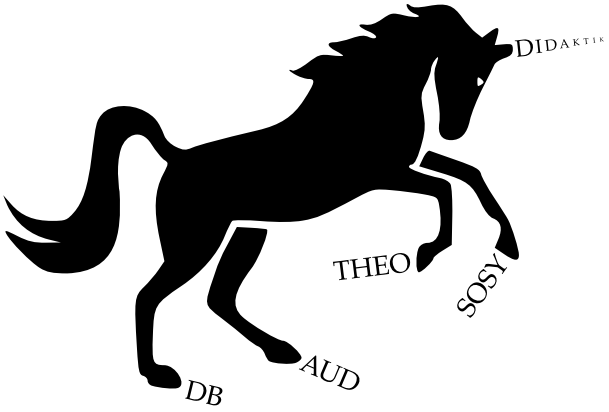
Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

Fach Informatik




Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

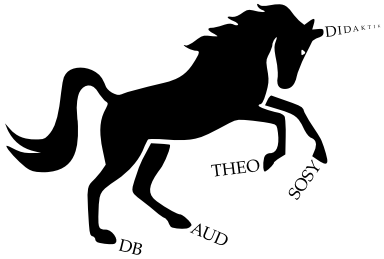
# Frühjahr 2021

66116

Datenbanksysteme / Softwaretechnologie (vertieft)

Aufgabenstellungen mit Lösungsvorschlägen

# 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 [private, package-private und protected]

(a) Definieren Sie die Bedeutung der Sichtbarkeiten *private*, *package-private* und *protected* von Feldern in Java-Klassen. Erklären Sie diese kurz (je ein Satz).

**private** Das Feld ist innerhalb einer Klasse zugreifbar.

**package-private** Das Feld ist innerhalb der Klassen eines Packets zugreifbar.

**protected** Das Feld ist in allen Subklassen und in der eigenen Klasse zugreifbar.

(b) Benennen Sie jeweils einen Grund für den Einsatz der Sichtbarkeiten *private*, *package-private* und *protected* von Feldern in Java-Klassen.

**private** Datenkapselung, Verbergen der internen Implementation. So kann die interne Implementation geändert werden, ohne dass die öffentlichen Schnittstellen sich ändern.

**package-private** Um die Felder zwar innerhalb deines Packets für alle Klassen zugänglich zu machen aber nicht außerhalb, z. B.in einem größeren Projekt.

**protected** Wenn eine Klasse vererbt werden soll, z. B.abstrakte Klassen.

### Aufgabe 2 [Verstoß gegen die Prinzipien guter objektorientierter Programmierung]

Lesen Sie die folgenden Beispielcodes gründlich. Identifizieren Sie für jeden Beispielcode den jeweiligen wesentlichen Verstoß gegen die Prinzipien guter objektorientierter Programmierung. Benennen und erklären Sie jeweils den Verstoß (Fehler) in einem Satz und erläutern Sie für jeden Beispielcode, welche Probleme aus dem jeweiligen Fehler resultieren können, ebenfalls in einem Satz.

(a)
```
class Rectangle {
  private int width;
  private int length;

  Rectangle(int w, int l) {
    width = w;
    length = l;
  }

  public int getWidth() {
    return this.width;
  }

  public int getLength() {
    return this.length;
  }
```

```
}

class RectangleDemo {
  public static void main(String args[]) {
    Rectangle rectangle1 = new Rectangle(10, 20);
    Rectangle rectangle2 = new Rectangle(3, 90);
    Rectangle example = new Rectangle(1, 2);
    int area;

    // Compute area of first box
    area = rectangle1.getWidth() * rectangle1.getLength();
    System.out.println("Area is " + area);

    // Compute area for second box
    area = rectangle2.getWidth() * rectangle2.getLength();
    System.out.println("Area is " + area);

    // Compute area for third box
    area = example.getWidth() * example.getLength();
    System.out.println("Area is " + area);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Rectangle.java`

Es sollte die Methode computeArea in der Klasse Rectangel implementiert werden.

(b)
```
public class CalculateSpeed {
  private double kilometers;
  private double minutes;

  public CalculateSpeed(double k, double m) {
    this.kilometers = k;
    this.minutes = m;

  }

  // Display the speed
  void speed() {
    double speed;
    speed = kilometers / (minutes / 60);
    System.out.println("A car traveling " + kilometers + " kilometers in " + minutes +
    ↪  " minutes travels at " + speed
        + " kilometers per hour");
  }

  public static void main(String args[]) {
    CalculateSpeed car = new CalculateSpeed(110.0, 120.0);

    // Display car speed
    car.speed();

    // Display bicycle speed
```

```
    double speed;

    speed = 20.0 / (80.0 / 60);
    // So steht es in der Angabe
    // System.out.println("A bicycle traveling " + kilometers + " kilometers in " +
    ↪  minutes + " minutes travels at "
    // + speed + " kilometers per hour");
    // Ohne Fehler:
    System.out.println("A bicycle traveling " + car.kilometers + " kilometers in " +
    ↪  car.minutes + " minutes travels at "
        + speed + " kilometers per hour");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/CalculateSpeed.java`

> Klassen sollten nach Objekten modelliert werden und nicht nach Tätigkeiten (berechne Geschwindigkeit) Besser wäre der Name SpeedCalculator gewesen. Außerdem sind die beiden Attribute kilometer und miutes in der Main Methode so nicht ansprechbar, weil sie nicht statisch sind.

(c)

```
class Stack {

  int stck[] = new int[3];
  public int top;

  // Initialize top of stack
  Stack() {
    top = -1;
  }

  // Push an item on the stack
  void push(int item) {
    if (top == 2) {
      System.out.println("Stack is full.");
    } else {
      stck[++top] = item;
    }
  }

  // Pop an item from the stack
  int pop() {
    if (top < 0) {
      throw new IllegalStateException("Stack is empty.");
    } else {
      return stck[top--];
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Stack.java`

Das Feld stck sollte private sein. So wird die interne Implemtation verborgen.

### Aufgabe 3 [Elementtypen UML-Klassendiagramm]

Wählen Sie bis zu fünf unterschiedliche Elementtypen aus folgendem Diagramm aus und benennen Sie diese Elemente und ihre syntaktische (nicht semantische) Bedeutung.

**Klasse** ConcreteCreator

**Interface** Produkt

**Abstrakte Klasse** Creator

**Kommentar** return new ConcreteProdukt()

### Aufgabe 4 [Klasse Box]

(a) Schreiben Sie ein Programm in einer objektorientierten Programmiersprache Ihrer Wahl, das den folgenden Anweisungen entspricht.

(i) Es gibt eine Klasse mit dem Namen JBox.

(ii) Alle Zahlen sind Fließkommazahlen.

(iii) JBox wird mit einem Argument instanziiert, dessen Wert einer Variable namens Jlength zugewiesen wird.

```java
public Box(double length) {
  this.length = length;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java`

(iv) JBox hat eine Methode ohne Argumente namens Jsize, welche den Wert von Jlength zurückgibt.

```java
public double size() {
  return length;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java`

(v) Eine weitere Methode namens Jsize hat genau ein Argument namens Jwidth. Diese zweite Methode namens Jsize gibt das Produkt aus Jwidth und Jlength zurück. Eine weitere Methode namens Jsize hat genau zwei Argumente, nämlich eine Zahl Jnum und einen Faktor Jf. Es wird Jlength minus das Produkt aus Jnum und Jf zurückgegeben.

```java
  public double size(double width) {
    return this.length * width;
  }

  public double size(double num, double f) {
    return this.length - num * f;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java`

(vi) Schreiben Sie eine `main`-Methode, die eine Box namens `example` mit einer Länge von 15 anlegt.

(vii) Führen Sie die Methode `size` in der `main`-Methode wie unten angegeben drei Mal aus.

(viii) Speichern Sie hierbei das Ergebnis jeweils in einer Variable `mysize`. Geben Sie das Ergebnis jeweils in einer eigenen Zeile des Ausgabemediums `System.out` aus.

- Mit keinen Argumenten
- Mit dem Argument `10`
- Mit den beiden Argumenten `5` und `2`

```java
  public static void main(String[] args) {
    Box example = new Box(15);

    double mysize = example.size();
    System.out.println(mysize);

    mysize = example.size(10);
    System.out.println(mysize);

    mysize = example.size(5, 2);
    System.out.println(mysize);
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java`

```java
public class Box {
  double length;

  public Box(double length) {
    this.length = length;
  }

  public double size() {
    return length;
  }

  public double size(double width) {
    return this.length * width;
  }

  public double size(double num, double f) {
```

```
    return this.length - num * f;
  }

  public static void main(String[] args) {
    Box example = new Box(15);

    double mysize = example.size();
    System.out.println(mysize);

    mysize = example.size(10);
    System.out.println(mysize);

    mysize = example.size(5, 2);
    System.out.println(mysize);
  }

}
```

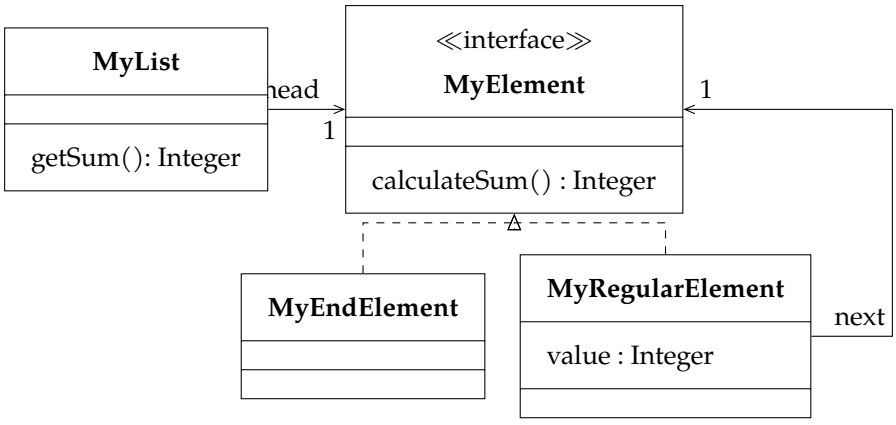
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java`

(b) Notieren Sie die Ausgabe der `main`-Methode.

- 15
- 150
- 5

### Aufgabe 5 [MyList Kompositium]

Die folgende Abbildung stellt den Entwurf der Implementierung einer verketteten Liste dar, welche Integer-Werte als Elemente enthalten kann.




Die Klassse `MyList` stellt die Methode `getSum()` zur Verfügung, welche die Summe über alle in einer Liste befindlichen Elemente berechnet. Ein Ausschnitt der Implementierung sieht folgendermaßen aus:

```
public class MyList {
  private MyElement head;
```

```
  public MyList() {
    this.head = new MyEndElement();
  }

  public int getSum() {
    // ..
  }
}
```

Gehen Sie im Folgenden davon aus, dass bereits Methoden existieren, welche Elemente in die Liste einfügen können.

(a) Implementieren Sie in einer objektorientierten Programmiersprache Ihrer Wahl, z. B.Java, die Methode `calculateSum()` der Klassen `MyEndElement` und `MyRegularElement`, so dass rekursiv die Summe der Elemente der Liste berechnet wird.

Als Abbruchbedingung darf hierbei nicht das Feld `MyRegluarElement.next` auf den Wert `null` überprüft werden.

Hinweis: Gehen Sie davon aus, die Implementierung von `MyList` garantiert, dass `MyRegluarElement.next` niemals den Wert `null` annimmt, sondern das letzte hinzugefügte `MyRegularElement` auf eine Instanz der Klasse `MyEndElement` verweist. Es gibt immer nur eine Instanz der Klasse `MyEndElement` in einer Liste.

Hinweis: Achten Sie auf die Angabe einer korrekten Methodensignatur.

