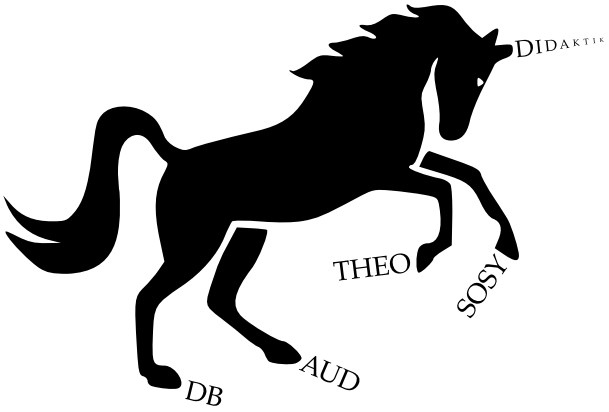
66116
Herbst
2016

Datenbanksysteme / Softwaretechnologie (vertieft)

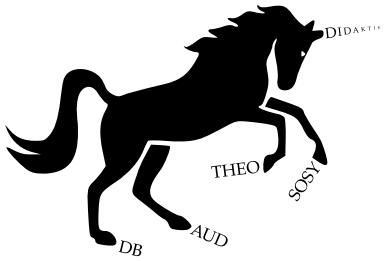
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Teilaufgabe Nr. 1

Aufgabe 1: SQL [Personalverwaltung]

Gegeben sind folgende Relationen aus einer Personalverwaltung:

`Mitarbeiter : {[ MitarbeiterID, Vorname, Nachname, Vorgesetzter[Mitarbeiter], AbteilungsID[Abteilung], Telefonnummer, Gehalt ]}`

`Abteilung : {[ AbteilungsID, Bezeichnung ]}`

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

Lösungsvorschlag

```
SELECT Vorname, Nachname
FROM Mitarbeiter m, Abteilung a
WHERE
  m.AbteilungsID = a.AbteilungsID AND
  a.Bezeichnung = 'Buchhaltung'
ORDER BY m.MitarbeiterID DESC;


 vorname | nachname
---------+----------
 Till    | Fuchs
 Hans    | Meier
(2 rows)
```

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

Lösungsvorschlag

```
SELECT m.Nachname AS Mitarbeiter, v.Nachname AS Vorgesetzter
FROM Mitarbeiter m LEFT OUTER JOIN Mitarbeiter v
ON m.Vorgesetzter = v.MitarbeiterID;


 mitarbeiter | vorgesetzter
-------------+--------------
 Meier       | Müller
 Wolitz      | Müller
 Müller      |
 Fuchs       | Wolitz
 Hase        | Müller
```

```
 Navratil    |
 Schmidt     | Navratil
(7 rows)
```

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

Lösungsvorschlag

```
CREATE VIEW Durchschnittsgehälter AS
SELECT Abteilung.AbteilungsID, Bezeichnung,
  AVG (Gehalt) AS Durchschnittsgehalt
FROM Mitarbeiter, Abteilung
WHERE Mitarbeiter.AbteilungsID = Abteilung.AbteilungsID
GROUP BY Abteilung.AbteilungsID, Bezeichnung;

SELECT a.Bezeichnung, a.Durchschnittsgehalt, COUNT (*) AS Rang
FROM Durchschnittsgehälter a, Durchschnittsgehälter b
WHERE a.Durchschnittsgehalt <= b.Durchschnittsgehalt
GROUP BY a.AbteilungsID, a.Bezeichnung, a.Durchschnittsgehalt
HAVING COUNT(*) <= 10
ORDER BY Rang ASC;


 bezeichnung | durchschnittsgehalt | rang
-------------+---------------------+------
 Managment   |              6514.5 |    1
 Buchhaltung |                2340 |    2
 Vertrieb    |              1283.5 |    3
 Produktion  |                 654 |    4
(4 rows)
```

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

Lösungsvorschlag

```
 vorname | nachname | gehalt
---------+----------+--------
 Lea     | Müller   |   5875
 Gerd    | Navratil |   7154
(2 rows)

SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;
```

```
UPDATE Mitarbeiter
SET Gehalt = 1.05 * Gehalt
WHERE AbteilungsID = 42;

SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;


 vorname | nachname |      gehalt
---------+----------+------------------
 Lea     | Müller   |           6168.75
 Gerd    | Navratil | 7511.700000000001
(2 rows)
```

