Hinweise zur Bearbeitung der Klausur zum Kurs 1664 „Implementierungskonzepte für Datenbanksysteme“

Bitte lesen Sie sich diese Hinweise vollständig und aufmerksam durch, bevor Sie mit der Bearbeitung der Klausur beginnen.

1. Die Klausurdauer beträgt 2 Stunden.
2. Prüfen Sie bitte die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst
   - 2 Deckblätter
   - diese Hinweise
   - 1 Formblatt für eine Teilnahmebescheinigung zur Vorlage beim Finanzamt
   - 5 Aufgaben
3. Bevor Sie mit der Bearbeitung der Klausuraufgaben beginnen, füllen Sie bitte die folgenden Teile der Klausur aus:

   (a) *sämtliche* Deckblätter mit Name, Anschrift sowie Matrikelnummer. Markieren Sie vor der Abgabe auf allen Deckblättern die von Ihnen bearbeiteten Aufgaben.

   (b) die Teilnahmebescheinigung, falls Sie diese wünschen.
4. Schreiben Sie Ihre Lösungen auf Ihr eigenes Papier (DIN A4) und nicht auf die Seiten mit den Aufgabenstellungen. Heften Sie vor Abgabe der Klausur die Deckblätter, die Klausuraufgaben und ggf. die Teilnahmebescheinigung an Ihre Bearbeitung. **Sie dürfen die Klausuraufgaben nicht mitnehmen!**
5. Schreiben Sie bitte auf jedes Blatt oben links Ihren Namen und oben rechts Ihre Matrikelnummer. Nummerieren Sie Ihre Seiten bitte durch.
6. Als Hilfsmittel sind nur unbeschriebenes Konzeptpapier und Schreibzeug (kein Bleistift!) zugelassen.

Es sind insgesamt 100 Punkte erreichbar. Wenn Sie mindestens 50 Punkte erreicht haben, haben Sie die Klausur bestanden.

Aufgabe 1 (Datenbanksysteme) ***23 Punkte***

(a) Welche Anforderungen werden an moderne Datenbanksysteme gestellt? *10 Punkte*

(b) Wie setzen sich die Kosten für die Übertragung vom externen Speicher zusammen? *3 Punkte*

(c) Nennen Sie die drei grundlegenden Formen der Dateiorganisation in Datenbanksystemen und geben Sie ihre jeweiligen Stärken und Schwächen an. *10 Punkte*

Aufgabe 2 (Dynamisches Hashing) ***26 Punkte***

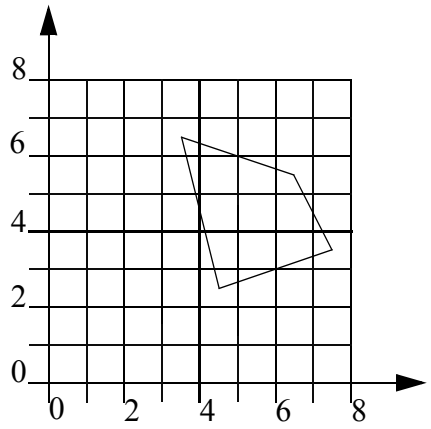
In dieser Aufgabe soll eine Menge von Tupeln mittels dynamischen Hashings indiziert werden. Wir nehmen an, dass ein Behälter lediglich ein einzelnes Element aufnehmen kann. Die verwendete Hashfunktion liefert die unten angegebenen Werte. Fügen Sie die Elemente in der gegebenen Reihenfolge in eine Hashtabelle ein, die mittels dynamischen Hashings verwaltet wird. Geben Sie den Inhalt der Indexstruktur nach jedem Einfügen an. Leere Behälter und deren Vorgänger müssen nicht mit angegeben werden.

Element	Hashwert
E_1	10001
E_2	01011
E_3	10011
E_4	11100
E_5	00111
E_6	10101

Aufgabe 3 (z-Ordnung) ***25 Punkte***

Gegeben sei eine z-Ordnung auf dem Bereich $[0, 8] \times [0, 8]$, in der die Gitterzellen der tiefsten Ebene eine Ausdehnung von 1×1 Koordinateneinheiten haben.

(a) Stellen Sie das Viereck mit den Eckpunkten (4.5, 2.5), (7.5, 3.5), (6.5, 5.5) und (3.5, 6.5) durch seine z-Elemente dar (siehe Skizze). Ermitteln Sie die zugehörigen Bitstrings dabei durch Bitverschachtelung (*bit interleaving*). *13 Punkte*



(b) Fügen Sie die *z*-Elemente aus Teil (a) anhand der Bitstrings in einen B^+-Baum der Ordnung $m = 2$ (für innere und Blattknoten) ein und führen Sie eine Bereichsanfrage für das achsenparallele Rechteck *f* mit den gegenüberliegenden Eckpunkten (5, 5) und (7, 8) durch. *12 Punkte*

Aufgabe 4 (Join-Algorithmus) *25 Punkte*

In dieser Aufgabe sollen Sie einen Algorithmus für einen Equi-Join entwerfen. Dieser soll nicht auf den Relationen selbst, sondern auf darüber angelegten Indexen durchgeführt werden. Diese besitzen Iteratoren, die einen geordneten Durchlauf über das indizierte Attribut erlauben und die folgenden Funktionen anbieten:

`void init()` : setzt den Iterator **vor** das erste Element im Index

`bool next()` : geht zum nächsten Element -- falls dieses nicht existiert, ist die Rückgabe dieses Operators `false`

`:=` : der übliche Zuweisungsoperator

`AttrType key()` : liefert den aktuellen Attributwert

`Tuple tuple()` : holt das aktuelle Tupel aus der Relation

Weiterhin gibt es die folgenden Operationen:

`Tuple concat(Tuple` t_1`, Tuple` t_2`)` : hängt t_2 an t_1 an

`append(Relation r, Tuple t)`: fügt das Tupel `t` in `r` ein

Geben Sie einen Algorithmus an, der unter Ausnutzung von Indexen auf zwei Relationen einen Equi-Join berechnet. Dieser soll in $O(n + m + e)$ Zeit durchführbar sein, wobei *m* und *n* die Tupelanzahlen der beteiligten Relationen sind und *e* die Er-

gebnisgröße ist. Als Eingabe erhält der Algorithmus Iteratoren für die Indexe. Beachten Sie, dass in beiden Indexen Einträge mehrfach auftreten können.

Aufgabe 5 Deckblatt *1 Punkt*

Lesen Sie sich die „Hinweise zur Bearbeitung“ sorgfältig durch. Füllen Sie den dort aufgeführten Anweisungen entsprechend beide Deckblätter vollständig und korrekt aus.