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 2024/25 — 16.05.2025

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

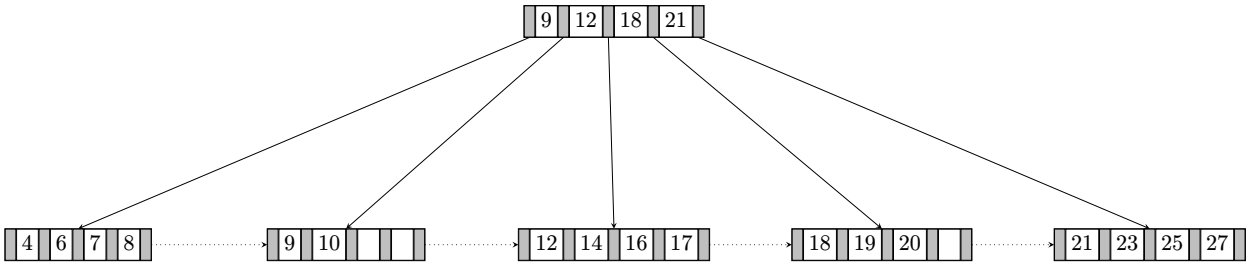
Aufgabe 2 - *Statisches Hashing*. 1 Punkt

Auf der folgenden Tabelle soll ein **Hash Index** konstruiert werden. Als Schlüssel wird das Attribut *Account Nr* verwendet. Der Hashwert ist die **erste Ziffer** des Attributwerts. Es können **3 Tupel pro Bucket** gespeichert werden. Bucket Overflows werden durch **Overflow Chaining** aufgelöst, wobei ein Zeiger auf ein Overflow Bucket einen Eintrag im Bucket benötigt. **Illustrieren** Sie den **Hash-Index**.

Owner Name	*Account Nr*	*Balance*
Donovan	579976	2.467
Kermit	585989	7.824
Solomon	489384	6.824
Gavin	579331	3.850
Kelly	630468	8.949
Angelica	676246	6.452
Fredericka	589374	8.888
Caesar	682535	2.776
Chanda	304225	2.014
Patricia	886712	7.726

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

Gegeben ist ein B^+-Baum mit $m = 5$. Zeichnen Sie den B^+-Baum, der nach dem Löschen von **10** entsteht.



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

Betrachten Sie eine Relation $R[A, B, C, D]$ und die folgenden vier Anfrageprädikate. Geben Sie die Attributordnung von (bis zu) zwei geordneten Indizes mit kombiniertem Suchschlüssel an, sodass alle vier Anfragen mithilfe dieser Indizes effizient ausgeführt werden können.

1. **WHERE** $B = 42$ **AND** $C < 100$
2. **WHERE** $A < 50$ **AND** $B = 84$
3. **WHERE** $B = 42$ **AND** $C < 100$ **AND** $A = 100$
4. **WHERE** $C = 100$ **AND** $D > 50$ **AND** $B = 42$

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 - *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 bis zu 20 Suchschlüssel speichern kann. Der B^+-Baum Index hat maximale Höhe.

Aufgabe 7 - *Effiziente Anfragebearbeitung*. 1 Punkt

Gegeben ist die Relation $R[A, B]$. Auf $R.A$ existiert ein **sparse $\mathbf{B^+}$-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 - *Anfrageoptimierung*. 1 Punkt

Betrachte die folgenden Relationen:

(B)oats(bid, name, color)
(S)ailors(sid, name, rating, age)
(R)eservations(bid, sid, day)

Weiters sei die folgende SQL-Anfrage gegeben:

```
SELECT DISTINCT B.name
  FROM Boats B, Sailors S, Reservations R
 WHERE S.age < 40
   AND B.color = 'blue'
   AND B.bid = R.bid
   AND S.sid = R.sid;
```

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**. **(0.5 Punkte)**

Aufgabe 9 — 1 Punkt

Betrachten Sie die folgende Schedule S:

T1:	T2:	T3:	T4:
		write(D)	
	read(A)		
		write(A)	
write(A)			
	write(B)		
	read(C)		
		read(C)	
	commit		
		read(D)	
		commit	
			read(A)
commit			
			read(B)
			commit

Entscheiden Sie für jede der folgenden Aussagen, ob sie wahr (**W**) oder falsch (**F**) ist. *Falsche Antworten führen zu Punkteabzug.*

1. Es gibt nur eine äquivalente serielle Schedule zu S. ☐
2. Der Precedence-Graph von S hat sechs Kanten. ☐
3. Die Schedule S ist recoverable. ☐
4. Die Schedule S ist cascadeless. ☐

Aufgabe 10 — 1 Punkt

Kann die folgende Historie (Schedule) Ausgabe eines Schedulers mit **Strict Two-Phase-Locking** sein? Falls ja, geben Sie alle notwendigen Lock/Unlock-Operationen an. Erklären Sie ansonsten, warum diese Historie unmöglich ist.

```
T1:         T2:         T3:
──────────────────────────────
            read(A)
            read(B)
                        read(A)
                        write(A)
            write(B)
            COMMIT
                        read(A)
                        read(B)
read(B)
read(A)
                        COMMIT
COMMIT
──────────────────────────────
```