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 2025/26 — 28.01.2026

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.
- **Multiple-Choice:** Falsche Antworten führen zu Punktverlust!
- 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 - *Pufferverwaltung*. 1 Punkt

Ein System besitzt einen (initial leeren) Puffer, in den **4** Seiten passen. Als Ersetzstrategie wird **MRU** (most recently used) eingesetzt. Es wird auf die Seiten A,B,C,D,E in folgender Reihenfolge zugegriffen.

A B D D E A E C A B C A E D

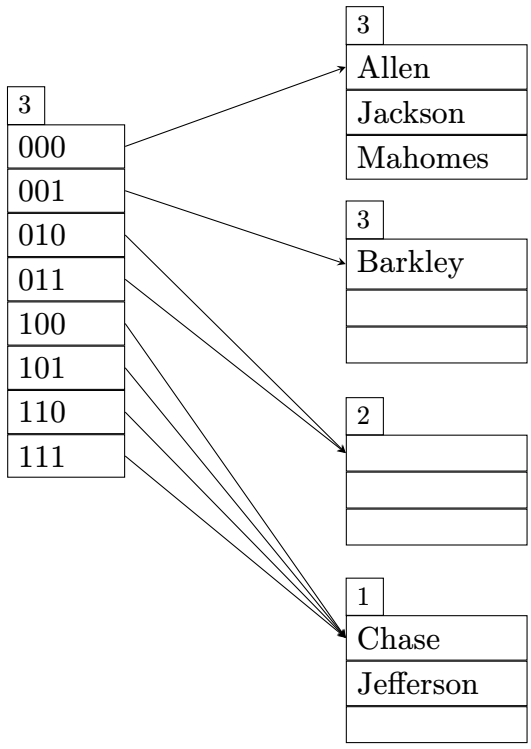
Wie viele Plattenzugriffe finden tatsächlich statt? Welche Blöcke befinden sich nach der Ausführung der Zugriffe im Puffer? Vervollständigen Sie dafür die folgende Tabelle. Puffertreffer werden mit "*" markiert.

Pufferseite 1	A												
Pufferseite 2		B											
Pufferseite 3			D	*									
Pufferseite 4													

Aufgabe 2 - *Erweiterbares Hashing*. 1 Punkt

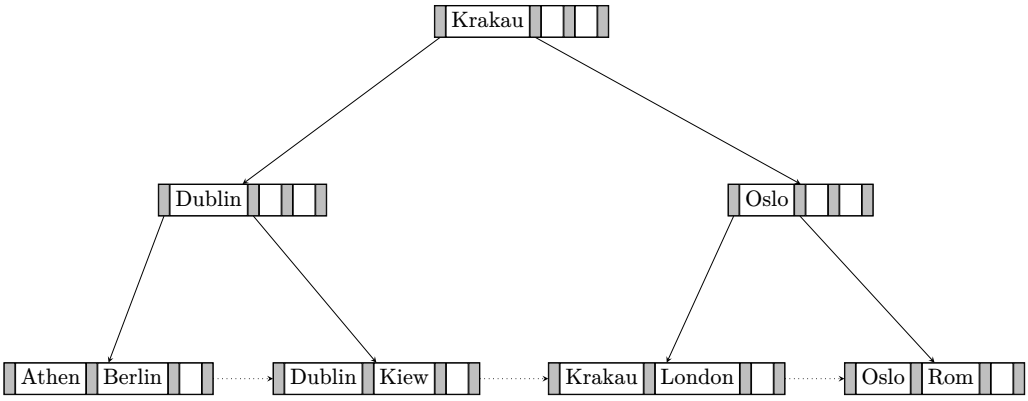
Die Hashfunktion $h(x)$ liefert die in der Tabelle angegebenen Binärwerte. Es soll das Tupel **Jackson** aus dem gegebenen Hashcontainer gelöscht werden. Ein Bucket im Hashcontainer kann bis zu 3 Tupel speichern. Das Verzeichnis soll so klein wie möglich gehalten werden. **Illustrieren Sie den resultierenden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

x	$h(x)$
Allen	00001
Barkley	00101
Chase	10010
Henry	01101
Jackson	00010
Jefferson	10101
Mahomes	00011



Aufgabe 3 - 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 4 - *Indexstrukturen mit kombiniertem Suchschlüssel.* 1 Punkt

Gegeben ist ein geordneter Index mit kombiniertem Suchschlüssel auf die Attribute *(BrName, AccName, Bal)* in dieser Reihenfolge. Geben Sie für die folgenden Anfrageprädikate an, ob eine effiziente Nutzung des Index zur Beantwortung der Anfrage möglich ist, und begründen Sie jeweils kurz Ihre Antwort.

