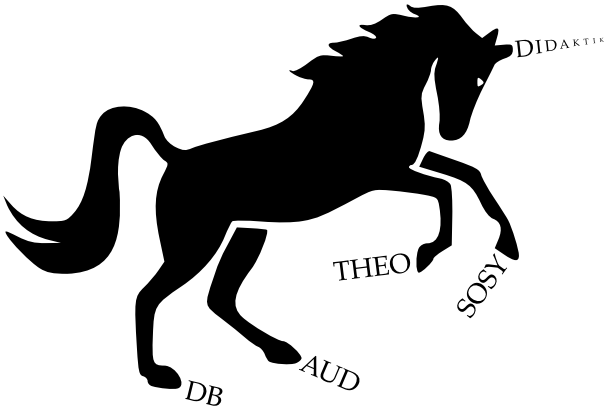
46116
Frühjahr
2013

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

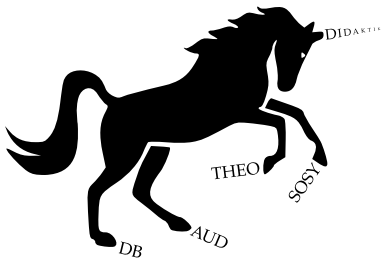
Aufgabenstellungen mit Lösungsvorschlägen



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 2 [modernen Softwaretechnologie. 3 Begriffe in 3 Sätzen]

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

Lösungsvorschlag

Pair Programming Beim Pair Programming entwickeln zwei Personen die Software zusammen. Es gibt einen sogenannten Driver - die Person, die tippt - und einen sogenannten Navigator - die Person, die kritisch den Code überprüft. Das Pair Programming ist wichtiger in agilen Entwicklungsmethoden wie z. B dem Exterme Programming (XP).

Refactoring Unter Refactoring versteht man die Verbesserung der Code- und Systemstruktur mit dem Ziel einer besseren Wartbarkeit. Dabei sollen Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit verbessert werden, mit dem Ziel, den jeweiligen Aufwand für Fehleranalyse und funktionale Erweiterungen deutlich zu senken. Im Refactoring werden keine neuen Funktionalitäten programmiert.

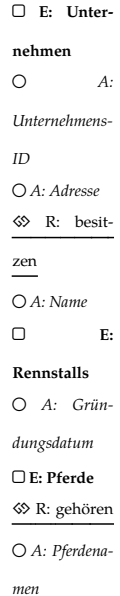
agile Softwareentwicklung Agile Entwicklungsprozesse gehen auf das Agile Manifest, das im Jahr 2001 veröffentlicht wurde, zurück. Dieses Agile Manifest bildet die Basis für agile Entwicklungsprozesse wie eXtreme Programming oder SCRUM. Das Manifest beinhaltet 12 Grundprinzipien für die moderne agile Software-Entwicklung. Hauptzielsetzung ist, rasch eine funktionsfähige Software-Lösung auszuliefern, um den Kundenwunsch bestmöglich zu erfüllen. Dabei spielt die Interaktion mit dem Kunden eine zentrale Rolle.

Teilaufgabe Nr. 2

Aufgabe [Rennstall]

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

- **Unternehmen** werden ihre eindeutige *Unternehmens-ID* identifiziert. Sie haben eine *Adresse* und besitzen Rennställe.
- Der *Name* eines **Rennstalls** ist nur innerhalb eines Unternehmens eindeutig. Für jeden Rennstall wird das *Gründungsdatum* gespeichert.
- **Pferde** gehören immer zu einem Rennstall. *Pferdenamen* werden in einem Rennstall nur jeweils maximal einmal vergeben.
- **Jockeys** sind in einem Rennstall beschäftigt. Jeder Rennstall vergibt seine eigenen *Personalnummern*. Für jeden Jockey werden *Vorname* und *Name* gespeichert.



