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 2020/2021

Prüfung
31.05.2021

Name: _____ **Matrikelnummer:** _____

Hinweise

- Bitte überprüfen Sie die Vollständigkeit des Prüfungsbogens (11 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 Clustered Index** 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: $2^{13} = 8192$ Bytes,
- Adressierungstyp: **Byte-Adressierung** (es kann jedes Byte adressiert werden)

In dieser Slotted Page werden **3 Tupel** O, P, Q gespeichert:

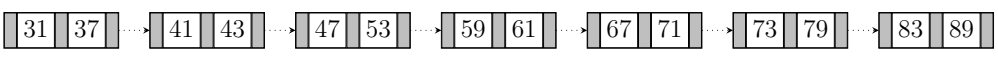
- d_1: $|O| = 64$ Bytes
- d_2: $|P| = 128$ Bytes
- d_3: $|Q| = 256$ Bytes

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

a	f	g_3	p_3	g_2	p_2	g_1	p_1	$\ldots$	d_1	d_2	d_3
									O	P	Q

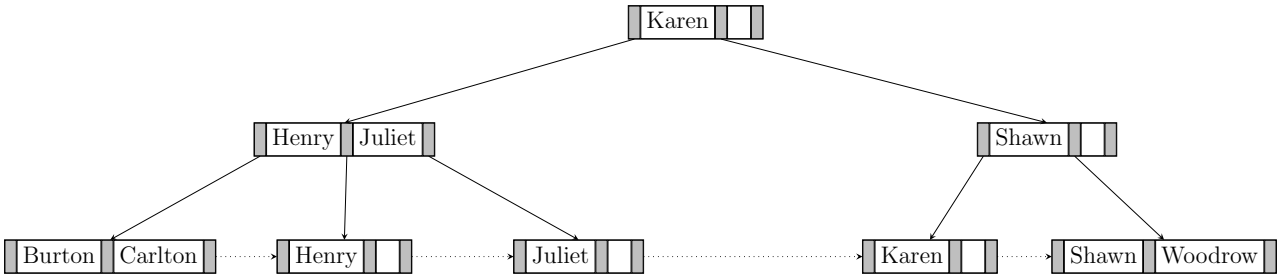
Aufgabe 3 - *B^+-Baum-Konstruktion.* 1 Punkt

Gegeben sind die Blätter eines B^+-Baumes ($m = 3$). 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

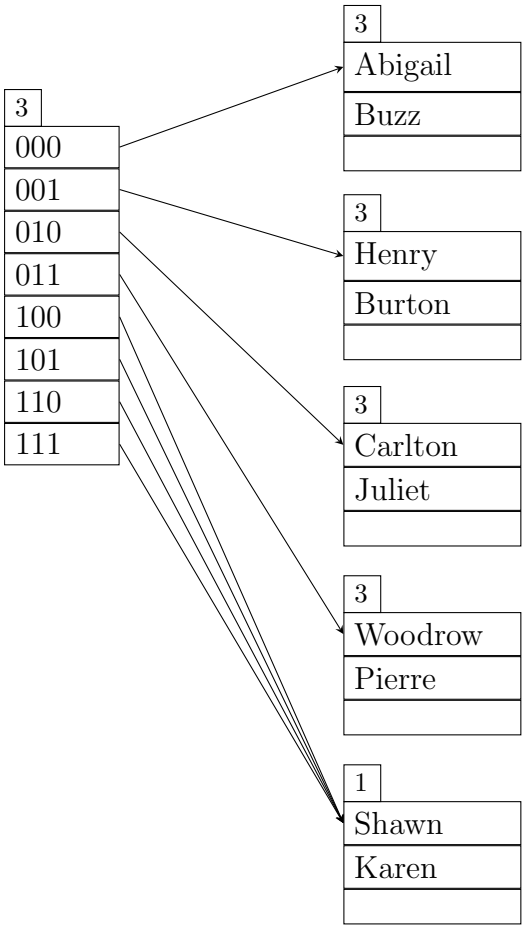
Gegeben ist ein B$^+$-Baum mit $m = 3$. Zeichnen Sie den B$^+$-Baum der nach dem Löschen von **Karen** entsteht.