```
int calculateSum() {
  return value + next.calculateSum();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/my_list/MyElement.java`

```
int calculateSum() {
  return 0;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/my_list/MyEndElement.java`

(b) Nennen Sie den Namen des Entwurfsmusters, auf welchem das oben gegebene Klassendiagramm basiert, und ordnen Sie dieses in eine der Kategorien von Entwurfsmustern ein.

Hinweis: Es genügt die Angabe eines Musters, falls Sie mehrere Muster identifizieren sollten.

Kompositium (Strukturmuster)

### Aufgabe 6 [Wissen Erbauer]

(a) Erläutern Sie den Zweck (Intent) des Erzeugungsmusters Erbauer in max. drei Sätzen, ohne dabei auf die technische Umsetzung einzugehen.

Die Erzeugung komplexer Objekte wird vereinfacht, indem der Konstruktionsprozess in eine spezielle Klasse verlagert wird. Er wird so von der Repräsentation getrennt und kann sehr unterschiedliche Repräsentationen zurückliefern.

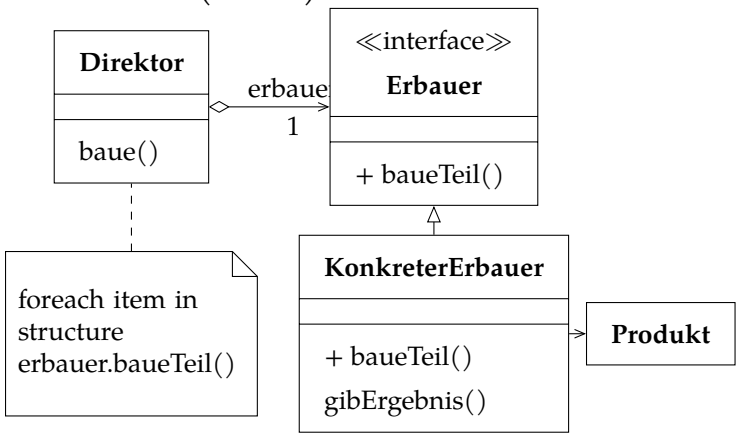
(b) Erklären Sie, wie das Erzeugungsmuster Erbauer umgesetzt werden kann (Implementierung). Die Angabe von Code ist hierbei NICHT notwendig!

(c) Nennen Sie jeweils einen Vor- und einen Nachteil des Erzeugungsmusters Erbauer im Vergleich zu einer Implementierung ohne dieses Muster.

**Vorteil** Die Implementierungen der Konstruktion und der Repräsentationen werden isoliert. Die Erbauer verstecken ihre interne Repräsentation vor dem Direktor.

**Nachteil** Es besteht eine enge Kopplung zwischen Produkt, konkretem Erbauer und den am Konstruktionsprozess beteiligten Klassen.

**Exkurs: Erbauer (Builder)**




**Erbauer** Der Erbauer spezifiziert eine abstrakte Schnittstelle zur Erzeugung der Teile eines komplexen Objektes.

**KonkreterErbauer** Der konkrete Erbauer erzeugt die Teile des komplexen Objekts durch Implementierung der Schnittstelle. Außerdem definiert und verwaltet er die von ihm erzeugte Repräsentation des Produkts. Er bietet auch eine Schnittstelle zum Auslesen des Produkts.

**Direktor** Der Direktor konstruiert ein komplexes Objekt unter Verwendung der Schnittstelle des Erbauers. Der Direktor arbeitet eng mit dem Erbauer zusammen: Er weiß, welche Baureihenfolge der Erbauer verträgt oder benötigt. Der Direktor entkoppelt somit den Konstruktionsablauf vom Klienten.

**Produkt** Das Produkt repräsentiert das zu konstruierende komplexe Objekt.

## Aufgabe [MyParser]

Lesen Sie die folgenden alternativen Codestücke.

(a)
```
public class MyParser {
  private InputStream input;

  public MyParser(String filePath) {

  }
}
```

(b)
```
public class MyParser {
  private InputStream input;