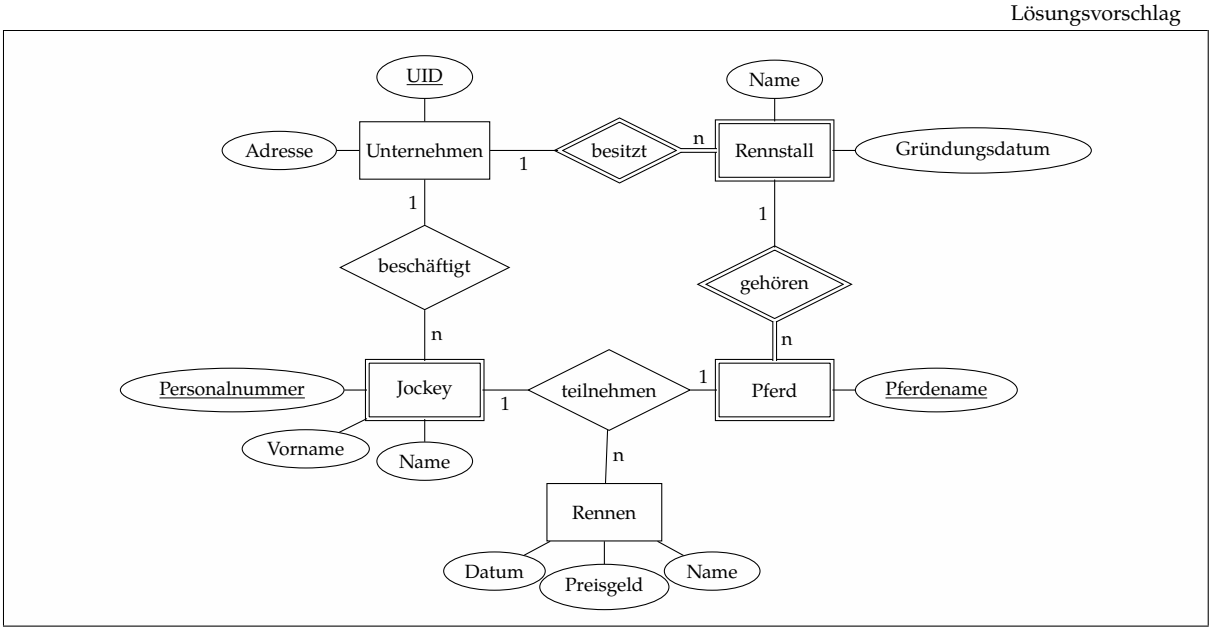
- **Rennen** haben ein *Datum*, ein *Preisgeld* und einen *Namen*, über den sie identifiziert werden.
- Unternehmen unterstützen Rennen finanziell mit einem bestimmten Betrag.
- Jockeys nehmen mit Pferden an Rennen teil. Im Rennen erreichen sie einen bestimmten Platz. Die Kombination aus Jockey und Pferd ist nicht fest, bei unterschiedlichen Rennen können Jockeys verschiedene Pferde reiten. Jockeys können auch mit Rennpferden von fremden Rennställen, die anderen Unternehmen gehören können, an Rennen teilnehmen.

□ **E: Rennen**
○ *A: Datum*
○ *A: Preisgeld*
○ *A: Namen*
◇ R: unterstützen
◇ R: nehmen

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

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

Lösungsvorschlag



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

```
Unternehmen (UnternehmensID, Addresse)
Rennstall (S_Name, UnternehmensID[Unternehmen], Gründungsdatum)
Jockey (PersNr, S_Name[Rennstall], ID[Unternehmen], Vorname, Name)
Rennen (R_Name, Datum, Preisgeld)
Pferd (P_Name, S_Namen, UnternehmensID[Unternehmen])
```

```
teilnehmen (R_Name[Rennen], PersNr[Jockey], S_Name1, ID1, P_Namen, S_Namen2, ID2,
↪  Platz)
unterstuetzen (UnternehmensID[Unternehmen], R_Name[Rennen], Betrag)
```

2. Anfragen [Browser-Online-Spiele]

2. Anfragen

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

```
Team : {[TNr, Name, Teamfarbel]}
Spieler : {[SNr, Name, Icon, TNr, EMail]}
Minispiel : {[MNr, Name, Kategorie, Schwierigkeit]}
Wettkampf : {[WNr, Sieger, Geschlagener, MNr, Dauer]}
```

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

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

(i) Geben Sie die *Namen* der *Minispiele* zurück, die zur *Kategorie „Denkspiele“* zählen.