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

Lösungsvorschlag

```
 vorname | nachname
---------+----------
 Hans    | Meier
 Fred    | Wolitz
 Lea     | Müller
 Till    | Fuchs
 Fred    | Hase
 Gerd    | Navratil
 Jürgen  | Schmidt
(7 rows)

 abteilungsid |    bezeichnung
--------------+--------------------
            1 | Buchhaltung
            2 | Vertrieb
           42 | Managment
            4 | Qualitätskontrolle
            5 | Produktion
(5 rows)

SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;

DELETE FROM Mitarbeiter
WHERE AbteilungsID IN (
  SELECT a.AbteilungsID
  FROM Abteilung a
  WHERE a.Bezeichnung = 'Qualitätskontrolle'
);
```

```
DELETE FROM Abteilung
WHERE Bezeichnung = 'Qualitätskontrolle';

SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;


 vorname | nachname
---------+----------
 Fred    | Wolitz
 Lea     | Müller
 Till    | Fuchs
 Gerd    | Navratil
 Jürgen  | Schmidt
(5 rows)

 abteilungsid | bezeichnung
--------------+-------------
            1 | Buchhaltung
            2 | Vertrieb
           42 | Managment
            5 | Produktion
(4 rows)
```

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

Lösungsvorschlag

Wir könnten eine VIEW erstellen, die zwar Namen und ID der anderen Mitarbeiter, sowie ihre Telefonnummern enthält (evtl. auch Abteilungsbezeichnung und ID), aber eben nicht das Gehalt: Mitarbeiter arbeiten auf eingeschränkter Sicht.

Alternativ mit `GRANT`:

explizit mit `SELECT` die Spalten auswählen, die man lesen können soll (auf nicht angegebene Spalten ist kein Zugriff möglich)

```
GRANT SELECT (Vorname, Nachname, Telefonnummer)
ON Mitarbeiter TO postgres;
```

Aufgabe 5 [Physische Datenorganisation]

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

Lösungsvorschlag

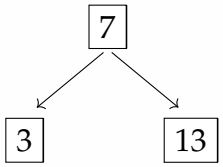
- Daten werden in Form von *Sätzen* auf der Festplatte abgelegt, um auf Sätze zugreifen zu können, verfügt jeder Satz über eine *eindeutige, unveränderliche Satzadresse*
- TID = Tupel Identifier: dient zur Adressierung von Sätzen in einem Segment und besteht aus zwei Komponenten:
  - Seitennummer (Seiten bzw. Blöcke sind größere Speichereinheiten auf der Platte)
  - Relative Indexposition innerhalb der Seite
- Satzverschiebung innerhalb einer Seite bleibt ohne Auswirkungen auf den TID. Wird ein Satz auf eine andere Seite migriert, wird eine „Stellvertreter-TID“ zum Verweis auf den neuen Speicherort verwendet. Die eigentliche TID-Adresse bleibt stabil.

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

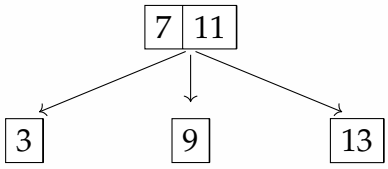
3, 7, 13, 11, 9, 10, 8

Lösungsvorschlag

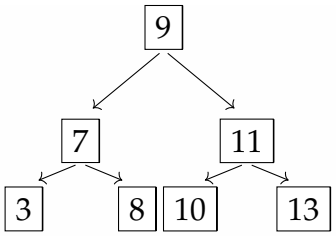
- 3 (einfaches Einfügen)
- 7 (einfaches Einfügen)
- 13 (Split)



- 11 (einfaches Einfügen)
- 9 (Split)



- 10 (einfaches Einfügen)
- 8 (Doppel-Split)



(c) Gegeben ist der folgende B-Baum:

Die folgenden Teilaufgaben sind voneinander unabhängig.

