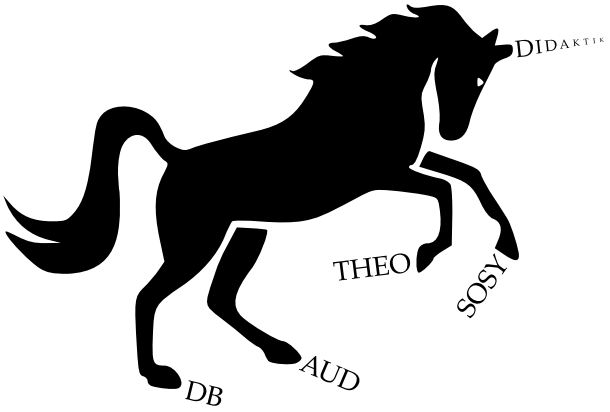
66116
Herbst
2015

Datenbanksysteme / Softwaretechnologie (vertieft)

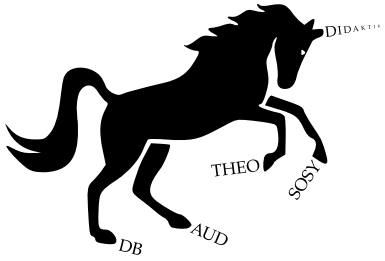
Aufgabenstellungen mit Lösungsvorschlägen



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

1. Modellierung [Online-Auktionshaus]

Wir wollen eine relationale Datenbankstruktur für ein Online-Auktionshaus modellieren.

Das Auktionshaus hat Mitglieder. Diese Mitglieder haben Kundennummern, Namen und Adressen. Sie können Verkäufer und/oder Käufer sein. Verkäufer können eine Laden-URL (eine Subdomain) erhalten, die zu einer Seite mit ihren aktuellen Auktionen führt. Verkäufer können neue Auktionen starten. Die Auktionen eines Verkäufers werden durchnummeriert. Diese Auktionsnummer ist nur je Verkäufer eindeutig. Jede Auktion hat ein Mindestgebot und eine Ablaufzeit.

Auf Auktionen können Gebote abgegeben werden. Die Gebote auf eine Auktion werden nach ihrem Eintreffen nummeriert. Diese Nummer ist nur innerhalb einer Auktion eindeutig. Zu einem Gebot werden noch die Zeit des Gebots sowie der gebotene Geldbetrag angegeben. Die Gebote werden von Käufern abgegeben. Jedes Gebot muss einem Käufer zugeordnet sein.

Jede Auktion besteht aus einer Menge von Artikeln, aber aus mindestens einem. Artikel haben eine Beschreibung und können und über ihre Artikel-ID identifiziert werden. Zur Katalogisierung der Artikel gibt es Kategorien. Kategorien haben eine eindeutige ID und einen Namen. Jede Kategorie kann Subkategorien besitzen. Auch Subkategorien sind Kategorien (und können damit weitere Subkategorien besitzen). Jeder Artikel kann beliebig vielen Kategorien zugeordnet werden.

Entwerfen Sie für das beschriebene Szenario ein ER-Diagramm. 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,
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

2. Normalformen [Relation A-H]

Gegeben sei folgendes verallgemeinerte Relationenschema in 1. Normalform:

$$R(\,A,B,C,D,E,F,G,H\,)$$

Für R soll die folgende Menge FD von funktionalen Abhängigkeiten gelten:

$\mathrm{FA} = \Big\{$

$$
\begin{aligned}
\{F\} &\rightarrow \{E\},\\
\{A\} &\rightarrow \{B, D\},\\
\{A, E\} &\rightarrow \{D\},\\
\{A\} &\rightarrow \{E, F\},\\
\{A, G\} &\rightarrow \{H\},
\end{aligned}
\Big\}
$$

Bearbeiten Sie mit diesen Informationen folgende Teilaufgaben. Vergessen Sie dabei nicht Ihr Vorgehen stichpunktartig zu dokumentieren und zu begründen.