(1) **WHERE** AccName="John" **AND** BrName="Salzburg"

(2) **WHERE** AccName="Smith" **AND** 10000<Bal<20000

(3) **WHERE** BrName="Munich" **AND** AccName="Doe" **AND** 10000<Bal<20000

Aufgabe 5 - *Externes Merge-Sort*. 1 Punkt

Führen Sie **externes Merge-Sort** auf der folgenden Relation $R[A]$ aus.
Jeder **Block fasst 2 Tupel**. Die Größe des **Puffer** beträgt **4 Blöcke**.

11
25
10
0
13
9
7
22
18
5
16
3
24
20
12
8
6
2
19
14
4
21
23
17
15
1

Aufgabe 6 - *Pipelining*. 1 Punkt

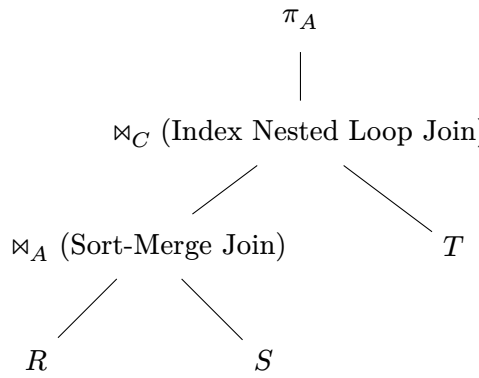
Betrachten Sie die Relationen $R[A, B]$, $S[A, C]$ und $T[C, D]$.

- R: 10^6 Tupel mit einem **sparse** B$^+$-Baum-Index auf A.
- S: $5 \cdot 10^5$ Tupel.
- T: $2 \cdot 10^6$ Tupel mit einem **dense** B$^+$-Baum-Index auf C.

Die logische Anfrage lautet: $\pi_A((R \bowtie_{R.A=S.A} S) \bowtie_{S.C=T.C} T)$.

Hinweis: Wie in der relationalen Algebra üblich (und im Gegensatz zu SQL), entfernt die Projektion π_A Duplikate.

Der Datenbank-Optimierer erstellt den folgenden physischen Anfrageplan inklusive Annotationen für die physischen Operatoren.



(1) Beschriften Sie im obigen Baum alle Kanten entweder als **blocking** oder **pipelining**. (Bei einer blockierenden Kante muss der konsumierende Operator warten, bis der produzierende Operator vollständig fertig ist, bevor er das erste Tupel ausgeben kann.)

(2) Wie können Tupel in π_A effizient dedupliziert werden?

(3) Nehmen Sie an, wir ersetzen den **Index Nested Loop Join** durch einen **Hash Join** für den Join $S \bowtie_C T$. Wie beeinflusst dies Ihre Antworten auf (1) und (2)?

Aufgabe 7 - *Effiziente Anfragebearbeitung.* 1 Punkt

Gegeben ist die Relation $R[A]$. Auf $R.A$ existiert ein sparse B^+-Baum Index. Was ist die **effizienteste Strategie** um Bereichsanfragen von folgendem Typ zu beantworten?

$$\sigma_{a<A<b}(R)$$

Geben Sie **alle notwendigen Schritte** an.

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

Gegeben seien 3 Relationen $R[A, B, C]$, $S[A, D, E]$, $T[D, E, F]$ 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, $V(S, A) = 50$, $V(S, D) = 200$, $V(S, E) = 300$
- $|T[D, E, F]| = 2000$ Tupel, $V(T, D) = 200$, $V(T, E) = 400$, $V(T, F) = 600$

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

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

Aufgabe 9 — 1 Punkt

Antworten Sie knapp. Geben Sie für jede untenstehende Anomalie an: (a) einen Schedule mit R-W Operationen und Commit/Rollback, (b) eine kurze Erklärung, warum dies problematisch sein kann, und (c) das niedrigste SQL-Isolationslevel, das die Anomalie verhindert (wählen Sie zwischen: Read Uncommitted, Read Committed, Repeatable Read, Serializable).

(1) Dirty read

(2) Non-repeatable read

Aufgabe 10 — 1 Punkt

Betrachten Sie den folgenden Schedule unter **Two-Phase-Locking**.

T1:	T2:	T3:
read(X)		
	write(X)	
		read(X)
write(X)		
		write(X)

Prüfen Sie, ob dieser Schedule zu einem Deadlock führt. Falls ja, geben Sie an, was gemäß dem Wound-Wait Deadlock-Vermeidungsprotokoll mit den Transaktionen passiert (nehmen Sie an, dass TS(T1) $<$ TS(T2) $<$ TS(T3) gilt).