  public MyParser(InputStream stream) {
  }
}
```

Beide Codestücke zeigen die Initialisierung einer Klasse namens `MyParser`. Das zweite Codestück nutzt jedoch hierfür eine Technik namens Abhängigkeits-Injektion (Dependency Injection).

(a) Erklären Sie den Unterschied zwischen beiden Initialisierungen. Hinweis: Sie können diese Aufgabe auch lösen, falls Sie die Technik nicht kennen.

> Die Abhängigkeit von einer Instanz der Klasse InputStream wird erst bei der Initialisierung des Objekt übergeben.

(b) Benennen Sie einen Vorteil dieser Technik.

> Die Kopplung zwischen einer Klasse und ihrer Abhängigkeit wird verringert.

### Aufgabe 8 [Client-Server-Modell]

Das Client-Server-Modell ist ein Architekturmuster. Nennen Sie zwei Vorteile einer nach diesem Muster gestalteten Architektur.

(a) Einfache Integration weiterer Clients

(b) Prinzipiell uneingeschränkte Anzahl der Clients [a]

(c) Es muss nur ein Server gewartet werden. Dies gilt z. B.für Updates, die einmalig und zentral auf dem Server durchgeführt werden und danach für alle Clients verfügbar sind. [b]

[a] https://www.karteikarte.com/card/164928/vorteile-und-nachteile-des-client-server-modells
[b] https://www.eoda.de/wissen/blog/client-server-architekturen-performance-und-agilitaet-fuer-data-s

### Aufgabe 9 [Client-Server-Technologien]

Betrachten Sie die folgende Liste von Technologien:

- Nodejs

- PHP
- CSS
- AJAX
- Python
- Java

Welche dieser Technologien laufen in einem Client-Server-System üblicherweise auf der Seite des Klienten und welche auf der Seite des Servers? Nehmen Sie hierzu an, dass der Client ein Browser ist.

Übertragen Sie die im Folgenden gegebene Tabelle in Ihren Bearbeitungsbogen und ordnen Sie die aufgelisteten Technologien anhand der Buchstaben in die Tabelle ein. Fortsetzung nächste Seite!

Hinweis: Mehrfachzuordnungen sind möglich.

| Client-seitige Technologien | Server-seitige Technologien |
|---|---|
| CSS | Nodejs |
| AJAX | PHP |
| Python | Python |
| Java | Java |

**Aufgabe 10 [AJAX]**

(a) Was bedeutet die Abkürzung AJAX?

Asynchronous JavaScript and XML

(b) Erklären Sie in max. drei Sätzen die grundlegende Funktion von AJAX.

Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden. [a]

[a] `https://de.wikipedia.org/wiki/Ajax_(Programmierung)`

**Aufgabe 11 [HTTP]**

(a) Was ist das Hypertext Transfer Protocol (HTTP) und wozu dient es?

Das Hypertext Transfer Protocol ist ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz. Es wird hauptsächlich eingesetzt, um Webseiten (Hypertext-Dokumente) aus dem World Wide Web (WWW) in einen Webbrowser zu laden. Es ist jedoch nicht prinzipiell darauf be-

schränkt und auch als allgemeines Dateiübertragungsprotokoll sehr verbreitet. [a]

[a] https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol

(b) Betrachten Sie die folgende Zeile Text. Um welche Art von Text handelt es sich?

`https://developer.mozilla.org/en-US/search?q=client+servertoverview`

Es handelt sich um eine HTTP-URL (Uniform Resource Locator). Die URL lokalisiert eine Ressource, beispielsweise eine Webseite, über die zu verwendende Zugriffsmethode (zum Beispiel das verwendete Netzwerkprotokoll wie HTTP oder FTP) und den Ort (engl. location) der Ressource in Computernetzwerken. [a]

[a] https://de.wikipedia.org/wiki/Uniform_Resource_Locator

(c) Was sind die vier wesentlichen Bestandteile des Texts aus der vorigen Teilaufgabe?

**Schema** `https://`

**Host** `developer.mozilla.org`

**Pfad** `/en-US/search`

**Query** `?q=client+servertoverview`

[a]

[a] https://de.wikipedia.org/wiki/Uniform_Resource_Locator

**Aufgabe 12 [Richtig-Falsch]**

Es gibt Softwaresysteme, welche auf peer-to-peer (P2P) Kommunikation basieren und eine entsprechende Architektur aufweisen.

(a) Bewerten Sie die folgenden Aussagen als entweder richtig oder falsch.

(i) Mithilfe des Befehls "lookup" können Peers sich gegenseitig identifizieren.

richtig

(ii) In einem P2P-System, wie auch bei Client-Server, sind alle Netzwerkteilnehmer gleichberechtigt.

falsch. Im Client-Servermodell sind nicht alle Netzwerkteilnehmer gleichberechtig. Der Server hat mehr Privilegien wie der Client.

(iii) Alle P2P-Systeme funktionieren grundsätzlich ohne einen zentralen Verwaltungs-Peer.

falsch. Es gibt zentralisierte P2P-Systeme (Beispiel: Napster), welche einen zentralen Server zur Verwaltung benötigen, um zu funktionieren. [a]

[a] https://de.wikipedia.org/wiki/Peer-to-Peer

(iv) P2P kann auch für eine Rechner-Rechner-Verbindung stehen.

richtig

(v) Es gibt strukturierte und unstrukturierte P2P-Systeme. In unstrukturierten P2P-Systemen wird zum Auffinden von Peers eine verteilte Hashtabelle verwendet (DHT).

richtig

(vi) In einem P2P-System sind theoretisch alle Peers gleichberechtigt, praktisch gibt es jedoch leistungsabhängige Gruppierungen.

richtig

(vii) Ein Peer kann sowohl ein Client wie auch ein Server für einen anderen Peer sein.

richtig

(b) Wählen Sie zwei falsche Aussagen aus der vorherigen Tabelle aus und berichtigen Sie diese in jeweils einem Satz.

Sie oben.

## Teilaufgabe Nr. 2

### Aufgabe 1 [Vermischte Fragen]

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort.

(a) Erläutern Sie die Begriffe Kardinalität und Partizipität. Welche Arten von Partizipität gibt es in der ER-Modellierung? Nennen und erklären Sie diese kurz.

**Kardinalitäten** Für die noch genauere Darstellung der Beziehungen im ER-Modell verwendet man Kardinalitäten (auch Grad der Beziehungen genannt). Diese geben an wie viele Entitätsinstanzen mit wie vielen Entitätsinstanzen einer anderen Entitätsinstanz in Beziehung stehen. [a]

**Partizipation** Die Partizipation eines Beziehungstyps (in einem Entity-Relationship-Modell) bestimmt, ob alle Entities eines beteiligten Entitätstyps an einer bestimmten Beziehung teilnehmen müssen. [b]

**totale Partizipation:** Wenn eine Beziehung Entität A und Entität B in Beziehung setzt, dann muss ein Eintrag in Entität A existieren, damit ein Eintrag in Entität B existiert und umgekehrt. Beide Entitäten müssen also an

der Relation teilnehmen. Eine Entitätsinstanz aus A kann also nicht ohne eine in-Beziehung-stehende Entitätsinstanz aus B existieren und umgekehrt.

**partielle Partizipation:** Wenn eine Beziehung Entität A mit Entität B in Beziehung setzt, dann muss kein Eintrag in Entität A existieren, damit ein Eintrag in Entität B existieren kann und umgekehrt. Die beiden Entitäten müssen also nicht an der Relation teilnehmen (enthalten sein). [c]

Die Kardinalität definiert, wie viele Entities eines Typs mit wie vielen Entities eines anderen Typs in Beziehung stehen können (siehe Schneider et al., S. 446)

Partizipität – ein anderer Begriff dafür ist Totalität – beschreibt den Grad der Teilnahmeverpflichtung zweier Entitäten an einer Beziehung. Sie kann partiell, total oder einseitig total sein.

a. Totale P.: Jede Entity A und Entity B besteht nur dann, wenn sie an dieser Beziehung teilnehmen.

b. Einseitige totale P.: Eine Entity A besteht nur dann, wenn sie an der Beziehung zu Entity B teilnimmt. Entitäten von B dagegen können, müssen aber nicht teilnehmen.

c. Partielle P.: Die Existenz der Entitäten ist unabhängig von der Teilnahme an dieser Beziehung. Die Teilnahme ist nicht verpflichtend.

[a] `https://usehardware.de/datenbanksysteme-iv-entity-relationship-modell-er-modell-datenbankda`
[b] `https://lehrbuch-wirtschaftsinformatik.org/glossar/kapitel03/Partizipation`
[c] `https://usehardware.de/datenbanksysteme-iv-entity-relationship-modell-er-modell-datenbankda`

(b) Mit welchen beiden Befehlen kann eine Transaktion beendet werden? Nennen Sie diese und erklären Sie den Unterschied.

Für den Abschluss einer Transaktion gibt es 2 Möglichkeiten:

- Den erfolgreichen Abschluss mit `commit`.
- Den erfolglosen Abschluss mit `abort`

(c) Erläutern Sie den Unterschied zwischen einer kurzen und einer langen Sperre.

**lange Sperren:** LOCKs werden erst nach dem commit zurückgegeben ($\rightarrow$ striktes 2PL)

**kurze Sperren:** LOCKs werden direkt nach dem schreiben/lesen zurückzugeben [a]

[a] `https://www.dbs.ifi.lmu.de/Lehre/DBSII/SS2015/vorlesung/DBS2-03-Synchronisation.pdf`

(d) Stellen Sie außerdem die Kompatibilitätsmatrix zur Umsetzung des ACID-Prinzips mit den richtigen Werten dar. S stehe dabei für eine Lese- und X für eine Schreibsperre.

Kompatibilitätsmatrix zur Umsetzung des ACID-Prinzips (Atomicity, Consistency, Isolation, Durability)

| | NL (no lock) | S (Lesesperre) | X (Schreibsperre) |
|---|---|---|---|
| S (Lesesperre) | ja | ja | nein |
| X (Schreibsperre) | ja | nein | nein |

(e) Nennen und erklären Sie kurz die Armstrong-Axiome. Sind diese vollständig und korrekt?

**Reflexivität:** Eine Menge von Attributen bestimmt eindeutig die Werte einer Teilmenge dieser Attribute (triviale Abhängigkeit), das heißt, $\beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$ .

**Verstärkung:** Gilt $\alpha \rightarrow \beta$, so gilt auch $\alpha\gamma \rightarrow \beta\gamma$ für jede Menge von Attributen $\gamma$ der Relation.

**Transitivität:** Gilt $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$, so gilt auch $\alpha \rightarrow \gamma$.

Die Armstrong-Axiome sind korrekt und vollständig: Diese Regeln sind gültig (korrekt) und alle anderen gültigen Regeln können von diesen Regeln abgeleitet werden (vollständig). [a]

[a] https://dbresearch.uni-salzburg.at/teaching/2019ss/db1/db1_06-handout-1x1.pdf

(f) Was versteht man unter einem (Daten-)Katalog (Data Dictionary) und was enthält dieser (es genügt eine Auswahl zu nennen)?

Bei einer relationalen Datenbank ist ein Datenkatalog eine Menge von Tabellen und Ansichten, die bei Abfragen nur gelesen werden. Das Data-Dictionary ist wie eine Datenbank aufgebaut, enthält aber nicht Anwendungsdaten, sondern Metadaten, das heißt Daten, welche die Struktur der Anwendungsdaten beschreiben (und nicht den Inhalt selbst).

Zu einem Data-Dictionary zur physischen Datenmodellierung gehören genaue Angaben zu:

- Tabellen und Datenfeldern
- Primär- und Fremdschlüsselbeziehungen
- Integritätsbedingungen, z. B.Prüfinformationen

[a]

[a] https://de.wikipedia.org/wiki/Data-Dictionary

(g) Erklären Sie das konservative und das strikte Zwei-Phasen-Sperrprotokoll.

**Konservatives 2-Phasen-Sperrprotokoll** Das konservative 2-Phasen-Sperrprotokoll (Preclaiming), bei welchem zu Beginn der Transaktion alle benötigten Sperren auf einmal gesetzt werden. Dies verhindert in jedem Fall Deadlocks, führt aber auch zu einem hohen Verlust an Parallelität, da eine Transaktion ihre erste Operation erst dann ausführen kann, wenn sie alle Sperren erhalten hat.

**Striktes 2-Phasen-Sperrprotokoll** Das strikte 2-Phasen-Sperrprotokoll, bei welchem alle gesetzten Write-Locks erst am Ende der Transaktion (nach der letzten Operation) freigegeben werden. Dieses Vorgehen verhindert den Schneeballeffekt, also das kaskadierende Zurücksetzen von sich gegenseitig beeinflussenden Transaktionen. Der Nachteil ist, dass Sperren häufig viel länger gehalten werden als nötig und sich somit die Wartezeit von blockierten Transaktionen verlängert. Die Read-Locks werden entsprechend dem Standard-2PL-Verfahren entfernt.

(h) Erklären Sie die Begriffe „Steal/NoSteal“ und „Force/NoForce“ im Kontext der Systempufferverwaltung eines DBS.

**No-Steal** Schmutzige Seiten dürfen nicht aus dem Puffer entfernt und in die Datenbank übertragen werden, solange die Transaktion noch aktiv ist. Die Datenbank enthält keine Änderungen nicht-erfolgreicher Transaktionen. Eine UNDO-Recovery ist nicht erforderlich. langen Änderungs-Transaktionen können zu Problemen führen, da große Teile des Puffers blockiert werden

**Steal** Schmutzige Seiten dürfen jederzeit ersetzt und in die Datenbank eingebracht werden. Die Datenbank kann unbestätigte Änderungen enthalten. Eine UNDO-Recovery ist erforderlich. Es handelt sich um eine effektivere Puffernutzung bei langen Transaktionen mit vielen Änderungen.

**Force** Alle geänderten Seiten werden spätestens bei EOT (vor COMMIT) in die Datenbank geschrieben. Bei einem Systemfehler ist keine REDO-Recovery erforderlich. Die Force-Strategie benötigt einen hohen I/O-Aufwand, da Änderungen jeder Transaktion einzeln geschrieben werden. Die Vielzahl an Schreibvorgängen führt zu schlechteren Antwortzeiten, länger gehaltenen Sperren und damit zu mehr Sperrkonflikten. Große Datenbank-Puffer werden schlecht genutzt.

**No-Force** Änderungen können auch erst nach dem COMMIT in die Datenbank geschrieben werden. Die Änderungen durch mehrere Transaktionen werden „gesammelt“. Beim COMMIT werden lediglich REDO-Informationen in die Log-Datei geschrieben. Bei einem Systemfehler ist eine REDO-Recovery erforderlich. Die Änderungen auf einer Seite über mehrere Transaktionen hinweg können gesammelt werden.

**Aufgabe 2 [Automobilproduktion]**

Erstellen Sie ein möglichst einfaches ER-Schema, das alle gegebenen Informationen enthält. Attribute von Entitäten und Beziehungen sind anzugeben, Schlüsselattribute durch Unter-

streichen zu kennzeichnen. Verwenden Sie für die Angabe der Kardinalitäten von Beziehungen die Min-Max-Notation. Führen Sie Surrogatschlüssel (künstlich definierte Schlüssel) nur dann ein, wenn es nötig ist, und modellieren Sie nur die im Text vorkommenden Elemente.

Zunächst gibt es **Autos**, die einen eindeutigen *Namen* (`AName`), 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* (`ZName`) sowie seine *E-Mail-Adresse* (`EMailAdresse`) 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 $m^2$).

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 davon 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.

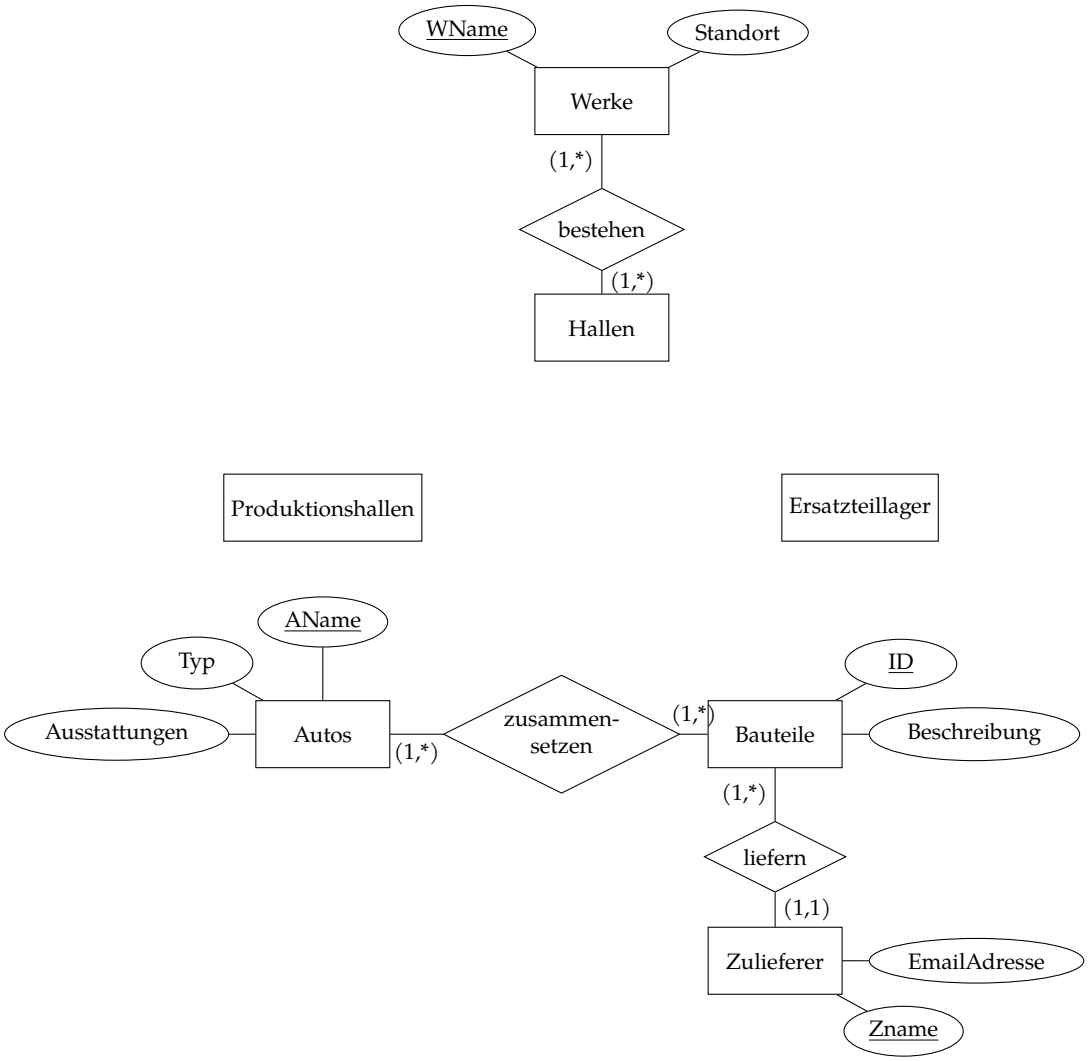
Mitarbeiter werden unter anderem in **Reinigungskräfte**, **Werksarbeiter** und **Ingenieure** unterteilt.

Zu den Ingenieuren 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.

☐ **E: Autos**
○ *A: Namen*
○ *A: Typ*
○ *A: Ausstattungen*
☐ **E: Bauteilen**
◇ R: zusammengesetzt
○ *A: ID*
○ *A: Beschreibung*
☐ **E: Zulieferer**
○ *A: Name*
○ *A: E-Mail-Adresse*
☐ **E: Werke**
○ *A: Namen*
○ *A: Standort*
◇ R: besteht
☐ **E: Hallen**
○ *A: nummeriert*
○ *A: Größe*
☐ **E: Produktionshallen**
☐ **E: Ersatzteillager**
◇ R: hergestellt
◇ R: festgehalten
☐ **E: Mitarbeiter**
○ *A: ID*
○ *A: Vor-*
○ *A: Nachname*
○ *A: Adresse (Straße, PLZ, Ort)*
○ *A: Gehalt*
○ *A: Geschlecht*
☐ **E: Reinigungskräfte**
☐ **E: Werksarbeiter**
☐ **E: Ingenieure**
○ *A: Hoch-*




**Aufgabe 3 [Relationen R1 und R2]**

(a) Gegeben seien die folgenden beiden Relationen R1 und R2:

**R1**

| P | Q | S |
|---|---|---|
| 10 | einfach | 5 |
| 15 | b | 8 |
| 13 | einfach | 6 |

**R2**

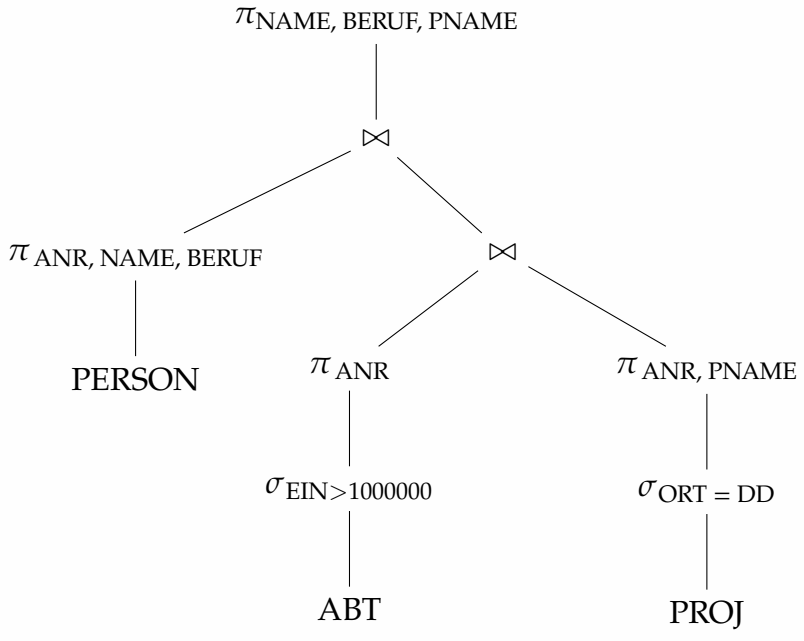
| A | B | C |
|---|---|---|
| 10 | b | 6 |
| 13 | c | 3 |
| 10 | b | 5 |

Geben Sie das Ergebnis des folgenden relationalen Ausdrucks an:

$$R_1 \bowtie_{R_1.P=R_2.A} R_2 \text{ (Equi-Join)}$$

| P | Q | S | A | B | C |
|---|---|---|---|---|---|
| 10 | einfach | 5 | 10 | b | 6 |
| 10 | einfach | 5 | 10 | b | 5 |
| 13 | einfach | 6 | 13 | c | 3 |

(b) Zeichnen Sie den Operatorbaum zu folgender Abfrage in relationaler Algebra:




(c) Ist der linke (bzw. rechte) Verbundoperator (Left- bzw. Right-Outer Join) assoziativ? Falls ja, beweisen Sie die Aussage, falls nein, geben Sie ein Gegenbeispiel an.

Nein. Beleg durch Gegenbeispiel:

**R1**

| A | B |
|---|---|
| 1 | 2 |
| 2 | 15 |

**R2**

| A | C |
|---|---|
| 1 | 35 |
| 2 | 12 |
| 13 | 5 |

**R3**

| B | C |
|---|---|
| 2 | 35 |
| 100 | 35 |

**(R1 LEFT OUTER JOIN R2) LEFT OUTER JOIN R3**

| R1.A | R1.B | R2.A | R2.C | R3.B | R3.C |
|---|---|---|---|---|---|
| 1 | 2 | 1 | 35 | 2 | 35 |
| 2 | 15 | 2 | 12 | NULL | NULL |

**R1 LEFT OUTER JOIN (R2 LEFT OUTER JOIN R3)**

| R1.A | R1.B | R2.A | R2.C | R3.B | R3.C |
|---|---|---|---|---|---|
| 1 | 2 | 1 | 35 | 2 | 35 |
| 2 | 15 | NULL | NULL | NULL | NULL |

(Nur wenn beide Tabellen leer sind, wären auch die Outer Joins assoziativ).

**Aufgabe 4 [Normalisierung]**

Gegeben ist das folgende Relationenschema in erster Normalform, bestehend aus zwei Relationen:

