

Klausur Datenbanksysteme II

Wintersemester 2015/2016

Matrikelnr.: _____

1	2	3	4	5	6	7	8	Σ
1	10	8	9	10	12	7	10	67

Hinweise:

- Die Bearbeitungszeit beträgt maximal drei Stunden (9.00 – 12.00 Uhr).
- Bitte notieren Sie auf *jedem* Blatt oben Ihre Matrikelnummer. Wir korrigieren die Klausur aufgabenweise und nehmen die Seiten dafür auseinander.
- Die Nutzung von Hilfsmitteln wie Skript, Taschenrechner, vorbeschriebenen Seiten und Büchern ist untersagt.
- Bitte schalten Sie Ihre Mobiltelefone aus, jedes Klingeln eines Mobiltelefons resultiert im Abzug von 1 Punkt.
- Für Antworten ist der dafür vorgesehene Freiraum zu nutzen. Sollte der für die Antwort vorgesehene Platz nicht ausreichen, nutzen Sie bitte *vornehmlich die Rückseite* des jeweiligen Aufgabenblattes. Am Ende der Klausur finden Sie zusätzlich drei leere Blätter, die Sie für Entwürfe oder Nebenrechnungen nutzen können. *Bitte verweisen Sie gegebenenfalls auf dem jeweiligen Aufgabenblatt auf Ihre Lösungen* auf der Rückseite bzw. auf relevante Notizen oder Lösungen auf den Zusatzblättern.
- Bitte schreiben Sie deutlich! Verwenden Sie nicht die Farbe rot und keinen Bleistift.
- Bei Unklarheiten bezüglich einer Aufgabe melden Sie sich bitte.
- Und vor allem: **Viel Erfolg!**

Aufgabe 1: Matrikelnummer

Tragen Sie auf jedem Blatt der Klausur Ihre Matrikelnummer ein. “Jedes Blatt” umfasst das Deckblatt, die Aufgabenblätter und die Zusatzblätter. Tragen Sie die Matrikelnummer auch ein, falls Sie ein (Zusatz-)Blatt nicht verwenden.

Hinweis: Lösen Sie diese Aufgabe sofort.

1 Punkt

Aufgabe 2: Multiple Choice

Beantworten Sie folgende Fragen, indem Sie jeweils die richtige Antwort ankreuzen. Pro Frage gibt es genau eine richtige Antwort und einen Punkt. **10 Punkte**

Physische Speicherstrukturen

1. Was ist die kleinste physische Leseinheit einer Festplatte?
 - Platte
 - Block
 - Sektor
 - Spur
2. Auf welchen Spuren einer Festplatte sollte man eine sehr große Datei positionieren, damit seltene, sequentielle Scans schnell durchgeführt werden können?
 - Innere Spuren
 - Äußere Spuren
 - Mittlere Spuren
 - Es macht keinen Unterschied
3. Welche Angabe bezeichnet den *wenigsten* Speicher?
 - 1 MB
 - 1 MiB
 - 2 KiB
 - 2 kB

Indexstrukturen

4. Ein dünnbesetzter Index setzt voraus, dass das indexierte Attribut sortiert ist.
 - Wahr
 - Falsch
5. Ein dünnbesetzter Index kann nicht durch einen B⁺-Baum implementiert werden.
 - Wahr
 - Falsch
6. Welche der folgenden Indexkonstruktionen ist *nicht* sinnvoll?
 - Ein dünnbesetzter Level-1-Index mit einem dünnbesetzten Level-2-Index.
 - Ein dichtbesetzter Level-1-Index mit einem dünnbesetzten Level-2-Index.
 - Ein dichtbesetzter Level-1-Index mit einem dichtbesetzten Level-2-Index.

