

Softwareentwurf und Programmierung – Lösungen

Aufgabe 1 – Informatikaufgabe, 20 Punkte

a) Mögliche Nachteile der gewählten Datenbankstruktur:

- Durch die teils redundant geführten Namen, wie z.B. der Titel „Datenbanken“, kann es zu Inkonsistenzen kommen
- Durch die teils redundant geführten Namen, wie z.B. der Titel „Datenbanken“, ist der Pflegeaufwand größer
- Durch die Nichtbeachtung der funktionalen Abhängigkeiten kann es hier möglich sein, voll funktional abhängige Daten voneinander getrennt einzugeben, z.B. die Eingabe unterschiedlicher Titel zur selben ISBN.
- Die eine Tabelle enthält alle Daten, was schnell unübersichtlich werden kann
- ...

b) Normalisierung:

- i. Tabellen dürfen nur atomare Einträge besitzen, es ist also keine Zusammensetzung mehr erlaubt. So würde die 1. Normalform erfüllt:

ISBN, Autor

ISBN, Titel, Filialbezeichnung, Filialstraße, HausNr, Filialplz, Filialort, Anzahl, Preis

- ii. Alle Attribute in den Tabellen müssen voll funktional abhängig von allen Schlüsselattributen sein. Offensichtlich ist z.B. die Filialbezeichnung nicht funktional abhängig von der ISBN. Daher muss die eine Tabelle nun in mehrere Tabellen aufgespalten werden:

ISBN, Autor

ISBN, Titel, Preis

Filialbezeichnung, Straße, HausNr, PLZ, Ort

ISBN, Filialbezeichnung, Anzahl

- iii. Bei Datenbankentwürfen, die die 3. Normalform erfüllen, müssen alle Attribute direkt und nicht transitiv voneinander abhängen. Der Entwurf aus ii. ist nicht in 3. Normalform, da der Filialort nur transitiv von der Bezeichnung abhängig ist. Wenn man seinen Entwurf also in 3. NF bringen möchte, kann man eine weitere Tabelle mit PLZ, Ort anlegen, wobei der Primärschlüssel die PLZ ist.

c)

<u>ISBN</u>	<u>Autor</u>
3110443759	Kemper
3110443759	Eickler
3897215675	Passig
3527707212	Haffner

<u>ISBN</u>	<u>Titel</u>	<u>Preis</u>
3110443759	Datenbanken	50
3897215675	Weniger schlecht Programmieren	25
3527707212	Lineare Algebra für Dummies	23

<u>Filialbezeichnung</u>	<u>Straße</u>	<u>HausNr</u>	<u>PLZ</u>	<u>Ort</u>
Pontfiliale	Pontstraße	121	52062	Aachen
Innenstadtfiliale	Buchkremerstraße	1-7	52062	Aachen
Lager	Matthiashofstraße	28	52064	Aachen

<u>ISBN</u>	<u>Filialbezeichnung</u>	<u>Anzahl</u>
3110443759	Pontfiliale	10
3110443759	Innenstadtfiliale	18
3897215675	Innenstadtfiliale	5
3527707212	Pontfiliale	9
3110443759	Lager	12

Aufgabe 2 – Algorithmus Aufgabe, 60 Punktea) Datenmodell:

Ein mögliches Datenmodell wäre es, die beiden ganzzahligen Koordinaten der jeweiligen Punkte in einer Klasse Punkt und die durch den Roboter angefahrenen Punkte, an denen er die Richtung ändert, in einer verketteten Liste von Elementen des Typs Punkt zu speichern. Entsprechende Deklaration:

```
public class Punkt {  
    private int x;  
    private int y;  
  
    /*  
     * Getter/Setter  
     */  
    public int getX() {  
        return x;  
    }  
    public void setX(int x) {  
        this.x = x;  
    }  
    public int getY() {  
        return y;  
    }  
    public void setY(int y) {  
        this.y = y;  
    }  
  
    /*  
     * Fuer Ausgabe des Punktes  
     */  
    public String toString(){  
        return String.format("(%d/%d)", x, y);  
    }  
}
```

Für Liste der besuchten Punkte: List<Punkt> punktListe = new LinkedList<>();

b) Methode zur Bestimmung eines beliebigen Weges zwischen zwei Punkten:

```
static List<Punkt> verbindePunkte(Punkt p1, Punkt p2){
    List<Punkt> punktListe = new LinkedList<>();
    Punkt aktuellerPunkt;

    Punkt neuerPunkt;
    punktListe.add(p1);
    aktuellerPunkt = p1;
    neuerPunkt = new Punkt();

    //Falls x-Koordinate von p1 nicht auf dem Raster
    //Zwischenpunkt einlegen
    if(aktuellerPunkt.getX()%6 != 0){
        neuerPunkt.setX(aktuellerPunkt.getX()/6 * 6);
        neuerPunkt.setY(aktuellerPunkt.getY());
        punktListe.add(neuerPunkt);
        aktuellerPunkt = neuerPunkt;
        neuerPunkt = new Punkt();
    }

    //x-Koordinate ist auf dem Raster, nach unten gehen
    neuerPunkt.setX(aktuellerPunkt.getX());
    neuerPunkt.setY(0);
    punktListe.add(neuerPunkt);
    aktuellerPunkt = neuerPunkt;
    neuerPunkt = new Punkt();

    //Falls x-Koordinate von p2 nicht auf dem Raster
    //Zwischenpunkt einlegen
    if(p2.getX()%6 != 0){
        neuerPunkt.setX(p2.getX()/6 * 6);
        neuerPunkt.setY(0);
        punktListe.add(neuerPunkt);
        aktuellerPunkt = neuerPunkt;
        neuerPunkt = new Punkt();
        //Jetzt hochfahren auf Hoehe von p2
        neuerPunkt.setX(aktuellerPunkt.getX());
        neuerPunkt.setY(p2.getY());
        punktListe.add(neuerPunkt);
    }
}
```

```
 }else{
    //x-Koordinate von p2 auf dem Raster,
    //gleich Hochfahren auf hoehe von p2
    neuerPunkt.setX(p2.getX());
    neuerPunkt.setY(0);
    punktListe.add(neuerPunkt);
}

punktListe.add(p2);

return punktListe;
}
```

Aufgabe 3 – Informatikaufgabe, 20 Punkte

Die drei hier genannten Speichermedien Festplatte, Cache und Arbeitsspeicher zeichnen sich u.a. durch verschiedene Speicherkapazitäten und Zugriffsgeschwindigkeiten aus. Caches sind hierbei am kleinsten und schnellsten, langsamer, aber größer ist der Arbeitsspeicher, wohingegen die Festplatten am größten und langsamsten sind. Es gilt also: Je geringer die Zugriffszeiten (was gut ist), desto geringer auch die Kapazität (was schlecht ist).

Möchte man also die Laufzeit seines Programmes reduzieren, sollte man darauf achten, dass man möglichst immer auf die Daten im Cache/Arbeitsspeicher zugreift und sehr selten von Festplatte nachlädt. Bei Verwendung eines Datums im Speicher, werden dessen Inhalt und auch die drumherum befindlichen Daten in den Cache gelegt, da man davon ausgeht, dass diese auch mit hoher Wahrscheinlichkeit bald benötigt werden. Da wie oben allerdings angeführt, der Cache nicht sehr viele Daten vorhalten kann, ist es sinnvoll, häufig verwendete und logisch benachbarte Daten schnell hintereinander abzurufen.