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 — 24.04.2026

**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 **LRU** (least 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

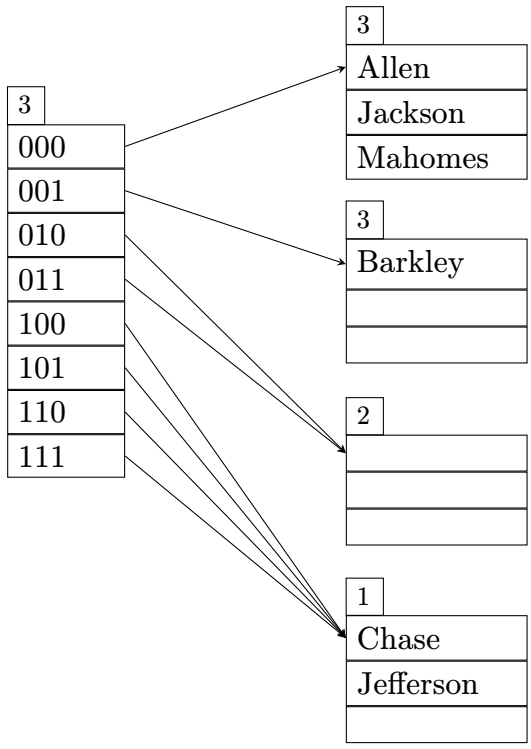
Wie viele Plattenzugriffe finden tatsächlich statt? Welche Blöcke befinden sich nach der Ausführung der Zugriffe im Puffer? Vervollständigen Sie dafür die folgende Tabelle. Puffertreffer werden mit "*" markiert.

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

Aufgabe 2 - *Erweiterbares Hashing*. 1 Punkt

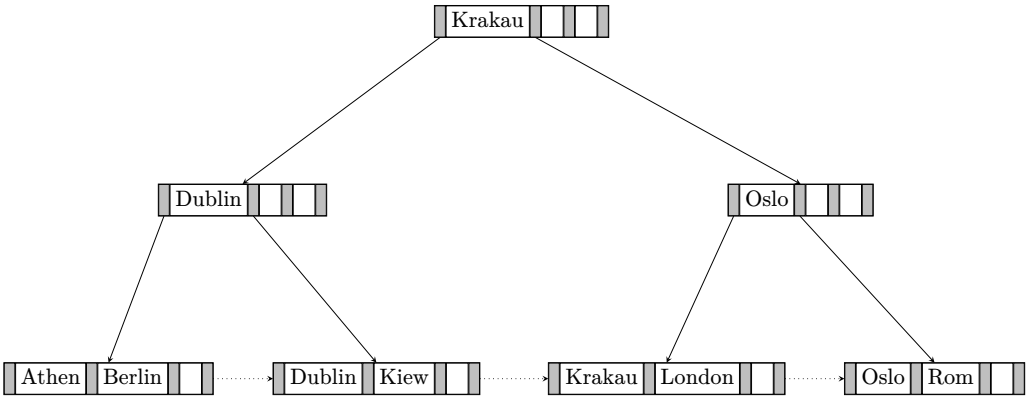
Die Hashfunktion $h(x)$ liefert die in der Tabelle angegebenen Binärwerte. Es soll das Tupel **Jackson** aus dem gegebenen Hashcontainer gelöscht werden. Ein Bucket im Hashcontainer kann bis zu 3 Tupel speichern. Das Verzeichnis soll so klein wie möglich gehalten werden. **Illustrieren Sie den resultierenden 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 - *Duplikate und Indexstrukturen.* 1 Punkt

Betrachten Sie die folgende (unsortierte) Relation $R[name]$ mit Duplikaten.

a) Zeichnen Sie einen **einstufigen, non-clustered Index** für Attribut *name* und lösen Sie Duplikate mittels **Buckets** auf. *Hinweis:* Obwohl die Relation Duplikate enthält, darf der erstellte Index keine Dupliakte in den Suchschlüsseln enthalten.

b) Duplikate können auch über **Tupel Identifier (TID)** aufgelöst werden. Erklären Sie, was ein **TID** ist, und **wie TIDs benutzt werden können** um in einem Index Duplikate in den Suchschlüsseln zu vermeiden.