(a) Bestimmen Sie alle Schlüsselkandidaten von R. Begründen Sie stichpunktartig, warum es außer den von Ihnen gefundenen Schlüsselkandidaten keine weiteren geben kann.

(b) Ist R in 2NF, 3NF?

(c) Berechnen Sie eine kanonische Überdeckung von FD. Es genügt, wenn Sie für jeden der vier Einzelschritte die Menge der funktionalen Abhängigkeiten als Zwischenergebnis angeben.

(d) Bestimmen Sie eine Zerlegung von R in 3NF. Wenden Sie hierfür den Synthesealgorithmus an.

Aufgabe 3 [Vater und Mutter]

Gegeben seien folgende Relationen:

`Mensch : {[ ID, MutterID, VaterID ]}`

`Mann : {[ ID ]}`

`Frau : {[ ID ]}`

Das zugehörige ER-Modell für dieses relationale Datenbankschema sieht folgendermaßen aus:

Bearbeiten Sie folgende Teilaufgaben:

(a) Finden Sie die Töchter der Frau mit ID 42.

Lösungsvorschlag

```sql
SELECT Mensch.ID
FROM Mensch, Frau
WHERE
  Mensch.MutterID = Frau.id AND
  Frau.ID = 42;
```

(b) Gibt es Männer, die ihre eigenen Großväter sind? Formulieren Sie eine geeignete SQL-Anfrage.

Lösungsvorschlag

```sql
SELECT Mensch.ID
FROM Mann, Mensch
WHERE
  Mensch.ID = Mann.id AND (
    Mensch.VaterID IN (SELECT v.ID FROM Mensch v WHERE v.VaterID = Mensch.ID)
    OR
    Mensch.MutterID IN (SELECT v.ID FROM Mensch v WHERE v.VaterID = Mensch.ID)
  );
```

(c) Definieren Sie eine View VaterKind (VaterID; KindID), die allen Vätern (VaterID) ihre Kinder (KinderID) zuordnet. Diese View darf keine NULL-Werte enthalten.

Lösungsvorschlag

```sql
-- Wir erzeuge bereits beim Erstellen der Datenbank diese View, damit
-- sie für spätere Aufgaben zur Verfügung steht.
DROP VIEW IF EXISTS VaterKind;
CREATE VIEW VaterKind AS
SELECT Mensch.VaterID, Mensch.ID as KindID
FROM Mensch
WHERE
  Mensch.VaterID IS NOT NULL;
SELECT * FROM VaterKind;
```

(d) Verwenden Sie die View aus c), um alle Väter zurückzugeben, absteigend geordnet nach der Anzahl ihrer Kinder.

Lösungsvorschlag

```sql
SELECT VaterID, COUNT(VaterID) as Anzahl
FROM VaterKind
GROUP BY VaterID
ORDER BY Anzahl DESC;
```

(e) Hugo möchte mit folgender Anfrage auf Basis der View aus c) alle kinderlosen Männer erhalten:

```sql
SELECT VaterID
FROM VaterKind
GROUP BY VaterID
HAVING COUNT(KindID) = 0
```

(i) Was ist das Ergebnis von Hugos Anfrage und warum?

Lösungsvorschlag

Die Anfrage liefert kein Ergebnis. Da die View laut Angabe keine Null-Werte enthalten darf, sind in der View nur Männer verzeichnet, die wirklich Väter sind.

(ii) Formulieren Sie eine Anfrage, die tatsächlich alle kinderlosen Männer zurückliefert.

