte Instanziierung des Klassendiagramms. Insbesondere sollen Links im Objektdiagramm Instanzen von gerichteten Assoziationen im Klassendiagramm sein. An den Enden der gerichteten Assoziationen sind die aus dem Objektdiagramm ersichtlichen Rollennamen anzugeben. Multiplizitäten können weggelassen werden. Die Attribute sind mit zu den Werten im Objektdiagramm passenden Datentypen zu versehen.

2. Es gibt keine zwei Klassen, die ein Attribut mit demselben Namen enthalten und es gibt auch keine zwei Klassen, die eine ausgehende gerichtete Assoziation mit demselben Rollennamen am gerichteten Assoziationsende haben.

Hinweis: Um Bedingung 2 zu erfüllen, sind geeignete Vererbungsbeziehungen im Klassendiagramm zu verwenden.

Fortsetzung nächste Seite!

Aufgabe 3 (Objektorientierte Implementierung)

Für die nächste Fußballweltmeisterschaft möchte ein Wettbüro ein Programm zur Verwaltung von Spielern, Vereinen und (National-)Mannschaften entwickeln. Dazu wurde bereits das folgende UML-Klassendiagramm entworfen.

Abbildung 4: UML-Klassendiagramm

Es kann angenommen werden, dass die Klasse `Verein` bereits implementiert ist. In den folgenden Implementierungsaufgaben können Sie eine objektorientierte Programmiersprache Ihrer Wahl verwenden. Die verwendete Sprache ist anzugeben. Zu beachten sind jeweils die im Klassendiagramm angegebenen Sichtbarkeiten von Attributen, Rollennamen, Konstruktoren und Operationen.

a) Es ist eine Implementierung der Klasse `Spieler` anzugeben. Der Konstruktor soll die Instanzvariablen mit den gegebenen Parametern initialisieren, wobei die Anzahl der Tore nach der Objekterzeugung gleich 0 sein soll. Ansonsten kann die Funktionalität der einzelnen Operationen aus deren Namen geschlossen werden.

b) Es ist eine Implementierung der Klasse `Mannschaft` anzugeben. Der Konstruktor soll das Land initialisieren, die Anzahl der Spieler auf 0 setzen und das Team mit einem noch "leeren" Array der Länge 20 initialisieren. Das Team soll mit der Methode `hinzufuegen` um einen Spieler erweitert werden. Die Methode `torschuetzenkoenig` soll den Namen eines Spielers aus dem Team zurückgeben, der die meisten Tore für die Mannschaft geschossen hat. Ist

Fortsetzung nächste Seite!

das für mehrere Spieler der Fall, dann kann der Name eines beliebigen solchen Spielers zurückgegeben werden. Ist noch kein Spieler im Team, dann soll der String "`Kein Spieler vorhanden`" zurückgegeben werden.

c) Schreiben Sie den Rumpf einer `main`-Methode, so dass nach Ausführung der Methode eine deutsche Mannschaft existiert mit zwei Spielern Namens "Hugo Meier" und "Frank Huber". Beide Spieler sollen zum selben Verein "FC Staatsexamen" gehören. "Hugo Meier" soll nach Aufnahme in die deutsche Mannschaft genau ein Tor geschossen haben, während "Frank Huber" noch kein Tor erzielt hat. (Wir abstrahieren hier von der Realität, in der ein 2-er Team noch gar nicht spielbereit ist.)

Teilaufgabe 2

1. Grundlagen

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort in jeweils ein bis zwei Sätzen.

a) Sind die Tupel in einer Relation in Einfügereihenfolge abgelegt?

b) Muss eine Transaktion die Datenbank in einem konsistenten Zustand hinterlassen?

c) Was versteht man unter Persistenz?

d) Aus wie vielen Attributen kann der Primärschlüssel einer Relation minimal und aus wie vielen maximal bestehen?

e) Was versteht man unter referentieller Integrität?

2. ER-Modellierung

Im Folgenden finden Sie die Beschreibung einer Terminverwaltungssoftware. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme (= Existenzabhängigkeit, Partizipität) von Entitytypen.

Ein Termin hat einen Startzeitpunkt und einen Endzeitpunkt, sowie eine Beschreibung. Eindeutig gekennzeichnet ist ein Termin durch seine ID. Ein Termin kann ein Serientermin sein. Serientermine haben zusätzlich zu den Attributen anderer Temine eine Wiederholungsperiode und eine Wiederholungsanzahl.

In der Software werden auch Personen verwaltet. Eine Person ist eindeutig durch ihre E-Mail-Adresse gekennzeichnet und hat einen Vornamen, einen Nachnamen sowie ein Geburtsdatum. Zu jeder Person ist auch noch das Alter abrufbar, das sich aus dem Geburtsdatum ergibt. Personen können an Terminen teilnehmen.

Weiterhin werden Räume verwaltet. Räume befinden sich in Gebäuden und haben innerhalb ihres Gebäudes eine eindeutige Raumnummer. Weiterhin wird zu Räumen deren Kapazität abgelegt. Gebäude haben eine eindeutige Adresse, die sich aus Postleitzahl, Straße und Hausnummer zusammensetzt. Personen können für Termine Räume buchen. Ein bestimmter Raum wird für einen bestimmten Termin immer von höchstens einer Person gebucht. Zu jeder Raumbuchung wird das Datum der Buchung gespeichert.

Fortsetzung nächste Seite!

3. Relationaler Entwurf

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema in dritter Normalform (3 NF) mit möglichst wenig 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,
Fremdschlüsselattribut1[Relation1],
(Fremdschlüssel2_Attribut1, Fremdschlüssel2_Attribut2)[Relation2])
```

Fortsetzung nächste Seite!

4. SQL

Gegeben sind folgende Relationen aus der Schulaufgaben-Planungssoftware einer Schule:

```
Schueler (ID, Nachname, Vorname)
Kurs(Bezeichnung, Jahrgang, Lehrername)
Mitglied(ID[Schueler], (Bezeichnung, Jahrgang)[Kurs])
Schulaufgabe((Bezeichnung, Jahrgang)[Kurs], Nummer, Datum)
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz nicht mehrfach aus.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle Schulaufgabe (inklusive Schlüsseln) anlegt. Verwenden Sie sinnvolle Datentypen.

b) Schreiben Sie eine SQL-Anweisung, die Datum, Bezeichnung, Jahrgang und Nummer aller Schulaufgaben ausgibt, an denen Schüler mit dem Nachnamen „Mustermann" teilnehmen. Die Ausgabe soll nach Datum absteigend sortiert sein.

c) Schreiben Sie eine SQL-Anweisung, die Nachname und ID aller Schüler ausgibt, die in mindestens einem Kurs Mitglied sind, in dem noch ein oder mehrere andere Schüler mit gleichem Vornamen und Nachnamen Mitglied sind.

d) Schreiben Sie eine SQL-Anweisung, die ID, Nachname und Vorname aller Schüler ausgibt, die in keinem Kurs Mitglied sind.

e) Schreiben Sie eine SQL-Anweisung, die alle Schulaufgaben ermittelt, die an einem Datum liegen, an dem mindestens einer der teilnehmenden Schüler noch mindestens eine weitere Schulaufgabe hat. Ausgegeben werden sollen Bezeichnung und Jahrgang des Kurses, die Nummer der Schulaufgabe, sowie die Anzahl der betroffenen Schüler.

f) Schreiben Sie eine SQL-Anweisung, die alle Kurse löscht, die weder Mitglieder noch Schulaufgaben haben.

5. Normalformen

Gegeben ist die Relation $R(a_1, a_2, a_3, a_4, a_5)$. Sie besitze den Schlüsselkandidaten a_2a_4 und die funktionalen Abhängigkeiten $a_4 \rightarrow a_5$ und $a_3 \rightarrow a_1$.

a) Begründen Sie in ein bis zwei Sätzen, warum R sich nicht in zweiter Normalform befindet. Zerlegen Sie R so, dass die Ergebnisrelationen die zweite Normalform erfüllen. Erzeugen Sie dabei so wenig Relationen wie möglich.

b) Begründen Sie in ein bis zwei Sätzen, warum sich die Ergebnisrelationen der vorherigen Teilaufgabe nicht alle in der dritten Normalform befinden. Zerlegen Sie sie so weiter, dass alle Relationen die dritte Normalform erfüllen. Erzeugen Sie dabei so wenig Relationen wie möglich.

Thema Nr. 2

Teilaufgabe 1

Aufgabe 1: „Entwurfsmuster"

Das sogenannte **Vermittler**-Muster (*Mediator Pattern*) ist eines der ursprünglichen GoF-Verhaltensmuster.

a) Stellen Sie das allgemeine **Vermittler**-Muster als UML-Klassendiagramm dar und beschreiben Sie kurz die einzelnen Akteure des Musters zusammen mit ihrer Rolle in Ihrem Diagramm.

b) Nennen und begründen Sie kurz je einen Vorteil und einen Nachteil bei der Anwendung des **Vermittler**-Musters.

c) Beschreiben Sie kurz zwei Anwendungsbeispiele für das **Vermittler**-Muster. Welche Varianz oder nachträgliche Änderung an der Anwendungslogik könnte auftreten und wie bzw. wo würde man diese konkret implementieren?

d) Welches andere GoF-Entwurfsmuster ist dem **Vermittler**-Muster am ähnlichsten? Was sind die entscheidenden Unterschiede zwischen den beiden? Skizzieren und begründen Sie jeweils kurz, worin die Ähnlichkeiten bzw. Unterschiede bestehen.

Aufgabe 2: „Kopplung, Kohäsion, Komplexität"

a) Beschreiben Sie kurz jede der folgenden sechs Kopplungsarten nach IEEE 610 (Standard Glossary of Software Engineering Terminology):

   i. Pathologische Kopplung (pathological coupling)

   ii. Inhaltskopplung (content coupling)

   iii. Bereichskopplung (common-environment coupling)

   iv. Hybridkopplung (hybrid coupling)

   v. Kontrollkopplung (control coupling)

   vi. Datenkopplung (data coupling)

b) Beschreiben Sie kurz jede der folgenden Kohäsionsarten:

   i. Zeitliche Kohäsion

   ii. Prozedurale Kohäsion

   iii. Kommunikative Kohäsion

   iv. Sequentielle Kohäsion

Fortsetzung nächste Seite!

c) Die sogenannte zyklomatische Komplexität M nach McCabe stellt die Anzahl der linear unabhängigen Pfade auf dem Kontrollflussgraphen eines Moduls dar.

i. Geben Sie eine allgemeine Rechenvorschrift an, um M zu berechnen. Erklären Sie ggf. kurz die beteiligten Größen.

ii. Beschreiben Sie kurz den Zusammenhang zwischen M und der strukturellen Überdeckung (white-box coverage).

iii. Geben Sie die zyklomatische Komplexität der folgenden Methode ggT an:

```
public int ggT(int a, int b) {
   int result = 1;
   for (int i = 1; i <= Math.min(a,b); i++) {
      if ((a%i==0)&(b%i==0)) {
         result = i;
      }
   }
   return result;
}
```

Aufgabe 3: „Testen"

Gegeben sei folgende Methode und ihr datenflussannotierter Kontrollflussgraph:

```
public int ggT(int a, int b) {
   int result = 1;
   for (int i = 1; i <= Math.min(a,b); i++) {
      if ((a%i==0)&(b%i==0)) {
         result = i;
      }
   }
   return result;
}
```

Fortsetzung nächste Seite!

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, $C_{\infty,2}$) genügen würden.

b) Welche der vorangehend ermittelten Pfade sind mittels Testfälle tatsächlich überdeckbar („feasible")? Falls der Pfad ausführbar ist, geben Sie den Testfall (also die Werte der Parameter **a** und **b**) an – andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

   i. Falls Ihr vorangehend genannter Pfad für die C_1-Überdeckung nicht ausführbar ist, geben Sie nun einen anderen Pfad samt zugehörigem Testfall an, so dass Sie C_1-Überdeckung erreichen.

   ii. Geben Sie mindestens einen Pfad im Kontrollflussgraphen an, der *nicht* durch einen Testfall überdeckt werden kann („infeasible"). Begründen Sie kurz Ihre Antwort.

c) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen All-Uses-Überdeckung genügen würden. Nennen Sie bei jedem Pfad, welche def/use-Paare als Tripel (Variable, Knoten mit def, Knoten/Kante mit c/p-use!) Sie damit „überdecken".

   Zur Vereinfachung ist ein Pfad zusammen mit einem davon überdeckten Tripel vorgegeben – ergänzen Sie den Rest der Zusammenstellung:

   Pfad 1: ***Start-n1-n2-End***

   - *Variable* ***a****, def in* ***Start****, p-use in* ***(n2-End)***

   -

Teilaufgabe 2

Aufgabe 1: Allgemeine Fragen

a) Was versteht man unter der Anomalie Lost Update, die im Mehrbenutzerbetrieb auftreten kann? Geben Sie ein Beispiel (frei formuliert) an, in dem ein Lost Update auftritt.

b) Gegeben sei das Relationenschema $R = (\underline{A}, B, C, D)$. Über die Relation sei bekannt, dass A ein Schlüssel ist, und es keine weiteren Schlüsselkandidaten gibt. Über weitere funktionale Abhängigkeiten sei nichts bekannt, es kann aber weitere funktionale Abhängigkeiten geben. Nehmen Sie an, dass R die erste Normalform erfüllt.

   i. Welche Normalformen erfüllt die Relation R noch? Begründen Sie Ihre Aussage.

   ii. Welche Normalformen erfüllt die Relation R möglicherweise nicht? Begründen Sie Ihre Aussage.

Fortsetzung nächste Seite!

c) Gegeben seien die Relationen $R = (A,B,C)$ und $S = (C,D,E)$. Relation R enthalte 50 Tupel und Relation S enthalte 10 Tupel. Gegeben seien außerdem folgende Anfragen:

A_1:

```
SELECT *
FROM R, S;
```

A_2:

```
SELECT *
FROM R NATURAL JOIN S;
```

- Wieviele Ergebnistupel liefert die Anfrage A_1?
- Wieviele Attribute enthält die Ergebnisrelation von Anfrage A_1?
- Wieviele Ergebnistupel liefert die Anfrage A_2 **mindestens**?
- Wieviele Ergebnistupel liefert die Anfrage A_2 **höchstens?**
- Wieviele Attribute enthält die Ergebnisrelation von Anfrage A_2?

d) Was versteht man unter dem "Cursor-Konzept" und weshalb wird es eingesetzt?

Aufgabe 2: Tupel- und Bereichskalkül

Gegeben sei das folgende Datenbankschema, alle Attribute seien vom Datentyp String:

Linie(<u>LNr</u>, LName, LZielort)
Zug(<u>ZNr</u>, <u>LNr</u>, Abfahrt)
Fahrgast(<u>FGNr</u>, FGName, FGAlter)
Fahrt(<u>ZNr</u>, <u>FGNr</u>, Start)

Geben Sie für die folgende verbal formulierte Anfrage jeweils einen aquivalenten Ausdruck **a) im Tupelkalkül und b) im Bereichskalkül** an.

Gesucht sind Nummer, Name, und Alter aller Fahrgäste, <u>die niemals einen Zug nach Bielefeld genommen</u> haben.

Tupelkalül:

Bereichskalkül:

Fortsetzung nächste Seite!

Aufgabe 3: SQL-Anfragen und ER-Diagramm

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Schule entworfen wurde. Die Primärschlüssel-Attribute sind jeweils unterstrichen.
Die Relation *Lehrer* enthält allgemeine Daten zu den Lehrern, nämlich sein 1. und 2. Unterrichtsfach, sowie den Spitznamen, den er von den Schülern bekommt und das Kürzel, mit dem er unterschreibt. Lehrer unterrichten Klassen in jeweils maximal einem bestimmten Fach.

Eine Klasse wird eindeutig bestimmt durch die Jahrgangsstufe sowie einen Buchstaben und hat ein ihr zugewiesenes Klassenzimmer.

Jeder Schüler ist in genau einer Klasse, hat ein Lieblingsfach und eine Durchschnittsnote und ist eindeutig bestimmt durch seinen Namen.

Die Durchschnittsnote, die Dauer und das Datum einer Schulaufgabe hängen davon ab, welcher Lehrer sie in welcher Klasse stellt und die wievielte Schulaufgabe es ist. Für jede Schulaufgabe wird eine Anwesenheitsliste geführt, auf der steht, welche Schüler daran teilgenommen haben, in die auch alle Noten eingetragen werden.

Lehrer (<u>LName</u>, Fach1, Fach2, Spitzname, Kuerzel)
Schueler (<u>SName</u>, Durchschnittsnote, Lieblingsfach, Jhgstufe, Buchstabe)
Unterrichtet (<u>LName, Jhgstufe, Buchstabe,</u> Fach)
Klasse (<u>Jhgstufe, Buchstabe,</u> Klassenzimmer)
Schulaufgabe (<u>LName, Jhgstufe, Buchstabe, Nummer,</u> Durchschnitt, Dauer, Datum)
NimmtTeil (<u>SName, LName, Jhgstufe, Buchstabe, Nummer,</u> Note)

a) Erstellen Sie ein vollständiges Entity-Relationship-Diagramm!

Formulieren Sie die folgenden Anfragen in SQL.

b) Geben Sie die Anweisung in **SQL-DDL** an, die notwendig ist, um die Relation **NimmtTeil** zu erzeugen. Achten Sie dabei auf Fremdschlüsselbeziehungen. Es sollen nur die Noten 1, 2, 3, 4, 5 und 6 erlaubt sein.

c) Geben Sie den Namen des Schülers mit der besten Durchschnittsnote an.

d) Geben Sie an, wie viele Schüler Herr Meier insgesamt unterrichtet.

e) Geben Sie für alle Schüler der 8b den Namen und den jeweiligen Durchschnitt aller Noten von Schulaufgaben an, die seit dem 15.12.2017 geschrieben wurden.

Fortsetzung nächste Seite!

f) Gesucht ist eine Liste von Schulaufgaben, die Frau Sommer gestellt hat, die jeweils die Nummer und die vollständige Bezeichnung der Klasse, in der sie gehalten wurde, enthält, sowie die Durchschnittsnote der Schulaufgabe selbst und die Gesamtdurchschnittsnote der ganzen Klasse.

Aufgabe 4: Normalisierung

Gegeben sei das Relationenschema $R(A,B,C,D,E,F)$, sowie die Menge der zugehörigen funktionalen Abhängigkeiten F:

- $C \rightarrow B$
- $B \rightarrow A$
- $CE \rightarrow D$
- $E \rightarrow F$
- $CE \rightarrow F$
- $C \rightarrow A$

a) Bestimmen Sie den Schlüsselkandidaten der Relation R und begründen Sie, warum es keine weiteren Schlüsselkandidaten gibt.

b) Überführen Sie das Relationenschema R mit Hilfe des Synthesealgorithmus in die dritte Normalform. Führen Sie hierfür jeden der vier Schritte durch und kennzeichnen Sie Stellen, bei denen nichts zu tun ist.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Herbst 2018

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2018**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: 19

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 (Entwurf von Klassendiagrammen) (19 PUNKTE)

Jeder Kaufvertrag für eine Immobilie wird von einem Notar besiegelt und zwischen einem Verkäufer und einem Käufer abgeschlossen, die beide juristische Personen mit Namen und Anschrift sind. Eine juristische Person ist entweder eine Privatperson oder ein Unternehmen mit Gewerbenummer. Immobilien sind entweder Eigentumswohnungen oder Häuser. Jede Immobilie hat eine Adresse. Eigentumswohnungen haben eine Wohnungsnummer, während Häuser eine Flurnummer haben. Notare haben einen Namen und einen Amtssitz. In jedem Kaufvertrag wird der Verkaufspreis und die Urkundennummer des Vertrags festgehalten.

Um ein System zur Verwaltung von Immobilienkäufen zu entwickeln, soll ein Datenmodell in Form eines UML-Klassendiagramms entworfen werden. Konstruieren Sie ein Klassendiagramm für den oben beschriebenen Sachverhalt mit geeigneten (ggf. abstrakten) Klassen, Vererbungsbeziehungen und Assoziationen. Assoziationen sollen geeignet benannt werden und an allen Assoziationsenden sollen passende Multiplizitäten angegeben werden.

Fortsetzung nächste Seite!

Aufgabe 2 (Design-Pattern und objektorientierte Implementierung) (5+20+5=30 PUNKTE)

Das obige Klassendiagramm modelliert Grafikobjekte im zweidimensionalen Raum. Ein Grafikobjekt ist ein Punkt, ein Kreis oder eine Gruppe von Grafikobjekten. Jedes Grafikobjekt kann entlang eines Vektors, gegeben durch die Parameter *dx* und *dy* der Operation *verschieben*, in der Ebene verschoben werden. Die Lage eines Punktes wird durch seine x- und y-Koordinaten bestimmt, die Lage eines Kreises durch seinen Mittelpunkt und den Radius. Mit der Operation *add* kann ein Grafikobjekt zu einer Gruppe hinzugenommen werden. Eine Gruppe wird verschoben, indem alle ihre Elemente entsprechend verschoben werden. Der Datentyp *Real* steht für reelle Zahlen.

(a) Welches Design-Pattern wurde hier für die Modellierung von Grafikobjekten angewendet? Erläutern Sie Ihre Antwort, indem Sie den Namen des Patterns angeben und erklären, wie die für das Pattern typischen Ideen hier zum Einsatz kommen.

(b) Das obige Klassendiagramm ist in einer objektorientierten Programmiersprache Ihrer Wahl zu implementieren. Die verwendete Sprache ist zu nennen. Der Datentyp *Real* ist durch einen Typ für Gleitkommazahlen zu realisieren. Für die Realisierung von Mengen mit Elementen eines Typs `C` kann ein geeigneter Kollektionstyp `HashSet<C>` vorausgesetzt werden mit einem Konstruktor `HashSet<C>()`, der ein Objekt erzeugt, das eine leere Menge repräsentiert. Wir nehmen an, dass der Typ `HashSet<C>` eine Methode `add` zum Hinzufügen von Elementen zu einer Menge zur Verfügung stellt und dass man über alle Elemente `x` eines `HashSet<C>`-Objekts `hs` iterieren kann mit der Anweisung `for (C x : hs){<Anweisungen>}`. Der Konstruktor für Punkte soll die beiden Koordinaten eines Punkts initialisieren; der Konstruktor für Kreise soll den Radius und den Mittelpunkt initialisieren. Der Konstruktor für Gruppen soll eine Gruppe erzeugen, die (zunächst) kein Grafikobjekt enthält.