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

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

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

Teilaufgabe Nr. 2

Aufgabe 1 [Softwaresysteme: Begriffe und Konzepte]

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

Lösungsvorschlag

Im Rahmen der Zielbestimmung werden die Ziele des Produktes in einer Art Prioritätenliste exakt eingegrenzt, die in drei weitere Kategorien gegliedert ist, die als **Muss-**, **Wunsch-** und **Abgrenzungskriterien** bezeichnet werden.

Musskriterien Zu den Musskriterien zählen die Produktfunktionen, die für ein Funktionieren des Produktes unabdingbar sind. Ohne ihre Umsetzung geht einfach nichts, deshalb müssen sie erfüllt werden.

Wunschkriterien Die nächste Kategorie, die Wunschkriterien, sind zwar entbehrlich, wurden aber ausdrücklich vom Auftraggeber gefordert. Ihre Umsetzung im Produkt ist also genauso eine Pflicht, wie die der ersten Kategorie, das Produkt würde aber auch ohne ihre Implementierung seinen Betrieb korrekt ausführen.

Abgrenzungskriterien Die letzte Kategorie beschreiben die Abgrenzungskriterien. Sie sollen die Grenzen des Produktes klarmachen und sind von daher beinahe wichtiger als die beiden ersten Fälle denn sie hindern die Entwickler daran, über alle Ziele hinauszuschießen.

https://st.inf.tu-dresden.de/SalesPoint/v3.2/tutorial/stepbystep2/pflh.html

(b) Nennen Sie drei Einsatzziele von Softwaremaßen.

Lösungsvorschlag

Aus Baumann K. (1997) Softwaremaße. In: Unterstützung der objektorientierten Systemanalyse durch Softwaremaße. Beiträge zur Wirtschaftsinformatik, vol 23. Physica, Heidelberg. https://doi.org/10.1007/978-3-662-13273-9_2

Der Einsatz von Softwaremaßen kann vielfältige Vorteile mit sich bringen. Folgende Ziele können mit der Anwendung von Softwaremaßen verfolgt werden:

- Durch eine Bewertung kann überprüft werden, inwieweit einmal gestellte *Qualitätsanforderungen* erfüllt wurden.
- Eine Bewertung bereits erstellter oder noch zu erstellender Softwareprodukte kann Hilfestellung bei Entscheidungen, die für die *Projektplanung* oder das Entwicklungsmanagement zu treffen sind, leisten. So können Softwaremaße zur Einschätzung des notwendigen Aufwands, der zu erbringenden Kosten und des Personalbedarfs herangezogen werden.
- Ein möglichst frühzeitiges *Aufzeigen von Schwachstellen im Systemdesign*, die Beurteilung der Auswirkungen neuer Techniken oder Tools auf die Produktivität der Entwickler oder auf die Qualität des entwickelten Produkts sowie die Überprüfung der Einhaltung festgelegter Designstandards sind weitere mögliche Einsatzfelder für Softwaremaße.
- Darüber hinaus ist der Einsatz von Softwaremaßen *Teil umfassender Konzepte zum Softwarequalitätsmanagement*. Softwaremaße bzw. die zugehörige Meßergebnisse tragen zunächst dazu bei, die betrachteten Produkte oder Prozesse besser zu verstehen oder hinsichtlich interessierender Eigenschaften zu bewerten.

https://link.springer.com/chapter/10.1007%2F978-3-662-13273-9_2

(c) Welche Arten von Softwaremaßen werden unterschieden?

Lösungsvorschlag

Die Unterscheidung nach dem Messgegenstand ergibt folgende Arten von Maßen:

Externe Produktmaße. Diese beschreiben Merkmale des Produkts, die von außen sichtbar sind, wenngleich eventuell nur indirekt. Dazu gehören insbesondere Maße für Qualitätsattribute wie die Wartbarkeit, die Zuverlässigkeit, die Benutzbarkeit, die Effizienz etc.

Interne Produktmaße. Basis für die Charakterisierung externer Produktattribute sind interne Attribute, die sich zumindest zum Teil besser messen lassen. Hierzu gehören z. B. die Größe, die Modularität und die Komplexität.