```
Relation1(A, B, C, D, E)
Relation2(F, G, H, A, E)
```

In diesem Schema gelten die folgenden funktionalen Abhängigkeiten:

$$\text{FA} = \Big\{ \begin{aligned} \{A, B\} &\to \{C\}, \\ \{A, B, C\} &\to \{E\}, \\ \{A\} &\to \{D\}, \\ \{F, G\} &\to \{H, A\}, \\ \{G, H\} &\to \{E\}, \end{aligned} \Big\}$$

(a) Nennen Sie die Bedingungen, damit ein Schema in erster Normalform ist.

> Ein Schema ist in erster Normalform, wenn es ausschließlich atomare Attributwerte aufweist.

(b) Überprüfen Sie, ob das Schema in zweiter Normalform ist.

> Eine Relation ist in 2NF, wenn sie in 1NF ist und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist. Der Schlüsselkandidat ist (A, B) in Relation 1 sowie (F, G) in Relation 2.
>
> Das Nichtschlüsselartribut D in Relation 1 ist nicht voll funktional abhängig von (A, B), sondern nur von A. Somit ist das Schema nicht in 2NF. Alle anderen Nichtschlüsselattribute sind voll funktional abhängig.

(c) Wenden Sie den Synthesealgorithmus an, um das Schema in ein Schema in dritter Normalform zu überführen.

> (i) **Kanonische Überdeckung**
>
> — *Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden.* ———
>
> i. **Linksreduktion**
>
> — *Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob $A$ überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$.* ———
>
> $$\text{FA} = \Big\{ \begin{aligned} \{A, B\} &\to \{C\}, \\ \{A, B\} &\to \{E\}, \\ \{A\} &\to \{D\}, \\ \{F, G\} &\to \{H, A\}, \\ \{G, H\} &\to \{E\}, \end{aligned} \Big\}$$
>
> ii. **Rechtsreduktion**
>
> — *Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist $B$ auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt.* ———
>
> nichts zu tun

iii. **Löschen leerer Klauseln**
— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* ——

nichts zu tun

iv. **Vereinigung**
— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* ——

$$\text{FA} = \Big\{ \begin{array}{r} \{ A, B \} \to \{ C, E \}, \\ \{ A \} \to \{ D \}, \\ \{ F, G \} \to \{ H, A \}, \\ \{ G, H \} \to \{ E \}, \end{array} \Big\}$$

(ii) **Relationsschemata formen**
— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.* ——

R1 (A, B, C, E) R2 (A, D) R3 (F, G, H, A) R4 (G, H, E)

(iii) **Schlüssel hinzufügen**
— *Falls eines der in Schritt 2. erzeugten Schemata $R_\alpha$ einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich $F_c$ enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* ——

R1 (A, B, C, E) R2 (A, D) R3 (F, G, H, A) R4 (G, H, E) R5 (B, F, G)
als Verbindung von R1 bis R4 (Attributhülle erhält alle Attribute, ist daher Schlüsselkandidat)

(iv) **Entfernung überflüssiger Teilschemata**
— *Eliminiere diejenigen Schemata $R_\alpha$, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* ——

nichts zu tun

(d) Sei nun das Relationenschema R(A,B,C,D) in erster Normalform gegeben. In R gelten die folgenden funktionalen Abhängigkeiten:

$$\text{FA} = \Big\{ \begin{array}{r} \{ A, B \} \to \{ D \}, \\ \{ B \} \to \{ C \}, \\ \{ C \} \to \{ B \}, \end{array} \Big\}$$

Welches ist die höchste Normalform, in der sich das Schema R befindet? Begründen Sie Ihre Entscheidung.

**Aufgabe 5 [Transaktionen T1 und T2]**

Gegeben sind die folgenden transaktionsähnlichen Abläufe. (Zunächst wird auf das Setzen von Sperren verzichtet.) Hierbei steht R(X) für ein Lesezugriff auf X und W(X) für einen Schreibzugriff auf X.

| T1 | T2 |
|---|---|
| R(A) | R(D) |
| A := A-10 | D := D-20 |
| W(A) | W(D) |
| R(C) | R(A) |
| R(B) | A := A+20 |
| B := B+10 | W(A) |
| W(B) | |

Betrachten Sie folgenden Schedule:

| T1 | T2 |
|---|---|
| R(A) | |
| | R(D) |
| | D := D-20 |
| | W(D) |
| | R(A) |
| | A := A+20 |
| | W(A) |
| A := A-10 | |
| W(A) | |
| R(C) | |
| R(B) | |
| B := B+10 | |
| W(B) | |

(a) Geben Sie die Werte von $A$, $B$, $C$ und $D$ nach Ablauf des Schedules an, wenn mit $A = 100$, $B = 200$, $C =$ true und $D = 150$ begonnen wird.

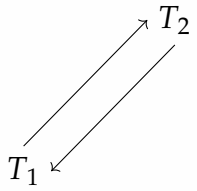
**A** 90 (A := A - 10 := 100 - 10) T2 schreibt 120 in A, was aber von T1 wiederüberschrieben wird.

**B** 210 (B wird nur in T1 gelesen, verändert und geschrieben)

**C** true (C wird nur in T1 gelesen)

**D** 130 (D wird nur in T2 gelesen, verändert und geschrieben)

(b) Geben Sie den Dependency-Graphen des Schedules an.




(c) Geben Sie alle auftretenden Konflikte an.

$R_1(A) < W_2(A)$ resultierende Kante $T_1 \rightarrow T_2$,

$R_2(A) < W_1(A)$ resultierende Kante $T_2 \rightarrow T_1$,

$W_2(A) < W_1(A)$ resultierende Kante $T_2 \rightarrow T_1$ (bereits vorhanden)

(d) Begründen Sie, ob der Schedule serialisierbar ist.

Nicht ohne den Einsatz der Lese- und Schreibsperren, denn der Dependency Graph enthält einen Zyklus, womit er nicht konfliktserialisierbar ist.

(e) Beschreiben Sie, wie die beiden Transaktionen mit LOCK Aktionen erweitert werden können, so dass nur noch serialisierbare Schedules ausgeführt werden können. Die Angabe eines konkreten Schedules ist nicht zwingend notwendig.

Hier führt die Verwendung des Zwei-Phasen-Sperrpotokolls zur gewünschten Serialisierbarkeit. (Es muss dabei weder die konservative, noch die strenge Variante verwendet werden, damit es funktioniert). T1 würde zu Beginn die und Lese- und Schreibsperre für A anfordern, den Wert verändern, zurückschreiben und anschließend die Sperren für A zurückgeben. Währendessen könnte T2 „ungestört“ die Schreib- und Lesesperren für D anfordern, D lesen, verändern und schreiben, und die Sperren zurückgeben. T2 bemüht sich nun um die Lesesperre für A, muss aber nun so lange warten, bis T1 die Schreibsperre zurückgegen hat. Dadurch kann man den Lost-Update-Fehler vermeiden und erhält allgemein einen serialisierbaren Schedule.

Beispiel für einen konkreten Schedule mit LOCKs (auch wenn nicht zwingend gefordert in der Aufgabenstellung):

| T1 | T2 |
|---|---|
| rLock(A) | |
| xLock(A) | |
| | rLock(D) |
| | xLock(D) |
| R1(A) | |
| | R2(D) |
| A := A-10 | |
| | D := D-20 |
| | W2(D) |
| | unLock(D) |
| | rLock(A) DELAY |
| W1(A) | |
| unLock(A) | |
| | R2(A) |
| | A := A+20 |
| | W2(A) |
| | unLock(A) |
| rLock(C) | |
| R1(D) | |
| unLock(C) | |
| | commit |
| rLock(B) | |
| xLock(B) | |
| R1(B) | |
| B := B+10 | |
| W1(B) | |
| unLock(B) | |
| commit | |

**Aufgabe 6 [Fußballweltmeisterschaft]**

Gegeben ist folgendes Relationenschema zur Verwaltung von Daten aus der Fußballweltmeisterschaft:

Die Tabelle Match wurde in Spiel umgenannt, da es sonst zu Konfliken mit der SQL-Syntax kommt, da match ein SQL Schlüsselwort ist.

Nation (Land, Kapitän, Trainer) Kapitän ist Fremdschlüssel zu Spieler_ID in Spieler.

Spiel (Spiel_ID, Ort, Datum, Team1, Team2, ToreTeam1, ToreTeam2) Team1 ist Fremdschlüssel zu Land in Nation. Team2 ist Fremdschlüssel zu Land in Nation.

Spieler (Spieler_ID, Name, Vorname, Wohnort, Land) Land ist Fremdschlüssel zu Land in Nation.

Platzverweise (Platzverweis_ID,Spiel_ID, Spieler_ID, Spielminute) Spiel_ID ist Fremdschlüsssel zu Spiel_ID in Spiel. Spieler_ID ist Fremdschlüssel zu Spieler_ID in Spieler.

Die Primärschlüssel der Relationen sind wie üblich durch Unterstreichen gekennzeichnet. Pro Ort und Datum findet jeweils nur ein Spiel statt.

Formulieren Sie folgende Abfragen in SQL. Vermeiden Sie nach Möglichkeit übermäßige Nutzung von Joins und Views.

(a) Ermitteln Sie die Anzahl der Platzverweise pro Spieler und geben Sie jeweils Name und Vorname des Spielers mit aus. Die Ausgabe soll nach der Anzahl der Platzverweise absteigend erfolgen.