Lösungsvorschlag

```
SELECT * FROM Mann
EXCEPT
SELECT VaterID
FROM VaterKind
GROUP BY VaterID;
```

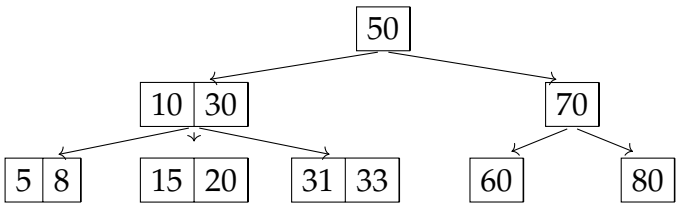
Hinweis: Denken Sie daran, dass SQL auch Mengenoperationen kennt.

Thema Nr. 2

Teilaufgabe Nr. 1

Aufgabe 6: B-Baum [B-Baum der Ordnung 3]

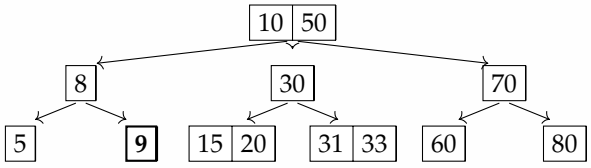
Gegeben ist der folgende B-Baum der Ordnung 3 (max. drei Kindknoten, max. zwei Schlüssel pro Knoten):



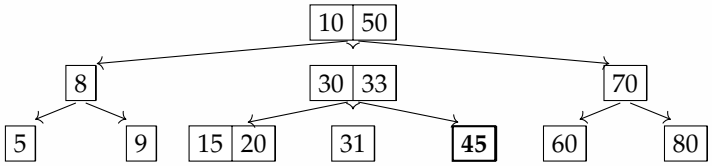
Fügen Sie die Werte 9 und 45 ein. Löschen Sie anschließend die Werte 30 und 70. Zeichnen Sie den Baum nach jeder Einfüge- bzw. Lösch-Operation.

Lösungsvorschlag

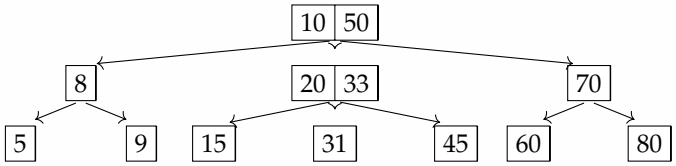
9 einfügen



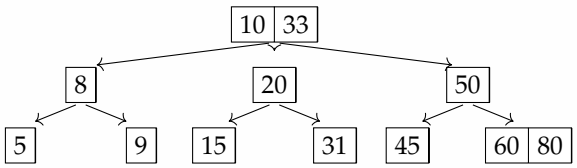
45 einfügen



30 löschen



70 löschen



Teilaufgabe Nr. 2

Aufgabe 3 [Methode „doubleFac()“: wp-Kalkül und Schleifeninvariante]

Gegeben Sei folgendes Programm:

```
long doubleFac (long n) {
  /* P */ long df = 1;
  for (long x = n; x > 1; x -= 2) {
    df *= x;
  } /* Q */
  return df;
}
```

sowie die Vorbedingung $P \equiv n \geq 0$ und Nachbedingung $Q \equiv (\mathrm{df} = n!!)$ wobei gilt

$$n!! := \begin{cases} 2^k \cdot k! & n \text{ gerade}, k := \frac{n}{2} \\ \frac{(2k)!}{2^k \cdot k!} & n \text{ ungerade}, k := \frac{n+1}{2} \end{cases}$$

Exkurs: Fakultät
Die Fakultät ist eine Funktion, die einer natürlichen Zahl das Produkt aller natürlichen Zahlen (ohne Null) kleiner und gleich dieser Zahl zuordnet. Für alle natürlichen Zahlen n ist

$$n! = 1 \cdot 2 \cdot 3 \cdots n = \prod_{k=1}^{n} k$$

Exkurs: Doppelfakultät
Die seltener verwendete „Doppelfakultät“ oder „doppelte Fakultät“ ist für gerade n das Produkt aller geraden Zahlen kleiner gleich n. Für ungerade n ist es das Produkt aller ungeraden Zahlen kleiner gleich n. Sie ist definiert als:

$$n!! = \begin{cases} n \cdot (n-2) \cdot (n-4) \cdots 2 & \text{für } n \text{ gerade und } n > 0, \\ n \cdot (n-2) \cdot (n-4) \cdots 1 & \text{für } n \text{ ungerade und } n > 0, \\ 1 & \text{für } n \in \{-1, 0\} \end{cases}$$