Prozessmaße. Diese charakterisieren Eigenschaften des Produktionsprozesses, z. B.

die typische Produktivität, Kostenaufteilung auf bestimmte Arbeiten, Dauer von Arbeitsschritten etc.

Ressourcenmaße. Diese charakterisieren Eigenschaften der im Prozess verwendeten Ressourcen, z. B. die Auslastung von Rechnern, die typische Produktivität und Fehlerrate eines bestimmten Ingenieurs etc.

`http://page.mi.fu-berlin.de/prechelt/swt2/node5.html`

Was sind die 3 Arten der (einfachen) Maße zur Messung von Software?

https://quizlet.com/de/339799466/softwaretechnik-theorie-flash-cards/

- Größenorientiert/nach Größe
- Funktionsorientiert/nach Funktion
- Objektorientiert/nach Objekt

(d) Nennen Sie drei Merkmale evolutionärer Softwareentwicklung.

Lösungsvorschlag

Wie für jede Form der Software-Entwicklung stellt sich die Frage nach einem geeigneten methodischen Ansatz. Generell lässt sich der Prozess der Software-Entwicklung in die folgenden drei Abschnitte gliedern:

(i) von der Problembeschreibung bis zur Software-Spezifikation (auch Problemanalyse, Systemanalyse, Requirements Engineering bzw. frühe Phasen der Software-Entwicklung ge- nannt),

(ii) die Software-Entwicklung im engeren (technischen) Sinn,

(iii) die Integration der Software in den Anwendungszusammenhang.

Diese Einteilung gilt ebenso für die evolutionäre Software-Entwicklung, allerdings wird hier der Schwerpunkt anders gelegt. Steht beim klassischen Ansatz in der Regel die Software-Entwicklung im engeren Sinn im Mittelpunkt, so werden beim evolutionären Ansatz alle drei Abschnitte möglichst gleichgewichtig und im Zusammenhang betrachtet. Daraus resultiert eine natürliche Tendenz zu einer zyklischen Vorgehensweise.

Daneben gibt es weitere Querbezüge: Orientiert sich die Software-Entwicklung an einem festen Ziel, verfolgt dieses jedoch durch schrittweises Erweitern zunächst unvollkommener Teillösungen, so wird dieses Vorgehen inkrementell genannt. Unfertige Versuchsanordnungen oder Teilsysteme, die am Beginn einer solchen Entwicklung stehen, werden oft als Prototypen bezeichnet. Wählt man Prototyping als Vorgehensweise, so kann entweder das spätere Produkt inkrementell aus einem oder mehreren Prototypen weiterentwickelt werden oder man setzt nach einer (Wegwerf-) Prototypphase neu auf. Beides sind Formen evolutionärer Entwicklung.

`http://waste.informatik.hu-berlin.de/~dahme/edidahm.pdf`

(e) Nennen Sie drei Vorteile des Einsatzes von Versionsverwaltungssoftware.

Lösungsvorschlag

(i) Möglichkeit, auf ältere Entwicklungsversionen zurückzukehren, wenn sich die aktuelle als nicht lauffähig bzw. unsicher erwiesen hat (z. B. git revert)

(ii) Möglichkeit, dass mehrere Entwickler gleichzeitig an ein- und denselben Softwareprojekt arbeiten und Möglichkeit, dass ihre Entwicklungen durch das Versionsverwaltungssystem zusammengeführt werden können (z. B. git merge)

(iii) Möglichkeit, den Zeitpunkt oder den Urheber eines Softwarefehlers ausfindig zumachen (z. B. git blame)

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

Lösungsvorschlag

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

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

Lösungsvorschlag

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (*rightarrow* neue Programmversion).

Aufgabe 2 [PKI-System Lehrer Schüler]

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

Damit der Schüler seinem Lehrer die Hausaufgaben verschlüsselt per E-Mail übermitteln kann, bedarf es einer entsprechenden Infrastruktur. Nachdem beide Teilnehmer die notwendige Software installiert haben, erstellt der Lehrer zunächst ein sogenanntes Schlüsselpaar, bestehend aus einem „Öffentlichen“ (ÖS) und einem zugehörigen „privaten“ Schlüssel (PS). Anschließend veröffentlicht der Lehrer seinen ÖS durch Hochladen auf einen sogenannten Keyserver (Schlüsselverzeichnisdienst). Damit steht er jedem Schüler zur Verfügung, so dass dieser den ÖS jederzeit vom Keyserver herunterladen kann. Alternativ kann der Lehrer seinen ÖS auch direkt (z. B. per E-Mail oder USB-Stick) an den Schüler übermitteln.