```
SELECT COUNT(*) AS Anzahl, s.Name, s.Vorname
FROM Platzverweise p, Spieler s
WHERE p.Spieler_ID = s.Spieler_ID
GROUP BY s.Name, s.Vorname
ORDER BY Anzahl DESC;

 anzahl |    name    | vorname
--------+------------+---------
      2 | Matthäus   | Lothar
      1 | Bodden     | Olaf
      1 | Beckham    | David
      1 | Rizzitelli | Luca
      1 | Babel      | Markus
      1 | Häßler     | Thomas
(6 rows)
```

(b) Welches ist die maximale Anzahl an Toren, die eine Mannschaft insgesamt im Turnier erzielt hat? (Sie dürfen der Einfachheit halber annehmen, dass jede Mannschaft jeweils mindestens einmal als Team1 und Team2 angetreten ist.)

```
SELECT MAX(tmp2.Summe) FROM (
  SELECT Team, SUM(Summe) as Summe FROM (
    SELECT Team1 AS Team, SUM(ToreTeam1) AS Summe
      FROM Spiel
      GROUP BY Team1, ToreTeam1
    UNION
    SELECT Team2 AS Team, SUM(ToreTeam2) AS Summe
      FROM Spiel
      GROUP BY Team2, ToreTeam2
  ) AS tmp
  GROUP BY Team
) as tmp2;
```

```
 max
-----
   9
(1 row)
```

(c) Wie viele Tore sind im Turnier insgesamt gefallen?

```
SELECT SUM(ToreTeam1 + ToreTeam2) AS GesamtanzahlTore
FROM Spiel;
```

(d) Ermitteln Sie die Namen und Länder der fünf Spieler, die nach der kürzesten Spielzeit einen Platzverweis erhielten. Die Ausgabe soll nummeriert erfolgen (beginnend bei 1 für die kürzeste Spielzeit).

