Fachbereich Informatik

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

PARIS LODRON UNIVERSITÄT SALZBURG

Datenbanken II — Prüfung
Wintersemester 2023/2024 — 10.01.2024

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 - *Slotted Page*. 1 Punkt

In die folgende Tabelle werden Werte eingefügt:

```
CREATE TABLE books (
  bid INTEGER,
  btitle VARCHAR(20)
);
```

Das Speichern eines `INTEGER` benötigt 4 Byte. Strings, die als `VARCHAR` gespeichert werden, benötigen ein Byte pro Zeichen und zusätzlich ein Byte zur Terminierung. Beispielsweise benötigt die Zeichenfolge `DBMS` 5 Bytes zur Speicherung.

Die Tupel werden in eine Slotted Page mit folgenden Eigenschaften eingefügt:

- Größe: $2^{13} = 8192$ Bytes
- Adressierungstyp: **Byte-Adressierung** (es kann jedes Byte adressiert werden)

Die folgenden Operationen werden in dieser Reihenfolge durchgeführt:

INSERT INTO books VALUES (42, 'Categoriae'); – Tupel A
INSERT INTO books VALUES (13, 'Analytica-priora'); – Tupel B
INSERT INTO books VALUES (37, 'De-sophisticis-elenchis'); – Tupel C

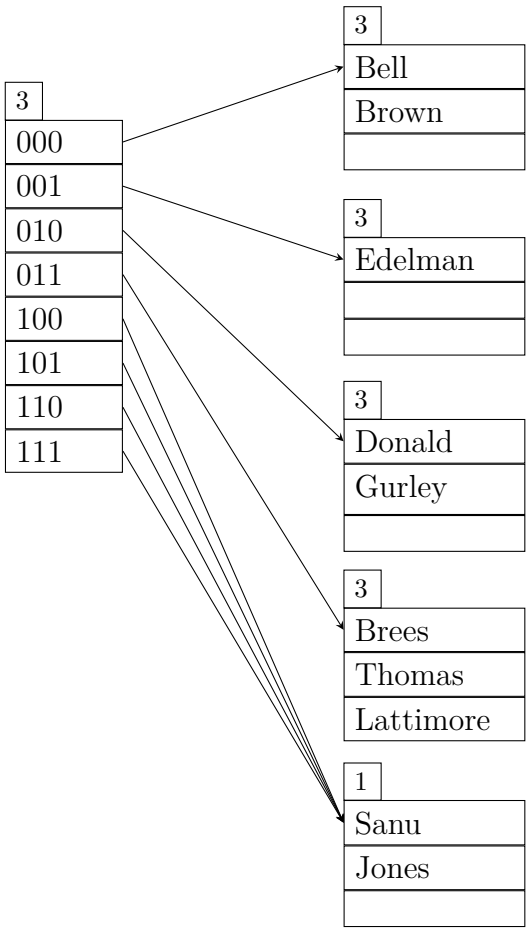
Ergänzen Sie die Slotted Page um die **fehlenden Werte/Adressen** (numerische Werte erwartet, Pfeile reichen nicht aus), wobei p_i und g_i sich auf den jeweiligen Datensatz d_i beziehen.

a	f	g_1	p_1	g_2	p_2	g_3	p_3	...	d_3	d_2	d_1
									C	B	A

Aufgabe 2 - *Erweiterbares Hashing*. 1 Punkt

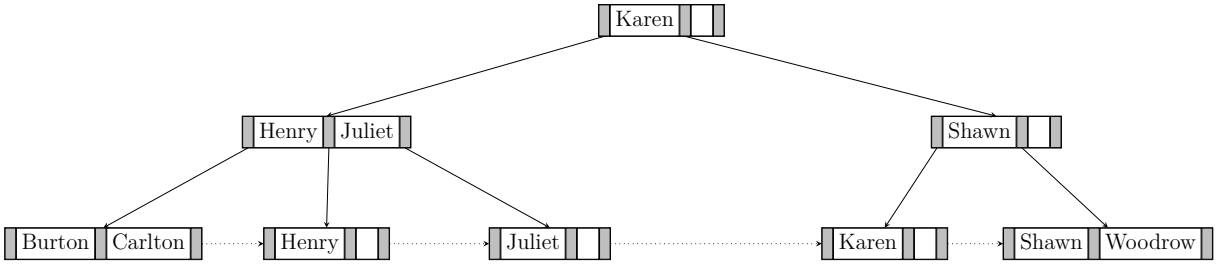
Die Hashfunktion $h(x)$ liefert die in der Tabelle angegebenen Binärwerte. Es soll das Tupel **Donald** aus dem gegebenen Hashcontainer gelöscht werden. Ein Bucket im Hashcontainer kann bis zu 3 Tupel speichern. Das Verzeichnis soll so klein wie möglich gehalten werden. **Illustrieren Sie den resultierenden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

