

66116 Herbst 2017

Datenbanksysteme / Softwaretechnologie (vertieft)

Aufgabenstellungen mit Lösungsvorschlägen

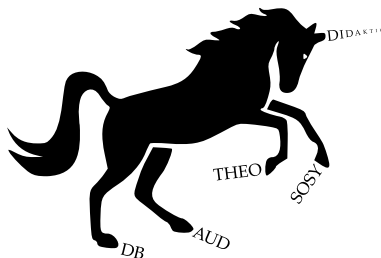


Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht

| | |
|---|----|
| Thema Nr. 1 | 3 |
| Teilaufgabe Nr. 1 | 3 |
| 5. Schemadefinition [SQL-Syntax-Überprüfung] | 3 |
| 6. Relationale Anfragen in SQL [Fluginformationssystem] | 4 |
| Teilaufgabe Nr. 2 | 6 |
| Aufgabe 3 [Code-Inspection bei Binärer Suche] | 6 |
| Thema Nr. 2 | 11 |
| Teilaufgabe Nr. 2 | 11 |



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.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Thema Nr. 1

Teilaufgabe Nr. 1

5. Schemadefinition [SQL-Syntax-Überprüfung]

Gegeben ist die folgende Definition zweier Tabellen:

```
CREATE TABLE R2 (  
  b integer not null,  
  c integer unique,  
  primary key (b)  
);  
  
CREATE TABLE R1 (  
  a integer not null,  
  b integer references R2,  
  primary key (a)  
);
```

Geben Sie jeweils an, ob das Statement syntaktisch korrekt ist und ob es von der gegebenen Datenbank ausgeführt werden kann.

Beantworten Sie jede der folgenden Fragen unabhängig von allen anderen, öes liegt immer das hier gezeigte Schema vor undd alle Relationen sind leer.

- (a) `DELETE FROM R1;`

Lösungsvorschlag

korrekt

- (b) `INSERT INTO R2 VALUES (1,1);`
`INSERT INTO R1 VALUES (1,1);`
`INSERT INTO R1 VALUES (2,1);`
`INSERT INTO R1 VALUES (3,1);`

Lösungsvorschlag

korrekt

- (c)

```
INSERT INTO R2 VALUES (1,1);  
INSERT INTO R2 VALUES (2,2);  
INSERT INTO R1 VALUES (1,1);  
DELETE FROM R2 WHERE b=a;
```

Lösungsvorschlag

falsch: Fehlermeldung column "a" does not exist

```
INSERT INTO R2 VALUES (1,1);  
INSERT INTO R2 VALUES (2,2);  
INSERT INTO R1 VALUES (1,1);  
-- Wir löschen von R1 weil R2 auf R1 referenziert
```

```
-- b kann nur mit Integer verglichen werden.  
DELETE FROM R1 WHERE b=1;
```

(d)

```
INSERT INTO R1 SELECT * FROM R1;
```

Lösungsvorschlag

korrekt

(e)

```
DROP TABLE R2 FROM DATABASE;
```

Lösungsvorschlag

falsch: Fehlermeldung ERROR: syntax error at or near "FROM"

Müsste so lauten:

```
-- Zuerst R1 löschen, wegen der Referenz  
DROP TABLE R1;  
DROP TABLE R2;
```

6. Relationale Anfragen in SQL [Fluginformationssystem]

Folgende Tabellen veranschaulichen eine Ausprägung eines Fluginformationssystems:

Flughäfen

| Code | Stadt | Transferzeit (min) |
|------|---------------|--------------------|
| LHR | London | 30 |
| LGW | London | 20 |
| JFK | New York City | 60 |
| EWB | New York City | 35 |
| MUC | München | 30 |
| FRA | Frankfurt | 45 |

Verbindungen

| ID | Von | Nach | Linie | Abflug (MEZ) | Ankunft (MEZ) |
|-----|-----|------|-------|---------------------|---------------------|
| 410 | MUC | FRA | LH | 2016-02-24 07:00:00 | 2016-02-24 08:10:00 |
| 411 | MUC | FRA | LH | 2016-02-24 08:00:00 | 2016-02-24 09:10:00 |
| 412 | FRA | JFK | LH | 2016-02-24 10:50:00 | 2016-02-24 19:50:00 |

