Fachbereich für Computerwissenschaften

Prof. Dr. Nikolaus Augsten
Jakob-Haringer-Str. 2
5020 Salzburg, Austria
Telefon: +43 662 8044 6347
E-Mail: nikolaus.augsten@sbg.ac.at

UNIVERSITÄT SALZBURG

Datenbanken II
Wintersemester 2018/2019

Prüfung
06.02.2019

Name: _____ **Matrikelnummer:** _____

Hinweise

- Bitte überprüfen Sie die Vollständigkeit des Prüfungsbogens (12 nummerierte Seiten).
- Schreiben Sie Ihren Namen und Ihre Matrikelnummer auf jedes Blatt des Prüfungsbogens und geben Sie alle Blätter ab.
- Grundsätzlich sollten Sie alle Antworten auf den Prüfungsbogen schreiben.
- Sollten Sie mehr Platz für eine Antwort benötigen, bitte einen klaren Verweis neben die Frage auf die Seitennummer des zusätzlichen Blattes setzen.
- Keinen Bleistift verwenden. Keinen roten Stift verwenden.
- Verwenden Sie die Notation und die Lösungsansätze, die während der Vorlesung besprochen wurden.
- Aufgaben mit mehr als einer Lösung werden nicht bewertet.
- Als Unterlage ist ein beliebig (auch beidseitig) beschriftetes A4-Blatt erlaubt.
- Zeit für die Prüfung: 90 Minuten

Unterschrift _____

Korrekturabschnitt — Bitte frei lassen

Aufgabe	1	2	3	4	5	6	7	8	9	10	Summe
Maximale Punkte	1	1	1	1	1	1	1	1	1	1	10
Erreichte Punkte											

Aufgabe 1 - *Indexstrukturen.* 1 Punkt

Zeichnen Sie für die folgende Tabelle einen **2-stufigen Sekundärindex** auf dem Attribut `Player`. **Beide Indexstufen** sollen **sparse** sein. Ein **Datenblock** kann **2 Tupel** speichern. In einen **Indexblock** können **5 Einträge** gespeichert werden.

`Player`	`Team`
Bell	Steelers
Brees	Saints
Brown	Steelers
Donald	Rams
Edelman	Patriots
Fournette	Jaguars
Gurley	Rams
Houston	Chiefs
Jones	Falcons
Kuechly	Panthers
Lattimore	Saints
Leonard	Colts
Mack	Bears
Michel	Colts
Sanu	Falcons
Thomas	Saints

Aufgabe 2 - *Slotted Page*. 1 Punkt

Gegeben sei eine Slotted Page mit folgenden Eigenschaften:

- Größe: $2^{12} = 4096$ Bytes,
- Adressierungstyp: **Word-Adressierung** (d.h. es kann nur jedes 2. Byte adressiert werden)

In dieser Slotted Page werden **3 Tupel** U, C, K gespeichert:

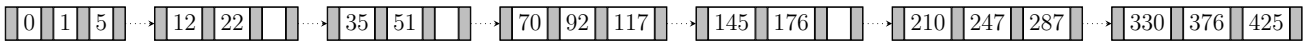
- d_1: $|U| = 827$ Bytes
- d_2: $|C| = 722$ Bytes
- d_3: $|K| = 242$ Bytes

Ergänzen Sie die Slotted Page um die **fehlenden Werte/Adressen**, wobei p_i und g_i sich auf den jeweiligen Datensatz d_i beziehen. **(0.125 Punkte pro Wert/Adresse)**

a	f	g_1	p_1	g_2	p_2	g_3	p_3	$\ldots$	d_1	d_2	d_3
									U	C	K

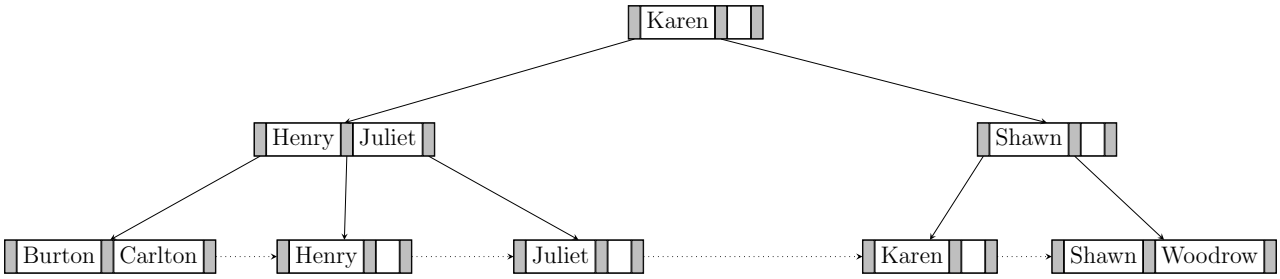
Aufgabe 3 - *B^+-Baum-Konstruktion.* 1 Punkt

