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

PARIS LODRON UNIVERSITÄT SALZBURG

Datenbanken II — Prüfung
Wintersemester 2021/2022 — 22.02.2022

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

Gegeben sei eine Slotted Page mit folgenden Eigenschaften:

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

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

- d_1: $|U| = 222$ Bytes
- d_2: $|C| = 333$ Bytes
- d_3: $|K| = 444$ Bytes

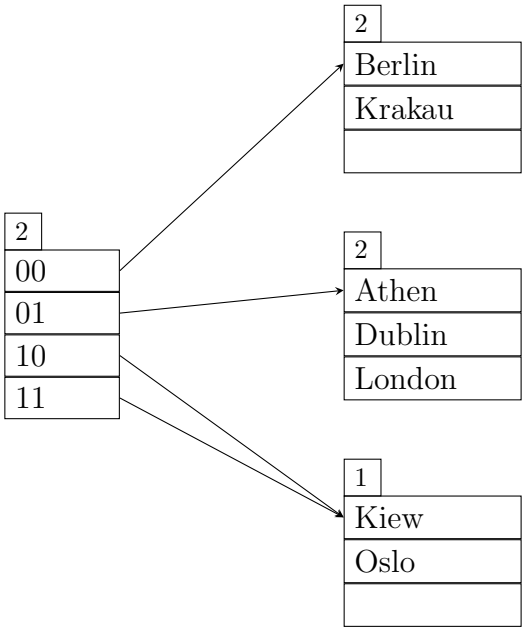
Ergänzen Sie die Slotted Page um die **fehlenden Werte/Adressen**, wobei p_i und g_i sich auf den jeweiligen Datensatz d_i beziehen. **(0.125 Punkte pro Wert/Adresse)**

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 2 - *Erweiterbares Hashing*. 1 Punkt

Die Hashfunktion $h(x)$ liefert die in der Tabelle angegebenen Binärwerte. Es soll das Tupel **Rom** in den gegebenen Hashcontainer eingefügt 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	$h(x)$
Athen	0100
Berlin	0010
Dublin	0100
Kiew	1100
Krakau	0011
London	0110
Oslo	1110
Rom	0101



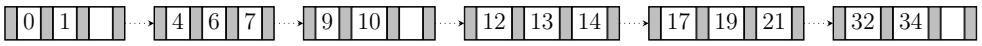
Aufgabe 3 - *Indexstrukturen*. 1 Punkt

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

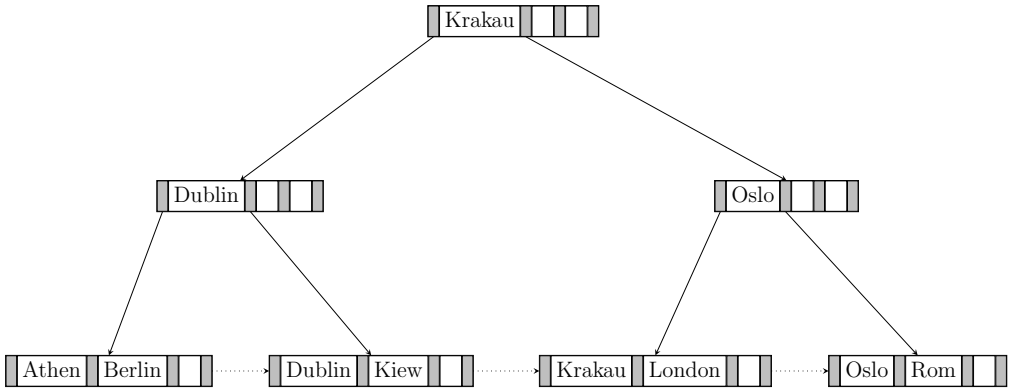
Aufgabe 4 - *B^+-Baum-Konstruktion.* 1 Punkt

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 5 - *B⁺-Baum-Löchen.* 1 Punkt

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



Aufgabe 6 - *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 7 - *Join-Algorithmen*. 1 Punkt

Gegeben seien zwei Relationen mit folgenden Eigenschaften:

- $R[A]$ mit Werten $1, 2, 3, 4, 5, 6, 7, 8, 9$,
- $S[A]$ mit Werten $2, 5, 6, 8, 9, 11, 13, 15, 16, 18, 19, 20$,
- die Größe des Puffer beträgt $M = 4$ Blöcke,
- jeder Block fasst 1 Tupel.

Finden Sie eine geeignete Hash-Funktion und zeichnen Sie die Partitionen für einen Hash Join für beide Relationen.

Aufgabe 8 - *Effiziente Anfragebearbeitung.* 1 Punkt

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

- $|R| = 10^7$ Tupel,
- Pro Datenblock werden 32 Tupel gespeichert,
- Attribut A hat ganzzahlige Werte gleichverteilt im Bereich $[1; 4 \cdot 10^4]$,
- Attribut B hat ganzzahlige Werte gleichverteilt im Bereich $[1; 3 \cdot 10^4]$,
- Duplikate werden mittels Tuple Identifier (TID) aufgelöst,
- Folgende Indizes existieren:
  - Clustered B$^+$-Baum-Index auf Attribut A, $m = 64$
  - Dense B$^+$-Baum-Index auf Attribut B, $m = 64$

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{A<8000 \wedge B>15000}(R)$$

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 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 $R[A, B, C]$, $S[A, D, E]$ und $T[A, F, G]$:

- $|R| = 1.200$ 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
```

1. Zeichnen Sie die **algebraische Normalform als Operatorbaum** für die gegebene SQL-Anfrage. **(0.5 Punkte)**
2. Wenden Sie die **heuristische Optimierung** an, um den Operatorbaum zu optimieren. **Es soll auch die Join-Reihenfolge optimiert werden! (0.5 Punkte)**