(c) Schreiben Sie eine *main*-Methode mit Kopf `public static void main(String[] args)`, so dass am Ende der Methodenausführung folgende drei Objekte existieren: ein Punkt `p` mit Koordinaten `(7.0,-4.0)`, ein Kreis `k` mit Mittelpunkt `p` und Radius `3.14` und eine Gruppe, die `p` und `k` enthält.

Fortsetzung nächste Seite!

Aufgabe 3 (Verhaltens-Modellierung mit Zustandsdiagrammen) (23 PUNKTE)

Eine Digitaluhr kann alternativ entweder die Zeit (Stunden und Minuten) oder das Datum (Tag, Monat und Jahr) anzeigen. Zu Beginn zeigt die Uhr die Zeit an. Sie besitzt drei Druckknöpfe **A**, **B** und **C**. Mit Knopf **A** kann zwischen Zeit- und Datumsanzeige hin und her gewechselt werden.

Wird die Zeit angezeigt, dann kann mit Knopf **B** der Reihe nach erst in einen Stundenmodus, dann in einen Minutenmodus und schließlich zurück zur Zeitanzeige gewechselt werden. Im Stundenmodus blinkt die Stundenanzeige. Mit Drücken des Knopfes **C** können dann die Stunden schrittweise inkrementiert werden. Im Minutenmodus blinkt die Minutenanzeige und es können mit Hilfe des Knopfes **C** die Minuten schrittweise inkrementiert werden.

Die Datumsfunktionen sind analog. Wird das Datum angezeigt, dann kann mit Knopf **B** der Reihe nach in einen Tagesmodus, Monatsmodus, Jahresmodus und schließlich zurück zur Datumsanzeige gewechselt werden. Im Tagesmodus blinkt die Tagesanzeige. Mit Drücken des Knopfes **C** können dann die Tage schrittweise inkrementiert werden. Analog blinken mit Eintritt in den entsprechenden Einstellmodus der Monat oder das Jahr, die dann mit Knopf **C** schrittweise inkrementiert werden können.

Wenn sich die Uhr in einem Einstellmodus befindet, hat das Betätigen des Knopfes **A** keine Wirkung. Ebenso wirkungslos ist Knopf **C**, wenn gerade Zeit oder Datum angezeigt wird.

Beschreiben Sie das Verhalten der Digitaluhr durch ein UML-Zustandsdiagramm. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. Deren Bedeutung soll durch die Verwendung von sprechenden Namen klar sein. Für die Inkrementierung von Stunden, Minuten, Tagen etc. brauchen keine konkreten Berechnungen angegeben werden. Der kontinuierliche Zeitfortschritt des Uhrwerks ist nicht zu modellieren.

Zustände sind, wie in der UML üblich, durch abgerundete Rechtecke darzustellen. Sie können unterteilt werden in eine obere und eine untere Hälfte, wobei der Name des Zustands in den oberen Teil und eine in dem Zustand auszuführende Aktivität in den unteren Teil einzutragen ist.

Fortsetzung nächste Seite!

Aufgabe 4 (Verifikation funktionaler Programme) (18 PUNKTE)

Gegeben sei der folgende polymorphe Datentyp *'a list* zur Darstellung endlicher Listen mit Elementen eines beliebigen Typs *'a*.

datatype *'a list* = *nil* | *append* **of** *'a* * *'a list*

Die Länge einer Liste wird durch die folgende rekursive Funktion *length* definiert.

fun *length*(*nil*) = 0

| *length*(*append*(x, L)) = *length*(L) + 1

Die Konkatenation (d.h. das Zusammenhängen) zweier Listen wird durch folgende rekursive Funktion *conc* definiert.

fun *conc*(*nil*, L) = L

| *conc*(*append*(x, L1), L2) = *append*(x, *conc*(L1, L2))

Es wird behauptet, dass für alle Ausdrücke L1 und L2 des Typs *'a list* folgendes gilt:

$$\mathit{length}(\mathit{conc}(\mathrm{L1}, \mathrm{L2})) = \mathit{length}(\mathrm{L1}) + \mathit{length}(\mathrm{L2})$$

Beweisen Sie diese Behauptung durch strukturelle Induktion gemäß des Aufbaus des Ausdrucks L1. Geben Sie zunächst an, welche Fälle für die Gestalt von L1 unterschieden werden müssen und welcher Fall der Basisfall der Induktion ist. Für den Nicht-Basisfall ist die Induktionsvoraussetzung und die Induktionsbehauptung zu nennen. Es ist genau anzugeben, wo die Induktionsvoraussetzung im Beweis verwendet wird.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1 (Vermischte Fragen) (2+2+2+2+2+2+2+2=16 PUNKTE)

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort in jeweils ein bis zwei Sätzen.

(a) Kann ein Tupel mehrfach in einer Relation enthalten sein?

(b) In welcher Reihenfolge erhält man die Tupel bei einer SQL-Anfrage zurück, wenn man keine `ORDER-BY`-Klausel angibt?

(c) Worin besteht der Unterschied, wenn ein Attribut vom Typ String einen leeren String enthält und wenn es `NULL` enthält?

(d) Dürfen Fremdschlüsselwerte `NULL` sein?

(e) Aus wie vielen SQL-Statements besteht eine Datenbanktransaktion mindestens und aus wie vielen höchstens?

(f) Was ist der Effekt des Aufrufs von abort innerhalb einer Transaktion?

(g) Dürfen während einer Transaktion inkonsistente Zustände auftreten?

(h) Was muss die Recovery-Komponente eines Datenbanksystems, das ACID garantiert, beim Wiederanlauf nach einem Systemabsturz sicherstellen?

Aufgabe 2 (ER-Modellierung) (16 PUNKTE)

Im Folgenden finden Sie die Beschreibung eines Systems zur Verwaltung von Freizeitparks. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme (= Existenzabhängigkeit, Partizipität) von Entitytypen.

Der Freizeitpark ist in mehrere Gebiete eingeteilt. Ein Gebiet hat einen eindeutigen Namen und eine Beschreibung. In jedem Gebiet gibt es eine oder mehrere Attraktionen. Diese verfügen über eine innerhalb ihres Gebiets eindeutige Nummer. Außerdem gibt es zu jeder Attraktion einen Namen, eine Beschreibung und ein oder mehrere Fotos. Der Freizeitpark hat Mitarbeiter. Zu diesen werden jeweils eine eindeutige ID, der Vorname und der Nachname gespeichert. Weiterhin hat jeder Mitarbeiter ein Geburtsdatum, das sich aus Tag, Monat und Jahr zusammensetzt. Die Arbeit im Freizeitpark ist in Schichten organisiert. Eine Schicht kann eindeutig durch das Datum und die Startzeit identifiziert werden. Jede Schicht hat weiterhin eine Dauer. Mitarbeiter können in Schichten an Attraktionen arbeiten. Dabei wird die Aufgabe gespeichert, die der Mitarbeiter übernimmt. Pro Schicht kann der selbe Mitarbeiter nur an maximal einer Attraktion arbeiten.

Fortsetzung nächste Seite!

Aufgabe 3 (Relationaler Entwurf) (15 PUNKTE)

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. Wenn ein Attribut zur korrekten Abbildung des ER-Diagramms als `UNIQUE` oder `NOT NULL` ausgezeichnet werden muss, geben Sie dies an.

Beispiel:

```
Relation1 (Primärschlüssel, Attribut1, Attribut2,
  Fremdschlüsselattribut1[Relation2],
  (Fremdschlüssel2_Attribut1, Fremdschlüssel2_Attribut2)[Relation3]);
Attribut1 UNIQUE
```

Aufgabe 4 (SQL) (5+3+5+6+7+4=30 PUNKTE)

Gegeben sind folgende Relationen aus einem Kundenverwaltungssystem:

```
Kunde(ID, Vorname, Nachname, PLZ)
Produkt(GTIN, Bezeichnung, Bruttopreis, MWStSatz)
Kauf(ID[Kunde], GTIN[Produkt], Datum, Menge)
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz, also eine Entity, nicht mehrfach aus.

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Kauf" anlegt. Gehen Sie davon aus, dass die Tabellen „Kunde" und „Produkt" bereits existieren.

(b) Schreiben Sie eine SQL-Anweisung, die Vorname und Nachname aller Kunden mit der Postleitzahl 20251 ausgibt, absteigend sortiert nach Nachname und bei gleichen Nachnamen, absteigend nach Vorname.

(c) Schreiben Sie eine SQL-Anweisung, die zu jedem Einkauf mit mehr als 10 unterschiedlichen Produkten den Nachnamen des Kunden und den Bruttogesamtpreis des Einkaufs ausgibt. Ein Einkauf ist dabei definiert als Menge aller Produkte, die ein bestimmter Kunde an einem bestimmten Datum kauft.

(d) Schreiben Sie eine SQL-Anweisung, die die GTINs aller Produkte ausgibt, die an mindestens einen in der Datenbank enthaltenen PLZ-Bereich noch nie verkauft worden sind. Als in der Datenbank enthaltener PLZ-Bereich gelten alle in der Tabelle „Kunde" enthaltenen PLZs. Ein Produkt gilt als an einen PLZ-Bereich verkauft, sobald es von mindestens einem Kunden aus diesem PLZ-Bereich gekauft wurde. Produkte, die bisher noch gar nicht verkauft worden sind, müssen nicht berücksichtigt werden.

(e) Schreiben Sie eine SQL-Anweisung, die die Top-Ten der am meisten verkauften Produkte ausgibt. Ausgegeben werden sollen der Rang (1 bis 10) und die Bezeichnung des Produkts. Gehen Sie davon aus, dass es keine zwei Produkte mit gleicher Verkaufszahl gibt und verwenden Sie **keine** produktspezifischen Anweisungen wie beispielsweise `rownum`, `top` oder `limit`.

(f) Schreiben Sie eine SQL-Anweisung, die alle Produkte löscht, die noch nie gekauft wurden.

Fortsetzung nächste Seite!

Aufgabe 5 (Normalformen) (2+2+3+6=13 PUNKTE)

(a) Erklären Sie in ein bis zwei Sätzen, was ein Superschlüssel (Oberschlüssel) ist.

(b) Erklären Sie in ein bis zwei Sätzen, was die funktionale Abhängigkeit $\alpha \rightarrow \beta$ auf einer Relation R für die in ihr enthaltenen Tupel bedeutet.

(c) Erklären Sie in ein bis zwei Sätzen, wann eine Attributmenge α transitiv funktional abhängig von einer Attributmenge β ist.

(d) Gegeben ist folgende vollständige Ausprägung (= Extension) der Relation „Prüfer" mit dem Primärschlüssel „Prüfernummer".

Prüfernummer	Nachname	Vorname	Fakultät	Dekan
9	Konrad	Franz	Medizinische Fakultät	Sonntag
10	Schmidt	Franz	Technische Fakultät	Maier
12	Schmidt	Hans	Technische Fakultät	Maier
16	Schmidt	Leon	Medizinische Fakultät	Sonntag
42	Konrad	Hans	Technische Fakultät	Maier

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

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 (Projektmanagement) (13 PUNKTE)

(a) Nennen Sie jeweils vier technische und menschliche Risiken für Projekte.

(b) Nennen Sie die vier Stadien der Team-Entwicklung in der Reihenfolge, in der sie üblicherweise durchlebt werden.

Aufgabe 2 (Softwareprozesse) (5 PUNKTE)

Übernehmen Sie die untenstehende Grafik auf Ihre Bearbeitung und tragen Sie dort die Phasen des Wasserfallmodells ein.

Aufgabe 3 (Softwareentwurf) (7 PUNKTE)

Erläuterns Sie, welche Probleme bei einem System entstehen können, welches eine starke Kopplung und eine schwache Kohäsion hat.

Definieren Sie Kohäsion und Kopplung dafür kurz!

Fortsetzung nächste Seite!

Aufgabe 4 (Design Patterns) (9 PUNKTE)

Es soll eine Software für einen Eisladen und dessen Produkte erstellt werden:
Der Eisladen bietet drei unterschiedliche Eisarten an: einfaches Eis, Eis mit heißen Früchten und Eis mit Schokoflocken.

Zeichen Sie ein UML-Klassendiagramm, in dem Sie die verschiedenen Eisarten als Klassen repräsentieren und außerdem das Decorator-Pattern passend verwenden.

Hinweise:

- Machen Sie Gebrauch von Vererbung.
- Methoden und Felder/Attribute müssen nicht modelliert werden.

Aufgabe 5 (Implementierung) (26 PUNKTE)

(a) Geben Sie an, welche Ausgaben das folgende Java-Programm produziert:

```
1  public class Computation {
2   int z = 2;
3
4    public void computation(int x, int y) {
5      if (x == y) {
6        System.out.println(z * x);
7        z = z * 3;
8      } else if (x < y) {
9        computation(x, (int) y / 2);
10     }
11     if (x > y) {
12       System.out.println(x * y);
13     }
14   }
15
16   public static void main(String[] args) {
17     Computation comp = new Computation();
18     comp.computation(4, 8);
19     comp.computation(3, 5);
20     comp.computation(2, 2);
21   }
22
23 }
```

(b) Implementieren Sie einen Ringpuffer in Java, der auf einer einfach verketteten Liste basiert und zur Speicherung von ganzen Zahlen verwendet werden kann. Bei der Initialisierung des Ringpuffers soll festgelegt werden können, wie viele Elemente in dem Puffer gespeichert werden können. Soll ein Element in dem Puffer gespeichert werden, obwohl seine maximale Größe erreicht ist, ist das älteste der Elemente zu überschreiben. Generell sollen die folgenden Operationen von der Datenstruktur unterstützt werden:

- add: Fügt eine ganze Zahl in die Datenstruktur ein.
- getOldestElement: Gibt das älteste in der Liste gespeicherte Element zurück.
- numberOfElements: Gibt die Anzahl der Elemente in der Datenstruktur zurück.

Der Ringpuffer soll auf einer Liste basieren, deren Node-Klasse aus dem nachfolgendem Quelltext besteht!

```
1  public class Node {
2
3    int content = 0;
4       Node next = null;
5
6    public Node(int content) {
7      this.content = content;
8  }
9
10 public void setNext(Node newNode) {
11      Node next = this.next;
12      newNode.next = next;
13      this.next = newNode;
14 }
15
16 public int getContent(){
17      return content;
18 }
19
20 public void setContent(int newValue) {
21      this.content = newValue;
22 }
23
24 }
```

Aufgabe 6 (Entwicklung und Wartung) (8 PUNKTE)

Sie haben von einem Kunden eine Fehlermeldung bekommen, dass die von Ihrem Vorgänger entwickelte Mathebibliothek einen Programmabsturz verursacht, wenn ganzzahlige Division auftritt. Sie vermuten den Fehler in der unten aufgeführten Methode. Sie müssen jetzt die Methode debuggen, um den konkreten Fehler zu identifizieren. Beschreiben Sie Ihre Schritte während des Debuggings. Nennen Sie den Fehler und beschreiben Sie passende Eingaben für das Debugging.