Gegeben sind die Blätter eines B^+-Baumes ($m = 4$). Konstruieren Sie die **darüberliegenden Ebenen** (d.h. die inneren Knoten) so, dass ein **gültiger B^+-Baum mit minimaler Höhe** entsteht.



Aufgabe 4 - *B⁺-Baum-Löschen.* 1 Punkt

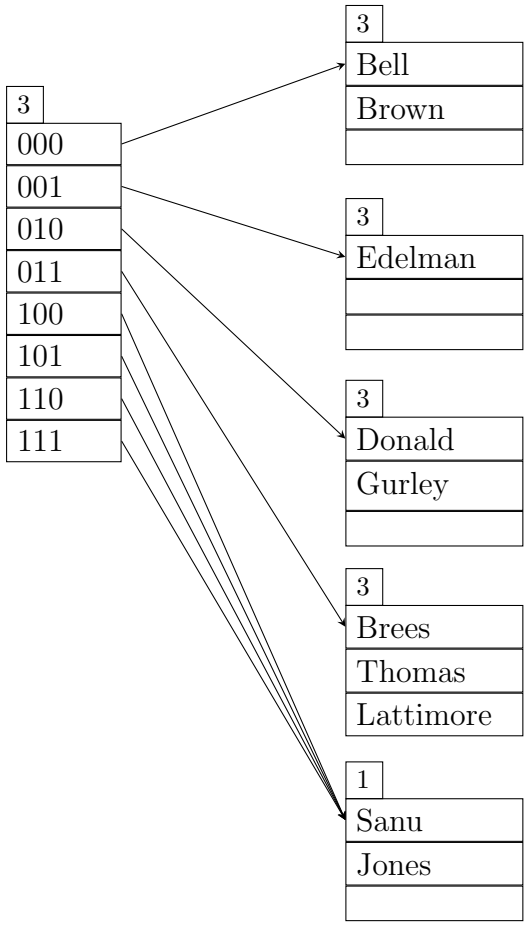
Gegeben ist ein B$^+$-Baum mit $m = 3$. Zeichnen Sie den B$^+$-Baum der nach dem Löschen von **Karen** entsteht.



Aufgabe 5 - *Erweiterbares Hashing*. 1 Punkt

Die Hashfunktion $h(x)$ liefert die in der Tabelle angegebenen Binärwerte. Es soll das Tupel *Donald* aus dem gegebenen Hashcontainer gelöscht werden. Ein Bucket im Hashcontainer kann bis zu 3 Tupel speichern. Das Verzeichnis soll so klein wie möglich gehalten werden. **Illustrieren Sie den resultierenden Hashcontainer**.

x	$h(x)$
Bell	0000
Brees	0110
Brown	0001
Donald	0100
Edelman	0011
Gurley	0101
Jones	1110
Lattimore	0111
Sanu	1100
Thomas	0110



Aufgabe 6 - *Externes Merge-Sort*. 1 Punkt

Die Relation R soll mittels **externem Merge-Sort** sortiert werden. Gegeben ist die Formel zur Berechnung der Anzahl an Blockzugriffen.

$$Kosten = \overbrace{b_R}^{(a)} \cdot (\underbrace{2}_{(b)} \cdot \overbrace{\left\lceil log_{M-1}(\underbrace{\frac{b_R}{M}}_{(d)})\right\rceil}^{(c)} + \underbrace{1}_{(e)})$$

Erklären Sie die mit Klammern gekennzeichneten Teile der Formel:

(a)

(b)

(c)

(d)

(e)

Aufgabe 7 - *Effiziente Anfragebearbeitung.* 1 Punkt

Gegeben sei eine Relation $R[A, B, C]$ mit folgenden Eigenschaften:

- $|R| = 4 \cdot 10^9$ Tupel,
- Pro Datenblock werden $8 \cdot 10^3$ Tupel gespeichert,
- Attribut A hat ganzzahlige Werte gleichverteilt im Bereich $[1; 4 \cdot 10^9]$,
- Attribut B hat ganzzahlige Werte gleichverteilt im Bereich $[1; 4 \cdot 10^6]$,
- Attribut C hat ganzzahlige Werte gleichverteilt im Bereich $[1; 4 \cdot 10^3]$,
- Duplikate werden mittels Tuple Identifier (TID) aufgelöst,
- Folgende Indizes existieren:
  - flacher sparse Index auf Attribut A mit 12.500 Einträgen pro Indexblock,
  - dense B$^+$-Baum-Index auf Attribut B, $m = 2.048$, minimale Höhe,
  - dense B$^+$-Baum-Index auf Attribut C, $m = 1.024$, minimale Höhe.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{B=4\cdot10^3 \wedge A \geq 4\cdot10^3}(R)$$

Geben Sie die **Strategie (0.5 Punkte)** an und berechnen Sie die **Anzahl der Blockzugriffe (0.5 Punkte)** um die Anfrage **möglichst effizient** zu beantworten (1 Knotenzugriff im B$^+$-Baum entspricht 1 Blockzugriff).

Aufgabe 8 - *Join-Algorithmen.* 1 Punkt

Welcher Join Algorithmus (**Hash Join**, **Index Nested Loop Join**) generiert die minimalen Kosten für das folgende Szenario? Geben Sie in der Lösung die **Algorithmen und** die **dazugehörigen Kosten** an.

Berechnen Sie einen Natural Join zwischen zwei Relationen $R[A, B]$ und $S[B, C]$, wobei $|R| = 4000$ Tupel und $|S| = 1000$ Tupel. Die Relationen sind auf $b_R = 500$ bzw. $b_S = 1000$ hintereinander liegenden Blöcken gespeichert. Der Buffer hat Platz für $M = 21$ Blöcke. Es existiert ein sparse B^+-Baum Index auf $R.B$, wobei jeder Knoten im B^+-Baum 20 Schlüssel speichern kann.

Aufgabe 9 - *Bitmap Index Scan.* 1 Punkt

Gegeben sei eine Relation $R[A, B, C]$ mit folgenden Eigenschaften:

- $|R| = 10^6$ Tupel gespeichert auf $b_R = 10^3$ Blöcken,
- Non-clustering B^+-Baum-Index auf Attribut B, $m = 1.024$, maximale Höhe (d.h. die Knoten sind minimal befüllt),
- Duplikate werden mittels Tuple Identifier (TID) aufgelöst,
- Attribut B hat insgesamt 10^3 mal den Wert 2.500, verteilt auf 100 Datenblöcke.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{B=2.500}(R)$$

Geben Sie die Anzahl an **Datenblockzugriffen (worst case)**

a. **ohne Anwendung** eines Bitmap Index Scans **(0.5 Punkte)**,
b. **unter Anwendung** eines Bitmap Index Scans **(0.5 Punkte)**

an.

Aufgabe 10 - *Anfrageoptimierung, Join-Reihenfolge.* 1 Punkt

Gegeben seien 3 Relationen $R[A, B, C]$, $S[C, D, E]$ und $T[C, F, G]$ mit folgenden Eigenschaften:

- $|R| = 10^5$ Tupel, $V(R, A) = 10^3$, $V(R, B) = 4 \cdot 10^3$, $V(R, C) = 10^4$
- $|S| = 10^7$ Tupel, $V(S, C) = 2 \cdot 10^3$, $V(S, D) = 5 \cdot 10^3$, $V(S, E) = 8 \cdot 10^3$
- $|T| = 10^9$ Tupel, $V(T, C) = 10^9$, $V(T, F) = 6 \cdot 10^3$, $V(T, G) = 9 \cdot 10^3$

Weiters sei folgende SQL-Anfrage gegeben:

```
SELECT R.C, S.E, T.F
FROM   R, S, T
WHERE  R.C = S.C
       AND S.C = T.C
```

a. Zeichnen Sie die **algebraische Normalform als Operatorbaum** für die gegebene SQL-Anfrage. **(0.5 Punkte)**
b. Wenden Sie **heuristische Optimierung** an, um den **Operatorbaum zu optimieren**. Im resultierenden Operatorbaum soll die **Join-Reihenfolge optimal** sein (d.h. es soll zuerst der Join mit dem kleinsten Zwischenergebnis durchgeführt werden). **(0.5 Punkte)**