Aufgabe 5 - *Erweiterbares Hashing*. 1 Punkt

Die Hashfunktion $h(x)$ liefert die in der Tabelle angegebenen Binärwerte. Es soll das Tupel *Henry* 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	$h(x)$
Abigail	0000
Burton	0010
Buzz	0001
Carlton	0100
Henry	0011
Juliet	0101
Karen	1110
Pierre	0111
Shawn	1100
Woodrow	0110



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

Betrachten Sie eine Relation $R[A]$ mit 2000 Tupel. Ein Block hat Platz für zwei Tupel. Der Puffer kann 10 Blöcke speichern.

Wieviele Blöcke müssen gelesen/geschrieben werden um einen externen Merge-Sort auf Relation R auszuführen? Zählen Sie dabei den letzten Schreib-Schritt, der das Ergebnis wieder auf die Festplatte schreibt, **nicht** mit.

Aufgabe 7 - *Effiziente Anfragebearbeitung*. 1 Punkt

Gegeben ist die Relation $R[A, B]$. Auf $R.A$ existiert ein sparse B$^+$-Baum Index und auf $R.B$ existiert ein dense Hash-Index. Die Werte für Attribut B sind eindeutig. Was ist die **effizienteste Strategie** um Anfragen von folgendem Typ zu beantworten?

$$\sigma_{A<a \,\vee\, B=b}(R)$$

Geben Sie **alle notwendigen Schritte** an.

Aufgabe 8 - *Join-Algorithmen.* 1 Punkt

Gegeben seien **zwei Relationen** mit folgenden Eigenschaften:

$R[A, B, C]$:

- $|R| = 10^7$ Tupel, gespeichert auf $b_R = 20 \cdot 10^3$ Datenblöcken
- **dense** B$^+$-Baum-Index auf Attribut A, $m = 2^8$
- **sparse** B$^+$-Baum-Index auf Attribut B, $m = 2^7$
- Die B$^+$-Bäume besitzen **maximale Höhe**.

$S[B, D, F]$:

- $|S| = 5 \cdot 10^6$ Tupel, gespeichert auf $b_S = 4 \cdot 10^3$ Datenblöcken
- **einstufiger dense** Index (ISAM) auf Attribut B mit $5 \cdot 10^4$ Indexblöcken
- **einstufiger sparse** Index (ISAM) auf Attribut D mit 40 Indexblöcken

Es soll ein **Natural Join** $R \bowtie S$ unter Anwendung des **Indexed-Nested-Loop-Join-Algorithmus** durchgeführt werden.

Geben Sie hierfür die effizienteste Join-Reihenfolge ($R \bowtie S$ oder $S \bowtie R$) sowie die zugehörigen **Kosten (Blockzugriffe)** an. Ein Knotenzugriff im B$^+$-Baum entspricht einem Blockzugriff. Duplikate können für diese Aufgabe vernachlässigt werden.

Aufgabe 9 - *Join Kardinalität.* 1 Punkt

Gegeben sind folgende Relationen:

- $|R[A, B, C]| = 1000$, $V(R, A) = 100$, $V(R, B) = 200$, $V(R, C) = 300$
- $|S[A, D, E]| = 4000$, $V(S, A) = 50$, $V(S, D) = 200$, $V(S, E) = 300$
- $|T[D, E, F]| = 2000$, $V(T, D) = 200$, $V(T, E) = 400$, $V(T, F) = 600$

Die Werte in den Tupeln sind gleichverteilt und unabhängig.

Schätzen Sie die Kardinalität des Ergebnisses der folgenden Anfrage ab ($\sigma_{A=100}(R) \neq \emptyset$).

$$(\sigma_{A=100}(R)) \bowtie S \bowtie T$$

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

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

- $|R| = 2000$ Tupel, $V(R, A) = 100$, $V(R, C) = 20$, $V(R, D) = 80$
- $|S| = 300$ Tupel, $V(S, C) = 200$, $V(S, E) = 50$
- $|T| = 500$ Tupel, $V(T, B) = 20$, $V(T, C) = 50$, $V(T, F) = 30$

Weiters sei folgende SQL-Anfrage gegeben:

```
SELECT R.A, S.C, T.B
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)**