```
1 int dividiereGanzzahlig (int dividend, int divisor)   {
2     int result = 0;
3     result = dividend  /  divisor;
4     return result;
5 }
```

Aufgabe 7 (Refactoring und Versionskontrolle) (4 PUNKTE)

Nennen Sie zwei Gründe, warum funktionierender Code refaktorisiert werden sollte.

Aufgabe 8 (Verifikation, Validierung und Testing) (12 PUNKTE)

(a) Gegeben ist folgende Fehlerbeschreibung von der Website eines Pizza-Lieferservices:

> Wir betrachten ein Bestellformular eines Online-Pizzalieferanten. Bei der Eingabe der Adresse sieht der Kunde fehlerhafterweise nicht die Adresse, die er eingibt, sondern eine Reihe des Buchstaben "X", bspw. "XXXXXXX"; das heißt, der Kunde tippt die Adresse blind ein. Nach der Eingabe aller weiterer notwendigen und richtigen Daten kann die Bestellung erfolgreich abgeschlossen werden und die Pizza wird entsprechend der Bestellung geliefert.

Ordnen Sie die gegebene Fehlerbeschreibung den sechs folgenden Fehlerklassen I–VI zu und begründen Sie Ihre Antwort kurz. Es können mehrere Klassen zutreffen.

> (I) Transient (II) Permanent (III) Recoverable (IV) Unrecoverable (V) Non-corrupting (VI) Corrupting

(b) Kann man durch Testen sicherstellen, dass ein Programm korrekt läuft? Begründen Sie Ihre Antwort.

Aufgabe 9 (Softwarequalität) (3 PUNKTE)

Nennen Sie drei Merkmale, an denen die Qualität einer Software festgemacht werden kann.

Aufgabe 10 (Softwaremaße) (3 PUNKTE)

Nennen Sie drei Kategorien von Softwaremaßen.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1 (Allgemeine Fragen) (3+2+2+3+1+2=13 PUNKTE)

Beantworten Sie die folgenden allgemeinen Fragen zu Datenbanksysteme.

(a) Was sind die primären Aufgaben eines Datenbanksystems?

(b) Was ist ein **Datenbankschema**?

(c) Was ist ein **Datenbankzustand**?

(d) Gegeben sei eine beliebige klassische, relationale Datenbank. Erfüllt die Datenbank eine Normalform und wenn ja, welche Normalform ist mindestens erfüllt. Begründen Sie Ihre Antwort.

(e) Um Anfragen und Operationen effizient durchführen zu können, setzt die interne Ebene des Datenbanksystems geeignete Datenstrukturen und Speicherungsverfahren (Indexstrukturen) ein. Nennen Sie drei mögliche Anforderungen an solche Indexstrukturen.

(f) Welches Problem kann bei redundanten Daten auftreten?

(g) Nennen Sie vier Aggregationsfunktionen in SQL.

Fortsetzung nächste Seite!

Aufgabe 2 (Relationale Algebra) (4+6+3=13 PUNKTE)

Gegeben seien folgende Relationen:

R:

A	B	C	D	E	F
6	8	1	7	3	7
5	3	4	4	5	7
0	6	3	0	1	7

S:

A	C	X	Z
7	8	6	1
0	3	0	0
2	3	0	5
0	6	1	6
6	7	1	7
7	1	2	2
1	8	8	0
5	1	5	5
7	3	0	2
4	8	2	7

T:

X	Y
5	3
0	5
8	6
3	6
5	7
2	8

Geben Sie die Ergebnisrelation folgender Ausdrücke der relationalen Algebra als Tabellen an. Begründen Sie Ihr Ergebnis, gegebenenfalls durch Zwischenschritte.

(a) $\sigma_{A>6}(S) \underset{S.X=T.Y}{\bowtie} \pi_Y(T)$

(b) $\pi_{A,C}(S) - (\pi_A(R) \times \pi_C(\sigma_{X=1}(S)))$

(c) $(\pi_D(R) \times \pi_E(R)) \div \pi_E(R)$

Fortsetzung nächste Seite!

Aufgabe 3 (Anfragen) (6+4+13+15+4=42 PUNKTE)

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Schule entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen.

Die Relation *Schüler* enthält allgemeine Daten zu den Schülerinnen und Schülern. Schülerinnen und Schüler nehmen an Prüfungen in verschiedenen Unterrichtsfächern teil und erhalten dadurch Noten. Diese werden in der Relation *Noten* abgespeichert. Prüfungen haben ein unterschiedliches Gewicht. Beispielsweise hat ein mündliches Ausfragen oder eine Extemporale das Gewicht 1, während eine Schulaufgabe das Gewicht 2 hat.

Schüler:

SchülerID	Vorname	Nachname	Klasse
1	Laura	Müller	4A
2	Linus	Schmidt	4A
3	Jonas	Schneider	4A
4	Liam	Fischer	4B
5	Tim	Weber	4B
6	Lea	Becker	4B
7	Emilia	Klein	4C
8	Julia	Wolf	4C

Noten:

SchülerID	Schulfach	Note	Gewicht	Datum
1	Mathematik	3	2	23.09.2017
1	Mathematik	1	1	03.10.2017
1	Mathematik	2	2	15.10.2017
1	Mathematik	4	1	11.11.2017

In dem durch die Tabellen gegebenen Datenbank-Schema sind Primärschlüssel unterstrichen und Fremdschlüssel überstrichen.

Fortsetzung nächste Seite!

(a) Geben Sie die SQL-Befehle an, die notwendig sind, um die oben dargestellten Tabellen in einer SQL-Datenbank anzulegen.

(b) Entscheiden Sie jeweils, ob folgende Einfügeoperationen vom gegebenen Datenbanksystem (mit der angegebenen Ausprägung) erfolgreich verarbeitet werden können und begründen Sie Ihre Antwort kurz.

- `INSERT INTO Schüler (SchülerID, Vorname, Nachname, Klasse) VALUES (6, "Johannes", "Schmied", "4C");`
- `INSERT INTO Noten VALUES (9, "Chemie", 1, 2);`

(c) Geben Sie die Befehle für die folgenden Aktionen in SQL an. Beachten Sie dabei, dass die Befehle auch noch bei Änderungen des oben gegebenen Datenbankzustandes korrekte Ergebnisse zurückliefern müssen.

- Die Schule möchte verhindern, dass in die Datenbank mehrere Kinder mit dem selben Vornamen in die gleiche Klasse kommen. Dies soll bereits auf Datenbankebene verhindert werden. Dabei sollen die Primärschlüssel nicht verändert werden. Geben Sie den Befehl an, der diese Änderung durchführt.
- Der Schüler Tim Weber (SchülerID: 5) wechselt die Klasse. Geben Sie den SQL-Befehl an, der den genannten Schüler in die Klasse "4C" überführt.
- Die Schülerin Laura Müller (SchülerID: 1) zieht um und wechselt die Schule. Löschen Sie die Schülerin aus der Datenbank. Nennen Sie *einen* möglichen Effekt, welcher bei der Verwendung von Primär- und Fremdschlüsseln auftreten kann.
- Erstellen Sie eine View "DurchschnittsNoten", die die folgenden Spalten beinhaltet: Klasse, Schulfach, Durchschnittsnote
  Hinweis: Beachten Sie die Gewichte der Noten.
- Geben Sie den Befehl an, der die komplette Tabelle "Noten" löscht.

(d) Formulieren Sie die folgenden Anfragen in SQL. Beachten Sie dabei, dass die SQL-Befehle auch noch bei Änderungen der Ausprägung die korrekten Anfrageergebnisse zurückgeben sollen.

- Gesucht ist die durchschnittliche Note, die im Fach Mathematik vergeben wird. Hinweis: Das Gewicht ist bei dieser Anfrage nicht relevant.
- Berechnen Sie die Anzahl der Schüler, die im Fach Mathematik am 23.09.2017 eine Schulaufgabe (d.h. Gewicht=2) geschrieben haben.
- Geben Sie die SchülerID aller Schüler zurück, die im Fach Mathematik mindestens drei mal die Schulnote 6 geschrieben haben.
- Gesucht ist der Notendurchschnitt bezüglich jedes Faches der Klasse "4A".

(e) Geben Sie jeweils an, welchen Ergebniswert die folgenden SQL-Befehle für die gegebene Ausprägung zurückliefern.

- `SELECT count(distinct Klasse) FROM Schüler NATURAL JOIN Noten;`
- `SELECT count(all Klasse) FROM Noten, Schüler;`
- `SELECT count(Note) FROM Schüler NATURAL LEFT OUTER JOIN Noten;`
- `SELECT count(*) FROM Schüler NATURAL LEFT OUTER JOIN Noten;`

Fortsetzung nächste Seite!

Aufgabe 4 (Tupelkalkül) (2+4=6 PUNKTE)

Gegeben sei das folgende Datenbankschema, wobei Primärschlüssel unterstrichen und Fremdschlüssel überstrichen sind. Die von einem Fremdschlüssel referenzierte Relation ist in eckigen Klammern nach dem Fremdschlüsselattribut angegeben.

Schüler(SchülerID, SVorname, SNachname, $\overline{\text{KlassenID}}$[Klassen], Geburtsdatum)

Lehrer(LehrerID, LVorname, LNachname)

Klassen(KlassenID, Klassenstufe, Buchstabe)

Unterrichtsfächer(FachID, Name)

Noten(NotenID, $\overline{\text{SchülerID}}$[Schüler], $\overline{\text{FachID}}$[Unterrichtsfächer], Note, Gewicht)

LehrerUnterrichtet($\overline{\text{LehrerID}}$[Lehrer], $\overline{\text{KlassenID}}$[Klassen], $\overline{\text{FachID}}$[Unterrichtsfächer])

Formulieren Sie die folgenden Anfragen im Tupelkalkül. Geben Sie zudem das Schema aller freier Variablen an.

(a) Bestimmen Sie die Namen (Vor- und Nachname) der Schüler, die maximal 10 Jahre alt sind. Sie können dafür eine Funktion "age" verwenden, die aus dem Geburtsdatum das Alter berechnet.

(b) Bestimmen Sie die Nachnamen der Lehrer, die die Klasse 4A unterrichten.

Fortsetzung nächste Seite!

Aufgabe 5 (Enity Relationship Modell) (16 PUNKTE)

Überführen Sie das Datenbankschema aus Aufgabe 4 in ein ER-Diagramm. Verwenden Sie hierfür die bereits eingezeichneten Entity-Typen und Relationship-Typen. Weisen Sie die Relationen zu und schreiben Sie deren Namen in die dazugehörigen Felder. Fügen Sie, falls erforderlich, Attribute hinzu und beschriften Sie die Beziehungen. Markieren Sie Schlüsselattribute durch unterstreichen.

Hinweis: Übernehmen Sie die untenstehende Grafik auf Ihre Bearbeitung und vervollständigen Sie sie dort.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2019

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____ Kennwort: _____ Arbeitsplatz-Nr.: _____	**Frühjahr 2019**	**46116**

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologie/Datenbanksystme**

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

Anzahl der Druckseiten dieser Vorlage: **11**

Bitte wenden!

Thema Nr. 1

Teilaufgabe I: Softwaretechnik

Aufgabe 1 (Anforderungserhebung)

Es soll eine Webanwendung zur automatischen Überwachung und zum Handel von Aktien entwickelt werden. Sie sind für die Anforderungserhebung zuständig.

a) Erklären Sie, was an der folgenden Anforderung nicht optimal ist:
„Die Aktien sollen in einer relationalen Datenbank gespeichert werden."
Gehen Sie dabei davon aus, dass das Domänenkonzept Aktien an anderer Stelle schon definiert ist.

b) Kategorisieren Sie die Anforderungen in funktional oder nicht-funktional. Kreuzen Sie die richtige Antwort an.

Anforderung	funktional	Nicht-funktional
Der Kunde soll Aktien zu einer Liste von überwachten Aktien hinzufügen können.		
Die Antwortzeit aller Operationen muss unter 1 Sekunde liegen.		
Das System verkauft Aktien und informiert den Nutzer, sobald gewisse Aktienkurse einen vordefinierten Schwellenwert über- oder unterschreiten.		
Die Software soll barrierefrei sein und dies durch eine «Access for all» Zertifizierung nachweisen.		
Die Antwortzeit darf nicht schneller als 1 Millisekunde sein, da dies in Deutschland sonst unter den erlaubnispflichtigen Hochfrequenzhandel fallen könnte.		
Die Software muss den Regeln des deutschen Börsengesetzes folgen.		

Hintergrundinformationen für die Aufgaben 2-5: Modellierung und Implementierung eines Programms

Ein autonomes Auto hat einen Namen und fährt mit einer bestimmten Geschwindigkeit (gemessen in Meter/Sekunde). Es bekommt von verschiedenen Sensoren (z. B. Kameras) jede Zehntelsekunde neue Informationen über seine Umwelt. Wir betrachten hier nur die Situation, wenn es gerade eine neue Information bekommen hat. Ein Sensor hat einen Namen und eine Aussage über seine Funktionsfähigkeit (ausgeschaltet, schlechte Funktionsfähigkeit, gute Funktionsfähigkeit). Außerdem liefert er zwei Aussagen, ob eine rote Ampel bzw. ob eine gelbe Ampel gefunden wurde (jeweils als Boolescher Wert). Weiterhin wird der Abstand zur Ampel zurückgegeben: als Zahl in Metern.

Fortsetzung nächste Seite!

Wenn mindestens ein Sensor mit guter Funktionsfähigkeit eine rote Ampel meldet, soll das Fahrzeug bremsen. Wenn das nicht der Fall ist und mindestens ein Sensor mit guter Funktionsfähigkeit eine gelbe Ampel findet, soll das Fahrzeug abhängig von seiner Position und Geschwindigkeit entweder bremsen oder normal weiterfahren. Die Regel ist: Es soll bremsen, wenn gilt:
(Geschwindigkeit in m/s * 2 Sekunden) < Abstand_zur_Ampel (in Meter).
Im Folgenden soll ein Klassendiagramm für das Auto und die Sensoren beschrieben werden. Das Auto soll eine Methode "bremsen" haben, die abhängig vom Zustand des Autos und der Sensoren einen Booleschen Wert "wahr" (es soll bremsen) oder "falsch" (es soll weiterfahren) zurückgibt.

Aufgabe 2 (UML-Modellierung: Klassendiagramm)

Geben Sie ein minimales UML-Klassendiagramm an, das alle angegebenen Informationen enthält. Hinweis: Bei den Aufgaben 4 und 5 wird Konsistenz des Aktivitätsdiagramms bzw. des Codes mit dem Klassendiagramm verlangt.

Aufgabe 3 (Pseudocode und Aktivitätsdiagramm)

a) Geben Sie Pseudo-Code für die Methode „bremsen“ an. Benutzen Sie dazu eine separat definierte Hilfsmethode für die Entscheidung, ob das Auto bei gelber Ampel bremsen soll. Achten Sie auf die Konsistenz zum Klassendiagramm (Inkonsistenzen führen zu Punktabzügen).
b) Geben Sie zu dem Pseudo-Code aus Aufgabe 3 a zwei syntaktisch korrekte UML-Aktivitätsdiagramme an. Schleifen und Verzweigungen müssen durch strukturierte Knoten dargestellt werden (d. h. kein graphisches „goto“).

Aufgabe 4 (oo-Programmierung)

Implementieren Sie die Methode „bremsen“ der Klasse Auto in einer objektorientierten Programmiersprache (z. B. Java oder andere mit Nennung). Sie sollen nur den Code für die Methoden angeben. Sie brauchen keinen Code für Klassendefinitionen angeben, sondern können sich auf das UML-Klassendiagramm aus Aufgabe 2 beziehen.

Aufgabe 5: (JUnit Test)

Beim Programmieren ist es häufig nützlich, vor der Implementierung einige Tests mit Eingabe-Ausgabe-Spezifikationen zu schreiben, was die Methode leisten soll. Implementieren Sie dazu einen Test (z. B. in Java (JUnit) oder einer anderen Programmiersprache) für das korrekte Verhalten der Methode „bremsen“, der automatisch ausgeführt werden kann.

Teilaufgabe II: Datenbanken

Aufgabe 1: ER-Modellierung

***Hinweis**: Bei Wahl dieser Aufgabe wird Wissen über das erweiterte Entity-Relationship-Modell (beispielsweise schwache Entity-Typen, Vererbung) sowie die Verfeinerung eines relationalen Schemas vorausgesetzt.*

Gegeben seien folgende Informationen:

- Universitäten bieten Fachrichtungen an, deren Name nur innerhalb der Universität eindeutig ist. Eine Universität hat neben einem eindeutigen Namen auch eine Adresse und ein Gründungsjahr.
- Fachrichtungen wiederum bestehen aus mehreren Lehrstühlen, deren Name ebenfalls nur innerhalb der Fachrichtung eindeutig ist.
- An jedem Lehrstuhl arbeitet genau ein Professor.
- Universitäten bezahlen Angestellte. Diese werden über eine PersID identifiziert und besitzen zudem einen Namen. Angestellte werden unter anderem unterschieden in Professor, Hilfswissenschaftler (Hiwi) und wissenschaftliche Mitarbeiter, wobei ein Angestellter immer nur eines davon sein kann.
- Für einen Professor wird zudem die Anzahl seiner Veröffentlichungen gespeichert, für einen Hiwi wird die Anzahl an Semestern gespeichert. Jeder wissenschaftliche Mitarbeiter wird von genau einem Professor betreut. Jeder Hiwi unterstützt wissenschaftliche Mitarbeiter.
- Eine Universität kann zusammen mit einem Lehrstuhl einen Antrag stellen, bei dem genau ein Professor beteiligt ist.

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. Die einzelnen Schritte müssen angegeben werden.

Fortsetzung nächste Seite!

Aufgabe 2: (SQL)

Gegeben sei der folgende Ausschnitt des Schemas eines Schulverwaltungssystems:

Schueler : {[
ID : INTEGER,
Name : VARCHAR(255),
Wohnort : VARCHAR(255)
]}

Zaubererschule : {[
Name : VARCHAR(255),
Standort : VARCHAR(255)
]}

Unterrichtsfach : {[
Name : VARCHAR(100),
Jahr : INTEGER,
Lehrer : VARCHAR(255),
AnzahlSchueler : INTEGER,
wird_angeboten_von : VARCHAR(255)
]}

besucht : {[
Schueler : INTEGER,
Schulname : VARCHAR(255),
von : DATE,
bis : DATE
]}

setzt_voraus : {[
GrundlageName : VARCHAR(100),
GrundlageJahr : INTEGER,
FortgeschrittenName : VARCHAR(100),
FortgeschrittenJahr : INTEGER,
Mindestnote : {1, 2, 3, 4, 5}
]}

Die Tabelle *Schueler* enthält Informationen über Schüler. Die Tabelle *Zaubererschule* beschreibt Zaubererschulen anhand ihres Namens und des Standortes. Die Tabelle *besucht* gibt an, welcher Schüler zu welchem Zeitpunkt welche Zaubererschule besucht hat. Die Tabelle *Unterrichtsfach* enthält Informationen über die Unterrichtsfächer bezüglich deren Name und Jahr, den Namen des Lehrers, die Anzahl der teilnehmenden Schüler und den Namen der Schule als Referenz auf Zaubererschule, die dieses Unterrichtsfach anbietet. Die Tabelle *setzt_voraus* beinhaltet Informationen darüber, welche Unterrichtsfächer welche anderen Unterrichtsfächer mit welcher Mindestnote voraussetzen, jeweils mit Referenz auf Unterrichtsfach.

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

a) Schreiben Sie eine SQL-Anweisung, welche die Tabelle Unterrichtsfach mit allen Constraints (Primärschlüssel-, Fremdschlüssel- und Check-Constraints) anlegt. Stellen Sie dabei sicher, dass die Obergrenze für die Anzahl der Schüler bei 20 liegt.

b) Schreiben Sie eine SQL-Anweisung, die für jede Zaubererschule den Namen des Unterrichtsfaches zurückgibt, an dem die meisten Schüler teilnehmen (unabhängig von Jahr).

c) Schreiben Sie eine SQL-Anweisung, welche die ID und den Namen aller Schüler bestimmt, die an Zaubererschulen waren, deren Standort an ihrem Wohnort ist.

d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Unterrichtsfächer zurückgibt, welche mehr als ein anderes Unterrichtsfach mit einer Mindestnote von „2“ voraussetzen.

Fortsetzung nächste Seite!

e) Schreiben Sie eine SQL-Anweisung, die die ID und den Namen folgender Schüler ermittelt: Entweder besucht dieser Schüler keine Schule oder an allen von ihm besuchten Schulen wird das Unterrichtsfach mit dem Namen „Zaubertränke" (unabhängig von Jahr) nicht angeboten.

Aufgabe 3: (Entwurfstheorie)

Gegeben sei folgendes relationales Schema R in erster Normalform:

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

und die Zerlegung $\rho = \{R1, R2\}$ von R mit $R_1 = \{C, D\}$ und $R_2 = \{A, B, D\}$.
Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$BC \rightarrow AD;$$
$$B \rightarrow A;$$
$$D \rightarrow C$$

$\}$

a) Bestimmen Sie alle Kandidatenschlüssel/Schlüsselkandidaten 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 und/oder 3NF ist.

c) Zeigen oder widerlegen Sie, dass die Zerlegung ρ von R abhängigkeitserhaltend bzgl. FD ist.

d) Zeigen oder widerlegen Sie, dass die Zerlegung ρ von R verlustfrei bzgl. FD ist.

e) Bestimmen Sie 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).

f) Zerlegen Sie R mit FD_C mithilfe des Synthesealgorithmus in 3NF. Geben Sie zudem alle funktionalen Abhängigkeiten der erzeugten Relationenschemata an.

Thema Nr. 2

Teilaufgabe I:

Aufgabe 1 (Programmierrichtlinien)

Entscheiden Sie jeweils, ob folgende Aussagen zutreffend sind oder nicht. Begründen Sie jeweils Ihre Antwort.

a) Lesbarkeit

Programmierrichtlinien sind ein Mittel, die Lesbarkeit von Quelltext zu erhöhen.

b) Vollständigkeit

Formatieren zwei Programmierer den gleichen Quelltext nach denselben Richtlinien, wird das Ergebnis immer identisch sein.

c) Objektivität

Für eine gegebene Programmiersprache gibt es objektiv betrachtet genau eine beste Programmierrichtlinie.

d) Fehlervermeidung

Einheitlich nach einer Programmierrichtlinie formatierter Quelltext hilft, Programmierfehler zu vermeiden.

e) Teamarbeit

Ein Programmierer, der Fehler in den Programmierrichtlinien eines Projekts findet, sollte diese allein beheben und umsetzen.

Fortsetzung nächste Seite!

Aufgabe 2: (Programmierung)

Das im Folgenden dargestellte Diagramm ist das von Aufgaben 2a) - e) referenzierte Diagramm.

Das im Diagramm dargestellte Klassenmodell stellt einen einfachen Entwurf zur objektorientierten Modellierung von homogenen Namensobjekten dar. Ein homogener Name ist ein Bezeichner, der aus mehreren gleichartigen Komponenten besteht. Eine Komponente hat den Basistyp String. Beispiele für homogene Namen sind Domänennamen, z.B. "www.google.com" oder Java-Paket- und Klassennamen, z.B. "java.lang.Object". Es gilt Folgendes für den Entwurf:

- Der Entwurf ist nicht vollständig; es fehlen Operationen.
- Trennzeichen, z. B. '.', sind nicht modelliert worden; nehmen Sie an, dass fix der Punkt '.' verwendet wird.
- Die Klasse AbstractName definiert keine Felder, welche den Namen im Speicher repräsentieren. Nur die Unterklassen machen dies.

a) Schnittstellen

Erläutern Sie den technischen (nicht fachlichen) Zweck der Schnittstelle Name.

b) Abstrakte Oberklasse

Erläutern Sie den technischen (nicht fachlichen) Zweck der abstrakten Oberklasse AbstractName.

c) Instanziierbarkeit

Muss die Klasse StringName die Methode getComponent(int) implementieren, um instanziierbar zu sein? Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

d) Speicher

Welche der beiden Unterklassen StringName und StringArrayName verbraucht bei gleichem Inhalt mehr Speicher? Begründen Sie Ihre Antwort.

e) Zugriff

Was ist die Komplexität des Zugriffs auf eine beliebige Namenskomponente jeweils für StringName und StringArrayName? Verwenden Sie die O-Notation und begründen Sie Ihre Antwort.

Aufgabe 3 (Modellierung)

Ein Code-Review-System ist ein Werkzeug zur Unterstützung von Code-Reviews in Softwareentwicklungsprozessen.
Gegeben ist die folgende Beschreibung eines Teils eines solchen Systems:
"Nutzer des Systems können Autoren oder Reviewer sein. Autoren reichen veränderte Dateien ein, Reviewer begutachten die Änderungen. Zu jeder Datei soll deren Name mitgeführt werden. Jede Änderung an einer Datei hat ein Datum (Datentyp Date) und muss zuvor einen Code-Review durchlaufen, welcher von einem Reviewer ausgeführt wird. Ein Autor darf seine eigenen Änderung nicht selbst begutachten. Zu jedem Review soll der Erstellungszeitpunkt des Reviews mitgeführt werden. Das System wählt automatisch eine bestimmte Anzahl an Reviewern aus, die ihr Reviewergebnis (Enumeration Accepted, Rejected) mit einem Zeitstempel (Datentyp Date) abgeben."

Erstellen Sie ein UML-Klassendiagramm, welches die gegebene Beschreibung des Code-Review-Systems getreu wiedergibt.

Aufgabe 4 (Modellierung)

Gegeben ist die folgende Beschreibung eines Softwaresystems zur Bibliotheksverwaltung:
"Nutzende des Systems sind Studierende und Bibliothekare oder Bibliothekarinnen. Nutzende können das Inventar der Bibliothek über eine Suchmaske in einer Web-basierten Benutzungsschnittstelle durchsuchen. Das Suchergebnis zeigt die gefundenen Bücher sowie deren Verfügbarkeit an. Ein Nutzer oder eine Nutzerin kann ein Buch zur Ausleihe vormerken, sofern dieses verfügbar ist und er oder sie über ein Nutzerkonto verfügt. Beim Ausleihvorgang wird das Nutzerkonto überprüft und die Ausleihe von einem Bibliothekar oder einer Bibliothekarin autorisiert, welche das Buch aushändigt. Bei der Rückgabe des Buches wird ebenfalls das Nutzerkonto geprüft; wurde das Buch zu spät zurückgegeben, wird eine Strafgebühr erhoben."

Modellieren Sie die beschriebenen Anwendungsfälle in einem UML-Anwendungsfalldiagramm.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1: ER-Modellierung

Wir betrachten eine Datenbank, die einem Webshop zur Verwaltung seiner Daten dient. Sie verfügt über die folgenden Eigenschaften:

Für jedes Produkt kann der Name sowie das Gewicht erfasst werden. Jedes Produkt hat eine eindeutige EAN-Nummer und einen Nettopreis.

Es gibt verschiedene Steuerklassen. Für diese wird jeweils ein Bezeichner erfasst (z. B. *reduzierte MwSt.*) sowie die Steuerhöhe (z. B. *0.07*) und eine eindeutige Nummer. Jedes Produkt ist genau einer Steuerklasse zugeordnet.

Produkte können als passend zu anderen deklariert werden. Bspw. kann zu einem *Notebook* ein passendes *Netzteil* und ein passender *Akku* hinterlegt werden.

Ein Kunde ist charakterisiert durch seinen Vornamen, Nachnamen, die Postadresse sowie das Geburtsjahr. Jeder Kunde hat eine eindeutige Kundennummer.

Für jede Bestellung wird das Datum abgespeichert. Jede Bestellung ist genau einem Kunden zugeordnet. Die Bestellung enthält eine Bestellnummer, die zusammen mit der Kundennummer eindeutig ist. Eine Bestellung besteht aus beliebig vielen Artikeln, zu denen jeweils die Artikelanzahl abgespeichert wird. So können in einer Bestellung z. B. gleich zwei *Netzteile* gekauft werden.

Mögliche Dateneinträge sind *kursiv* hervorgehoben und dienen einzig der Veranschaulichung.

a) Modellieren Sie das oben genannte Datenbanksystem möglichst vollständig in einem erweiterten ER-Diagramm. Kennzeichnen Sie die Schlüssel der Entity-Typen und geben Sie für die Relationship-Typen jeweils Funktionalitätsbedingungen (in N:M-Notation) oder Kardinalitätsbedingungen (in (MIN, MAX)-Notation) an. Sollte die o. g. Spezifikation nicht ausreichend sein, geben Sie Ihre genutzten Annahmen kurz an.

b) Übertragen Sie Ihr ER-Modell in das relationale Datenmodell. Erstellen Sie hierzu die Tabellen der Entities Steuerklasse, Produkt und Bestellung mit Hilfe von `CREATE TABLE` Statements aus SQL. Verwenden Sie sinnvolle Datentypen.

c) Geben Sie ein passendes `CREATE TABLE` SQL-Statement zur Erstellung der passt_zu-Tabelle an. Definieren Sie, wenn möglich, Fremdschlüssel und geben Sie die `ON UPDATE`- und die `ON DELETE`-Modi an.

Fortsetzung nächste Seite!

Aufgabe 2: (SQL)

Gegeben sind folgende Relationen aus einer Musiksammlung:

`INTERPRET`(MID, Name, Label)
`ALBUM`(AID, MID[`INTERPRET`], Name, Verkaeufe)
`SONG`(AID[`ALBUM`], TID, Titel, Laenge)

Zur Vereinfachung enthält also jedes Album ausschließlich Titel desselben Interpreten.

a) Schreiben Sie eine SQL-Anweisung, die einen neuen Interpreten namens *Nena* vom Label *NDW* anlegt.

b) Schreiben Sie eine SQL-Anweisung, die einen View erzeugt, der zu jedem Interpretennamen die Gesamtzahl all seiner verkauften Alben enthält.

c) Schreiben Sie eine SQL-Anweisung, die alle Alben löscht, zu denen kein Song vorliegt.

d) Schreiben Sie eine SQL-Anweisung, die alle Songtitel zurückgibt, deren Länge größer ist als die durchschnittliche Länge aller Songs. Die Songs sollen absteigend nach ihrer Länge ausgegeben werden. Sie können davon ausgehen, dass die Songlänge in Sekunden als Ganzzahl hinterlegt ist.

e) Schreiben Sie eine SQL-Anweisung, die zu jedem Label die Anzahl der gelisteten Interpreten zurückliefert.

f) Schreiben Sie eine SQL-Anweisung, die die Verkaufszahlen aller Alben um eins erhöht.

g) Diskutieren Sie kurz, ob Name statt MID ein geeigneter Primärschlüssel für `INTERPRET` gewesen wäre.

Aufgabe 3: (Entwurftheorie)

Gegeben sei das Relationenschema $R = (U, F)$ mit der Attributmenge U und der Menge F von funktionalen Abhängigkeiten:

$U = \{A, B, C, D, E\}$,
$F = \{ \{B, E\} \rightarrow \{A, D\}, \{B, C\} \rightarrow \{A, E\}, \{E\} \rightarrow \{C\}\}$.

Geben Sie für alle Antworten jeweils Begründungen an.

a) Erklären Sie die bekannten Anomalien anhand von R.

b) Geben Sie alle Schlüssel sowie die Nichtschlüsselattribute für das Relationenschema R an.

c) Zeigen Sie, dass R in 3NF ist.

d) Ist R auch in BCNF?
Geben Sie – falls R nicht in BCNF ist – eine BCNF-Zerlegung von R an.

e) Geben Sie eine Basis G von F an und führen Sie damit die 3NF-Synthese aus.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Herbst 2019

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2019**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologie/Datenbanksystme**

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

Anzahl der Druckseiten dieser Vorlage: **14**

Bitte wenden!

Thema Nr. 1

Aufgabengruppe

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 (Klassen- und Objektdiagramme) (30 Punkte)

In einem Fußballturnier treten Mannschaften in Spielen gegeneinander an. An jedem Spiel sind zwei Mannschaften beteiligt. Jedes Spiel hat ein Ergebnis und genau einen Sieger. Das Turnier besteht aus mehreren Runden. In der ersten Runde treten alle Mannschaften an. Der Sieger erreicht jeweils die nächste Runde. Ein Spielplan legt fest, welche Mannschaften gegeneinander antreten. Das Turnier endet mit einem Endspiel (kein Spiel um den dritten Platz).

Beispiel: Endrunde der WM 2018 ab Viertelfinale

a) Erstellen Sie ein UML-Klassendiagramm zur Modellierung eines Fußballturniers, an dem 2^n Mannschaften teilnehmen können ($n \geq 1$ beliebig). Geben Sie für Attribute jeweils den Namen und den Datentyp an. Geben Sie für Assoziationen den Namen und für jedes Ende den Rollennamen und die Multiplizität an. Nutzen Sie ggf. Vererbung, Komposition oder Aggregation. Verzichten Sie auf Operationen und Sichtbarkeiten.

b) Erstellen Sie für das Klassendiagramm aus a) und die Endrunde aus der Aufgabenstellung ein Objektdiagramm. Beschränken Sie sich dabei auf die Mannschaften und Spiele im Halbfinale und im Finale. Geben Sie Rollennamen nur an, wenn es notwendig ist, um die Enden eines Links (Instanz einer Assoziation) zu unterscheiden.

Fortsetzung nächste Seite!

Aufgabe 2 (Entwurfsmuster) (30 Punkte)

Ein Kurznachrichtendienst versendet Nachrichten, die aus einer Rubrik (Politik, Wirtschaft oder Sport) und einer Zeichenkette bestehen. Abonnenten können sich beim Kurznachrichtendienst registrieren. Der Kurznachrichtendienst versendet alle Nachrichten an alle Abonnenten. Die Abonnenten entscheiden darüber, wie sie die Nachricht verarbeiten. Ein Abonnent für Sportnachrichten gibt auf der Standardausgabe die Überschrift SPORT, gefolgt vom Nachrichtentext aus; Abonnenten für die übrigen Rubriken verhalten sich analog.

a) Implementieren Sie diese Anwendung in einer Programmiersprache Ihrer Wahl mit Hilfe des Beobachter-Musters; geben Sie diese Sprache an. Bei den Abonnenten beschränken Sie sich dabei auf die Rubrik Sport.

b) Visualisieren Sie folgenden Ablauf mit Hilfe eines Sequenzdiagramms, wobei Sie annehmen, dass das Senden von Nachrichten von einem Testprogramm ausgelöst wird:

   1. Der Abonnent `politik` registriert sich.
   2. Der Abonnent `sport1` registriert sich.
   3. Das Testprogramm ruft die Versende-Operation des Kurznachrichtendienstes auf.
   4. Der Abonnent `sport1` deregistriert sich.
   5. Der Abonnent `sport2` registriert sich.
   6. Der Abonnent `wirtschaft` registriert sich
   7. Das Testprogramm ruft die Versende-Operation des Kurznachrichtendienstes auf.

Hinweis: Verzichten Sie auf Aktivierungsbalken, Aufrufparameter und Rücksprünge. Verwenden Sie als Grundlage Ihre Lösung a). Setzen Sie das Testprogramm als gegeben voraus.

Fortsetzung nächste Seite!

Aufgabe 3 (Zustandsbasiertes Testen) (15 Punkte)

Gegeben sei der folgende endliche Automat zur Erkennung von Dualzahlen mit optionalem Vorzeichen und optionalem gebrochenen Anteil mit Dezimalpunkt:

a) Erläutern Sie die Begriffe Zustandsüberdeckung und Transitionsüberdeckung beim zustandsbasierten Testen.

b) Geben Sie eine minimale Testmenge für die Zustandsüberdeckung an.

c) Geben Sie eine minimale Testmenge für die Transitionsüberdeckung an.

d) Welches der beiden Überdeckungskriterien ist schärfer? Begründen Sie Ihre Antwort.

Hinweis: Eine Testmenge ist minimal, wenn es keine andere Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität braucht nicht bewiesen zu werden.

Aufgabe 4 (CPM-Netzwerke)

(15 Punkte)

Gegeben sei folgendes CPM-Netzwerk („Critical Path Method"):

a) Berechnen sie für jedes Ereignis den frühesten Zeitpunkt (Projektstart zum Zeitpunkt 0).

b) Berechnen Sie für jedes Ereignis den spätesten Zeitpunkt (= frühester Zeitpunkt des letzten Ereignisses).

c) Berechnen Sie für jedes Ereignis die Pufferzeit.

d) Geben Sie einen kritischen Pfad für das Netzwerk an.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1 (Wissensfragen) (5 Punkte)

Antworten Sie kurz und prägnant.

a) Beschreiben Sie kurz den Zweck einer Datenbank, eines Datenbankmanagementsystems und eines Datenbankensystems.

b) Nennen Sie zwei verschiedene Notationen, die man zur Darstellung von Mengenrelationen in der ER-Modellierung verwendet. Geben Sie zusätzlich jeweils die möglichen Ausprägungen der Notation an.

Aufgabe 2 (ER-Modellierung) (35 Punkte)

Entwerfen Sie ein ER-Diagramm, welches eine imaginäre Fernsehserie abbildet:

In dieser Serie gibt es Figuren mit nicht eindeutigen Namen, einem Sozialstatus und einem Wohlstand. Sie gehören entweder zum Volk, zu den Adeligen oder zu den Regierenden.
Ein Adeliger oder ein Regierender ist zwangsläufig Teil einer Familie. Diese Familie ist eindeutig gekennzeichnet durch den Namen, und hat zusätzlich noch eine Anzahl an Mitgliedern. Es gibt weiterhin Ortschaften, welche einen eindeutigen Namen haben und entweder ein Dorf, eine Ruine, eine Festung oder ein Hafen sein können. Jede existierende Ortschaft ist genau einer Familie zugeordnet, wobei Familien bereits ausgestorben sein können und daher keine Ortschaften oder Figuren zugeordnet haben.
Weiterhin gibt es sieben Königreiche. Jedes Königreich hat einen gewissen Reichtum und hat zugeordnete Familien, wobei einem Königreich mindestens fünf Familien zugehören müssen. Weiterhin regiert eine Familie ein oder mehrere (bis zu allen 7) Königreiche.
Außerdem gibt es Armeen, die abhängig von einer Familie sind. Da eine Familie mehrere Armeen haben kann, sind diese zusätzlich durch ihren Rufnamen identifiziert. Dabei kann eine Familie eine Armee besitzen, muss aber nicht. Die Armeen haben verschiedene Größen und sind unterschiedlich stark.
Zuletzt gibt es besondere Figuren, die einen Drachen fliegen können. Dieser existiert nur in Beziehung mit seinem Besitzer und hat einen Namen.

a) Modellieren Sie ein ER-Diagramm. Überlegen Sie sich, welche Entitäten und welche Arten von Beziehungen in diesem Beispiel relevant sind, bestimmen Sie Primärschüssel und fügen Sie die beschriebenen Attribute hinzu. Falls kein eindeutiger Primärschlüssel gegeben ist, führen Sie einen Surrogatschlüssel ein.

b) Geben Sie im Diagramm die Funktionalitäten in der Min-Max-Notation an!

Fortsetzung nächste Seite!

Aufgabe 3 (SQL) (30 Punkte)

Formulieren Sie folgende Anfragen in SQL gegen die angegebene Zauberer-Datenbank.

Schüler (Id, Name, Patronus, Haarfarbe, Aktiv, Gesamtnote)
teil_von (Id, Name,
FK (Id) references Schüler(Id),
FK (Name) references Haus(Name))
Haus (Name, Geist, Vertrauenslehrer, Wappen)
Jahrgang (Schuljahr, Einschulung, Größe)
gehört_zu (Id, Einschulung
FK (Id) references Schüler(Id),
FK (Einschulung) references Jahrgang(Einschulung))
Quidditch (Haus, Captain, Tropäen,
FK (Haus) references Haus(Name))

a) Welche Schüler (Name ist gesucht) hat eine "Hirschkuh" als Patronus?

b) Wie viele Schüler gehören jeweils zu den verschiedenen Häusern?

c) Geben Sie eine Liste der im Haus Griffindor existierenden Patronie an.

d) Gesucht sind die besten 5 Schüler aus dem 4. Schuljahr. Geben Sie den Namen, die Note und die Platzierung geordnet nach der Platzierung aus.

e) Erstellen Sie eine neue Tabelle namens "Familie" mit folgendem Schema.

Familie (Name, Familiensymbol, Oberhaupt)

Alle Attribute müssen gesetzt werden.

f) Ändern Sie die Tabelle Schüler so, dass jeder Schüler einer Familie angehören kann, aber nicht muss.

g) Geben Sie die Namen der Schüler an, bei welchen das Familiensymbol nicht identisch zu irgendeinem Hauswappen eines Hauses ist.

h) Löschen Sie alle Schüler aus Jahrgängen mit weniger als 20 Schülern.

Fortsetzung nächste Seite!

Aufgabe 4 (Relationale Algebra)

(10 Punkte)

Gegeben ist das Datenbankenschema aus Aufgabe 3.

Übertragen Sie die folgenden Ausdrücke in die relationale Algebra. Beschreiben Sie diese Ausdrücke umgangssprachlich, bevor Sie die Ausdrücke umformen.

a) $\{s \mid s \in \text{Schüler} \wedge \neg\exists t \in \text{teil_von}\ (t.\text{Id} = s.\text{Id})\}$

b) $\{s \mid s \in \text{Schüler} \wedge \exists t \in \text{teil_von}\ (t.\text{Id} = s.\text{Id}) \wedge$
$\exists h \in \text{Haus}\ (t.\text{Name} = h.\text{Name}) \wedge \exists q \in \text{Quidditch}\ (h.\text{Name} = q.\text{Haus} \wedge q.\text{Captain} =$
"Harry Potter")$\}$

Aufgabe 5 (Kanonische Überdeckung und Normalfall) (10 Punkte)

Gegeben sei $\mathcal{R} = (A,B,C,D,E,F,G,H)$ mit folgenden funktionalen Abhängigkeiten.

$F \rightarrow AH$
$E \rightarrow FG$
$AG \rightarrow BF$
$BC \rightarrow DH$

a) Geben Sie die Hülle von E an.

b) Geben Sie alle Schlüsselkandidaten an.

c) Überführen Sie die Relation $\mathcal{R}$ in die 3. NF, indem Sie den Synthesealgorithmus anwenden.

Thema Nr. 2

Aufgabengruppe

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

(90 Punkte)

Ein Testfall besteht neben Informationen zur HW/SW-Konfiguration des Testsystems aus einer Eingabe und einer erwarteten Ausgabe oder der Charakterisierung einer erwarteten Ausgabe für das Programm. Für ein Programm, das die Quadratwurzel berechnet, sind mögliche Testfälle (4.0,2.0), (-2.3,error) oder für die Ausgabe x des Programms (0.04,0.19<=x<=0.21).

a) [3 Punkte] Warum ist der erste Testfall problematisch und eine Lösung in der Art des *dritten Testfalls vorzuziehen?*

b) [2+2 Punkte] Warum kann es allgemein sinnvoll sein, nicht konkrete Ausgabewerte als erwartete Ausgabe anzugeben, sondern stattdessen eine Menge möglicher Werte zu charakterisieren? Nennen Sie zwei Gründe, die unterschiedlich von der Begründung zu a) sind.

Wir vernachlässigen im Folgenden das Problem der Bestimmung der erwarteten Ausgabe, indem wir als erwartete Ausgabe konstant „nicht abgestürzt" bzw. „keine Exception" definieren. Bei der Testeingabeselektion wird der Eingaberaum üblicherweise in k Blöcke partitioniert und aus jedem Block dieselbe Anzahl n an Testdaten zufällig gleichverteilt ausgewählt. Eingaben können bzgl. einer hier angenommenen Spezifikation fehlerverursachend oder korrekt sein. Ein Ziel des Testens ist es, Tests mit möglichst vielen fehlerverursachenden Eingaben zu definieren.

Nun interessiert uns die Frage nach der Güte dieses sogenannten partitionsbasierten Testens, also der k*n Testfälle. Nehmen wir an, dass das Gütemaß „Wahrscheinlichkeit, mindestens einen Fehler zu finden" ist, also mindestens einen Testfall mit einer fehlerverursachenden Eingabe definiert zu haben.

c) [14 Punkte] Wie groß ist dann i. A. die Güte einer Testsuite aus n*k Testfällen für k Partitionen, wenn wir annehmen, dass die Testfälle unabhängig voneinander ausgewählt wurden, derselbe Test mehrfach ausgewählt werden kann und Block i der Partition c_i korrekte und w_i fehlerverursachende Eingaben enthält, so dass die Wahrscheinlichkeit eines Elements des Blocks i, fehlerverursachend zu sein, gleich der Fehlerrate r_i=w_i/(c_i+w_i) ist? Geben Sie die Güte der Testsuite als Wahrscheinlichkeitsfunktion in Abhängigkeit von r_i, k und n an.

   Hinweis: Machen Sie sich klar, dass die Wahrscheinlichkeit, keine inkorrekte Eingabe auszuwählen, das Komplement der Wahrscheinlichkeit ist, mindestens eine inkorrekte Eingabe auszuwählen und dass die Wahrscheinlichkeiten, in den einzelnen Blöcken voneinander zu finden, voneinander statistisch unabhängig sind.

Fortsetzung nächste Seite!

d) [10 Punkte] Eine Alternative zum partitionsbasierten Testen ist das gleichverteilt zufällige Testen. Damit die Werte vergleichbar sind, nehmen Sie an, dass diese zufällige Testsuite aus N=k*n Testfällen und der Eingaberaum aus C=c_1+...c_k korrekten und W=w_1+...+w_k fehlerverursachenden Eingaben besteht und die Wahrscheinlichkeit jeder Eingabe gleich der Fehlerrate R=W/(C+W) ist, fehlerverursachend zu sein. Wie groß ist die Güte dieser zufälligen Testsuite?

Partitionsbasiertes Testen ist i. A. aufwändiger als zufälliges Testen, weil die Partitionen errechnet und Testfälle entsprechend selektiert werden müssen. Deswegen sollte partitionsbasiertes Testen nur angewendet werden, wenn es gemäß unserem Gütemaß „besser" als zufälliges Testen ist.

e) [15 Punkte] Identifizieren Sie für C+W=100, W=8, n=1, N=2 und k=2 drei Situationen, also Verteilungen von fehlerverursachenden Eingaben über die 100 Elemente des Eingaberaums, in denen partitionsbasiertes Testen einmal besser als zufälliges Testen, einmal schlechter als zufälliges Testen und einmal ebenso gut wie zufälliges Testen ist. Berechnen Sie die Güte der drei Testsuiten.

Aufgabe e) hat eine fundamentale Konsequenz: Da man i. A. nicht a-priori sagen kann, wo sich im Eingaberaum die fehlerverursachenden Eingaben befinden, kann man i. A. auch nicht voraussetzen, dass partitionsbasiertes Testen gemäß unserem Gütemaß immer besser als zufälliges Testen ist, was aus dieser theoretischen Perspektive das zufällige Testen als sinnvolle Alternative erscheinen lässt!

f) [10 Punkte] Unter welchen Bedingungen wird partitionsbasiertes Testen i. A. besser sein als zufälliges Testen? Charakterisieren Sie abstrakt die dazu notwendige Verteilung der fehlerverursachenden Eingabedaten in den Blöcken der Partition.

g) [7 Punkte] Wie sehen i. A. Testauswahlstrategien aus, die solche Partitionen induzieren? Begründen Sie Ihre Antwort.

Anforderungsbasiertes Testen bezeichnet eine Strategie, bei der Tests gemäß der Programmspezifikation ausgewählt werden: Pro in der Spezifikation beschriebener Funktionalität (use case) wird eine definierte Anzahl von Testfällen ausgewählt. Funktionalitäten bilden jeweils unterschiedliche Eingabemengen auf Klassen von Ausgaben ab und induzieren also eine Partition des Eingabedatenraums. Die vorherigen Teilaufgaben zeigen deswegen, dass anforderungsbasiertes Testen besser als, gleich gut wie oder schlechter als zufälliges Testen sein kann – und das führt zu dem überraschenden Resultat, dass man, wenn man das hier erarbeitete Modell zugrundelegt, aus Kostengründen eher zufällig als anforderungsbasiert testen sollte.

Fortsetzung nächste Seite!

h) [8 Punkte] Warum kann es sinnvoll sein, trotzdem anwendungsbasiert zu testen? Warum ist ein Großteil des Testens in der Praxis anforderungsbasiert?

Eine andere Art des Testens ist das Grenzwerttesten. Dabei werden aus der Spezifikation für geordnete Datentypen relevante Intervalle extrahiert und als Grundlage für das Testen verwendet, in dem an den Rändern sowie aus der Mitte des Intervalls Tests ausgewählt werden. Eine Möglichkeit unter anderen ist, dass je ein Test genau links der linken/rechten Intervallgrenze, je ein Test genau auf der linken/rechten Intervallgrenze, je ein Test genau rechts der linken/rechten Intervallgrenze und ein Test an beliebiger anderer Stelle des Intervalls ausgewählt wird. Wenn also beispielsweise eine Spezifikation besagt, dass bei Temperaturen zwischen 4 und 80 Grad eine grüne Lampe leuchten soll und andernfalls eine rote, sind typische Grenzwerttests (3, rot), (4, grün), (5, grün), (42, grün), (79, grün), (80, grün), (81, rot). Hintergrund ist, dass empirisch Programmierer häufiger Fehler in „Randbereichen" machen.

i) [6 Punkte] Wenn wir also den Rändern von Intervallen von im Vergleich zu den sonstigen Eingaben erhöhten Fehlerwahrscheinlichkeiten ausgehen können, wie sieht dann allgemein eine partitionsbasierte Teststrategie mit 3 Blöcken und 3 Tests für ein Intervall [i...j] mit j>i+3 aus? Können wir davon ausgehen, dass diese Teststrategie i. A. besser sein wird als zufälliges Testen?

100 % Pfadabdeckung eines deterministischen Programms bedeutet, dass alle Pfade von Startknoten zu Endknoten im Kontrollflussgraphen ausgeführt werden. Wir gehen im Folgenden davon aus, dass alle Pfade endlich sind, alle Schleifenzähler also eine uns vorab bekannte konstante obere Schranke besitzen.

j) [2 Punkte] Kann man für jedes Programm eine solche obere Schranke für alle Schleifen berechnen, wenn es sie gibt, und umgekehrt herausfinden, ob jede Schleife über eine solche obere Schranke verfügt?

k) [1+1 Punkte] Das Testkriterium Pfadabdeckung induziert offenbar eine Partition des Eingabedatenraums. Charakterisieren Sie umgangssprachlich diese Partition und diskutieren Sie anhand des oben entwickelten Modells, ob Sie einem Praktiker diese Form der Pfadabdeckung empfehlen würden.

l) [3 Punkte] Wenn (endliche) Pfadabdeckung im obigen Sinn als Testselektionskriterium verwendet wird, kann damit die Korrektheit von Programmen bewiesen werden? Begründen Sie Ihre Antwort.

Das hier diskutierte Modell zur Bemessung der Qualität einer Testsuite basiert auf diversen Annahmen.

m) [3+3 Punkte] Fassen Sie diese Annahmen zusammen und diskutieren Sie sowohl diese Annahmen als auch das gewählte Gütemaß. Wie könnte umgangssprachlich ein besseres Gütemaß aussehen?

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1 (ER-Modellierung) (25 Punkte)

Erstellen Sie ein möglichst einfaches ER-Schema, das alle unten angegebenen Informationen enthält. Attribute von Entitäten und Beziehungen sind anzugeben, Schlüsselattribute durch Unterstreichen zu kennzeichnen. Verwenden Sie für die Angabe der Kardinalitäten von Beziehungen die Min-Max-Notation. Führen Sie Surrogatschlüssel nur dann ein, wenn es nötig ist, und modellieren Sie nur die im Text vorkommenden Elemente.

Zunächst gibt es Autos, die einen eindeutigen Namen, einen Typ sowie eine Liste an Ausstattungen besitzen. Autos werden aus Bauteilen zusammengesetzt. Diese besitzen eine ID sowie eine Beschreibung. Jedes Bauteil wird von genau einem Zulieferer geliefert. Zu jedem Zulieferer werden sein Name sowie seine Email-Adresse gespeichert.

Weiter gibt es Werke, die einen eindeutigen Namen sowie einen Standort besitzen. Jedes Werk besteht aus Hallen, welche werksintern eindeutig nummeriert sind. Zudem besitzt eine Halle noch eine gewisse Größe (in m²). Es gibt genau zwei Typen von Hallen: Produktionshallen und Ersatzteillager. In jeder Produktionshalle wird mindestens ein Auto hergestellt. Zu den Ersatzteillagern wird festgehalten welche Bauteile und wie viele sich dort befinden.
Zu jedem Mitarbeiter werden eine eindeutige ID, sein Vor- und Nachname, die Adresse (Straße, PLZ, Ort), das Gehalt sowie das Geschlecht gespeichert. Jeder Mitarbeiter besitzt genau einen Vorgesetzten, der selbst Mitarbeiter ist.

Mitarbeiter werden unter anderem in Reinigungskräfte, Werksarbeiter und Ingenieure unterteilt. Zu letzteren wird zusätzlich der Hochschulabschluss gespeichert. Ingenieure sind genau einem Werk zugeordnet, Werksarbeiter einer Halle. Eine Reinigungskraft reinigt mindestens eine Halle. Jede Halle muss regelmäßig gereinigt werden. Weiter sind Ingenieure Projekten zugeteilt. Zudem wird zu jedem Projekt genau ein Ingenieur als Projektleiter festgehalten.

Fortsetzung nächste Seite!

Aufgabe 2 (Relationaler Entwurf und Normalisierung) (22 Punkte)

Gegeben ist folgendes unsaubere ER-Modell. Das Attribut „foerdert" gibt an, ob die Stadt die Theater in ihrer Stadt fördert, d. h. eine Stadt fördert jeweils alle oder kein Theater dieser Stadt.

a) [7 Punkte] Transformieren Sie das ER-Diagramm in ein relationales Modell. Geben Sie die Herkunft der Fremdschlüssel an und markieren Sie die Primärschlüssel durch Unterstreichen.

b) [4 Punkte] Geben Sie jeweils die Definition für die erste und zweite Normalform an.

c) [7 Punkte] Welche Normalform erfüllen Ihre in Teilaufgabe a) erstellten Relationen jeweils? Stellen Sie ihre Ergebnisse in folgender Form dar und begründen Sie Ihre Antwort.

Relation	1NF	2NF	3NF	BCNF
Tabelle1				
Tabelle2				
...				

d) [4 Punkte] Transformieren Sie, sofern nötig, Ihr Schema in die dritte Normalform. Geben Sie alle hierfür notwendigen Zwischenschritte an.

Fortsetzung nächste Seite!

Aufgabe 3.1 (SQL-Abfragen) (20 Punkte)

Gegeben sind folgende Relationen aus einem Verwaltungssystem für die jährlichen Formel 1-Rennen:

```
Strecke(Strecken_ID, Streckenname, Land, Länge)
Fahrer(Fahrer_ID, Fahrername, Nation, Rennstall)
Rennen(Strecken_ID, Jahr, Wetter)
            FK (Strecken_ID) referenziert Strecke(Strecken_ID)
Rennteilnahme(Fahrer_ID, Strecken_ID, Jahr, Rundenbestzeit, Gesamtzeit, disqualifiziert)
            FK (Fahrer_ID) referenziert Fahrer(Fahrer_ID)
            FK (Strecken_ID, Jahr) referenziert Rennen(Strecken_ID, Jahr)
```

Der Einfachheit halber wird angenommen, dass Fahrer den Rennstall nicht wechseln können. Das Attribut „disqualifiziert" kann die Ausprägungen „ja" und „nein" haben.
Formulieren Sie folgende Abfragen in SQL. Vermeiden Sie nach Möglichkeit übermäßige Nutzung von Joins und Views.

a) [3 Punkte] Geben Sie die Anzahl der Fahrer aus, die 2017 am Rennen in „Monaco" teilgenommen haben.

b) [4 Punkte] Geben Sie für jeden Fahrer, der in den Jahren 2005-2017 mindestens einmal disqualifiziert wurde, seinen Namen, seine ID sowie die Anzahl seiner Disqualifikationen in diesem Zeitraum an. Ordnen Sie die Ausgabe absteigend nach der Anzahl der Disqualifikationen.

c) [4 Punkte] Gesucht sind alle Länder aus denen noch nie ein Fahrer disqualifiziert wurde.

d) [9 Punkte] Schreiben Sie eine SQL-Anweisung, die aus der Tabelle Strecke alle Strecken löscht, bei denen das Wetter öfter „regnerisch" als „sonnig" war.

Aufgabe 3.2 (Tabellen mit SQL erstellen) (5 Punkte)

Erstellen Sie in SQL eine Tabelle Person und setzen Sie die folgenden Constraints. Wählen Sie geeignete Datentypen.