Lösungsvorschlag

$$\pi_{\text{Name}}(\sigma_{\text{Kategorie='Denkspiele'}}(\text{Minispiele}))$$

(ii) Geben Sie die *Namen* und *E-Mail-Adressen* aller Spieler zurück, die in einem Minispiel des Typs *„Geschicklichkeitsspiel“* gewonnen haben.

Lösungsvorschlag

$$\pi_{\text{Spieler.Name,Spieler.Email}}(\sigma_{\text{Kategorie='Geschicklichkeitsspiel'}}(\text{Minispiele}) \bowtie_{\text{Minispiel.MNr=Wettkampf.MNr}} \text{Wettkampf} \bowtie_{\text{Wettkampf.Sieger=Spieler.SNr}} \text{Spieler})$$

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

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

Lösungsvorschlag

```
SELECT DISTINCT Kategorie
FROM Minispiel
WHERE Schwierigkeit = 'sehr schwer';
```

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

Lösungsvorschlag

```
SELECT WNr
FROM Wettkampf
WHERE Dauer < (
  SELECT AVG(Dauer) FROM Wettkampf
);
```

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

```
SELECT
  SNr,
  Name,
  (
    SELECT COUNT(*)
    FROM Wettkampf
    WHERE Wettkampf.Sieger = SNr
  ) AS Anzahl_Siege,
  (
    SELECT AVG(Dauer)
    FROM Wettkampf
    WHERE Wettkampf.Sieger = SNr
  ),
  (
    SELECT COUNT(DISTINCT Team.TNr)
    FROM Team, Spieler, Wettkampf
    WHERE
      Team.TNr = Spieler.TNr AND
      Wettkampf.Geschlagener = Spieler.SNr AND
      Wettkampf.Sieger = SNr
  )
FROM Spieler;
```

Lösungsvorschlag

```
SELECT
  s.SNR,
  s.Name,
  COUNT(*) AS AnzahlSiege,
  AVG(w.Dauer) AS DurchschnittlicheWettkampfdauer,
  COUNT(DISTINCT g.TNr) AS TeamsBesiegt
FROM Spieler s, Wettkampf w, Spieler g
WHERE
  s.SNr = w.Sieger AND
```

```
  g.SNr = w.Geschlagener
GROUP BY s.SNR, s.Name;
```

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

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

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

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

Thema Nr. 2

Teilaufgabe Nr. 1

Aufgabe 1 [CreditCard, Order]

Aufgabe 1

Gegeben sei folgendes Klassendiagramm:

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

```java
public class Check extends Payment {
  String bankName;
  long bankId;
  public boolean authorized() {
    return true;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Check.java`

```java
@SuppressWarnings("unused")
public class Item {
  private double weight;
  public String description;

  public double getPrice() {
    return 42;
  }

  public double getWeight() {
    return 23;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Item.java`

```java
import java.util.Date;

public class Order {
  public static double VAT = 0.19;

  protected Date date;

  protected boolean shipped;
```

```
  public double calcPrice() {
    return 0.1d;
  }

  public double calcWeight() {
    return 0.2d;
  }

  public double calcTax(double tax) {
    return 0.3d;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Order.java`

```
@SuppressWarnings("unused")
public class OrderDetail {
  private long quantity;

  Item item;

  public double calcPrice() {
    return 0.1d;
  }

  public double calcWeight() {
    return 0.2d;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/OrderDetail.java`

```
public abstract class Payment {
  double amount;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Payment.java`

(b) Erstellen Sie ein Sequenzdiagramm (mit konkreten Methodenaufrufen und Nummerierung der Nachrichten) für folgendes Szenario:

(i) „Erika Mustermann“ aus „Rathausstraße 1, 10178 Berlin“ wird als neue Kundin angelegt.

(ii) Frau Mustermann bestellt heute 1 kg Gurken und 2 kg Kartoffeln.

(iii) Sie bezahlt mit ihrer Visa-Karte, die im August 2014 abläuft und die Nummer „1234 567891 23456“ hat — die Karte erweist sich bei der Prüfung als gültig.

(iv) Am Schluss möchte sie noch wissen, wie viel ihre Bestellung kostet — dabei wird der Anteil der Mehrwertsteuer extra ausgewiesen.