Häufig werden anstelle der Doppelfakultät Ausdrücke mit der gewöhnlichen Fakultät verwendet. Es gilt $(2k)!! = 2^k k!$ und $(2k-1)!! = \frac{(2k)!}{2^k k!}$

Zur Vereinfachung nehmen Sie im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

(a) Welche der folgenden Bedingungen ist eine zum Beweisen der Korrektheit der Methode mittels wp-Kalkül (Floyd-Hoare-Kalkül) sinnvolle Schleifeninvariante?

(i) $\mathrm{df} = n!! - x!! \wedge x \geq 1$

(ii) $\mathrm{df} = (n-x)!! \wedge x \geq 1$

(iii) $\mathrm{df} \cdot x!! = n!! \wedge x \geq 0$

(iv) $(\mathrm{df} + x)!! = n!! \wedge x \geq 0$

Zunächst wird der Code in einen äquivalenten Code mit while-Schleife umgewandelt:

```
long doubleFac (long n) {
  /* P */ long df = 1;
  long x = n ;
  while (x > 1) {
    df = df * x;
    x = x - 2;
  } /* Q */
  return df;
}
```

(i) $\mathrm{df} = n!! - x!! \wedge x \geq 1$

(ii) $\mathrm{df} = (n - x)!! \wedge x \geq 1$

Die ersten beiden Bedingungen sind unmöglich, da z. B. für $n = 2$ nach der Schleife $x = 0$ gilt und daher $x \geq 1$ verletzt wäre.

(iii) $\mathrm{df} \cdot x!! = n!! \wedge x \geq 0$

Nach dem Ausschlussprinzip ist es daher die dritte Bedingung: $I \equiv (\mathrm{df} + x)!! = n!! \wedge x \geq 0$.

(iv) $(\mathrm{df} + x)!! = n!! \wedge x \geq 0$

Die letzte kann es auch nicht sein, da vor der Schleife $\mathrm{df} = 1$ und $x = n$ gilt, $ð(\mathrm{df} + x)!! = (1 + n)!!$. Jedoch ist offenbar $(1 + n)!! \neq n!!$.

$\Rightarrow$ Die Schleifeninvariante lautet: $\mathrm{df} \cdot x!! = n!! \wedge x \geq 0$

(b) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung unmittelbar vor Beginn der Schleife gilt, wenn zu Beginn der Methode die Anfangsbedingung P gilt.

Zu zeigen $P \Rightarrow \mathrm{wp}(\texttt{"Code vor der Schleife"}, I)$

$$
\begin{aligned}
\mathrm{wp}(\texttt{"Code vor der Schleife"}, I) &\equiv \mathrm{wp}(\texttt{"df = 1; x = n;"}, (\mathrm{df} \cdot x)!! = n!! \wedge x \geq 0) \\
&\equiv \mathrm{wp}(\texttt{"df = 1;"}, (\mathrm{df} \cdot n)!! = n!! \wedge n \geq 0) \\
&\equiv \mathrm{wp}(\texttt{""}, (1 \cdot n)!! = n!! \wedge n \geq 0) \\
&\equiv n!! = n!! \wedge n \geq 0 \\
&\equiv n \geq 0 \\
&\equiv P
\end{aligned}
$$

Insbesondere folgt damit die Behauptung.

(c) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung tatsächlich eine Invariante der Schleife ist.

Lösungsvorschlag

zu zeigen: $I \wedge \text{Schleifenbedingung} \Rightarrow \text{wp}(\texttt{"Code in der Schleife"}, I)$

Bevor wir dies beweisen, zeigen wir erst $x \cdot (x-2)!! = x!!$.