```
SELECT s.Name, s.Land, COUNT(*) AS Rang
FROM Spieler s, Platzverweise p1, Platzverweise p2
WHERE
  s.Spieler_ID = p2.Spieler_ID AND
  p1.Spielminute <= p2.Spielminute
GROUP BY s.Name, s.Land, p2.Spieler_ID
HAVING COUNT(*) < 6
ORDER BY Rang;
```

Der Erstplatzierte kommt durch die WHERE-Bedingungen nur einmal in der Relation vor, weil sein Eintrag genau einmal mit sich selbst vorkommt. Alle anderen Einträge, bei denen die `p2.Spieler_ID`, der Spieler mit der „geringsten Minute“ ist, werden ja eliminiert, da ja nur die Einträge behalten werden, die der Bedingung `p1.Spielminute <= p2.Spielminute` entsprechen.

# Thema Nr. 2

## Teilaufgabe Nr. 1

### Aufgabe 1 [Projektmanagement]

Gegeben seien folgende Tätigkeiten mit ihren Abhängigkeiten und Dauern:

| Task | Dauer (in h) | Abhängigkeiten |
|---|---|---|
| T1 | 3 | / |
| T2 | 6 | / |
| T3 | 2 | T1 |
| T4 | 2 | T2 |
| T5 | 5 | T1 |
| T6 | 3 | T4, T5 |
| T7 | 6 | T3 |
| T8 | 7 | T4 |
| T9 | 4 | T6, T8 |
| T10 | 1 | T7, T9 |

(a) Zeichnen Sie ein CPM-Diagramm basierend auf der gegebenen Aufgabenliste. Benutzen Sie explizite Start- und Endknoten.

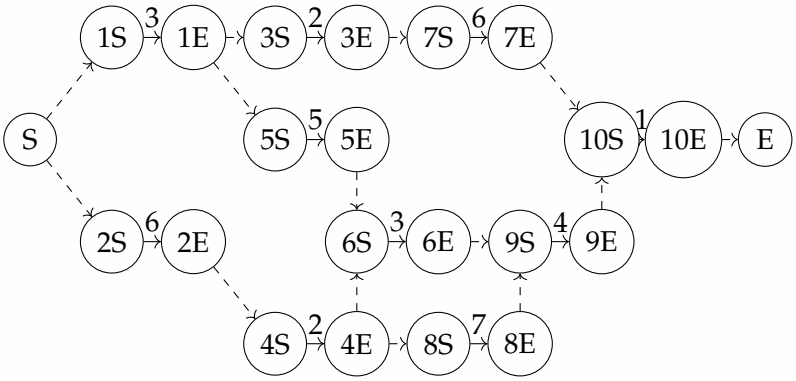
**Abkürzungen**

**S** Start

**1S** Start von T1

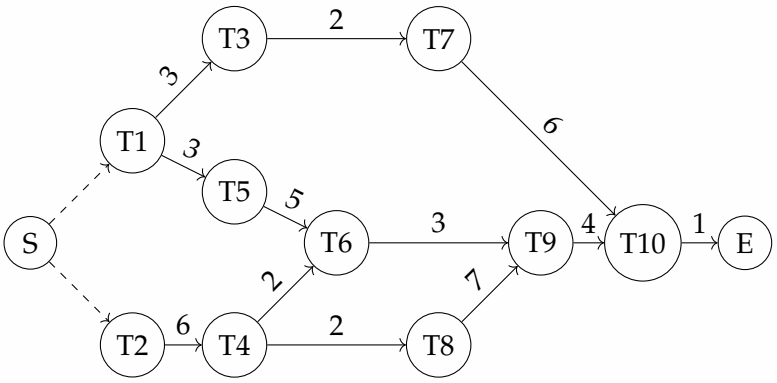
**1E** Ende von T1

**E** Ende

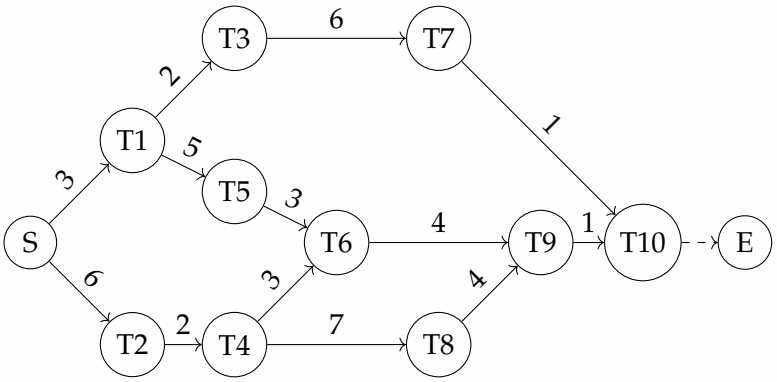


Teilen wir einen Task in zwei Knoten auf, so wird das Diagramm sehr unübersichtlich. Wir verwenden pro Task nur einen Knoten. Es gibt zwei Möglichkeiten:

**Knoten sind Anfang der Tasks**




**Knoten sind Ende der Tasks**




(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 ergänzen Sie ihn in Ihrem Diagramm.

**Knoten sind Anfang der Tasks**

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:** $i$: Ereignis $i$; $FZ_i$: Frühester Zeitpunkt, zu dem Ereignis $i$ eintreten kann. —

| $i$ | Nebenrechnung | $FZ_i$ |
|---|---|---|
| T1 | | 0 |
| T2 | | 0 |
| T3 | | 3 |
| T4 | | 6 |
| T5 | | 3 |
| T6 | $\max(8_{T4}, 8_{T5})$ | 8 |
| T7 | | 5 |
| T8 | | 8 |
| T9 | $\max(11_{T6}, 15_{T4})$ | 15 |
| T10 | $\max(19_{T9}, 11_{T7})$ | 19 |
| E | | 20 |

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:** $i$: Ereignis $i$; $SZ_i$: Spätester Zeitpunkt, zu dem Ereignis $i$ eintreten kann. ———————————

| $i$ | Nebenrechnung | $SZ_i$ |
|---|---|---|
| E | | 20 |
| T10 | | 19 |
| T9 | | 15 |
| T8 | | 8 |
| T7 | | 13 |
| T6 | | 12 |
| T5 | | 7 |
| T4 | $\min(12_{T6}, 6_{T8})$ | 6 |
| T3 | | 11 |
| T2 | | 0 |
| T1 | $\min(8_{T3}, 4_{T5})$ | 4 |

| $i$ | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $FZ_i$ | 0 | 0 | 3 | 6 | 3 | 8 | 5 | 8 | 15 | 19 | 20 |
| $SZ_i$ | 4 | 0 | 11 | 6 | 7 | 12 | 13 | 8 | 15 | 19 | 20 |
| GP | 4 | 0 | 8 | 0 | 4 | 4 | 8 | 0 | 0 | 0 | 0 |

**Knoten sind Ende der Tasks**

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:** $i$: Ereignis $i$; $FZ_i$: Frühester Zeitpunkt, zu dem Ereignis $i$ eintreten kann.

| $i$ | Nebenrechnung | $FZ_i$ |
|---|---|---|
| T1 | | 3 |
| T2 | | 6 |
| T3 | | 5 |
| T4 | | 8 |
| T5 | | 8 |
| T6 | $\max(11_{T4}, 11_{T5})$ | 11 |
| T7 | | 11 |
| T8 | | 15 |
| T9 | $\max(15_{T6}, 19_{T8})$ | 19 |
| T10 | $\max(20_{T9}, 12_{T7})$ | 20 |
| E | | 20 |

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:** $i$: Ereignis $i$; $SZ_i$: Spätester Zeitpunkt, zu dem Ereignis $i$ eintreten kann.

| $i$ | Nebenrechnung | $SZ_i$ |
|---|---|---|
| E | | 20 |
| T10 | | 20 |
| T9 | | 19 |
| T8 | | 15 |
| T7 | | 19 |
| T6 | | 15 |
| T5 | | 12 |
| T4 | $\min(12_{T6}, 8_{T8})$ | 8 |
| T3 | | 13 |
| T2 | | 6 |
| T1 | $\min(11_{T3}, 7_{T5})$ | 7 |

| $i$ | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $FZ_i$ | 3 | 6 | 5 | 8 | 8 | 11 | 11 | 15 | 19 | 20 | 20 |
| $SZ_i$ | 7 | 6 | 13 | 8 | 12 | 15 | 19 | 15 | 19 | 20 | 20 |
| GP | 4 | 0 | 8 | 0 | 4 | 4 | 8 | 0 | 0 | 0 | 0 |

(c) Zeichnen Sie den kritischen Pfad in Ihr Diagramm ein oder geben Sie die Tasks des kritischen Pfades in der folgenden Form an: **Start** ! . . . ! **Ende**. Sollte es mehrere kritische Pfade geben, geben Sie auch diese an. Wie lange ist die Dauer des kritischen Pfades bzw. der kritischen Pfade?

Kritischer Pfad: **Start** ! T2 ! T4 ! T8 ! T9 ! T10 ! **Ende**

Dauer: 20 h

**Aufgabe 2 [Entwicklungsprozesse]**

(a) Erklären Sie den Unterschied zwischen *iterativen* und *inkrementellen* Entwicklungsprozessen. Nennen Sie zudem je ein Prozessmodell als Beispiel.

**Iterativ:** Ein Entwicklungszyklus wird immer wieder durchlaufen:

Planung → Implementierung → Testung → Evaluation.

Mit jeder Iteration wird das Produkt verfeinert. Das Wasserfallmodell in seiner normalen Form würde dies nicht erfüllen, da hier alle Phasen nur einmal durchlaufen werden.

**Beispiel:** Agile Programmierung, erweitertes Wasserfallmodell

**Inkrementell:** Das Projekt wird in einzelne Teile zerlegt. An jedem Teilprojekt kann bestenfalls separat gearbeitet werden. In den einzelnen Teilprojekten kann dann ebenfalls wieder iterativ gearbeitet werden, die beiden Methoden schließen sich gegenseitig also nicht aus.

**Beispiel:** Agile Programmierung, V-Modell XT, Aufteilung in Teilprojekte, die alle mit Wasserfallmodell bearbeitet werden

(b) Nennen und erklären Sie kurz die vier Leitsätze der agilen Softwareentwicklung laut *Agilem Manifest*.

(i) **Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge**
Ein festgelegter Prozess, der nicht sinnvoll von den beteiligten Personen umgesetzt werden kann, ist nicht sinnvoll und sollte nicht durchgeführt werden. Es geht darum, die Stärken der Mitarbeiter einzusetzen und sich nicht sklavisch an Abläufen zu orientieren. Es geht darum, Freiräume zu schaffen, um das Potential des Einzelnen voll entfalten zu lassen.

(ii) **Funktionierende Software ist wichtiger als umfassende Dokumentation**
Der Kunde ist in erster Linie an einem funktionierenden Produkt interessiert. Es soll möglichst früh ein Prototyp entstehen, der dann im weiteren Prozess an die Bedürfnisse des Kunden angepasst wird. Eine umfassende Dokumentation kann am Ende immer noch ergänzt werden.

(iii) **Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlung**
Anforderungen und Spezifikationen können sich flexibel ändern, dadurch kann der Kunde direkt Rückmeldung geben, ðes ist nicht wichtig im Vertrag kleinste Details zu regeln, sondern auf die Bedürfnisse des Kunden einzugehen. Daraus folgt direkt der letzte Punkt:

(iv) **Reagieren auf Veränderung ist wichtiger als das Befolgen eines Plans**
Softwareentwicklung ist ein dynamischer Prozess. Das starre Befolgen eines Plans widerspricht der Grundidee der agilen Programmierung.

(c) Beschreiben Sie die wesentlichen Aktivitäten in *Scrum* (agiles Entwicklungsmodell) inklusive deren zeitlichen Ablaufs. Gehen Sie dabei auch auf die *Artefakte* ein, die im Verlauf der Entwicklung erstellt werden.

(i) **Initiale Projekterstellung:**
Festlegen eines Product Backlog durch den Product Owner, Erstellen von ersten User Stories. Diese entstehen gegebenenfalls durch den Kontakt mit den Stakeholdern, falls der Product Owner nicht selbst der Kunde ist.

(ii) **Sprint planning:**
Der nächste Sprint (zwischen 1 und 4 Wochen) wird geplant. Dabei wird ein Teil des Product Backlog als *Sprint Backlog* definiert und auf die einzelnen Entwickler verteilt. Es wird festgelegt, *was* implementiert werden soll (Product Owner anwesend) und *wie* das geschehen soll (Entwicklerteam). Während des Sprints findet jeden Tag ein *Daily Scrum* statt, ein fünfzehnminütiger Austausch zum aktuellen Stand. Am Ende des Sprints gibt es ein *Sprint Review* und das *Product Inkrement* wird evaluiert und das *Product Backlog* gegebenenfalls angepasst in Absprache mit *Product Owner* und *Stakeholdern*.

**Artefakte:**

- Product Backlog
- Sprint Backlog
- Product Increment

(d) Nennen und erklären Sie die *Rollen* in einem Scrum-Team.

**Product Owner** Verantwortlich für das Projekt, für die Reihenfolge der Bearbeitung und Implementierung, ausgerichtet auf Maximierung und wirtschaftlichen Erfolg. Er führt und aktualisiert das *Product Backlog*.

**Scrum Master** Führungsperson, die das Umsetzen des Scrum an sich leitet und

begleitet. Er mischt sich nicht mit konkreten Arbeitsanweisungen ein, sondern moderiert und versucht Hindernisse zu beseitigen, die einem Gelingen der Sprintziele im Weg sind.

**Entwickler** Sie entwicklen und implementieren die einzelnen Produktfunktionalitäten, die vom Product Owner festgelegt wurden, in eigenverantwortlicher Zeit. Sie müssen die Qualitätsstandards beachten.

**Aufgabe 4 [Wasserstandsmesser]**

(a) Nennen und erklären Sie kurz die drei Kategorien der Organisation von klassischen Entwurfsmustern. Geben Sie zu jeder Kategorie ein Beispiel-Pattern an.

(b) Erstellen Sie ein Klassendiagramm zu den Komponenten einer Beobachtungsstation, welche aus einem einzigen Wasserstandsmesser besteht. Dabei sollen bei Änderungen des Wasserstandes der aktuelle Wasserstand sowohl auf der Konsole als auch in eine Log-Datei geschrieben werden. Der momentane Wasserstand soll dabei mittels einer Variablen des Datentyps double dargestellt werden. Die verschiedenen Anzeigearten (Konsolenanzeige, Logger) sollen als verschiedene Klas- sen modelliert werden und enthalten jeweils nur eine Methode zum Anzeigen bzw. Schreiben des aktuellen Wasserstandes. Verwenden Sie für die Realisierung dieses Klassendiagramms die pas- senden Entwurfsmuster.

Hinweise:

- Getter und Setter müssen nicht eingezeichnet werden.
- Fügen Sie auch Methoden ein, welche durch die einzelnen Klassen implementiert werden.

**Aufgabe 5 [Webshop]**

(a) Nennen Sie vier Programmierparadigmen.

- Imperative Programmierung
- Prozedurale Programmierung
- Funktionale Programmierung
- Objektorientiere Programmierung

[a]

[a] https://de.wikipedia.org/wiki/Programmierparadigma#Strukturierte_Programmierung

(b) Erlautern Sie die Begriffe Overloading und Overriding, sowie deren Unterschiede.

- Beim Überladen muss die Methode eine andere Signatur haben, beim Überschreiben dieselbe Signatur.
- Die Intention beim Überladen ist, Methode zu erweitern, beim Überschreiben die Methode vollständig zu ersetzen.

- Überladen der Methode wird verwendet, um den Polymorphismus der Kompilierzeit zu erreichen. Das Überschreiben der Methode wird verwendet, um einen Laufzeit-Polymorphismus zu erreichen.

  In Methoden- / Funktionsüberladung weiß der Compiler, welches Objekt welcher Klasse zum Zeitpunkt der Kompilierung zugewiesen wurde. In der Methodenüberschreibung sind diese Informationen jedoch erst zur Laufzeit bekannt.
- Das Überladen von Methoden findet in derselben Klasse statt, während das Überschreiben in einer von einer Basisklasse abgeleiteten Klasse stattfindet.

*a*

[a] https://gadget-info.com/difference-between-method-overloading

(c) Erlautern Sie, wie sich zentrale und dezentrale Versionsverwaltung unterscheiden.

Beim der dezentralen Versionsverwaltung hat jeder EntwicklerIn die komplette Repository mit seiner kompletten History lokal gespeichert und kann diese dann mit anderen Repositories abgleichen.

Bei der zentralen Versionsverwaltung gibt es einen zentralen Server, der die komplette History vorhält.

(d) Erstellen Sie ein Sequenzdiagramm zur Methode `main` der Klasse Webshop.

Hinweise:

- Arithmetische Operationen müssen nicht weiter aufgelöst werden.
- Listenoperationen müssen nicht explizit dargestellt werden.
- Auf das Zeichnen einer passiven Lebenslinie muss nicht geachtet werden.
- Übertragen Sie das untenstehende Diagramm als Ausgangspunkt in Ihren Bearbeitungsbogen.