Der Schüler kann nun seine Nachricht (z. B. seine Lösung) mit dem ÖS des Lehrers verschlüsseln und mit einer E-Mail versenden. Empfängt der Lehrer eine solche E-Mail, so kann er (und nur er) mit seinem PS die Nachricht wieder entschlüsseln. Umgekehrt kann der Lehrer mit

seinem PS beliebige Informationen (z. B. die Note) „digital signieren". Diese unterschriebenen Daten übermittelt der Lehrer dann zusammen mit der digitalen Signatur per E-Mail an den Schüler. Der Schüler kann mit dem ÖS des Lehrers prüfen, ob die übermittelte Nachricht unverändert und tatsächlich vom unterschreibenden Lehrer stammt.

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

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

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

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

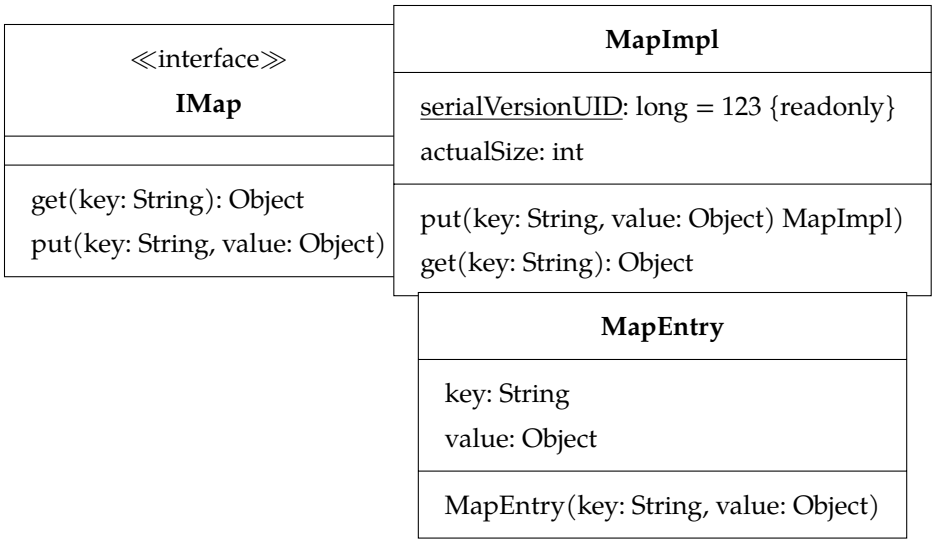
(iii) Jeder Lehrer hat höchstens ein Schlüsselpaar, aber jeder Schlüssel gehört genau einem Lehrer. Der Schüler hingegen kommuniziert mit mehreren Lehrern und kennt daher mehrere E-Mail-Adressen.

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

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

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

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



Lösungsvorschlag

Interface „IMap“

```java
interface IMap {
  Object get(String key);

  void put(String key, Object value);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/IMap.java`

Klasse „MapImpl“

```java
import java.util.ArrayList;
import java.util.List;

class MapImpl implements IMap {
  final static long serialVersionUID = 123;

  private List<MapEntry> entries;

  MapImpl() {
    entries = new ArrayList<MapEntry>();
  }

  public Object get(String key) {
    return new Object();
  }

  public void put(String key, Object value) {
    // Nicht verlangt in der Aufgabenstellung.
    entries.add(new MapEntry(key, value));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/MapImpl.java`