```
Person(SSN, Name, Vorname, Geschlecht, Geburtsdatum, Gehalt)
```

Constraints:

- SSN ist der Primärschlüssel der Relation
- Gehalt muss zwischen 2000 und 3000 liegen
- Geschlecht darf nur die Ausprägungen 'm', 'w' und 'd' besitzen

Fortsetzung nächste Seite!

Aufgabe 4 (Vermischte Fragen) (18 Punkte)

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antworten.

a) [2 Punkte] Nennen Sie den Unterschied zwischen einer Spezialisierungshierarchie und einem Spezialisierungsbaum.

b) [2 Punkte] Sind Chen- und min-max-Notation äquivalent?

c) [2 Punkte] Gibt es Sachverhalte, die in der ER-Modellierung nicht dargestellt, in SQL allerdings umgesetzt werden können? Falls ja, geben Sie konkrete Beispiele an, falls nein, begründen Sie ausführlich!

d) [3 Punkte] Was versteht man unter Unionkompatibilität? Nennen Sie drei SQL-Operatoren, welche Unionkompatibilität voraussetzen.

e) [3 Punkte] Nennen Sie die drei Teilsprachen von SQL und beschreiben Sie kurz deren Aufgaben.

f) [2 Punkte] Erklären Sie den Begriff „Datenbankindex“ und nennen Sie zwei häufige Arten.

g) [4 Punkte] Nennen Sie zwei Möglichkeiten, die Laufzeit einer SQL-Anfrage zu verbessern. Illustrieren Sie diese an je einem kurzen Beispiel.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2020

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	**Frühjahr 2020**	**46116**
Arbeitsplatz-Nr.: _____		

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: 13

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 (Aktivitätsprogramm) [15 PUNKTE]

In dieser Aufgabe wird der fiktive Messenger XYZ für Smartphones betrachtet, welcher auch mit einer Smartwatch interagieren kann, falls diese mit dem Handy verbunden ist. Wir gehen davon aus, dass maximal eine Smartwatch mit dem Handy verbunden ist. Erstellen Sie ein Aktivitätsdiagramm für den im Folgenden beschriebenen Umgang des Messenger XYZ mit neu ankommenden Nachrichten.

Erhält der Messenger XYZ auf dem Handy eine neue Nachricht, überprüft er zuerst, ob das Handy stumm geschaltet ist. Falls das Handy stumm geschaltet ist, zeigt der Messenger XYZ nur eine kurze Benachrichtigung über eine neue Nachricht auf dem Handy an. Ist das Handy hingegen nicht auf stumm geschaltet, so überprüft der Messenger XYZ, ob eine Smartwatch verbunden ist. Ist keine Smartwatch mit dem Handy verbunden, so wird die komplette Nachricht auf dem Handy angezeigt. Sollte eine Smartwatch verbunden sein, so überprüft der Messenger, ob auf dieser ebenfalls der Messenger XYZ installiert ist. Falls ja, so wird die komplette Nachricht auf der Smartwatch angezeigt. Falls nein, so wird die komplette Nachricht auf dem Handy angezeigt.

Aufgabe 2 (UML-Klassendiagramm) [37 PUNKTE]

Modellieren Sie die wie folgend spezifizierte Autovermietung mit Hilfe eines UML-Klassendiagramms.

In dieser Aufgabe soll der Fuhrpark einer Autovermietung modelliert werden.

Die Autovermietung besitzt eine Reihe von Fuhrparks. Ein Fuhrpark ist einer Stadt zugeordnet und besitzt eine Reihe von benachbarten Fuhrparks, von denen Fahrzeuge geliehen werden können.

Es gibt zwei Arten von Fahrzeugen: PKWs und Kombis. Jedes Fahrzeug ist gekennzeichnet durch einen Modelltyp und eine interne ID. Für Kombis ist zusätzlich die Größe des Frachtraums relevant. Ein Fuhrpark besitzt jedes Fahrzeug in einer bestimmten Anzahl.

Jedes Fahrzeug hat mehrere Mietpreise: Einen Jahrespreis, für den das Fahrzeug für ein Jahr geliehen werden kann; alternativ gibt es zwei Arten von Tagespreisen, bei denen jeweils eine bestimmte Rate pro Tag gezahlt werden muss. Es gibt zusätzlich zum Standardpreis, bei dem die normale Rate gezahlt werden muss, noch einen Frühbucherpreis, bei dem der Preis sich aus einem prozentualen Rabatt und der gleichen Rate wie beim Standardpreis berechnet.

Hinweis: Achten Sie auf das Vorhandensein von beschrifteten Assoziationen, Attributen und Methoden und nutzen Sie Vererbungsstrukturen, wenn dies angemessen erscheint. Sichtbarkeiten müssen nicht modelliert werden.

Fortsetzung nächste Seite!

Aufgabe 3 (Design-Patterns) [20 PUNKTE]

Adapter

3.1 Einsatz [2 Punkte]

Unter welchen Umständen wird das Adaptermuster eingesetzt?

3.2 Typen [2 Punkte]

Nennen Sie die beiden Typen des Adaptermusters und zeichnen Sie jeweils das Klassendiagramm. Geben Sie dabei explizit an, wie die adaptierte Methode aufgerufen wird.

Aufgabe 4 (Implementierung) [18 PUNKTE]

Gegeben sei folgender Java Code:

```
1  // does the main stuff
2  public class Class{
3  static int a,b;
4  static final int ONE = 1;
5
6  /**
7   * Computes
8   *
9   * @param x number
10  * @return b result
11  */
12 public static long compute(int x) {
13 if (x <= ONE) return x;
14 else return compute(x-1) + compute(x-2);
15 }
16
17 /**
18  * Computes for all
19  *
20  * @param a lower
21  * @param b upper
22  */
23 public static void something(int a, int b) {
24 for (int i = a; i <= b; i++)
25 System.out.println(i + ": " + compute(i));
26 }
27
28 /**
29  * Performs the task.
30  */
31 public static void main(String[] args) {
32 a = Integer.parseInt(args[0]);
33 b = Integer.parseInt(args[0]);
34 something(a, b)
35 }
36 }
```

Fortsetzung nächste Seite!

4.1 Funktion [2 Punkte]

Benennen Sie, was ausgegeben wird, wenn die `main()`-Funktion dieser Klasse aufgerufen wird.

4.2 Verbesserungen [16 Punkte]

Benennen Sie *vier* Möglichkeiten, wie der obige Code hinsichtlich Stil und Lesbarkeit verbessert werden kann. Geben Sie jeweils an

(i) warum dies im jetzigen Zustand suboptimal ist

(ii) ein betroffenes Beispiel im Code

(iii) wie dies behoben werden kann und

(iv) wie die korrigierte Version aussieht.

Teilaufgabe II: Datenbanken

Aufgabe 1 (Grundwissen Datenbanksysteme) [3+4+3+7 PUNKTE]

1. Nennen Sie drei Gründe für den Einsatz eines Datenbanksystems. (**3 Stichworte**)
2. Nennen Sie die vier transaktionalen Garantien an Datenbankensyteme, die unter dem Namen ACID zusammengefasst werden. (**nur Stichpunkte, keine Erklärung nötig**)
3. Nennen Sie drei Vorteile, die man mit einem RAID-Verbund von Festplatten erzielen kann. (**nur Stichpunkte**)

4.a) Was ist der Unterschied zwischen einer Punktanfrage (Point Query) und einer Bereichsanfrage (Range Query)?

b) Geben Sie je eine beispielhafte SQL Anfrage für eine Punkt- und eine Bereichsanfrage an. Benutzen Sie hierzu die Tabelle *elements* mit den Spalten *Key* und *Value*.

c) Nennen Sie für beide Arten von Anfragen jeweils eine geeignete Indexstruktur, um die Zugriffe auf die Tabelle zu beschleunigen. Die beiden Indexstrukturen sollen möglichst effizient für die jeweilige Anfrageart sein. **(nur Namen nennen, keine Erklärung nötig)**

Aufgabe 2 (Relationale Modellierung) [2+4+2 PUNKTE]

Fortsetzung nächste Seite!

a) Welche partiellen Funktionen gelten?

b) Setzen Sie das ER-Modell in eine – oder falls nötig mehrere – Relation(en) um.
(Form: RelationName : Attribut 1, Attribut 2, ...).
Bestimmen Sie einen Schlüssel für die Beziehung R, so dass möglichst viele Einschränkungen aus dem ER-Modell auch in der Relation für die Beziehung modelliert werden.

Aufgabe 3 (Relationales Modell) [6+6 PUNKTE]

Gegeben ist folgendes Universitätsschema (eine beispielhafte Ausprägung hängt der Klausur an):

$$
\begin{aligned}
\text{Studenten} &: \{[\underline{\text{MatrNr : integer}}, \text{Name : string}, \text{Semester : integer}]\} \\
\text{Vorlesungen} &: \{[\underline{\text{VorlNr : integer}}, \text{Titel : string}, \text{SWS : integer}, \text{gelesenVon : integer}]\} \\
\text{Professoren} &: \{[\underline{\text{PersNr : integer}}, \text{Name : string}, \text{Rang : string}, \text{Raum : integer}]\} \\
\text{Assistenten} &: \{[\underline{\text{PersNr : integer}}, \text{Name : string}, \text{Fachgebiet : string}, \text{Boss : integer}]\} \\
\text{hören} &: \{[\underline{\text{MatrNr : integer}, \text{VorlNr : integer}}]\} \\
\text{voraussetzen} &: \{[\underline{\text{Vorgänger : integer}, \text{Nachfolger : integer}}]\} \\
\text{prüfen} &: \{[\underline{\text{MatrNr : integer}, \text{VorlNr : integer}}, \text{PersNr : integer}, \text{Note : integer}]\}
\end{aligned}
$$

1. Geben Sie alle Vorlesungen an (VorlNr und Titel zurückgeben), die der Student Xenokrates gehört hat. Formulieren Sie die Anfrage in **relationaler Algebra**.
2. Finden Sie alle Studenten (**nur** Name zurückgeben), die in mindestens einer Vorlesung durchgefallen sind (Note > 4.0). Formulieren Sie die Anfrage im **Tupelkalkül**.

Aufgabe 4 (Relationale Entwurfstheorie) [2+2+7+3 PUNKTE]

Betrachten Sie ein abstraktes Relationenschema $\mathcal{R} = \{M, N, V, T, P, PN\}$ mit den FDs

- $M \rightarrow M$
- $M \rightarrow N$
- $V \rightarrow T, P, PN$
- $P \rightarrow PN$

1. Bestimmen Sie alle Kandidatenschlüssel.
2. In welcher Normalform befindet sich die Relation? (Keine Begründung nötig)
3. Bestimmen Sie zu den angegebenen FDs die kanonische Überdeckung.
4. Falls nötig, überführen Sie die Relation verlustfrei und abhängigkeitsbewahrend in die dritte Normalform mithilfe des Synthesealgorithmus.

   Verwenden Sie die folgenden Schritte:

Fortsetzung nächste Seite!

a) Linksreduktion
b) Rechtsreduktion
c) Zusammenfassen von FDs
d) Entfernung von leeren Klauseln auf der rechten Seite

Geben Sie die Zwischenergebnisse nach jedem Schritt an.

Aufgabe 5 (SQL) [2+7+7+7 PUNKTE]

Gegeben ist das folgende Universitätsschema (eine beispielhafte Ausprägung hängt der Klausur an):

$$
\begin{aligned}
\text{Studenten} &: \{[\underline{\text{MatrNr : integer}}, \text{Name : string}, \text{Semester : integer}]\} \\
\text{Vorlesungen} &: \{[\underline{\text{VorlNr : integer}}, \text{Titel : string}, \text{SWS : integer}, \text{gelesenVon : integer}]\} \\
\text{Professoren} &: \{[\underline{\text{PersNr : integer}}, \text{Name : string}, \text{Rang : string}, \text{Raum : integer}]\} \\
\text{Assistenten} &: \{[\underline{\text{PersNr : integer}}, \text{Name : string}, \text{Fachgebiet : string}, \text{Boss : integer}]\} \\
\text{hören} &: \{[\underline{\text{MatrNr : integer}, \text{VorlNr : integer}}]\} \\
\text{voraussetzen} &: \{[\underline{\text{Vorgänger : integer}, \text{Nachfolger : integer}}]\} \\
\text{prüfen} &: \{[\underline{\text{MatrNr : integer}, \text{VorlNr : integer}}, \text{PersNr : integer}, \text{Note : integer}]\}
\end{aligned}
$$

Geben Sie ein SQL-92 Statement an, welches jeweils die gestellte Frage beantwortet.

1. Wie viele Vorlesungen gibt es? (Ausgabe der Anfrage soll die Anzahl der Vorlesungen sein.)
2. Geben Sie die Namen der Professoren an, die Xenokrates aus Vorlesungen kennt.
3. Welche Vorlesungen werden von Studenten im Grundstudium (1.-4. Semester) gehört? Geben Sie die Titel dieser Vorlesungen an.
4. Finden Sie alle Studenten, die drei oder mehr Vorlesungen hören.

Aufgabe 6 (Baum) [16 PUNKTE]

Fügen Sie in einen anfänglich leeren B-Baum mit $k = 2$ die Zahlen 1 bis 20 in aufsteigender Reihenfolge ein. Geben Sie den Zustand des Baums nach jedem Einfügen an, bei dem ein neuer Knoten hinzukommt.

Beispielausprägung

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2
25403	4630	2137	5

Abbildung 1: Beispielausprägung für eine Universitäts-Datenbank

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 [7 PUNKTE]

Bewerten Sie die Korrektheit der folgenden Aussagen und begründen Sie Ihre Antwort kurz.

1. Ein gut entworfenes und implementiertes System braucht kein Testing.
2. Nachdem eine Software ausgeliefert wurde, muss sie nicht mehr angepasst werden und ihr Lebenszyklus ist abgeschlossen.
3. Das Wasserfallmodell ist sinnvoller als das Spiralmodell.

Aufgabe 2 [10 PUNKTE]

1. In welchem der beiden folgenden Entwicklungsprozesse wird der Kunde stärker eingebunden: Im *Wasserfallmodell* oder bei *eXtreme Programming*? Begründen Sie Ihre Antwort.
2. Nennen und beschreiben Sie drei Vorteile einer inkrementellen Produktauslieferung.

Aufgabe 3 [30 PUNKTE]

Zur Modellierung der Mensa ist bereits eine UML-Modellierung vorhanden, die unten gegeben ist. Fertigen Sie dazu eine Implementierung in einer objektorientierten Programmiersprache, beispielsweise Java, an. Beachten Sie bei der Implementierung folgende Vorgaben:

1. Jeder Kunde der Mensa besitzt eine Mensakarte (kurz Karte).
2. Jede Karte besitzt ihren aktuellen Kontostand und es ist möglich, mittels Transaktionen Beträge abzubuchen oder hinzuzufügen. Außerdem kann so auch überprüft werden, ob ein Betrag überhaupt abbuchbar ist.
3. Es wird zwischen Studentenkarten und Mitarbeiterkarten unterschieden, wobei Mitarbeiter das 1,2-Fache der Studentenpreise zahlen.
4. Die Mensa überwacht die Anzahl der ausgegebenen Karten.
5. Eine Speise kann zusätzlich weitere Teilspeisen (z. B. mehrere Beilagen) enthalten. Der Preis einer Speise berechnet sich aus dem Preis der Speise und der Summe der Preise der zusätzlichen Teilspeisen. Hinweis: Eine Beilage kann aber auch als einzelne Speise verkauft werden.

Fortsetzung nächste Seite!

Aufgabe 4

[21 PUNKTE]

Entwerfen Sie ein UML-Klassendiagramm zur folgenden Anforderungsbeschreibung. Verwenden Sie bei der Modellierung des Systems ein passendes Design-Pattern. Nennen Sie das verwendete Design-Pattern.

Es soll ein Sensornetzwerk modelliert werden, in das einzelne Sensoren, aber auch Sensorstationen eingebunden werden können. Für eine Sensorstation soll im System der Name, die Position und die Liste aller zur Station gehörigen Sensoren gespeichert werden, die mittels Get-/Set-Methoden abgerufen und gesetzt werden können. Über die Sensorstation können auch einzelne Sensoren abgerufen oder hinzugefügt werden. Ein Sensor wiederum besitzt einen Namen, seine Position und die Kenntnis, in welcher Station er angebracht ist (mit jeweils dazugehörigen Get-/Set-Methoden).

Für die Speicherung der von den Sensoren erhobenen Daten gibt es zwei Arten von Loggern, die die eingehenden Daten der Sensorstation oder Sensoren in eine Datei bzw. auf die Konsole ausgeben können. Die Logger können sich bei einzelnen Sensorstationen und Sensoren registrieren oder auch diese Registrierung wieder entfernen.

Hinweise:

- Die Position einer Station oder eines Sensors kann als String gespeichert werden.
- Achten Sie bei der Modellierung auf sinnvolle Kapselung, Vererbungsbeziehungen und Relationen zwischen den verschiedenen Klassen.

Fortsetzung nächste Seite!

Aufgabe 5 [11 PUNKTE]

1. Ein Softwaresystem wurde unter der Verwendung des MVC-Patterns erstellt. Zur Vereinfachung heißen die relevanten Komponenten in der Aufgabenstellung: `Model`, `View` und `Controller`. Erstellen Sie ein UML-Sequenzdiagramm, das folgendes Szenario abbildet:
   - Ein Nutzer füllt das Registrierungsformular der Anwendung aus, um sich im System zu registrieren.
   - Diese Eingaben werden direkt im `View` validiert, bevor sie an den `Controller` übergeben werden.
   - Der `Controller` gibt die Informationen an das `Model` weiter, das dafür sorgt, dass ein neues `Nutzerobjekt` erstellt wird, die eingegebenen Daten darin gespeichert werden sowie das Nutzerobjekt in der `Datenbank` gespeichert wird.

   Gehen Sie davon aus, dass alle Aktionen erfolgreich durchgeführt werden können.

Aufgabe 6 [11 PUNKTE]

1. Erläutern Sie die Unterschiede zwischen zentraler und dezentraler Versionsverwaltung und beschreiben Sie beide Arten der Versionsverwaltung.
2. Was versteht man unter Black-Box-Testing und White-Box-Testing? Beschreiben Sie beide Testing-Arten und gehen Sie auf die Unterschiede ein. Berücksichtigen Sie hierbei, dass die Tests ohne Zugabe weiterer Informationen eingesetzt werden.
3. Erklären Sie den Unterschied zwischen Verifikation und Validierung.

Teilaufgabe II: Datenbanken

Aufgabe 1 (ER-Modellierung) [15+15 PUNKTE]

Hinweis: *Bei Wahl dieser Aufgabe wird Wissen über das erweiterte Entity-Relationship-Modell (beispielsweise schwache Entity-Typen, Vererbung) sowie die Verfeinerung eines relationalen Schemas vorausgesetzt.*

Gegeben seien folgende Informationen:

- Eine Messe findet in einem Messezentrum statt. Dabei wird eine Messe über die Kombination aus Jahr und Name identifiziert, während ein Messezentrum über den Ort identifiziert wird und zudem einen Namen hat.
- Ein Messezentrum besteht aus Gebäuden. Diese Gebäude haben eine Nummer, die nur jeweils im Zusammenhang mit dem Messezentrum eindeutig ist. Weiter hat ein Gebäude Räume, die über eine Raumnummer auch nur innerhalb des Gebäudes eindeutig identifizierbar sind. Zudem soll für einen Raum gespeichert werden, wie viele Personen darin Platz finden.
- Räume lassen sich unter anderem in Büros und Ausstellungshallen unterscheiden, wobei ein Raum nicht beides sein kann. Für eine Ausstellungshalle wird zudem ihre Fläche in Quadratmetern gespeichert.

Fortsetzung nächste Seite!

- Angestellte arbeiten für eine Firma und werden über eine Personalnummer identifiziert. Eine Firma hat eine eindeutige FirmenID und zudem eine Adresse.
- Auf einer Messe sind Angestellte eingeteilt. Außerdem kann eine Messe einen Hauptraum haben.
- Eine Firma stellt innerhalb einer Ausstellungshalle für eine Messe aus. Dabei wird auch die Standnummer gespeichert.

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 (existenzabhängige Beziehungen) 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. Die einzelnen Schritte müssen angegeben werden.

Aufgabe 2 (SQL) [5+6+4+7+8+5 PUNKTE]

Gegeben sei der folgende Ausschnitt des Schemas eines Schulverwaltungssystems:

Familie : {[
ID : INTEGER,
Name : VARCHAR(255)
]}

besucht : {[
KindID : INTEGER,
BildungseinrichtungID : INTEGER,
von : DATE,
bis : DATE
]}

Bildungseinrichtung : {[
ID : INTEGER,
Name : VARCHAR(255),
Ort : VARCHAR(255),
Typ: VARCHAR(255)
]}

Kind : {[
ID : INTEGER,
Name : VARCHAR(255),
gehört_zu : INTEGER
]}

Die Tabelle *Kind* enthält Informationen über Kinder, wobei die Spalte *gehört_zu* auf die Familie referenziert. *Bildungseinrichtung* beschreibt verschiedene Einrichtungen anhand ihrer ID, des Namens, des Orts und des Typs der Einrichtung. Die Tabelle *besucht* gibt an, welche Kinder zu welchem Zeitpunkt welche Einrichtung besuchen oder besucht haben. *Familie* enthält Informationen über die Familien bezüglich deren ID und Namen.

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

Fortsetzung nächste Seite!

1. Schreiben Sie eine SQL-Anweisung, welche alle Einträge in der Tabelle *Bildungseinrichtung* löscht, die von keinem Kind besucht werden (weder aktuell noch früher).

2. Schreiben Sie eine SQL-Anweisung, welche die Namen aller Familien bestimmt, bei denen mindestens ein Kind eine Bildungseinrichtung vom Typ 'Gymnasium' besucht hat oder besucht.

3. Schreiben Sie die SQL-Anweisung aus Teilaufgabe b) um, sodass nun die Namen aller Familien bestimmt werden, bei denen mindestens zwei Kinder eine Bildungseinrichtung vom Typ 'Gymnasium' besucht haben oder besuchen.

4. Schreiben Sie eine SQL-Anweisung, welche die IDs und Namen aller Kinder bestimmt, die vor dem Besuch eines Gymnasiums auch eine Mittelschule besucht haben.

5. Bestimmen Sie für die folgenden SQL-Anweisungen die **minimale** und **maximale** Anzahl an Tupeln im Ergebnis. Beziehen Sie sich dabei auf die Größe der einzelnen Tabellen.

   Verwenden Sie für die Lösung folgende Notation:
   — Familie — beschreibt die Größe der Tabelle *Familie*

   a)

   ```
   SELECT *
   FROM Kind
   EXCEPT
   SELECT k.*
   FROM Kind k
   LEFT OUTER JOIN besucht b ON k.id = b.kindid
   ```

   b)

   ```
   SELECT bi.ort
   FROM Bildungseinrichtung bi, besucht b, Kind k
   GROUP BY bi.ort
   ```

6. Beschreiben Sie den Effekt der folgenden SQL-Query in natürlicher Sprache:

   ```
   SELECT k.gehört_zu, COUNT(DISTINCT bi.id)
   FROM besucht b, Kind k, Bildungseinrichtung bi
   WHERE k.id = b.kindid
   AND b.bildungseinrichtungid = bi.id
   GROUP BY k.gehört_zu
   ```

Aufgabe 3 (Entwurfstheorie) [6+12+7 PUNKTE]

Gegeben sei folgendes relationales Schema R in erster Normalform:

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

und die Zerlegung $\rho = \{R_1, R_2\}$ von R mit $R_1 = \{A, C, E\}$ und $R_2 = \{B, C, D, E\}$.
Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$\begin{aligned} ACD &\rightarrow B, \\ CE &\rightarrow AD, \\ CD &\rightarrow BE, \\ ABE &\rightarrow E \end{aligned}$$

$\}$

1. Bestimmen Sie alle Kandidatenschlüssel/Schlüsselkandidaten von R mit FD.
   Begründen Sie Ihre Antwort. Begründen Sie zudem, warum es keine weiteren Kandidatenschlüssel/Schlüsselkandidaten gibt.
   Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.
2. Prüfen Sie, ob R mit FD in
   - BCNF,
   - 3NF oder
   - 2NF ist.
3. Zeigen oder widerlegen Sie, dass die Zerlegung ρ von R verlustfrei bzgl. FD ist.

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Herbst 2020

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	Herbst 2020	46116
Arbeitsplatz-Nr.: _____		

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: 23

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 (Entwicklungsprozesse) [28 PUNKTE]

a) Nennen Sie drei Phasen, die im V-Modell mit steigendem Detailgrad durchlaufen werden sowie das Phasenergebnis und den jeweiligen Test des Phasenergebnisses.

b) Nennen Sie zwei Vorteile und zwei Nachteile der agilen Softwareentwicklung gegenüber dem V-Modell, jeweils mit Begründung.

c) Bestimmen Sie zu folgenden (voneinander unabhängigen) Wartungsvorgängen jeweils die genaue Art der Wartung (adaptiv, korrektiv, perfektionierend) und begründen Sie kurz Ihre Entscheidung:

   i. Das Softwaresystem einer Firma muss nach einem Hardware-Upgrade angepasst werden.

   ii. Ein Softwaresystem zur Berechnung des Firmenumsatzes wird auf ein neues Steuergesetz umgestellt.

   iii. Ein Studentenprojekt soll in ein Startup überführt werden. Dazu wird zwar ein Großteil der Software übernommen, aber mehrere Subroutinen enthalten schwer lesbaren Code und werden daher neu geschrieben.

   iv. Ein Studentenprojekt soll in ein Startup überführt werden. Dazu wird zwar ein Großteil der Software übernommen, aber mehrere Subroutinen werden auf Grund von Patentfragen neu entwickelt.

   v. Ein Chip-Hersteller liefert auf Grund eines fehlerhaften Chip-Designs neue Steuersoftware an alle Kunden.

d) Eine Webseite besteht typischerweise aus drei Komponenten: HTML legt den Aufbau fest, CSS bestimmt das Erscheinungsbild und JavaScript bietet Interaktionsmöglichkeiten. Erklären Sie anhand dieses Beispiels, warum „Separation of Concerns“ eine wichtige Rolle in der Softwareentwicklung einnimmt.

e) Begründen Sie anhand zweier Beispiele, warum die Verwendung einer höheren Programmiersprache einen positiven Einfluss auf die Entwicklung eines größeren Softwaresystems hat!

f) Zeigen Sie anhand zweier Beispiele von Software-Maßen, dass diese ungeeignet zum Messen des Projektfortschritts sind.

Fortsetzung nächste Seite!

Aufgabe 2 (Projektmanagement) [13 PUNKTE]

Betrachten Sie die folgende Tabelle zum Projektmanagement:

Arbeitspaket	**Dauer (Tage)**	**abhängig von**
A1	5	
A2	5	A1
A3	15	A2
A4	30	A1
A5	5	A1, A3
A6	15	
A7	15	A2, A3
A8	15	A4, A5, A6
A9	15	A6
A10	10	A7

a) Erstellen Sie einen Netzplan (CPM-Diagramm) nach der Methode des kritischen Pfades, der die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.
Das Diagramm muss nicht maßstabsgetreu sein, muss aber jede Information aus der gegebenen Tabelle enthalten.

b) Bestimmen Sie die Gesamtdauer des Projektes und geben Sie alle kritischen Pfade an.

c) Bestimmen Sie die Gesamtdauer des Projektes und geben Sie alle kritischen Pfade an, wenn das Arbeitspaket A4 nur halb so lange dauert.

d) Bestimmen Sie die Gesamtdauer des Projektes und geben Sie alle kritischen Pfade an, wenn das Arbeitspaket A5 sieben Tage mehr Zeit benötigt.

e) Beschreiben Sie kurz eine allgemeine Strategie, die während der Projektplanung eingesetzt wird, um Verzögerungen des Projektes zu vermeiden.

Fortsetzung nächste Seite!

Aufgabe 3 (Use-Case-Diagramm) [18 PUNKTE]

Im Folgenden ist eine Systembeschreibung für eine „Schul-Software“ angegeben.

> Für eine Schule soll eine neue Verwaltungssoftware entwickelt werden und Sie sind daran beteiligt. Zur Schulfamilie gehören allgemein mehrere Personengruppen: Lehrer, das Direktorat, der Hausmeister, natürlich Schüler und deren Eltern. Damit alle Beteiligten entsprechend Zugriff haben, soll das System vollständig über den Webbrowser benutzbar sein. Grundsätzlich sind die Hilfe und der aktuellen Speiseplan der Kantine öffentlich verfügbar, um jedem einen schnellen Überblick zu geben. Die Mitglieder der Schulfamilie können sich am System anmelden und dann weitere Aktionen durchführen. Schüler (und deren Eltern) sollen Zugang zum jeweiligen Stundenplan haben. Lehrer können sowohl ihren eigenen als auch die klassenspezifischen Stundenpläne aufrufen. Lehrer können Noten eintragen, welche von den Schülern abgerufen werden können. Es soll die Möglichkeit geben, dass sich beliebige Teilnehmer Nachrichten schicken. Lehrer und das Direktorat können zusätzlich Elternbriefe verschicken, die an eine größere Anzahl an Schülern (bzw. deren Eltern) gerichtet sind. Meldungen über Schäden am Inventar können von den Lehrern an den Hausmeister gemeldet werden. Dieser kann sich die Liste aller Schäden anzeigen lassen. Sollte ein Schaden bereits gemeldet worden sein, wird dem Lehrer ein entsprechender Hinweis angezeigt. Das Direktorat hat die Möglichkeit, den Stundenplan zu erstellen und Zeugnisse drucken zu lassen.

a) Geben Sie die im Text erwähnten Akteure für das beschriebene System an.

b) Identifizieren Sie zwei weitere Stakeholder und nennen Sie dazu je einen unterschiedlichen Anwendungsfall des Systems, in den diese involviert sind.

c) Geben Sie sechs verschiedene Anwendungsfälle für das beschriebene System an.

d) Erstellen Sie aus Ihren vorherigen Antworten ein Use-Case-Diagramm für das beschriebene System, in dem die Akteure und Anwendungsfälle inkl. möglicher Generalisierungen und Beziehungen eingetragen sind. Achten Sie insbesondere auf mögliche ⟨⟨include⟩⟩- und ⟨⟨extends⟩⟩- Beziehungen und Bedingungen für Anwendungsfälle.

Fortsetzung nächste Seite!

Aufgabe 4 (Objektorientierte Analyse) [14 PUNKTE]

Betrachten Sie das folgende Szenario:

> Es ist Freitagabend und Sie überlegen sich, zur Studentenparty im Freundeskreis zu gehen. Nachdem Sie Ihre **Jacke angezogen** haben, **gehen Sie zur Party**. Dort angekommen, wird Ihnen auch schnell klar, warum Sie nicht zu Hause geblieben sind. Es gibt lustige Trinkspiele. Natürlich alles nur mit Mineralwasser, da Alkohol ein echter Stimmungskiller ist. Das Spiel geht so:
>
> - Als Erstes wird per Zufall ein langer oder kurzer **Strohhalm gezogen**. Sofern der kurze Strohhalm gezogen wurde, muss nun so lange mit drei **Würfeln geworfen** werden, bis die Summe der Augen nicht größer als 15 ist.
> - Ist die Summe nicht größer als 15, so wird ein Becher leckeres **Mineralwasser getrunken** und die Runde ist beendet.
> - Wurde ein langer Strohhalm gezogen, so muss die Aufgabe erfüllt werden, gleichzeitig eine vorgegebene **Strecke zu laufen** und dabei einen Becher mit **Mineralwasser zu trinken**. Bevor dies ausgeführt werden kann, muss jedoch dem Veranstalter für die gute Feier ein **Dank ausgesprochen** werden, welcher dies durch ein lautes Lachen bestätigt.
> - Ist nun die eine oder andere Runde beendet, so muss der Teilnehmer noch ein **Lied singen**. Eine Jury **bewertet das Lied** als gut oder schlecht. Gute Lieder führen zum Sieg und damit zum Ende des Spiels. Schlechte Lieder müssen wiederholt und erneut bewertet werden.

a) Erstellen Sie ein UML-Aktivitätsdiagramm für den gegebenen Sachverhalt. Achten Sie dabei auf eine vernünftige Benennung der Aktivitäten und Bedingungen.

b) Sowohl das UML-Zustandsdiagramm wie auch UML-Aktivitätsdiagramm beschreiben Verhaltensaspekte eines Systems. Erklären Sie kurz, wie diese beiden Sichten zusammenhängen.

Fortsetzung nächste Seite!

Aufgabe 5 (UML und Entwurfsmuster) [17 PUNKTE]

Betrachten Sie das folgende Szenario:

Ein *Sandwich* besteht in seiner einfachsten Form lediglich aus einer *Brotscheibe*. Es kann aber auch ein Brot mit Belag (ein sog. *BelegtesBrot*) sein; der Einfachheit halber gehen wir in unserer zu modellierenden Welt davon aus, dass es nur mit Schinken, mit einer Scheibe Gurke und/oder Käse belegte Sandwiches gibt. Die entsprechenden Gerichte heißen dann *Schinkenbrot*, *Gurkenbrot* bzw. *Käsebrot*. Man kann allerdings beliebig viele Beläge kombinieren, also auch mehrfach verwenden. Der oberste Belag bestimmt, ob es sich um ein Schinkenbrot, ein Gurkenbrot oder ein Käsebrot handelt. Mit anderen Worten: Durch Auflegen einer Scheibe Käse auf ein Schinkenbrot wird aus einem vorhandenen Sandwich ein Käsebrot. Für den Verkauf kann man von jedem Sandwich mit *lieferePreis()* den Preis ermitteln (der sich aus der Summe der Preise für die Brotscheibe und die Beläge zusammensetzt).

a) Welches Design-Pattern bietet sich an, um den beschriebenen Sachverhalt zu modellieren?

b) Geben Sie ein UML-Klassendiagramm an, das den beschriebenen Sachverhalt modelliert. Achten Sie auf Multiplizitäten bei Komposition und Aggregation.

c) Gegeben sei ein Sandwich-Objekt *sw*, das aus einer Scheibe Brot, zwei Scheiben Schinken, einer Scheibe Käse und obendrauf einer Scheibe Gurke besteht (in dieser Reihenfolge!). Geben Sie ein UML-Objektdiagramm an, das diese Situation darstellt.

d) Begründen Sie, ob das Objektdiagramm für den gegebenen Term basierend auf Ihrem Klassendiagramm eindeutig definiert ist.

e) Jeder einzelne Bestandteil eines Sandwiches holt sich bei Bedarf den aktuell gültigen Preis aus einer zentralen Datenbank-Instanz *db*. Die Datenbank soll folgenden Datensatz enthalten: {*Brotscheibe: 2€, Scheibe Schinken: 0,50€, Scheibe Käse: 0,40€, Scheibe Gurke: 0,10€*}. Erstellen Sie ein vollständig beschriftetes Sequenzdiagramm für die (erfolgreiche) Preisberechnung des genannten Sandwiches *sw* (aus Teilaufgabe c) und geben Sie den Gesamtpreis Ihrer Berechnung an.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanksysteme

Aufgabe 1 (Entwurfstheorie: Entity-Relationship-Modellierung) [26 PUNKTE]

Gegeben sind zwei Entity-Relationship-Diagramme, siehe Abbildungen 1 und 2.

Abbildung 1: Firma

Abbildung 2: Berufsgenossenschaft

a) Beschreiben Sie den Zusammenhang der Entitäts- und Relationshiptypen in Abbildung 1.

b) Beschreiben Sie den Zusammenhang der Entitäts- und Relationshiptypen in Abbildung 2.

c) Untersuchen Sie die beiden Abbildungen auf redundante Relationships. Erläutern Sie, ob und ggf. wo es Redundanz gibt und wie diese aufgelöst werden kann.

d) Überführen Sie das ER-Modell in Abbildung 2 in ein verfeinertes relationales Schema. Wählen Sie geeignete Attribute (mind. zwei pro Entitätstyp) und kennzeichnen Sie Schlüssel durch Unterstreichen. Wählen Sie außerdem für jedes Attribut einen geeigneten Datentyp.

Fortsetzung nächste Seite!

Aufgabe 2 (SQL) [30 PUNKTE]

Gegeben ist das folgende Relationenschema zur Verwaltung der Fußball-Bundesliga:

Bundesliga: {[Mannschaft , Spieldatum, Heimspiel, Heimtore, Gasttore]}

Ferner existiert der folgende Ausschnitt einer Relationeninstanz:

Bundesliga	Mannschaft	Spieldatum	Heimspiel	Heimtore	Gasttore
	FC Augsburg	17.08.2019	0	5	1
	Bayern München	24.08.2019	0	0	3
	Freiburg	31.08.2019	1	1	2
	Werder Bremen	01.09.2019	1	3	2
	Bayern München	14.09.2019	0	1	1
	Freiburg	21.09.2019	1	1	1
	FC Augsburg	28.09.2019	1	0	3
	Werder Bremen	06.10.2019	0	2	2
	...	...	...	...	...

Das Attribut *Heimspiel* hat den Wert 1, falls es sich um ein Heimspiel der jeweiligen Mannschaft handelt, ansonsten 0. *Heimtore* beschreibt die Anzahl erzielter Tore der Heimmannschaft, *Gasttore* die entsprechenden Tore der Gastmannschaft.

Bearbeiten Sie die folgenden Teilaufgaben:

a) Erläutern Sie kurz, warum das Attribut *Spieldatum* Teil des Primärschlüssels ist.

b) Schreiben Sie eine SQL-Anweisung, welche die Tabelle *Bundesliga* anlegt. Verwenden Sie geeignete Datentypen (kein Boolean). Stellen Sie über Constraints sicher, dass das Attribut *Heimspiel* nur die Werte 0 bzw. 1 annehmen kann und dass weder *Heimtore* noch *Gasttore* negativ sein können. Sie brauchen keine Daten "einfügen".

c) Erstellen Sie mittels SQL eine View *Auswaertserfolge*, die alle Einträge von Auswärtsspielen enthält, die mit einem Sieg oder einem Unentschieden für die Gastmannschaft enden.

d) Schreiben Sie eine SQL-Anweisung zum Ändern des Ergebnisses von *Werder Bremen* vom *06.10.2019* in der View *Auswaertserfolge* aus Aufgabe c):
Die Gastmannschaft hat 2:0 verloren.

Fortsetzung nächste Seite!

e) Schreiben Sie eine SQL-Anweisung, welche für jede Mannschaft die Zeit in Tagen, die zwischen dem ersten und dem letzten Spiel liegt, ermittelt.

f) Ermitteln Sie mit einer SQL-Anweisung die *Mannschaft*, *Heimtore* und das *Spieldatum* für solche Heimspiele, an denen der Gastgeber weniger Tore als der Mittelwert aller Heimtore der Mannschaften erzielt hat. Sortieren Sie das Ergebnis bezüglich der Tore absteigend.

Aufgabe 3 (Entwurfstheorie) [14 PUNKTE]

Bearbeiten Sie die folgenden Teilaufgaben:

a) Gegeben sei das folgende Relationenschema **R**:{[A, B, C, D]} (in 1. Normalform) mit folgender Ausprägung:

R	A	B	C	D
	a1	b1	c1	d1
	a1	b1	c2	d1
	a2	b2	c1	d2
	a2	b2	c2	d2
	a3	b2	c2	d3
	a4	b1	c2	d4
	a4	b1	c4	d4
	a4	b1	c5	d4

Welche der folgenden funktionalen Abhängigkeiten sind in obiger Ausprägung der Relation erfüllt? Geben Sie jeweils eine kurze Begründung an.

i. $A \rightarrow B$
ii. $C \rightarrow B$
iii. $AC \rightarrow D$
iv. $A \rightarrow D$
v. $D \rightarrow C$
vi. $BC \rightarrow A$

Fortsetzung nächste Seite!

b) Eine Relation **S** (A, B, C, D, E, F) (in 1. Normalform) habe folgende funktionalen Abhängigkeiten:

$FD = \{$

$$\begin{aligned} BC &\rightarrow C \\ C &\rightarrow AD \\ D &\rightarrow CE \\ E &\rightarrow BC \\ F &\rightarrow D \end{aligned}$$

$\}$

i. Bestimmen Sie alle Kandidatenschlüssel von **S** mit FD. Begründen Sie Ihre Antwort. Begründen Sie zudem, warum es keine weiteren Kandidatenschlüssel gibt.
Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Punktabzügen.

ii. Prüfen Sie, ob **S** mit FD in 2. Normalform ist. Begründen Sie Ihre Antwort.

iii. Prüfen Sie, ob **S** mit FD in 3. Normalform ist. Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

Aufgabe 4 (Transaktionen) [20 PUNKTE]

Bearbeiten Sie die folgenden Teilaufgaben:

a) Beschreiben Sie das *Lost Update* Problem. Geben Sie hierfür ein geeignetes Beispiel in Form von zwei nebenläufigen Transaktionen an, welche bei unkontrolliertem Mehrbenutzerbetrieb zu einem Lost Update führen. Welche ACID-Eigenschaft wird im Lost Update Problem verletzt?

b) Erklären Sie den Unterschied zwischen *Livelock* und *Deadlock* im Zusammenhang mit sperrbasierten Synchronisationsmethoden.

c) Gegeben sind vier Transaktionen T_1, T_2, T_3 und T_4. Jede Transaktion T_i, $i \in \{1, 2, 3, 4\}$, möchte wie folgt ein Objekt X lesen und anschließend schreiben.

$$
\begin{array}{lcllllll}
T_1 & : & l_1(A); & r_1(A); & l_1(B); & w_1(B); & u_1(A); & u_1(B); \\
T_2 & : & l_2(C); & r_2(C); & l_2(A); & w_2(A); & u_2(C); & u_2(A); \\
T_3 & : & l_3(B); & r_3(B); & l_3(C); & w_3(C); & u_3(B); & u_3(C); \\
T_4 & : & l_4(D); & r_4(D); & l_4(A); & w_4(A); & u_4(D); & u_4(A);
\end{array}
$$

Dabei gilt:

- $l_i(X)$ ist ein Lock auf das Element X
- $r_i(X)$ möchte X lesen
- $w_i(X)$ möchte X schreiben
- $u_i(X)$ ist ein Unlock von X

Fortsetzung nächste Seite!

Nachstehend ist der Beginn eines Schedules für obige Transaktionen gegeben.

	T_1	T_2	T_3	T_4
1.	$l_1(A); r_1(A)$			
2.		$l_2(C); r_2(C)$		
3.			$l_3(B); r_3(B)$	
4.				$l_4(D); r_4(D)$
5.		$l_2(A);$		
6.			$l_3(C);$	
7.				$l_4(A);$
8.	$l_1(B);$			

Zeichnen Sie den dazugehörigen Wartegraphen zur Erkennung eines Deadlocks und erläutern Sie anhand des Graphs, ob eine Verklemmung vorhanden ist oder nicht.

Definition Wartegraph: Die Knoten eines Wartegraphen entsprechen den Kennungen der Transaktionen. Wann immer eine Transaktion T_i auf die Freigabe einer Sperre durch eine Transaktion T_j wartet, wird die Kante $T_i \rightarrow T_j$ eingefügt. Die Kanten sind gerichtet. Entsteht ein Zyklus, liegt eine Verklemmung vor.

d) Falls ein Deadlock vorhanden ist, wie kann dieser aus Sicht des Transaktionsmanagers aufgelöst werden?

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 (Projektplanung) [15 PUNKTE]

Gegeben sei folgendes CPM-Netzwerk ("Critical-Path-Method"):

a) Geben Sie zunächst an, was die Knoten und Kanten in dieser Diagrammart hinsichtlich Aktivitäten im Projekt bedeuten.

b) Welche Pseudoaktivität(en) kann man ohne Informationsverlust aus dem Netzwerk entfernen? Geben Sie auch den Grund dafür an!

c) Welche zwei Knoten liegen – neben Start- und Endknoten – auf jeden Fall auf dem kritischen Pfad (ohne Berechnung der Zeiten)? (Keine weitere Begründung erforderlich!)

d) Geben Sie für jedes Ereignis den frühesten Zeitpunkt unter der Annahme an, dass das Projekt zum Zeitpunkt 0 beginnt.

e) Berechnen Sie für jedes Ereignis den spätesten Zeitpunkt. Setzen Sie für das letzte Ereignis den spätesten Zeitpunkt gleich dem frühesten Zeitpunkt und berechnen Sie die übrigen Termine rückwärts.

f) Geben Sie alle kritischen Pfade durch das Netzwerk an.

g) Was heißt es für das Projektende, wenn sich ein Ereignis auf dem kritischen Pfad um 3 Zeiteinheiten verspätet?

Fortsetzung nächste Seite!

Aufgabe 2 (Entwurfsmuster) [30 PUNKTE]

Ihnen sei ein **UML Klassendiagramm** zu folgendem Szenario gegeben. Kunden können einen Newsfeed abonnieren bzw. ein Abo kündigen. Dementsprechend werden sie in dem Newsfeed vermerkt bzw. aus der Kundenliste wieder entfernt. Sobald eine neue Nachricht im Newsfeed erstellt wird, werden alle Abonnenten benachrichtigt.

a) Im angegebenen Klassendiagramm werden **zwei** unterschiedliche Entwurfsmuster verwendet. Um welche Muster handelt es sich? Geben Sie jeweils den Namen des Musters sowie die Elemente des Klassendiagramms an, mit denen diese Muster im Zusammenhang stehen. ACHTUNG: Es handelt sich dabei **nicht** um das *Interface*- oder das Vererbungsmuster.

b) Nennen Sie zwei generelle Vorteile von Entwurfsmustern.

c) Geben Sie eine Implementierung der Klasse `NewsFeed` an. Die Methode `nachrichtErstellen()` soll lediglich die Methode `kundenBenachrichtigen()` mit der entsprechenden Nachricht aufrufen.

d) Geben Sie eine Implementierung der Klasse `Kunde` an. Die Methode `benachrichtigen()` soll den Kundennamen sowie die Nachricht auf der Konsole ausgeben.

Hinweis: Die Implementierungen **müssen** sowohl dem Klassendiagramm, als auch den Konzepten der verwendeten Muster entsprechen. Verwenden Sie eine objektorientierte Programmiersprache, vorzugsweise Java. Sie müssen sich an der Testmethode und ihrer Ausgabe auf der nächsten Seite orientieren. Die Testmethode muss mit Ihrer Implementierung ausführbar sein und sich semantisch korrekt verhalten.

Fortsetzung nächste Seite!

Quelltext der Testmethode:

```
1 public static void main(String[] args) {
2     IKunde alice = new Kunde("Alice");
3     IKunde bob = new Kunde("Bob");
4     alice.newsFeedAbonnieren(NewsFeed.gibInstanz());
5     NewsFeed.gibInstanz().nachrichtErstellen("Bayern hat gewonnen.");
6     bob.newsFeedAbonnieren(NewsFeed.gibInstanz());
7     NewsFeed.gibInstanz().nachrichtErstellen("Spritpreise sinken.");
8     alice.aboKuendigen(NewsFeed.gibInstanz());
9     NewsFeed.gibInstanz().nachrichtErstellen("Wahlergebnisse stehen fest.");
10 }
```

Konsolenausgabe:

```
Nachricht an Alice: Bayern hat gewonnen.
Nachricht an Alice: Spritpreise sinken.
Nachricht an Bob: Spritpreise sinken.
Nachricht an Bob: Wahlergebnisse stehen fest.
```

Fortsetzung nächste Seite!

Aufgabe 3 (White-Box-Testverfahren) [30 PUNKTE]

Die *(einfache) Quersumme* einer natürlichen Zahl ist die Summe aller ihrer Ziffern in ihrer Repräsentation als Dezimalzahl. Die Quersumme der Dezimalzahl 4242 ist beispielsweise $2 + 4 + 2 + 4 = 12$. Bei der *alternierenden Quersumme* werden die einzelnen Ziffern abwechselnd addiert und subtrahiert (beginnend mit der rechtesten Ziffer). Die alternierende Quersumme der Dezimalzahl 4242 ist beispielsweise $2 - 4 + 2 - 4 = -4$.

Folgende **Java-Methode** berechnet für eine übergebene Zahl die (alternierende) Quersumme. Eine Eingabe wird durch ein Paar `(a,b)` beschrieben; dabei beschreibt `a` die Zahl und `b` den Wahrheitswert, ob die alternierende Quersumme berechnet werden soll. Für die Eingabe `(42,true)` wird beispielsweise die alternierende Quersumme der Dezimalzahl 42 berechnet, für die Eingabe `(99,false)` die (einfache) Quersumme der Dezimalzahl 99. Falls `a` keine natürliche Zahl darstellt, wird `0` zurückgegeben.

```
1  public static int berechneQuersumme(int zahl, boolean alternierend) {
2      int resultat = 0;
3      if (zahl >= 0) {
4          if (zahl < 10) {
5              resultat = zahl;
6          } else {
7              int faktor = 1;
8              do {
9                  int ziffer = zahl % 10;
10                 zahl = zahl / 10;
11                 resultat = resultat + faktor * ziffer;
12                 if (alternierend) {
13                     if (faktor == 1) {
14                         faktor = -1;
15                     } else {
16                         faktor = 1;
17                     }
18                 }
19             } while (zahl > 0);
20         }
21     }
22     return resultat;
23 }
```

Beachten Sie bei der Bearbeitung der Aufgabe die Hinweise auf der nächsten Seite!

a) Geben Sie für die Methode einen **Kontrollflussgraphen** an, wobei Sie die Knoten mit den jeweiligen Zeilennummern im Quelltext beschriften.

b) Geben Sie eine **minimale Testmenge** an, die das Kriterium der *Anweisungsüberdeckung* (*Knotenüberdeckung*) erfüllt. Geben Sie für jeden Testfall den Kontrollfluss als Pfad in der Notation $1 \rightarrow 2 \rightarrow \ldots$ an.

c) Geben Sie eine **minimale Testmenge** an, die das Kriterium der *Zweigüberdeckung* (*Kantenüberdeckung*) erfüllt. Geben Sie für jeden Testfall den Kontrollfluss als Pfad in der Notation $1 \rightarrow 2 \rightarrow \ldots$ an; falls ein Testfall bereits in Teilaufgabe b) genannt und dessen Pfad beschrieben wurde, genügt es, bei dieser Teilaufgabe nur den Testfall ohne Pfad anzugeben.

d) Beschreiben Sie den konzeptionellen **Unterschied** zwischen *White-Box-Tests* (hier eingesetzt) und *Black-Box-Tests*.

Fortsetzung nächste Seite!

Hinweise:

- Diese Aufgabe betrachtet als *natürliche Zahlen* die Menge $\mathbb{N} = \{0, 1, \dots\}$ (also inklusive der Zahl 0).
- Der Operator % (vgl. Quelltext-Zeile 9) notiert den mathematischen Modulo-Operator, welcher für $a \bmod b$ den Rest der ganzzahligen Division von a durch b berechnet, z. B. $42 \bmod 10 = 2$.
- Eine *Testmenge* ist *minimal*, wenn es keine Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität muss nicht bewiesen werden.

Aufgabe 4 (Lebenszyklus) [15 PUNKTE]

a) Beschreiben Sie den Entwicklungszyklus bei testgetriebener Entwicklung.

b) Nennen Sie die Phasen des RUP und geben Sie stichpunktartig ein jeweiliges Ziel an.

c) Beschreiben Sie die Grundstuktur des V-Modells.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanksysteme

Aufgabe 1 (Modellierung) [19 PUNKTE]

Betrachten Sie folgenden Ausschnitt aus der Prüfungsverwaltung einer Hochschule:

Prüfungen sind durch eine eindeutige Prüfungsnummer (PrüfNr) gekennzeichnet. Sie erstrecken sich über ein bestimmtes Fach und finden an einem bestimmten Datum statt.

Dozenten haben eine eindeutige Personalnummer (PersNr) und einen Namen. Sie können Erstprüfer für beliebig viele Prüfungen sein, aber jede Prüfung hat genau einen Erstprüfer.

Räume werden durch ein Gebäudekürzel (GebK) und eine Raumnummer (RaumNr) eindeutig beschrieben. Außerdem wird ihre Größe erfasst.

In den Räumen befinden sich *Tische*. Diese werden durch eine Tischnummer (TischNr) gekennzeichnet, die innerhalb des jeweiligen Raumes eindeutig ist.

Studenten haben eine eindeutige Matrikelnummer (MatrNr) und einen Namen. Sie können an beliebig vielen Prüfungen teilnehmen; dabei wird ihnen jeweils ein Tisch zugewiesen, und sie erhalten anschließend eine Note. An jeder Prüfung können beliebig viele Studenten teilnehmen, aber wenn kein Student teilnimmt, wird die Prüfung abgesagt.

a) Modellieren Sie den beschriebenen Sachverhalt als Entity-Relationship-Diagramm. Geben Sie für jeden Gegenstandstyp sämtliche Attribute an und kennzeichnen Sie jeweils einen Primärschlüssel. Charakterisieren Sie die Beziehungstypen mit Funktionalitäten und der (min, max)-Notation entsprechend der Beschreibung.

   Übernehmen Sie hierfür folgendes Grundgerüst in Ihre Lösung:

Nicht in die Angabe schreiben!

Fortsetzung nächste Seite!

b) Übersetzen Sie das Entity-Relationship-Diagramm in ein relationales Schema. Fassen Sie dabei Relationen zusammen, soweit dies möglich ist. Typen für die Attribute müssen nicht angegeben werden. Kennzeichnen Sie aber in jeder Relation den Primärschlüssel.

c) Wäre es sinnvoll, zwischen *Prüfungen* und *Räume* noch eine Beziehung einzufügen, die angibt, in welchen Räumen welche Prüfungen stattfinden? Begründen Sie Ihre Antwort.

Aufgabe 2 (Entwurfstheorie) [18 PUNKTE]

Gegeben sei folgendes Relationenschema, das Informationen über Vorlesungen speichert. Beispielsweise liest Professor Codd, der die Personalnummer 2138, den Rang C4 und sein Büro im Raum K123 hat, die Vorlesung Datenbanken mit der Vorlesungsnummer 5432 am Dienstag um 08:15 Uhr im Hörsaal K014 und am Donnerstag um 10:00 Uhr im Hörsaal K003.

Vorlesungsverzeichnis								
PersNr	Name	Rang	Büro	VorlNr	VorlTitel	VorlTag	VorlZeit	Hörsaal
2138	Codd	C4	K123	5432	Datenbanken	Di	08:15	K014
2138	Codd	C4	K123	5432	Datenbanken	Do	10:00	K003
...	...	...	...	...	...	...	...	...

Dabei hat jeder Professor eine eindeutige Personalnummer, einen festgelegten Rang (Besoldungsgruppe) und ein Büro für sich alleine. Jede Vorlesung hat eine eindeutige Vorlesungsnummer, einen Titel (mehrere verschiedene Vorlesungen können den selben Titel tragen) und einen festen Professor, der sie liest. Eine Vorlesung kann an mehreren Terminen, aber nicht mehrfach an einem Tag, und in unterschiedlichen Hörsälen stattfinden. Und natürlich kann zu einem bestimmten Termin in einem Hörsaal nur eine Vorlesung stattfinden.

a) Geben Sie die geltenden funktionalen Abhängigkeiten in kanonischer Überdeckung an.

b) Bestimmen Sie den oder die Kandidatenschlüssel (ohne Begründung).

c) Ist das Schema in zweiter Normalform? Geben Sie eine kurze Begründung an.

Fortsetzung nächste Seite!

d) Überführen Sie das Schema mit Hilfe des Synthesealgorithmus in die dritte Normalform. Benennen Sie dabei jeden Schritt des Algorithmus und geben Sie das Zwischenergebnis nach jedem Schritt an. Kennzeichnen Sie im Endergebnis in jeder Relation einen Primärschlüssel.

e) Gegeben sei nun ein abstraktes Relationenschema $\mathcal{R} = \{A, B, C, D, E, F\}$ mit den funktionalen Abhängigkeiten

$$S = \{A \rightarrow DF,\ BF \rightarrow CDE,\ C \rightarrow ADF,\ E \rightarrow BF,\ F \rightarrow AB\}.$$

Berechnen Sie die kanonische Überdeckung von S. Benennen Sie dabei jeden Schritt und geben Sie das Zwischenergebnis nach jedem Schritt an.

Aufgabe 3 (SQL-Anfragen) [25 PUNKTE]

Die folgenden Fragen beziehen sich auf das Universitätsschema, welches Sie in einer Beispielausprägung auf Seite 23 finden. Die Anfragen sind in SQL zu formulieren – und zwar so, dass sie unabhängig von der Ausprägung der Datenbank korrekt ausgewertet werden. Die Ergebnisse sind ohne Duplikate zu berechnen, aber unnötige Duplikatelimination ist zu vermeiden.

a) Welche Studenten haben die Vorlesung „Ethik“ prüfen lassen? Geben Sie die Matrikelnummern und Namen dieser Studenten aus.

MatrNr	Name
25403	Jonas

b) Wie viele Studenten haben die Vorlesung „Ethik“ prüfen lassen?

count
1

c) Was sind die direkten Voraussetzungen ("Vorgänger") der Vorlesung „Ethik“? Geben Sie Vorlesungsnummer und Titel dieser Vorlesung(en) aus.

VorlNr	Titel
5001	Grundzüge

Fortsetzung nächste Seite!

d) In welchen Fächern ist die Durchschnittsnote schlechter als 1?
Geben Sie die Vorlesungsnummer und den Titel der entsprechenden Vorlesungen aus.

VorlNr	Titel
4630	Die 3 Kritiken
5041	Ethik

e) Welche Professoren lesen mehr als eine Vorlesung?

PersNr	Name
2137	Kant
2126	Russel
2125	Sokrates

f) Welche Studenten haben (noch) **keine** Prüfungen abgelegt?
Geben Sie die Matrikelnummern und Namen dieser Studenten aus.

MatrNr	Name
24002	Xenokrates
26120	Fichte
26830	Aristoxenos
29120	Theophrastos
29555	Feuerbach

g) Welche Studenten hören **alle** Vorlesungen? Also für welche Studenten existiert keine Vorlesung zu der kein passender hören-Eintrag existiert.

MatrNr	Name

(Leeres Ergebnis)

Fortsetzung nächste Seite!

h) Der Student *Fichte* (MatrNr 26120) hat die Prüfung in *Glaube und Wissen* (VorlNr 5022) mit der Note 1 erfolgreich abgelegt. Prüfer war der Assistent *Spinoza* (PersNr 3007). Geben Sie ein entsprechendes SQL-Statement an, um die Prüfung in die Uni-Datenbank einzufügen. Sie können die gegebenen MatrNr, VorlNr, und PersNr direkt in ihr Insert-Statement übernehmen, ohne sie nochmal mit extra SQL-Anfragen zu bestimmen.

i) Gegeben sei folgende SQL-Anfrage auf dem Universitätsschema (siehe Seite 23).

```
WITH VorlesungenKant AS (
  SELECT * FROM Professoren p, Vorlesungen v
   WHERE p.PersNr = v.gelesenVon AND p.Name = 'Kant'
)
SELECT count(*), sum(vk.SWS)
  FROM VorlesungenKant vk
 GROUP BY vk.PersNr
```

Geben Sie das Ergebnis der Anfrage tabellarisch an.

Fortsetzung nächste Seite!

Beispielausprägung

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2
25403	4630	2137	5

Abb. 1: Beispielausprägung für eine Universitäts-Datenbank

46116

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

Frühjahr 2021

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2021**	**46116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **15**

Bitte wenden!

Thema Nr. 1

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 **(6 Punkte)**

a) Erklären Sie ein Problem von Mehrfachvererbung in max. 2 Sätzen.

b) Erklären Sie einen Vorteil von Mehrfachvererbung in max. 2 Sätzen.

Aufgabe 2 **(13 Punkte)**

a) Sie sind dabei, eine Softwareanwendung für Gärtner zu entwickeln. Konkret wollen Sie einen Garten modellieren in dem es v. a. verschiedene Baumarten gibt. Zeichnen Sie ein UML Klassendiagramm auf Basis der folgenden Beschreibung. Halten Sie sich dafür exakt an die beschriebenen Details.

- Die Klasse Baum ist eine abstrakte Klasse.
- Die Klasse Baum hat eine abstrakte Methode reproduce.
- Die Klasse Baum ist eine Oberklasse der Klasse Nussbaum.
- Die Klasse Kiefer spezialisiert die Klasse Baum.
- Die Klasse Kischbaum ist eine Unterklasse der Klasse Baum.
- Die Klasse Nuss ist eine Implementierung der Schnittstelle Samen.
- Die Methode reproduce in Nussbaum instanziiert ein neues Nuss Objekt und gibt dieses zurück.

Ein Züchter kreuzt Kirschbaum und Nussbaum und erzeugt eine neue Baumart mit Namen KirschNussBaum. Tragen Sie auch diese Klasse in das Diagramm ein.

b) Kann das Klassenmodell in Java so implementiert werden? Begründen Sie Ihre Entscheidung in max. 2 Sätzen.

Aufgabe 3 **(8 Punkte)**

Der folgende Java-Programmcode berechnet die Anzahl an möglichen Permutationen für "dreimaliges Ziehen mit Zurücklegen". Der übergebende Parameter choices bestimmt dabei die Anzahl an Optionen pro Ziehung.