```
public class Webshop {
  public static void main(String[] args) {
    Bestellung b1 = new Bestellung();

    // ab hier soll modelliert werden
    Artikel a1 = new Artikel();
    a1.setName("Taschenrechner");
    a1.setPrice(10);

    b1.addArticle(a1);

    Bestellung b2 = new Bestellung();
    Artikel a2 = new Artikel();
    a2.setName("Lineal");
    a2.setPrice(2.5);

    Artikel a3 = new Artikel();
```

```
    a3.setName("Bleistift");
    a3.setPrice(0.7);

    b2.addArticle(a3);
    b1.addArticle(a2);

    b1.getSize();

    b2.getPrice();
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/webshop/Webshop.java

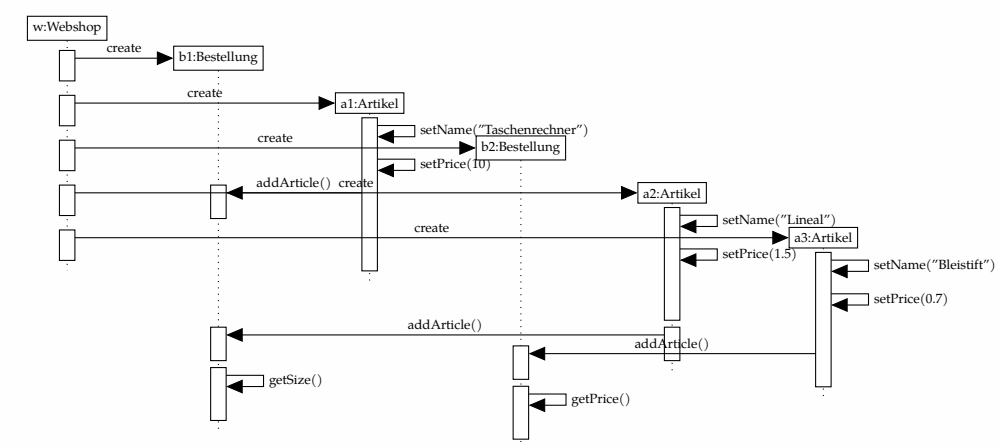
```
@SuppressWarnings({"unused"})
public class Artikel {
  private String name;

  private double price;

  public void setName(String name) {
    this.name = name;
  }

  public void setPrice(double price) {
    this.price = price;
  }

  public double getPrice() {
    return price;
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/webshop/Artikel.java

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

public class Bestellung {
  private List<Artikel> articles;
  // Anzahl an Artikeln
  private int size = 0;
  // Gesamtpreis der Bestellung
  private double price = 0;

  public Bestellung() {
    articles = new ArrayList<>();
  }

  public void addArticle(Artikel article) {
    // muss nicht weiter aufgelöst werden, siehe Hinweise
    articles.add(article);
    size++;
    // muss nicht weiter aufgelöst werden, siehe Hinweise
    price = article.getPrice() + price;
  }
```

```java
  public int getSize() {
    return size;
  }

  public double getPrice() {
    return price;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/webshop/Bestellung.java`




# Teilaufgabe Nr. 2

## Aufgabe 1 [Vermischte Fragen]

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort.

(a) Kann ein Tupel mehrfach im Ergebnis einer SQL-Anfrage enthalten sein?

Ja. Geben wir nur eine Teilmenge an Attributen aus (z. B.ohne Primärschlüssel), so kann ein Tupel mehrfach in der Ausgabe erscheinen.

Außerdem ist es möglich eine Tabelle ohne `PRIMARY KEY` anzulegen. In so einer Tabelle kann dann eine Tupel mehrmals gespeichert werden und über `SELECT * FROM ...` mehrmals ausgegeben werden. (getestet in MySQL und in PostgreSQL).

```sql
CREATE TABLE tmp (
  tmp INTEGER
);

INSERT INTO tmp VALUES
  (1),
  (1),
  (1);

SELECT * FROM tmp;
```

```
+------+
| tmp  |
+------+
|    1 |
|    1 |
|    1 |
+------+
```

Um die mehrfache Ausgabe zu verhindert, gibt es in SQL das Schlüsselwort `DISTINCT`. In der Relationalen Algebra hingegeben sind die Tupel einer Relation eindeutig.

(b) Was ist der Unterschied zwischen einem `INNER JOIN` und einem `OUTER JOIN`?

Ein `INNER JOIN` entspricht der Schnittmenge $A \cap B$.

Ein `OUTER JOIN` entspricht der Vereinigung $A \cup B$. Bei `OUTER JOIN`s können auch NULL-Werte vorkommen. [a]

[a] `https://stackoverflow.com/a/38578`

(c) Welche Auswirkung hat die Verwendung von `ON DELETE CASCADE` bei einem Fremdschlüsselattribut?

Ist `ON DELETE CASCADE` bei einem Fremdschlüsselattribut gesetzt, so wird der referenzierte Datensatz bei einem Löschvorgang mitgelöscht.

(d) Kann eine abgebrochene (aborted) Transaktion wieder fortgesetzt werden?

Eine Transaktion kann nicht fortgesetzt werden. Sie muss zurückgesetzt und wiederholt werden.

(e) Was versteht man unter einer `stored procedure` im Kontext einer Programmierschnittstelle für relationale Datenbanken (z.B JDBC)?

Eine `stored procedure` bildet eine Gruppe von SQL-Befehlen, die eine logische Einheiten bildet und einer bestimmten Aufgabe zugeordnet sind. `stored procedure` werden dazu benutzt eine mehrere Anweisungen und Abfragen zu koppeln

Beispielsweise können bei einer Angestellten-Datenbank die Aufgaben „einstellen“, „entlassen“, „befördern“ als `stored procedure` kompiliert werden und dann mit unterschiedlichen Parametern ausgeführt werden. [a]

[a] `https://docs.oracle.com/javase/tutorial/jdbc/basics/storedprocedures.html`

(f) Was sind `check constraints` und wie wirken sich diese aus?

Constraints definieren Bedingungen, die beim Einfügen, Ändern und Löschen von Datensätzen in der Datenbank erfüllt werden müssen. Wird beispielsweise eine Bedinung bei Einfügen eines Datensatzen nicht erfüllt, so kann dieser Datensatz

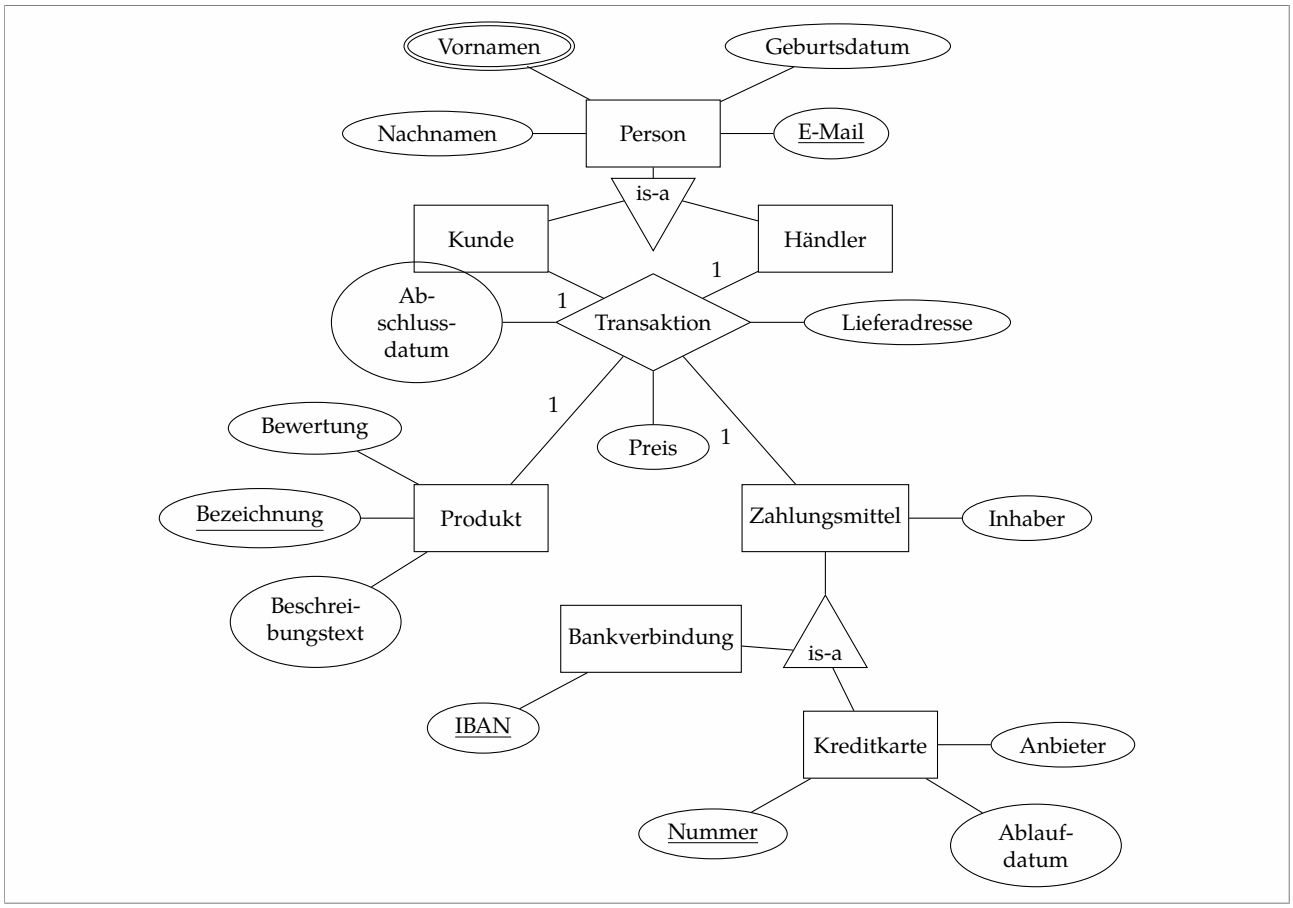
nicht gespeichert werden.

**Aufgabe 2 [Online-Marktplatze]**

Im Folgenden finden Sie die Beschreibung eines Online-Marktplatzes. 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 von Entity-Typen an Beziehungstypen.

Es gibt **Produkte**. Diese haben eine eindeutige *Bezeichnung*, einen *Beschreibungstext* und eine *Bewertung*. Außerdem gibt es **Personen**, die entweder **Kunde**, **Händler** oder beides sind. Jede Person hat einen *Nachnamen*, einen oder mehrere *Vornamen*, ein *Geburtsdatum* und eine *E-Mail-Adresse*, mit der diese eindeutig identifiziert werden kann.

Das System verwaltet außerdem **Zahlungsmittel**. Jedes Zahlungsmittel ist entweder eine **Kreditkarte** oder eine **Bankverbindung** für Lastschriften. Für das Lastschriftverfahren wird die international eindeutige *IBAN* und der Name des *Kontoinhabers* erfasst, bei Zahlung mit Kreditkarte der Name des *Karteninhabers*, die eindeutige *Kartennummer*, das *Ablaufdatum* sowie der *Kartenanbieter*. Es gibt Transaktionen. Jede Transaktion bezieht sich stets auf ein Produkt, einen Kunden, einen Händler und auf ein Zahlungsmittel, das für die Transaktion verwendet wird. Jede Transaktion enthält außerdem den *Preis*, auf den sich Kunde und Händler geeinigt haben, das *Abschlussdatum* sowie eine *Lieferadresse*, an die das Produkt versandt wird.




☐ **E: Produkte**
○ *A: Bezeichnung*
○ *A: Beschreibungstext*
○ *A: Bewertung*
☐ **E: Personen**
☐ **E: Kunde**
☐ **E: Händler**
○ *A: Nachnamen*
○ *A: Vornamen*
○ *A: Geburtsdatum*
○ *A: E-Mail-Adresse*
☐ **E: Zahlungsmittel**
☐ **E: Kreditkarte**
☐ **E: Bankverbindung**
○ *A: IBAN*
○ *A: Kontoinhabers*
○ *A: Karteninhabers*
○ *A: Kartennummer*
○ *A: Ablaufdatum*
○ *A: Kartenanbieter*
◇ R: Transaktionen
○ *A: Preis*
○ *A: Abschlussdatum*
○ *A: Lieferadresse*

### Aufgabe 3 [Autoverleih]

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, Fremdschlüsssel2 Attribut2) [Relation2]); Attribut1 UNIQUE Attribut2 NOT NULL
```

> **Exkurs: Enhanced entity-relationship model**
> https://en.wikipedia.org/wiki/Enhanced_entity-relationship_model https://www.tutorialride.com/dbms/enhanced-entity-relationship-model-eer-model.htm

```
Automarke : {[ AName, Name[Firma] ]}

Fahrzeug : {[ Kennzeichen, Mieter-Nr[Mieter], Name[Firma], Modell, Farbe ]}

Firma : {[ Name, Umsatz, Mieter-Nr ]}

Hersteller : {[ Name[Firma] ]}

Mieter : {[ Mieter-Nr ]}

Person : {[ Handynummer, Vorname, Nachname, Mieter-Nr[Mieter] ]}
```

### Aufgabe 4 [Mitarbeiter einer Abteilung]

Gegeben sind folgende Relationen:

```
Mitarbeiter (MitarbeiterID, Vorname, Nachname, Adresse, Gehalt, Vorgesetzter [Mitarbeiter] NOT

NULL, AbteilungsID[Abteilung])

Abteilung (AbteilungsID, Bezeichnung UNIQUE NOT NULL)
```

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

```sql
CREATE TABLE Abteilung(
  AbteilungsID INTEGER PRIMARY KEY,
  Bezeichnung VARCHAR(30) UNIQUE NOT NULL
```

```
);

CREATE TABLE Mitarbeiter(
  MitarbeiterID INTEGER PRIMARY KEY,
  Vorname VARCHAR(30),
  Nachname VARCHAR(30),
  Adresse VARCHAR(60),
  Gehalt DECIMAL(7, 2),
  Vorgesetzter INTEGER NOT NULL REFERENCES Mitarbeiter(MitarbeiterID),
  AbteilungsID INTEGER REFERENCES Abteilung(AbteilungsID)
);

INSERT INTO Abteilung VALUES
  (1, 'Buchhaltung'),
  (42, 'Vertrieb');

INSERT INTO Mitarbeiter VALUES
  (1, 'Karl', 'Landsbach', 'Sigmaringstraße 4, 87153 Farnbach', 2467.23, 1, 42),
  (2, 'Lisa', 'Grätzner', 'Scheidplatz 6, 18434 Tullach', 5382.2, 1, 42),
  (3, 'Sarah', 'Riedel', 'Am Angera 3, 79527 Töll', 7382.2, 1, 42),
  (4, 'Franz', 'Rudolf', 'Strewitzstraße 4, 45507 Strewith', 2382.2, 1, 42),
  (5, 'Sergej', 'Puschkin', 'Radolf 4, 12507 Radstadt', 1382.2, 1, 1);
```

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle Mitarbeiter anlegt. Gehen Sie davon aus, dass die Tabelle Abteilung bereits existiert.