Klasse „MapEntry“

```
class MapEntry {
  String key;
  Object value;

  MapEntry(String key, Object value) {
    // Nicht verlangt in der Aufgabenstellung.
    this.key = key;
    this.value = value;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/MapEntry.java`

Aufgabe 3 [Sort-Methode und datenflussorientierte Überdeckungskritierien]

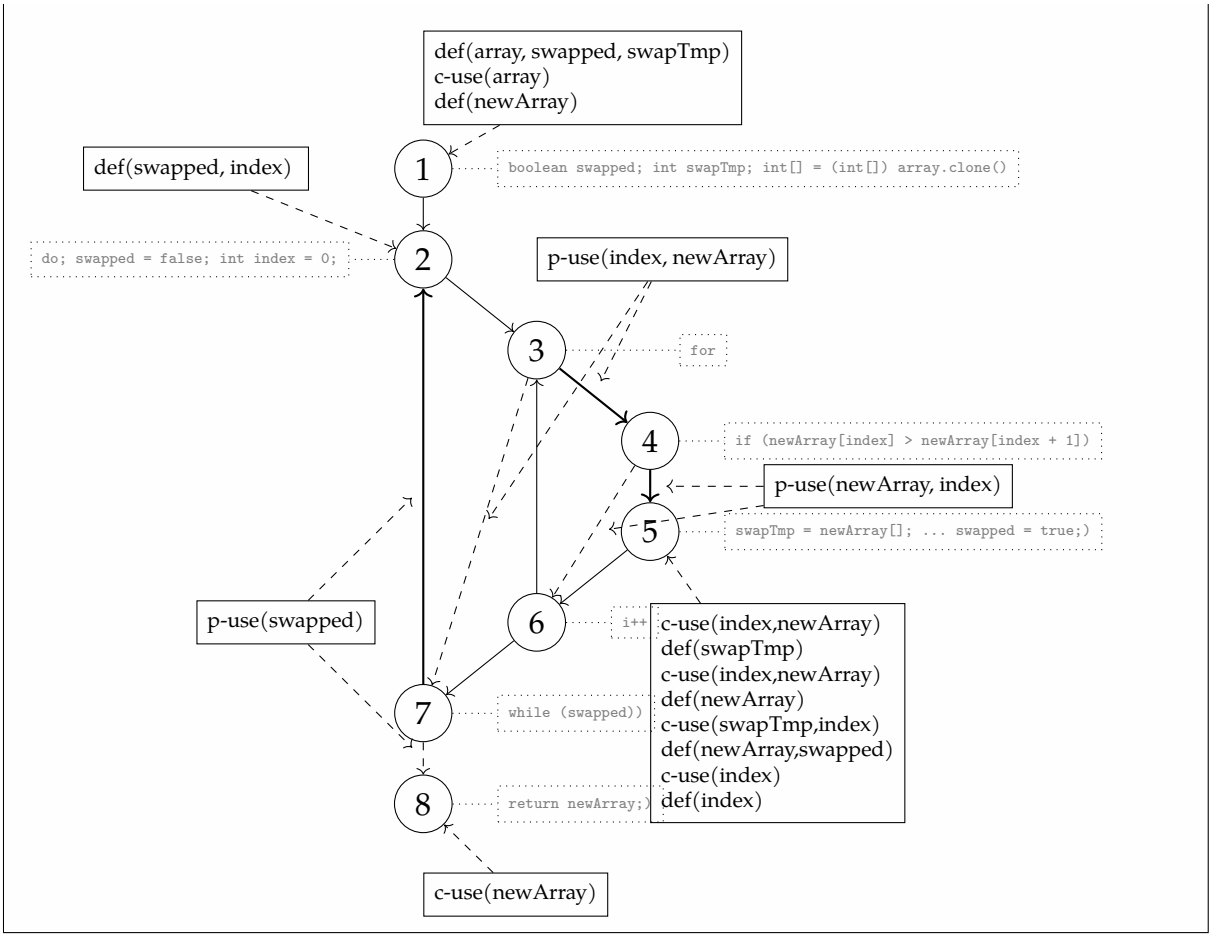
Gegeben Sei folgende Java-Methode `sort` zum Sortieren eines Feldes ganzer Zahlen:

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

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/BubbleSort.java`

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

Lösungsvorschlag



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

Lösungsvorschlag

Der Graph hat 8 Knoten und 10 Kanten. Daher ist die zyklomatische Komplexität nach McCabe gegeben durch 10 - 8 + 2 = 4.

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

Lösungsvorschlag

Die Eingabe muss mindestens ein Feld der Länge 3 sein. Ansonsten wäre das Feld schon sortiert bzw. bräuchte nur eine Vertauschung und die innere if-Bedingung wäre nicht zu 100% überdeckt. Daher wählt man beispielsweise array = [1,3,2].

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

Lösungsvorschlag

Das Kriterium all-uses ist das Hauptkriterium des datenflussorientierten Testens, denn es testet den kompletten prädikativen und berechnenden Datenfluss. Konkret: von jedem Knoten mit einem globalen def(x) einer Variable x existiert ein definitions-freier Pfad bzgl. x (def-clear(x)) zu jedem erreichbaren Knoten mit ei-

nem c-use(x) oder p-use(x)).

Thema Nr. 2

Teilaufgabe Nr. 1

Aufgabe 2 [Schulverwaltung]

Gegeben sei der folgende Ausschnitt aus dem Schema einer Schulverwaltung:

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

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

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

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