```
public class Permutate {

    // Three draws with replacement
   public int drawWithReplacement(int choices) {
      return choices * choices * choices;
   }
}
```

Fortsetzung nächste Seite!

Schreiben Sie eine Methode in einer objektorientierten Programmiersprache, welche die obige Methode überlädt. Diese überladende Methode soll die Anzahl an Permutationen für z. B. 2, 4, und 11-maliges Ziehen mit Zurücklegen berechnen.

Aufgabe 4 **(3 Punkte)**

Benennen Sie drei weitere grundlegende Prinzipien der Objektorientierung neben der Kapselung.

Aufgabe 5 **(3 Punkte)**

Betrachten Sie den folgenden Java-Programmcode. (Die Zeilennummern gehören nicht dazu.)

```
1    public class Person {
2       private String firstname;
3       private String lastname;
4       private int age;
5
6       Person (String f, String l, int a) {
7          firstname = f;
8          lastname = l;
9          age = a;
10      }
11   }
```

a) Benennen Sie die Art der Methode, die in den Zeilen 6-10 dargestellt ist.

b) Geben Sie das objektorientierte Prinzip an, das in den Zeilen 2, 3 und 4 demonstriert wird.

c) Welche Klassen werden im Beispielcode benannt?

Aufgabe 6 **(8 Punkte)**

Betrachten Sie die folgende Tabelle und beziehen Sie sich auf diese in den zwei folgenden Teilaufgaben.

package A; class Aa {	**package A; class Aa {**	**package A; class Ab {**	**package B; class Ac {**	**package B; class Ad extends Aa {**
private int i;	?	?	?	?
int j;	?	?	?	?
protected int k;	?	?	?	?
public int l;	Ja	Ja	Ja	Ja
}	}	}	}	}

a) Übernehmen Sie obige Tabelle in Ihre Bearbeitung und ersetzen Sie jedes '?' wie folgt: Wenn die in der jeweiligen Spaltenüberschrift genannte Klasse direkten Zugriff auf die in der jeweiligen Zeilenbeschriftung genannten Variable hat, schreiben Sie 'ja', ansonsten 'nein'.

Fortsetzung nächste Seite!

b) Nehmen Sie nun an, dass Klasse Aa aus obiger Tabelle private Sichtbarkeit hätte. Beschreiben Sie, inwiefern sich für Klassen aus Paket B der Zugriff auf die Variablen damit ändern würde.

Aufgabe 7 **(12 Punkte)**

a) Erklären Sie in einem Satz, was Kapselung im objektorientierten Sinne bedeutet.

b) Ist vereinfachte Wartung ein Vorteil von Kapselung? Begründen Sie in max. 2 Sätzen.

c) Ist reduzierte Komplexität ein Vorteil von Kapselung? Begründen Sie in max. 2 Sätzen.

d) Ist erhöhte Sicherheit (Safety) ein Vorteil von Kapselung? Begründen Sie in max. 2 Sätzen.

Aufgabe 8 **(14 Punkte)**

Teilaufgaben beziehen sich auf das Strukturmuster Adapter.

a) Erläutern Sie den Zweck (Intent) des Strukturmusters Adapter in max. 3 Sätzen, ohne dabei auf die technische Umsetzung einzugehen.

b) Erklären Sie die zwei Varianten (objektorientiert, klassenbasiert), das Strukturmuster Adapter umzusetzen. Schreiben Sie pro Variante max. 5 Sätze. Benennen Sie hierbei die verwendeten Klassen, z. B. "A" für Adapterklasse.

c) Nennen Sie einen Vorteil und einen Nachteil des Strukturmusters Adapter im Vergleich zu einer Implementierung, die nicht auf dieses Muster zurückgreift.

Fortsetzung nächste Seite!

Aufgabe 9 **(8 Punkte)**

Skizzieren Sie den Softwarelebenszyklus des traditionellen (nicht-agilen) Softwarelebenszyklus (Software Developement Life Cycle).

Aufgabe 10 **(15 Punkte)**

a) Geben Sie zwei traditionelle ("plan-getriebene") Methoden der Softwareentwicklung an.

b) Nennen Sie zwei Beispiele für eine agile Methode der Softwareentwicklung.

c) Was unterscheidet die agilen Methoden von traditionellen Methoden? Antworten Sie in drei unterschiedlichen Paaren (also in Gegensätzen zwischen den beiden Vorgehensmodellen, z. B. "Flexibilität im Vorgehen **statt** lange, umfangreiche Vorausplanung").

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1 (Vermischte Fragen) (24 Punkte)

a) Erläutern Sie die Begriffe Before und After Image. Erklären Sie deren Bedeutung für die Umsetzung des ACID-Prinzips.

b) Erläutern Sie Backward und Forward Recovery und grenzen Sie diese voneinander ab.

c) Beschreiben Sie ausführlich Funktionsweise und Aufgabe des Zwei-Phasen-Sperr-Protokolls.

d) Definieren Sie die Begriffe Entität und Entitätstyp.

e) Lässt sich jede tertiäre Beziehung durch drei binäre Beziehungen ausdrücken? Falls ja, begründen Sie dies. Falls nein, geben Sie ein Gegenbeispiel an.

f) Erklären Sie, wie sich ein ER-Modell in ein relationales Modell überführen lässt.

g) Beschreiben Sie, was man unter konkurrierenden Operationen versteht. Geben Sie ein Beispiel an.

Aufgabe 2 (22 Punkte)

Überführen Sie das im Folgenden gegebene ER-Modell in ein normalisiertes Relationenmodell. Vergessen Sie dabei nicht, Primär- und Fremdschlüssel entsprechend zu kennzeichnen. Geben Sie jeweils an, auf welche Spalte sich der Fremdschlüssel bezieht.

Erklärung der Notation:

- Teilmengen-Symbol: Stellt eine Vererbungsbeziehung dar, beispielsweise sind Privat und Geschäft Unterklassen von User.
- Das d bedeutet, dass die Vererbung disjunkt ist, d. h. ein Element der Oberklasse kann nicht zu mehreren Unterklassen gehören.
- Der Doppelstrich bei der Vererbung bedeutet, dass die Vererbung total ist, d. h. ein Element der Unterklasse ist zwingenderweise auch ein Element einer Oberklasse.
- Die doppelte Umrandung einer Entität und der dazugehörigen Relationship eine schwache Entität dar, d. h. diese Entität kann nicht ohne eine starke Entität existieren (Beispiel: Fahrzeug kann nicht ohne User existieren).
- Gestrichelt umrandete Attribute bezeichnen abgeleitete Attribute.
- Doppelt umrandete Attribute bezeichnen mehrwertige Attribute.
- Die hierarchische Struktur bei den Attributen stellen komposite Attribute dar.

Fortsetzung nächste Seite!

Aufgabe 3 (SQL-Abfragen) **(24 Punkte)**

Gegeben sind folgende Relationenschemata. Die Primärschlüssel der einzelnen Tabellen sind jeweils durch Unterstreichungen gekennzeichnet.

```
Interpret (Interpret_ID, Name, Alter, Nationalität)
Lied (Titel, Laenge, Interpret_ID)
      FK Interpret_ID REF Interpret
CD (ID, Name, Jahr)
Lied_auf_CD (ID, Titel, Position)
      FK ID REF CD
      FK Titel REF Lied
```

a) Schreiben Sie eine SQL-Anweisung, die die Namen aller Lieder des Künstlers „Luis Fonsi" ausgibt. Die Ausgabe soll nach der Länge absteigend geordnet erfolgen.

b) Schreiben Sie eine SQL-Anweisung, welche die ID aller CDs ausgibt, auf denen mindestens ein Interpret vertreten ist, der jünger als 16 Jahre ist.

c) Formulieren Sie einen Ausdruck in relationaler Algebra, der die Namen aller Interpreten ausgibt, die mindestens einen Titel auf der CD mit der ID 15472 besitzen und jünger als 25 Jahre alt sind.

d) Erklären Sie in Worten, was folgender Ausdruck in relationaler Algebra angibt:

$$\pi_{Name}(\sigma_{Alter \leq 20}(Interpret) \bowtie_{Interpret.Interpret_ID=Lied.Interpret_ID} \sigma_{Länge>5}(Lied))$$

Fortsetzung nächste Seite!

Aufgabe 4 (Normalisierung) **(9 Punkte)**

Im Nachfolgenden ist ein Relationenschema in erster Normalform gegeben. In diesem wird der Sachverhalt wiedergegeben, dass ein Kunde (beschrieben durch KundenNr und KundenName) eine oder mehrere Versicherungsverträge (beschrieben durch VNr und VArt) bei einer Versicherungsgesellschaft (beschrieben durch VName und VSitz) hat.

Gegeben sind die folgenden funktionalen Abhängigkeiten, die in der Menge FD zusammengefasst werden:

- `KundenNr` → `KundenName`
- `VNr` → `VArt`
- `VNr VSitz` → `KundenNr`
- `KundenNr VNr` → `VName`
- `VSitz` → `VName`

Die Menge FD ist bereits minimal.

a) Bestimmen Sie alle Schlüsselkandidaten der Relation Kunde unter Verwendung von FD.

b) Begründen Sie, weshalb das angegebene Schema Kunde nicht in dritter Normalform ist.

c) Überführen Sie das Schema Kunde in die dritte Normalform! Benennen Sie den Namen des verwendeten Algorithmus.

Aufgabe 5 (Transaktionsmanagement) **(11 Punkte)**

a) Zeichnen Sie den zugehörigen Serialisierungsgraph (es ist nicht notwendig, die Kanten des Graphen zu beschriften) zu folgender Historie:

$r_1(A)\ w_2(B)\ w_1(A)\ r_3(B)\ w_4(C) r_3(D) w_3(B) r_1(C) r_3(B) w_2(C) r_3(A) c_1 r_2(C) w_4(D) c_4 c_2 w_3(C) c_3$

b) Formulieren Sie das Serialisierungstheorem.

c) Ist die Historie aus Teilaufgabe a) serialisierbar? Begründen Sie kurz Ihre Antwort!

d) Erläutern Sie das ACID Prinzip.

Thema Nr. 2

Teilaufgabe I: Softwaretechnologie

Aufgabe 1 (Projektmanagement) (10 Punkte)

Gegeben seien folgende Aufgaben mit ihren jeweiligen Abhängigkeiten und ihrer jeweiligen Dauer:

Aufgabe	Abhängigkeiten	Dauer (Tage)
A	/	8
B	A	4
C	B	3
D	A	7
E	D	2
F	A, C, E	10

a) Zeichnen Sie ein CPM-Diagramm basierend auf der gegebenen Aufgabenliste. Zeichnen Sie hierbei auch den Start- und Endknoten ein.

b) Als *Slack* bezeichnet man die Zeit, um die eine Aufgabe bezüglich ihres frühesten Startzeitpunktes verzögert werden kann, ohne dass es Probleme bei der fristgerechten Fertigstellung des Projektes gibt. Berechnen Sie den Slack für alle Aktivitäten und tragen Sie die jeweiligen Werte in Ihr Diagramm ein.

c) Zeichnen Sie den kritischen Pfad ins Diagramm ein oder geben Sie die Aufgaben des kritischen Pfades in der folgenden Form an: **Start** → ··· → **Ende**. Sollte es mehrere kritische Pfade geben, geben Sie auch diese an. Geben Sie die Dauer des kritischen Pfades an.

Aufgabe 2 (Entwicklungsprozesse) (14 Punkte)

a) Zeichnen Sie das Wasserfallmodell und beschreiben Sie die einzelnen Phasen in kurzen Sätzen.

b) Nennen und erklären Sie je zwei Vor- und Nachteile des Wasserfallmodells.

Aufgabe 3 (Anforderungsanalyse) (15 Punkte)

Erstellen Sie ein Use-Case-Diagramm zur Modellierung einer Zeitungsredaktion, die aus Redakteuren, Chefredakteuren und Lesern besteht. Dabei sind folgende Angaben zu beachten:

/F010/ Ein Leser kann Zeitungsartikel kaufen.

/F020/ Leser, Redakteure und Chefredakteure können Zeitungsartikel lesen.

/F030/ Redakteure und Chefredakteure können Artikel verfassen.

/F040/ Chefredakteure können Artikel freigeben.

/F050/ Für alle Aktionen muss der Nutzer authentifiziert werden.

Hinweis: Verwenden Sie, falls sinnvoll, Vererbung zwischen den Akteuren.

Fortsetzung nächste Seite!

Aufgabe 4 (Entwurfsmuster) **(15 Punkte)**

a) Erstellen Sie ein Klassendiagramm, das verschiedene Strategien einen Baum zu travestieren, modelliert. Hierbei sollen genau die bekannten Strategien *Preorder und Postorder* angeboten werden. Der Rückgabetyp dieser Operationen ist `void`. Sie dürfen annehmen, dass bereits eine entsprechende Datenstruktur für den Baum namens `Tree` vorhanden ist.
Verwenden Sie für die Realisierung dieses Klassendiagramms das Entwurfsmuster *Strategy-Pattern*. Die Klasse `Serialization` (das Kontext-Objekt) kümmert sich um die Serialisierung des Baumes je nach ausgewählter Strategie. Die Auswahl der Strategie soll hierbei über eine get- und set-Methode geregelt werden. Außerdem sollten Sie in dieser Klasse mit einem Attribut die Baum-Instanz verwalten sowie eine Methode für die Ausführung des Algorithmus bereitstellen.

Hinweise:
- Die get- und set-Methode für die Baum-Instanz müssen nicht eingezeichnet werden.
- Sie müssen keine Konstruktoren angeben.

b) Geben Sie an, zu welcher Gruppe von Mustern das Entwurfsmuster *Strategy-Pattern* gehört.

c) Nennen Sie einen Vorteil des *Strategy-Patterns*.

Aufgabe 5 (Implementierung) **(21 Punkte)**

Gegeben ist der jeweilige Java-Programmcode der Klassen Verwaltung, Gehege und Tier.
Erstellen Sie unter Beachtung der Hinweise auf der nächsten Seite ein Sequenzdiagramm zur Methode `ausführen` der Klasse Verwaltung.

```
 1 public class Verwaltung {
 2
 3   public static void ausfuehren() {
 4       Gehege g1 = new Gehege();
 5       Gehege g2 = new Gehege();
 6
 7       // ab hier soll modelliert werden
 8       Tier t1 = new Tier();
 9       t1.setzeTierart("Zebra");
10       t1.setzeNamen("Daisy");
11       Tier t2 = new Tier();
12       t2.setzeTierart("Nilpferd");
13       t2.setzeNamen ("Hippo");
14
15       g1.fuegeTierHinzu(t1);
16
17       if (g1.besitztTierVonArt("Zebra")) {
18           g1.fuegeTierHinzu(t2);
19       } else {
20           g2.fuegeTierHinzu(t2);
21       }
22   }
23
24 }
```

Fortsetzung nächste Seite!

```
 1  public class Gehege {
 2
 3     private List<Tier> enthalteneTiere = new ArrayList<>();
 4
 5     public void fuegeTierHinzu(Tier tier) {
 6         enthalteneTiere.add(tier);
 7     }
 8
 9     public boolean besitztTierVonArt(String art) {
10         for (Tier tier : enthalteneTiere) {
11             if (tier.nenneArt().equals(art)) {
12                 return true;
13             }
14         }
15         return false;
16     }
17
18 }
```

```
 1  public class Tier {
 2     private String tierart;
 3     private String name;
 4
 5     public void setzeTierart(String tierart) {
 6         this.tierart = tierart;
 7     }
 8
 9     public void setzeNamen(String name) {
10         this.name = name;
11     }
12
13     public String nenneArt() {
14         return this.tierart;
15     }
16 }
```

Hinweise:

- Die Operationen auf Listen oder Strings müssen nicht gesondert dargestellt werden.
- Auf das Zeichnen einer passiven Lebenslinie muss nicht geachtet werden.
- Übertragen Sie das Diagramm von der nächsten Seite auf Ihren Bearbeitungsbogen und ergänzen Sie dieses im Folgenden.
- Lösen Sie if-Abfragen auf, sofern es möglich ist.
- Schleifen müssen nicht gesondert dargestellt werden, sondern betrachten Sie, falls notwendig, die konkrete Ausführung.
- Einzelne Rollen müssen nicht gesondert dargestellt werden.

Fortsetzung nächste Seite!

Aufgabe 6 (Testing) **(15 Punkte)**

a) Erklären Sie in wenigen Sätzen den Unterschied zwischen Fehlervermeidung und Fehlertoleranz.

b) Nennen und beschreiben Sie kurz die vier verschiedenen Testebenen.

c) Beschreiben Sie zwei Unterschiede zwischen Black-Box-Testing und White-Box-Testing. Nennen Sie sowohl zum Black-Box- als auch zum White-Box-Testing zwei Vorteile.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1 (Vermischte Fragen) (14 Punkte)

Beantworten Sie die folgenden Fragen und begründen bzw. erläutern Sie Ihre Antwort in jeweils ein bis zwei Sätzen.

a) Welches SQL-Schlüsselwort kann verwendet werden, um sicherzustellen, dass jedes Tupel nur einmal im Ergebnis auftaucht?

b) Welcher Unterschied besteht zwischen ALTER TABLE und UPDATE?

c) Welche Auswirkung hat die Verwendung von ON DELETE RESTRICT bei einem Fremdschlüsselattribut?

d) Kann eine abgeschlossene (*committed*) Transaktion wieder rückgängig gemacht werden? Warum (nicht)?

e) Was versteht man unter einem *prepared statement* im Kontext einer Programmierschnittstelle für relationale Datenbanken (z. B. JDBC)?

f) Worin besteht der Unterschied zwischen dem Zustand, dass ein Attribut vom Typ Integer 0 enthält, und dem Zustand, dass es NULL enthält?

g) Kann ein Fremdschlüssel auch auf ein anderes Attribut als den Primärschlüssel der referenzierten Tabelle zeigen?

Aufgabe 2 (ER-Modellierung) (23 Punkte)

Im Folgenden finden Sie die Beschreibung einer Software zur Notenverwaltung.
Erstellen Sie zu dieser Beschreibung ein ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme von Entity-Typen an Beziehungstypen.

Es gibt Studenten. Diese haben einen Namen und eine eindeutige Matrikelnummer. Daneben gibt es Prüfungen, die über eine eindeutige Prüfungsnummer sowie über eine Prüfungsart (z. B. mündlich oder schriftlich) verfügen. Dozenten haben einen eindeutigen Namen, einen Titel und sind einem Lehrstuhl zugeordnet. Jeder Student ist in mindestens einem Studiengang eingeschrieben. Ein Studiengang hat eine eindeutige Bezeichnung.
Studenten können Prüfungen in einem bestimmten Semester ablegen und erhalten dafür eine Note. Ein Semester hat ein eindeutiges Kürzel. In jedem Semester finden Prüfungen statt. Jede Prüfung gehört zu genau einer Vorlesung. Eine Vorlesung hat einen eindeutigen Namen und wird in mindestens einem Semester von mindestens einem Dozenten angeboten. Ein Dozent bietet mindestens eine Vorlesung an. Es kann mehrere Prüfungen für eine Vorlesung geben (bspw. Klausuren in unterschiedlichem Umfang), aber jede Prüfung gehört nur zu einer Vorlesung.

Fortsetzung nächste Seite!

Aufgabe 3 (Relationaler Entwurf) **(19 Punkte)**

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. Wenn ein Attribut zur korrekten Abbildung des ER-Diagramms als UNIQUE oder NOT NULL ausgezeichnet werden muss, geben Sie dies an.

Beispiel:

```
Relation1 (Primärschlüssel, Attribut1, Attribut2,
Fremdschlüsselattribut1[Relation1],
(Fremdschlüssel2_Attribut1, Fremdschlüssel2_Attribut2)[Relation2]);
Attribut1 UNIQUE
Attribut2 NOT NULL
```

Aufgabe 4 (SQL) **(25 Punkte)**

Gegeben sind folgende Relationen:

```
Verein(Name, Ort)
Athlet(ID, Verein[Verein], Vorname, Nachname)
Disziplin(Name, Beschreibung)
Wettbewerb(ID, Name, Ortsname[Ort]); Ortsname NOT NULL
Ort(Name, Land, Adresse)
Teilnahme(AthletID[Athlet], WettbewerbID[Wettbewerb], Name [Disziplin],
Ergebnis)
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz nicht mehrfach aus.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Wettbewerb* anlegt. Gehen Sie davon aus, dass alle anderen Tabellen bereits existieren.

b) Schreiben Sie eine SQL-Anweisung, die Vorname und Nachname aller Athleten ausgibt, die zu einem Verein aus Erlangen gehören, aufsteigend sortiert nach Nachname und bei gleichen Nachnamen, absteigend sortiert nach Vorname.

c) Schreiben Sie eine SQL-Anweisung, die für jede Disziplin das durchschnittliche Ergebnis ermittelt.

d) Schreiben Sie eine SQL-Anweisung, welche alle Athleten löscht, die noch nie an einem Wettbewerb teilgenommen haben.

e) Schreiben Sie eine SQL-Anweisung, welche die Athleten ausgibt, die an den meisten *unterschiedlichen* Wettbewerben bisher teilgenommen haben. Geben Sie zu jedem Athleten die Attribute AthletID, Vorname und Nachname an sowie die Anzahl der Wettbewerbe, an denen er teilgenommen hat.

Aufgabe 5 (Normalformen) **(9 Punkte)**

Gegeben ist die Relation $R(a_1, a_2, a_3, a_4, a_5)$. Sie besitzt den Schlüsselkandidaten $\{a1, a2\}$ und die funktionalen Abhängigkeiten $a_2 \rightarrow a_3$ und $a_4 \rightarrow a_5$.

a) Begründen Sie in ein bis zwei Sätzen, warum R sich nicht in zweiter Normalform befindet. Zerlegen Sie R so, dass die Ergebnisrelationen die zweite Normalform erfüllen. Erzeugen Sie dabei so wenige Relationen wie möglich.

b) Begründen Sie in ein bis zwei Sätzen, warum sich die Ergebnisrelationen der vorherigen Teilaufgabe nicht alle in der dritten Normalform befinden. Zerlegen Sie sie so weiter, dass alle Relationen die dritte Normalform erfüllen. Erzeugen Sie dabei so wenige Relationen wie möglich.