Hinweise

- Formulieren Sie alle Abfragen in SQL-92 (insbesondere sind LIMIT, TOP, FETCH FIRST, ROWNUM und dergleichen nicht erlaubt).
- Alle Datum/Zeit-Angaben erlauben arithmetische Operationen, beispielsweise wird bei der Operation `ankunft + transferzeit` die transferzeit auf den Zeitstempel `ankunft` addiert.
- Es müssen keine Zeitverschiebungen berücksichtigt werden. Alle Zeitstempel sind in MEZ.

```
CREATE TABLE Flughäfen (  
  Code VARCHAR(3) PRIMARY KEY,  
  Stadt VARCHAR(20),  
  Transferzeit integer  
);
```

```
CREATE TABLE Verbindungen (  
  ID integer PRIMARY KEY,  
  Von VARCHAR(3) REFERENCES Flughäfen(Code),  
  Nach VARCHAR(3) REFERENCES Flughäfen(Code),  
  Linie VARCHAR(20),  
  Abflug timestamp,  
  Ankunft timestamp  
);
```

```
INSERT INTO Flughäfen VALUES  
( 'LHR', 'London', 30),  
( 'LGW', 'London', 20),  
( 'JFK', 'New York City', 60),  
( 'EWR', 'New York City', 35),  
( 'MUC', 'München', 30),  
( 'FRA', 'Frankfurt', 45);
```

```
INSERT INTO Verbindungen VALUES  
(410, 'MUC', 'FRA', 'LH', '2016-02-24 07:00:00', '2016-02-24 08:10:00'),  
(411, 'MUC', 'FRA', 'LH', '2016-02-24 08:00:00', '2016-02-24 09:10:00'),  
(412, 'FRA', 'JFK', 'LH', '2016-02-24 10:50:00', '2016-02-24 19:50:00'),  
(413, 'MUC', 'LHR', 'LH', '2016-02-24 10:00:00', '2016-02-24 12:10:00'),  
(414, 'MUC', 'LGW', 'LH', '2016-02-24 11:00:00', '2016-02-24 13:20:00'),  
(415, 'MUC', 'LHR', 'LH', '2016-02-24 12:00:00', '2016-02-24 14:00:00');
```

- (a) Ermitteln Sie die Städte, in denen es mehr als einen Flughafen gibt.

Lösungsvorschlag

```
SELECT Stadt FROM Flughäfen  
GROUP BY Stadt  
HAVING count(Stadt) > 1;
```

- (b) Ermitteln Sie die Städte, in denen man mit der Linie „LH“ an mindestens zwei verschiedenen Flughäfen landen kann.

```

SELECT Stadt FROM Flughäfen
WHERE Stadt IN (
  SELECT Stadt FROM Flughäfen, Verbindungen
  WHERE
    Code = Nach AND
    Linie = 'LH'
  GROUP BY Stadt
)
GROUP BY Stadt
HAVING COUNT(Stadt) > 1;

```

- (c) Ermitteln Sie die Flugzeit des kürzesten Direktflugs von München nach London.

```

CREATE VIEW Flugdauer AS
  SELECT ID, Ankunft - Abflug AS Dauer FROM Flughäfen v, Flughäfen n,
  → Verbindungen
  WHERE
    n.Code = Nach AND
    v.Code = Von AND
    v.Stadt = 'München' AND
    n.Stadt = 'London';

SELECT a.Dauer FROM Flugdauer a, Flugdauer b
WHERE a.Dauer >= b.Dauer
GROUP BY a.Dauer
HAVING COUNT(*) <= 1;

```

- (d) Ermitteln Sie die kürzeste Roundtrip-Zeit (nur Direktflüge) zwischen den Flughäfen FRA und JFK (Transferzeit am Flughafen JFK beachten).

Teilaufgabe Nr. 2

Aufgabe 3 [Code-Inspection bei Binärer Suche]

Die folgende Seite enthält Software-Quellcode, der einen Algorithmus zur binären Suche implementiert. Dieser ist durch Inspektion zu überprüfen. Im Folgenden sind die Regeln der Inspektion angegeben.

| | | |
|-----|------------------|--|
| RM1 | (Dokumentation) | Jede Quellcode-Datei beginnt mit einem Kommentar, der den Klassennamen, Versionsinformationen, Datum und Urheberrechtsangaben enthält. |
| RM2 | (Dokumentation) | Jede Methode wird kommentiert. Der Kommentar enthält eine vollständige Beschreibung der Signatur so wie eine Design-by-Contract-Spezifikation. |
| RM3 | (Dokumentation) | Deklarationen von Variablen werden kommentiert. |
| RM4 | (Dokumentation) | Jede Kontrollstruktur wird kommentiert. |
| RM5 | (Formatierung) | Zwischen einem Schlüsselwort und einer Klammer steht ein Leerzeichen. |
| RM6 | (Formatierung) | Zwischen binären Operatoren und den Operanden stehen Leerzeichen. |
| RM7 | (Programmierung) | Variablen werden in der Anweisung initialisiert, in der sie auch deklariert werden. |
| RM8 | (Bezeichner) | Klassennamen werden groß geschrieben, Variablennamen klein. |

```

/**
 * BinarySearch.java
 *
 * Eine Implementierung der "Binaere Suche"
 * mit einem iterativen Algorithmus
 */
class BinarySearch {

    /**
     * BinaereSuche
     * a: Eingabefeld
     * item: zuzuchendesElement
     * returnValue: der Index des zu suchenden Elements oder -1
     *
     * Vorbedingung:
     * a.length > 0
     * a ist ein linear geordnetes Feld:
     * For all k: (1 <= k < a.length) ==> (a[k-1] <=a [k])
     *
     * Nachbedingung:
     * Wenn item in a, dann gibt es ein k mit a[k] == item und returnValue == k
     * Genau dann wenn returnValue == -1 gibt es kein k mit 0 <= k < a.length
     * und a[k]==item.
     */
    public static int binarySearch(float a[], float item) {

        int End; // exklusiver Index fuer das Ende des
                // zuzuchsuchenden Teils des Arrays