```sql
CREATE TABLE Person (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Wohnort VARCHAR(255),
  Typ CHAR(1) CHECK(Typ in ('S', 'L'))
);

CREATE TABLE Klasse (
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  Klassenlehrer INTEGER REFERENCES Person(ID),
  PRIMARY KEY (Klassenbezeichnung, Schuljahr)
);

CREATE TABLE Klassenverband (
```

```
  Schüler INTEGER REFERENCES Person(ID),
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  PRIMARY KEY (Schüler, Schuljahr)
);

CREATE TABLE Unterricht (
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  Lehrer INTEGER REFERENCES Person(ID),
  Fach VARCHAR(100),
  CONSTRAINT Unterricht_PK
    PRIMARY KEY (Klassenbezeichnung, Schuljahr, Lehrer, Fach)
);

INSERT INTO Person VALUES
  (1, 'Lehrer Ludwig', 'München', 'L'),
  (2, 'Schüler Max', 'München', 'S'),
  (3, 'Schülerin Maria', 'München', 'S'),
  (4, 'Schülerin Eva', 'Starnberg', 'S'),
  (5, 'Lehrerin Walter', 'München', 'L'),
  (6, 'Schüler Karl', 'München', 'S');

INSERT INTO Klasse VALUES
  ('1a', 2015, 1),
  ('1a', 2014, 1),
  ('1b', 2015, 5);

INSERT INTO Klassenverband VALUES
  (2, '1a', 2015),
  (3, '1a', 2015),
  (4, '1b', 2015),
  (6, '1a', 2015),
  (6, '1a', 2014);
```

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

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

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS Person(
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Wohnort VARCHAR(255),
  Typ CHAR(1) CHECK(Typ in ('S', 'L'))
);

CREATE TABLE IF NOT EXISTS Klasse(
```

```
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  Klassenlehrer INTEGER REFERENCES Person(ID),
  PRIMARY KEY (Klassenbezeichnung, Schuljahr)
);

CREATE TABLE IF NOT EXISTS Unterricht (
  Klassenbezeichnung VARCHAR(20) REFERENCES Klasse(Klassenbezeichnung),
  Schuljahr INTEGER REFERENCES Klasse(Schuljahr),
  Lehrer INTEGER REFERENCES Person(ID),
  Fach VARCHAR(100),
  CONSTRAINT Unterricht_PK
    PRIMARY KEY (Klassenbezeichnung, Schuljahr, Lehrer, Fach)
);
```

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

Lösungsvorschlag

Ich habe `REFERENCES` bei Unterricht Schuljahr vergessen, die referenzierten Tabellen `Person` und `Klasse` wurden in der Musterlösung auch nicht angelegt.

```
ALTER TABLE Person
  ADD CONSTRAINT TypLS
    CHECK(Typ IN ('S', 'L'));
```

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

Lösungsvorschlag

Falsch: `ORDER BY Anzahl;`. `DESC` vergessen.

```
SELECT k.Klassenbezeichnung, COUNT(*) AS Anzahl
FROM Klasse k, Klassenverband v
WHERE
  k.Schuljahr = 2015 AND
  k.Klassenbezeichnung = v.Klassenbezeichnung
GROUP BY k.Klassenbezeichnung
ORDER BY COUNT(*) DESC;
```