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



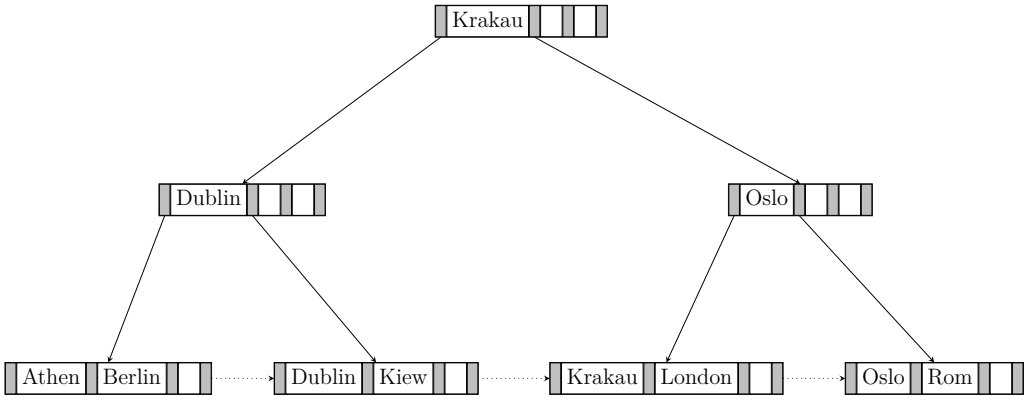
Aufgabe 3 - *B⁺-Baum Einfügen.* 1 Punkt

Gegeben ist ein B⁺-Baum mit $m = 3$. Zeichnen Sie den B⁺-Baum, der nach dem Einfügen von **Dexter** entsteht.



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

Gegeben ist ein B^+-Baum mit $m = 4$. Zeichnen Sie den B^+-Baum, der nach dem Löschen von **Krakau** entsteht.



Aufgabe 5 - *Indexstrukturen.* 1 Punkt

Zeichnen Sie für die folgende Tabelle einen **3-stufigen Sekundärindex** auf dem Attribut `Stadt`. Die **innere Indexstufe** soll **dense** und die **äußeren beiden Indexstufen** sollen **sparse** sein. In einen **Indexblock** können **3 Einträge** gespeichert werden.

`Stadt`	`KFZ`
Rom	I
London	GBM
Prag	CZ
Kiew	UA
Berlin	D
Athen	GR
Krakau	PL
Oslo	N
Dublin	IRL
Wien	A

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

Führen Sie **externes Merge-Sort** auf der folgenden Relation $R[A]$ aus.
Jeder **Block fasst 4 Tupel**. Die Größe des **Puffer** beträgt **4 Blöcke**.

100
35
10
45
95
20
25
5
30
90
40
0
15
50
80
60
55
75
65
70

Aufgabe 7 - *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 **natürlichen Join** zwischen zwei Relationen $R[A, B]$ und $S[B, C]$ mit $|R| = 1000$ Tupel und $|S| = 12000$ Tupel. Die Relationen sind auf $b_R = 250$ bzw. $b_S = 2000$ hintereinander liegenden Blöcken gespeichert. Der Buffer hat Platz für $M = 21$ Blöcke. Es existiert ein **sparse** B^+-Baum Index auf $S.B$, wobei jeder Knoten im B^+-Baum 20 Schlüssel speichern kann.

Aufgabe 8 - *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 9 - *Schätzung der Join-Kardinalität.* 1 Punkt

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

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

Die Werte in den Tupeln sind gleichverteilt und unabhängig. Schätzen Sie die Kardinalität der folgenden Abfrage ab $(\sigma_{A=100 \vee A=200}(R) \neq \emptyset)$:

$$(\sigma_{A=100 \vee A=200}(R)) \bowtie S \bowtie T$$

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

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

- $|R| = 1.200$ Tupel, $V(R, A) = 50$, $V(R, B) = 100$, $V(R, C) = 200$
- $|S| = 3.000$ Tupel, $V(S, A) = 20$, $V(S, D) = 1.000$, $V(S, E) = 600$
- $|T| = 5.000$ Tupel, $V(T, A) = 100$, $V(T, F) = 1.200$, $V(T, G) = 1.800$

Weiters sei die folgende SQL-Anfrage gegeben:

```
SELECT      R.A, S.D, T.G
FROM        R, S, T
WHERE       R.A = S.A
            AND R.A = T.A
```

a. Zeichnen Sie die **algebraische Normalform als Operatorbaum** für die gegebene SQL-Anfrage.
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).