Siehe oben

(b) Schreiben Sie eine SQL-Anweisung, die Vor- und Nachnamen der Mitarbeiter der Abteilung mit der Bezeichnung Vertrieb ausgibt, absteigend sortiert nach MitarbeiterID.

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

(c) Schreiben Sie eine SQL-Anweisung, die Vor- und Nachnamen sowie das Gehalt von Mitarbeitern ausgibt, die mehr verdienen als ihr Chef. Sortieren Sie die Ausgabe absteigend nach dem Gehalt.

```
SELECT m.Vorname, m.Nachname, m.Gehalt
FROM Mitarbeiter m, Mitarbeiter n
WHERE
  m.Vorgesetzter = n.MitarbeiterID AND
  m.Gehalt > n.Gehalt
ORDER BY m.Gehalt DESC;
```

(d) Schreiben Sie eine SQL-Anweisung, die das Gehalt von allen Mitarbeitern aus der Abteilung mit der ID 42 um 10% erhöht.

```sql
UPDATE Mitarbeiter
SET Gehalt = Gehalt * 1.1
WHERE AbteilungsID = 42;
SELECT * FROM Mitarbeiter;
```

(e) Schreiben Sie eine SQL-Anweisung, welche den Vornamen, die Nachnamen und das Gehalt der sieben bestbezahlten Mitarbeiter aus der Buchhaltung ausgibt. Standardkonforme Sprachkonstrukte, die eine Beschränkung der Ausgabe bewirken, sind erlaubt.

```sql
SELECT m.Vorname, m.Nachname, m.Gehalt
FROM Mitarbeiter m, Mitarbeiter n, Abteilung a
WHERE
  m.Gehalt <= n.Gehalt AND
  a.AbteilungsID = m.AbteilungsID AND
  a.AbteilungsID = n.AbteilungsID AND
  a.Bezeichnung = 'Buchhaltung'
GROUP BY m.Vorname, m.Nachname, m.Gehalt
HAVING COUNT(*) <= 7
ORDER BY m.Gehalt DESC;
```

(f) Schreiben Sie eine SQL-Anweisung, die für jede Abteilung die Mitarbeiter ermittelt, die am wenigsten verdienen. Dabei sollen Vorname, Nachname und die Abteilungsbezeichnung der Mitarbeiter ausgegeben werden.

```sql
SELECT m.Vorname, m.Nachname, m.Gehalt, a.Bezeichnung
FROM Mitarbeiter m, Mitarbeiter n, Abteilung a
WHERE
  m.Gehalt >= n.Gehalt AND
  m.AbteilungsID = n.AbteilungsID AND
  m.AbteilungsID = a.AbteilungsID
GROUP BY m.Vorname, m.Nachname, m.Gehalt, m.AbteilungsID, a.Bezeichnung
HAVING COUNT(*) <= 1
ORDER BY m.Gehalt DESC;
```

### Aufgabe 5 [Mitarbeiter einer Abteilung]

Formulieren Sie basierend auf den in der letzten Aufgabe gegebenen Relationen die geforderten Anfragen in der Relationenalgebra.

Mitarbeiter (MitarbeiterID, Vorname, Nachname, Adresse, Gehalt, Vorgesetzter [Mitarbeiter] NOT NULL, AbteilungsID[Abteilung])

Abteilung (AbteilungsID, Bezeichnung UNIQUE NOT NULL)

(a) Formulieren Sie eine Anfrage, welche die Vornamen und Nachnamen der Mitarbeiter ausgibt, die in der Buchhaltung arbeiten.

$$\pi_{\text{Vorname, Nachname}}\Big(\sigma_{\text{Bezeichnung = 'Buchhaltung'}}\big(\text{Mitarbeiter} \bowtie_{\text{Mitarbeiter.AbteilungsID = Abteilung.AbteilungsID}} \text{Abteilung}\big)\Big)$$

(b) Formulieren Sie eine Anfrage, welche die Vornamen und Nachnamen der Mitarbeiter ausgibt, die in keiner Abteilung arbeiten.

$$\pi_{\text{Vorname, Nachname}}(\text{Mitarbeiter}) - \pi_{\text{Vorname, Nachname}}(\text{Mitarbeiter} \bowtie \text{Abteilung})$$

**Aufgabe 6 [Relation Prüfung]**

Gegeben ist die Relation Prüfung (Prüfungsnummer, Fakultät, Prüfungsname, Dozent, Prüfungstyp, ECTS) mit den beiden Schlüsselkandidaten (Prüfungsnummer, Fakultät) und (Fakultät, Prüfungsname, Dozent).

Alle Attributwerte sind atomar. Es gelten nur die durch die Schlüsselkandidaten implizierten funktionalen Abhängigkeit.

Geben Sie die höchste Normalform an, die die Relation-Prüfung erfüllt. Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind und dass mindestens eine Bedingung der nächsthöheren Normalform verletzt ist. Beziehen Sie sich bei der Begründung auf die gegebene Relation und nennen Sie nicht nur die allgemeinen Definitionen der Normalformen.

$$\text{FA} = \Big\{ \{ \textit{Prüfungsnummer, Fakultät} \} \rightarrow \{ \textit{Prüfungsname, Dozent, Prüfungstyp, ECTS} \}, \{ \textit{Fakultät, Prüfungsname, Dozent} \} \rightarrow \{ \textit{Prüfungsnummer, Prüfungstyp, ECTS} \}, \Big\}$$

Höchste Normalform: 4NF
Siehe Taschenbuch Seite 449

**1NF** Alle Werte sind atomar.

**2NF** Ist in 1NF und jedes Attribut ist Teil des Schlüsselkandidaten (`Prüfungsnummer, Fakultät` oder `Fakultät, Prüfungsname, Dozent`) oder das Attribut ist von einem Schlüsselkandidaten voll funktional abhängig (`Prüfungsname, Dozent, Prüfungstyp, ECTS` oder `Prüfungsnummer, Prüfungstyp, ECTS`).

**3NF** Ist in 2NF und ein Nichtschlüsselattribut darf nur vom Schlüsselkandidaten abhängen (`Prüfungsname, Dozent, Prüfungstyp, ECTS` hängt von `Prüfungsnummer, Fakultät` ab) und (`Prüfungsnummer, Prüfungstyp, ECTS` hängt von `Fakultät, Prüfungsname, Dozent` ab).

**BCNF** Jede Determinate ist Schlüsselkandidat (`Prüfungsnummer, Fakultät` und `Fakultät, Prüfungsname, Dozent`).

**4NF** keine paarweise Unabhängigkeiten mehrwertigen Abhängigkeiten zwischen ihren Attributen.

### Aufgabe 7 [Optimierung]

(a) Erläutern Sie kurz, was Indizes sind und warum diese in Datenbanksystemen verwendet werden.

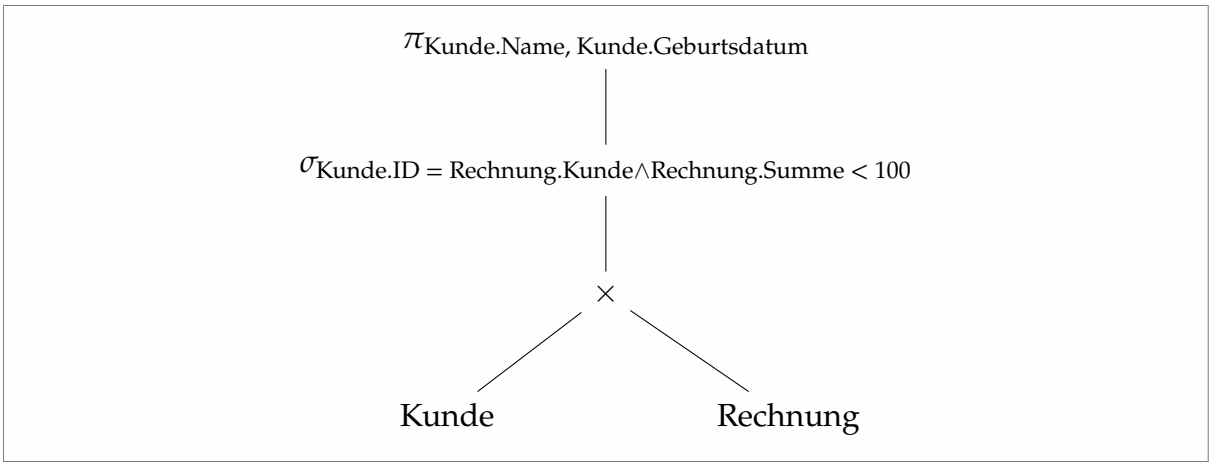
Ein Datenbankindex ist eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank, die die Suche und das Sortieren nach bestimmten Feldern beschleunigt.

Ein Index besteht aus einer Ansammlung von Zeigern (Verweisen), die eine Ordnungsrelation auf eine oder mehrere Spalten in einer Tabelle definieren. Wird bei einer Abfrage eine indizierte Spalte als Suchkriterium herangezogen, sucht das Datenbankmanagementsystem (DBMS) die gewünschten Datensätze anhand dieser Zeiger. In der Regel finden hier B+-Bäume Anwendung. Ohne Index müsste die Spalte sequenziell durchsucht werden, während eine Suche mit Hilfe des Baums nur logarithmische Komplexität hat. [a]

[a] `https://de.wikipedia.org/wiki/Datenbankindex`

(b) Übertragen Sie folgendes SQL-Statement in einen nicht optimierten algebraischen Term oder in einen Anfragegraphen.

```
SELECT Kunde.Name, Kunde.Geburtsdatum
FROM Kunde, Rechnung
WHERE Kunde.ID = Rechnung.Kunde
AND Rechnung.Summe < 100;
```




(c) Nennen Sie zwei Möglichkeiten, den algebraischen Term bzw. den Anfragegraphen aus der vorhergehenden Teilaufgabe logisch (d. h. algebraisch) zu optimieren. Beziehen Sie sich auf konkrete Stellen und Operatoren des von Ihnen aufgestellten algebraischen Ausdrucks.

Zuerst Selection und dann Join

Theta-Join