$R[name]$:

| |
|---|
| Noah |
| George |
| Noah |
| Ivy |
| Harry |
| Noah |
| George |
| Lily |
| Ivy |
| Florence |
| Noah |
| George |

## Aufgabe 5 - *Bitmap Index Scan.* 1 Punkt

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

- $|R| = 10^6$ Tupel gespeichert auf $b_R = 10^3$ Blöcken,
- Non-clustering $B^+$-Baum-Index auf Attribut $B$, $m = 1024$, 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 2500, verteilt auf 100 Datenblöcke.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{B=2500}(R)$$

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 6 - *Join-Algorithmen*. 1 Punkt

Welcher Join Algorithmus (**Merge 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, wobei $R.A$ und $S.B$ Schlüsselattribute sind. Die Relationen sind auf $b_R = 250$ bzw. $b_S = 2000$ hintereinander liegenden Blöcken gespeichert. Es existiert ein **sparse** $B^+$-Baum Index auf $S.B$, wobei jeder Knoten im $B^+$-Baum 20 Schlüssel speichern kann. Nehmen Sie an, dass das Sortieren von $b$ Blöcken $\lceil b \cdot \log_2(b) \rceil$ Blockzugriffe kostet.

Aufgabe 7 - *Effiziente Anfragebearbeitung.* 1 Punkt

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

- $|R| = 2.000.000$ Tupel.
- Pro Datenblock werden 400 Tupel gespeichert.
- Attribut $A$ hat ganzzahlige Werte gleichverteilt im Bereich $[1; 2.000.000]$.
- Attribut $B$ hat ganzzahlige Werte gleichverteilt im Bereich $[400.001; 500.000]$.
- Attribut $C$ hat ganzzahlige Werte gleichverteilt im Bereich $[100.001; 1.000.000]$.
- Es existieren folgende Indizes:
  - Sparse B$^+$-Baum-Index auf Attribut $A$, $m = 2^8 = 256$, minimale Höhe,
  - dense B$^+$-Baum-Index auf Attribut $B$, $m = 2^8 = 256$, minimale Höhe,
  - dense B$^+$-Baum-Index auf Attribut $C$, $m = 2^8 = 256$, minimale Höhe.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{A>1.800.000 \wedge B=450.000}(R)$$

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 - *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 Anfrage ab. $(\sigma_{A=100 \vee A=200}(R) \neq \emptyset)$.

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

## Aufgabe 9 — 1 Punkt

Betrachten Sie den folgenden Schedule.

(a) Welche Transaktion muss abbrechen, um ein **Cascading Rollback** aller anderen Transaktionen auszulösen?

(b) Zeigen Sie die Commit-Ordnung (indem Sie die Commit-Befehle in den Schedule einfügen), sodass der Schedule **cascadeless** ist.

| T1: | T2: | T3: | T4: |
|---|---|---|---|
| | read(B) | | |
| read(D) | | | |
| | write(A) | | |
| | | | read(A) |
| | | | write(C) |
| | | read(E) | |
| read(C) | | | |
| write(B) | | | |
| | | read(B) | |

## Aufgabe 10 — 1 Punkt

Betrachten Sie den folgenden Schedule und den **Two-Phase Locking** Scheduler.

(a) Nehmen Sie an, dass alle Transaktionen eine `write` Operation auf Item `A` in der folgenden Ordnung durchführen wollen: `T3`, `T2`, `T1`. Komplettieren Sie den Schedule, indem Sie Lock Requests (e.g., `R:lock-S(A)`) und Granted Locks (e.g., `G:lock-S(A)`) für alle verbleibenden `read` und `write` Operationen einfügen. Markieren Sie im Schedule, ob eine Transaktion aufgrund eines inkompatiblen Locks einer anderen Transaktion warten muss.

(b) Zeichnen Sie den Wait-For Graphen. Resultiert dieser Schedule in einem Deadlock? Begründen Sie Ihre Antwort.

```
T1:         T2:         T3:
_____________________________________
R:lock-S(A)
G:lock-S(A)
read(A)
_____________________________________


            read(A)
_____________________________________


_____________________________________


_____________________________________


_____________________________________
```