- Fall x ist gerade ($n!! = 2^k \cdot k!$ für $k := \frac{n}{2}$):
  $x \cdot (x-2)!! = x \cdot 2^{\frac{x-2}{2}} \cdot (\frac{x-2}{2})! = x \cdot \frac{1}{2} \cdot 2^{\frac{x}{2}} \cdot (\frac{x}{2}-1)! = 2^{\frac{x}{2}} \cdot (\frac{x}{2})! = x!!$
  Nebenrechnung (Division mit gleicher Basis: $x^{a-b} = \frac{x^a}{x^b}$):
  $2^{\frac{x-2}{2}} = 2^{(\frac{x}{2}-\frac{2}{2})} = \frac{2^{\frac{x}{2}}}{2^{\frac{2}{2}}} = \frac{2^{\frac{x}{2}}}{2^1} = \frac{2^{\frac{x}{2}}}{2} = \frac{1}{2} \cdot 2^{\frac{x}{2}}$
  Nebenrechnung ($n! = (n-1)! \cdot n$):
  $x \cdot \frac{1}{2} \cdot (\frac{x}{2}-1)! = \frac{x}{2} \cdot (\frac{x}{2}-1)! = \frac{x}{2}!$
- Fall x ist ungerade:

Dies benutzen wir nun, um den eigentlichen Beweis zu führen:

$$
\begin{aligned}
\text{wp}(\texttt{"Code vor der Schleife"}, I) &\equiv \text{wp}(\texttt{"df = df * x; x = x - 2;"}, (\text{df} \cdot x)!! = n!! \wedge x \geq 0) \\
&\equiv \text{wp}(\texttt{"df = df * x;"}, (\text{df} \cdot (x-2)))!! = n!! \wedge x - 2 \geq 0) \\
&\equiv \text{wp}(\texttt{""}, (\text{df} \cdot x \cdot (x-2)))!! = n!! \wedge x - 2 \geq 0) \\
&\equiv (\text{df} \cdot x)!! = n!! \wedge x \geq 2 \\
&\equiv (\text{df} \cdot x)!! = n!! \wedge x > 1 \\
&\equiv I \wedge x > 1 \\
&\equiv I \wedge \text{Schleifenbedingung}
\end{aligned}
$$

(d) Zeigen Sie formal mittels wp-Kalkül, dass am Ende der Methode die Nachbedingung Q erfüllt wird.

Lösungsvorschlag

z.z. $I \wedge \neg\text{Schleifenbedingung} \Rightarrow \text{wp}(\texttt{"Code nach der Schleife"}, Q)$

Wir vereinfachen den Ausdruck $I \wedge \neg$Schleifenbedingung:

$I \wedge \neg\text{Schleifenbedingung} \equiv I \wedge (x \leq 1) \equiv I \wedge ((x = 0) \vee (x = 1)) \equiv (I \wedge (x = 0)) \vee (I \wedge (x = 1)) \equiv (df \cdot 1 = n!!) \vee (df \cdot 1 = n!!) \equiv df = n!!$

Damit gilt:

$\text{wp}(\texttt{"Code nach der Schleife"}, Q) \equiv \text{wp}(\texttt{""}, df = n!!) \equiv df = n!! \equiv I \wedge \neg\text{Schleifenbedingung}$

(e) Beweisen Sie, dass die Methode immer terminiert. Geben Sie dazu eine Terminierungsfunktion an und begründen Sie kurz ihre Wahl.

Lösungsvorschlag

Sei $T(x) := x$. T ist offenbar ganzzahlig. Da x in jedem Schleifendurchlauf um 2 verringert wird, ist T streng monoton fallend. Aus der Schleifeninvariante folgt $x \geq 0$ und daher ist x auch nach unten beschränkt. Damit folgt $I \Rightarrow T \geq 0$ und T ist eine gültige Terminierungsfunktion.