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

Lösungsvorschlag

```
SELECT DISTINCT l.Name
FROM Person l
WHERE NOT EXISTS(
  SELECT DISTINCT *
  FROM Unterricht u, Klassenverband v, Person s
  WHERE
    u.Lehrer = l.ID AND
    u.Klassenbezeichnung = v.Klassenbezeichnung AND
    v.Schüler = s.ID AND
```

```
    l.Wohnort != s.Wohnort
);
```

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

Lösungsvorschlag

```
SELECT s.Name
FROM Person s, Klasse k, Klassenverband v
WHERE
  v.Klassenbezeichnung = k.Klassenbezeichnung AND
  k.Schuljahr = v.Schuljahr AND
  v.Schüler = s.ID
GROUP BY s.ID, s.Name
HAVING COUNT(*) = 1
```

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

Lösungsvorschlag

```
SELECT DISTINCT s1.ID, s2.ID
FROM Klassenverband s1, Klassenverband s2
WHERE
  s1.Schuljahr = s2.Schuljahr AND
  s1.Schueler <> s2.Schueler AND
  s1.Klassenbezeichung = s2.Klassenbezeichnung;
```

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

Lösungsvorschlag

$$\pi_{\text{Schüler}}\Big(\sigma_{\text{Name}=\text{'Ludwig Lehrer'}}(\text{Person}) \bowtie_{\text{Person.ID}=\text{Unterricht.Lehrer}} \text{Unterricht} \bowtie_{\text{Unterricht.Klassenbezeichnung}=\text{Klassenverband.Klassenbezeichnung}} \text{Klassenverband}\Big)$$

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

Lösungsvorschlag

$$\pi_{\text{Name,ID}}($$
$$($$
$$\pi_{\text{Lehrer,Schueler}}(\text{Unterricht} \bowtie \text{Klassenverband}) \div \pi_{\text{ID}}(\sigma_{\text{Typ}='\text{L}'}(\text{Person}))$$
$$)$$
$$\bowtie$$
$$\text{Person}$$
$$)$$

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

Teilaufgabe Nr. 2

Aufgabe 2 [Wahrheitsgehalt-Tabelle Software Engineering]

Ordnen Sie die folgenden Aussagen entsprechend ihres Wahrheitsgehaltes in einer Tabelle der folgenden Form an:

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

A Allgemein

- **A1** Im Software Engineering geht es vor allem darum qualitativ hochwertige Software zu entwickeln.
- **A2** Software Engineering ist gleichbedeutend mit Programmieren.

B Vorgehensmodelle

- **B1** Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.
- **B2** Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.
- **B3** Das Spiralmodell ist ein Vorläufer sogenannter Agiler Methoden.

C Anforderungserhebung

- **C1** Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z. B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zu, Vorschein kommen könnten.

C2 Ein Szenario beinhaltet eine Menge von Anwendungsfällen.

C3 Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.

D Architekturmuster

D1 Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.

D2 Das Black Board Muster ist besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.

D3 „Dependency Injection“ bezeichnet das Konzept, welches Abhängigkeiten zur Laufzeit reglementiert.

E UML

E1 Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.

E2 Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.

E3 Komponentendiagramme beschreiben die Struktur eines Systems.

F Entwurfsmuster

F1 Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.

F2 Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.

F3 Im Kommando Enwurfsmuster (engl. „Command Pattern“) werden Befehle in einem sog. Kommando-Objekt gekapselt, um sie bei Bedarf rückgängig zu machen.

G Testen

G1 Validation dient der Überprüfung von Laufzeitfehlern.

G2 Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.

G3 Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

Lösungsvorschlag

Kategorie	WAHR	FALSCH
A	A1	A2
B	B3	B1, B2
C	C3	C1, C2
D	D3	D1, D2
E	E1, E2, E3	
F	F2, F3	F1
G	G3	G1[a], G2[b]

[a] Validierung: Prüfung der Eignung beziehungsweise der Wert einer Software bezogen auf ihren Einsatzzweck: „Wird das richtige Produkt entwickelt?“

[b] Ein Softwaretest prüft und bewertet Software auf Erfüllung der für ihren Einsatz definierten Anforderungen und misst ihre Qualität.