```

```

int start = 1; // inklusiver Index fuer den Anfang der Suche
End = a.length;

// Die Schleife wird verlassen, wenn keine der beiden Haelften das
// Element enthaelt.
while(start < End) {

    // Teilung des Arrays in zwei Haelften
    // untere Haelfte: [0,mid[
    // obere Haelfte: ]mid,End[
    int mid = (start + End) / 2;

    if (item > a[mid]) {
        // Ausschluss der oberen Haelfte
        start = mid + 1;
    } else if(item < a[mid]) {
        // Ausschluss der unteren Haelfte
        End = mid-1;
    } else {
        // Das gesuchte Element wird zurueckgegeben
        return (mid);
    }
} // end of while

// Bei Misserfolg der Suche wird -1 zurueckgegeben
return (-1);
}
}

```

- (a) Überprüfen Sie durch Inspektion, ob die obigen Regeln für den Quellcode eingehalten wurden. Erstellen Sie eine Liste mit allen Verletzungen der Regeln. Geben Sie für jede Verletzung einer Regel die Zeilennummer, Regelnummer und Kommentar an, z. B. (07, RM4, while nicht kommentiert). Schreiben Sie nicht in den Quellcode.

Lösungsvorschlag

| Zeile | Regel | Kommentar |
|-------|-------|--|
| 3-8 | RM1 | Fehlen von Versionsinformationen, Datum und Urheberrechtsangaben |
| 11-26 | RM2 | Fehlen der Invariante in der Design-by-Contract-Spezifikation |
| 36,46 | RM5 | Fehlen des Leerzeichens vor der Klammer |
| 48 | RM6 | Um einen binären (zweistellige) Operator handelt es sich im Code-Beispiel um den Subtraktionsoperator: <code>mid-1</code> . Hier fehlen die geforderten Leerzeichen. |
| 32 | RM7 | Die Variable <code>End</code> wird in Zeile 32 deklariert, aber erst in Zeile initialisiert <code>End = a.length;</code> |
| 32 | RM8 | Die Variable <code>End</code> muss klein geschrieben werden. |

- (b) Entspricht die Methode `binarySearch` ihrer Spezifikation, die durch Vor-und Nachbedingungen angegeben ist? Geben Sie gegebenenfalls Korrekturen der Methode an.

Korrektur der Vorbedingung

Die Vorbedingung ist nicht erfüllt, da weder die Länge des Feldes `a` noch die Reihenfolge der Feldeinträge geprüft wurden.

```
if (a.length <= 0) {
    return -1;
}

for (int i = 0; i < a.length; i++) {
    if ( a[i] > a[i + 1]) {
        return -1;
    }
}
```

Korrektur der Nachbedingung

`int start` muss mit `0` initialisiert werden, da sonst `a[0]` vernachlässigt wird.

- (c) Beschreiben alle Kommentare ab Zeile 24 die Semantik des Codes korrekt? Geben Sie zu jedem falschen Kommentar einen korrigierten Kommentar mit Zeilennummer an.

| Zeile | Kommentar im Code | Korrektur |
|-------|--|--|
| 34-35 | // Die Schleife wird verlassen, wenn keine der beiden Haelften das Element enthaelt. | // Die Schleife wird verlassen, wenn keine der beiden Haelften das Element enthaelt oder das Element gefunden wurde. |
| 44 | // Ausschluss der oberen Haelfte | // Ausschluss der unteren Haelfte |
| 47 | // Ausschluss der unteren Haelfte | // Ausschluss der oberen Haelfte |
| 50 | // Das gesuchte Element wird zurueckgegeben | // Der Index des gesuchten Elements wird zurueckgegeben |

- (d) Geben Sie den Kontrollflussgraphen für die Methode `binarySearch` an.
- (e) Geben Sie maximal drei Testfälle für die Methode `binarySearch` an, die insgesamt eine vollständige Anweisungsüberdeckung leisten.

Die gegebene Methode: `binarySearch(a[], item)`

Testfall

(i) Testfall: $a[] = \{1, 2, 3\}$, $item = 4$

(ii) Testfall: $a[] = \{1, 2, 3\}$, $item = 2$

Thema Nr. 2

Teilaufgabe Nr. 2