Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2015

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: **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.

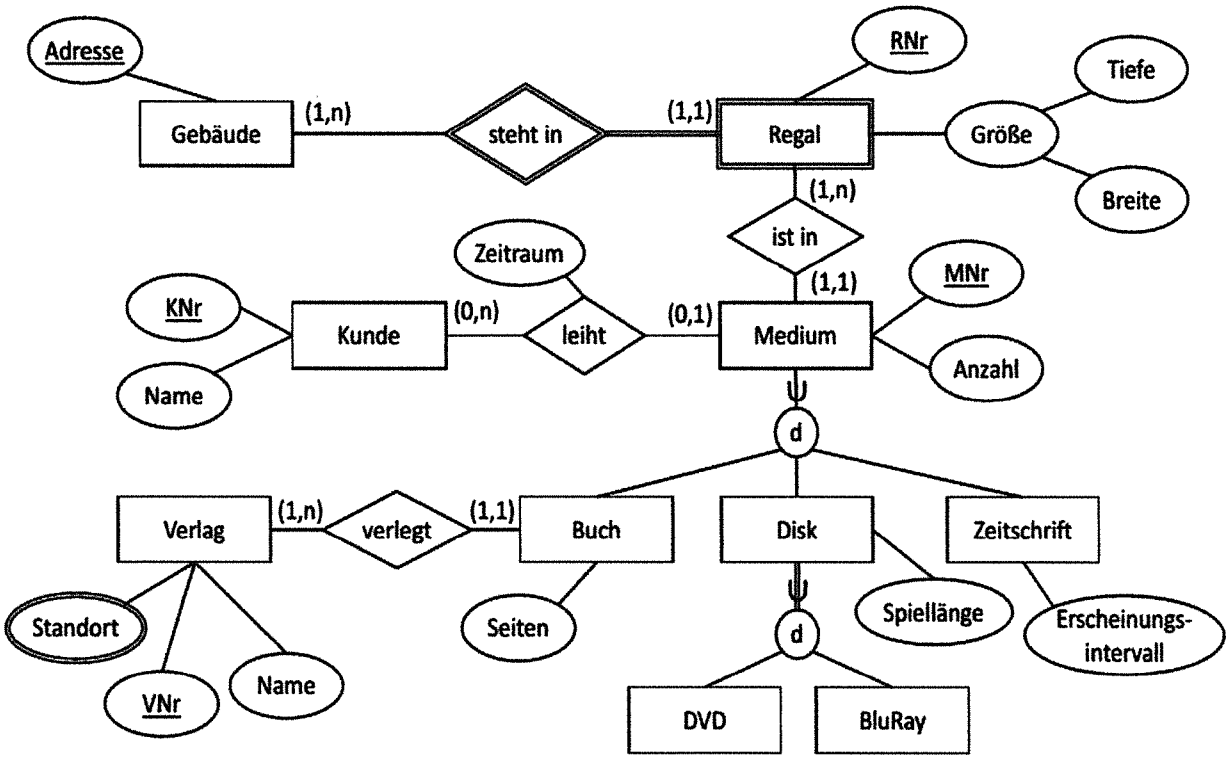
2.2
Erläutern Sie anhand eines selbstgewählten Beispiels, wann und warum es notwendig ist, bei rekursiven Beziehungen Rollennamen zu vergeben.

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:

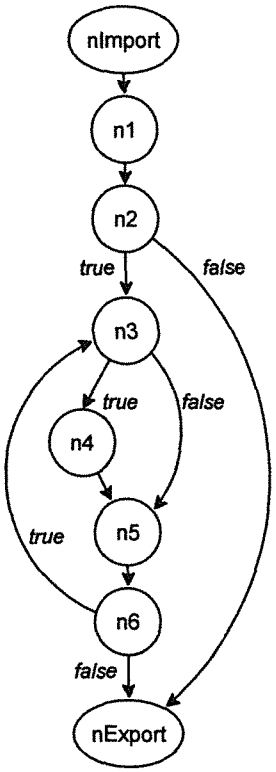
```
<klausur>
    <titel>Staatsexamen</titel>
    <aufgabe name="UML" punkte="42"/>
    <aufgabe name="Testen" punkte="47">
        <teilaufgabe name="a)" punkte="11">Geben Sie  </teilaufgabe>
    </aufgabe>
</klausur>
```

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).