Anfrageoptimierung

7. Welche der folgenden Operationen ist keine binäre Operation?
 - Join
 - Gruppierung
 - Differenz
 - Vereinigung
8. Es seien die Relationen R(a, b) und S(b, c) mit T(R) = 1.000, T(S) = 2000, V(R, a) = 100, V(R, b) = 40, V(S, b) = 200 und V(S, c) = 80 gegeben. Welche Schätzung ergibt sich?
 - V(R ⋈ S, b) = 40
 - V(R ⋈ S, b) = 80
 - V(R ⋈ S, b) = 100
 - V(R ⋈ S, b) = 200
 - V(R ⋈ S, b) = 10.000
 - V(R ⋈ S, b) = 20.000
 - V(R ⋈ S, b) = 25.000
 - V(R ⋈ S, b) = 50.000
 - V(R ⋈ S, b) = 2.000.000

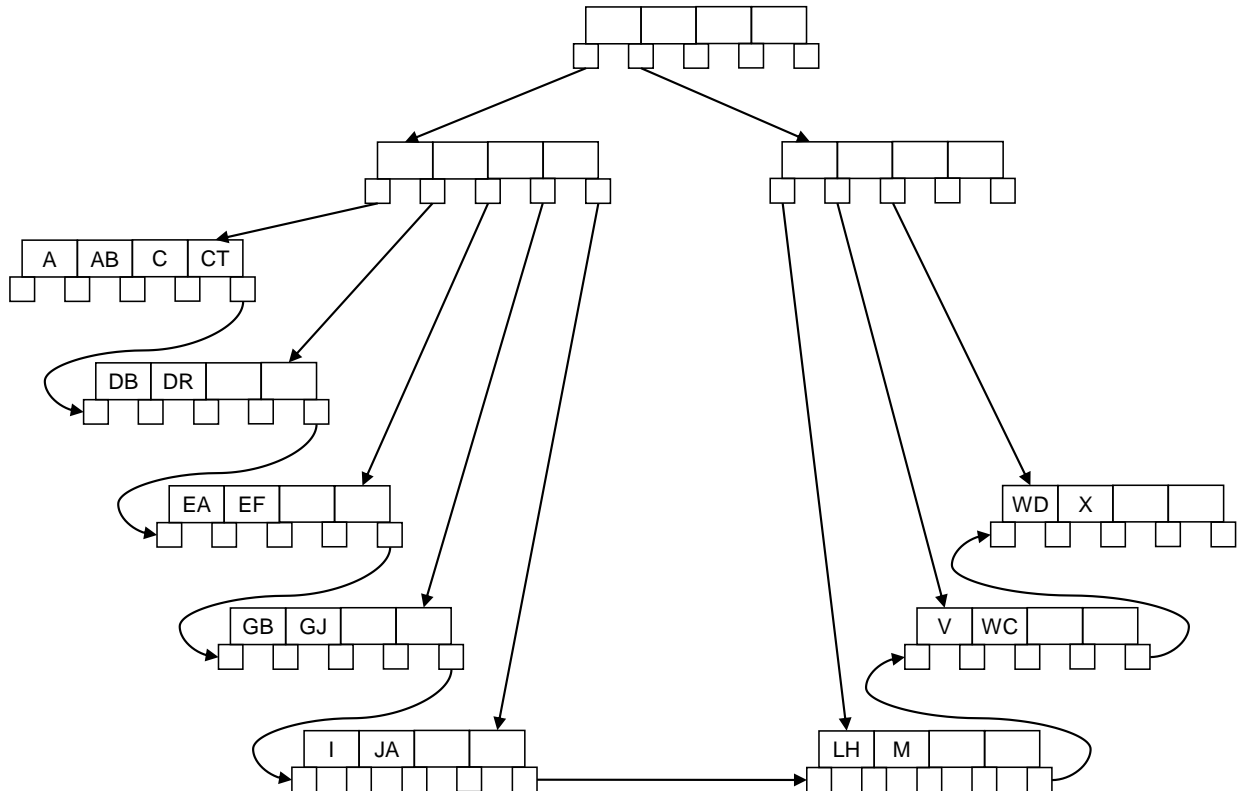
Recovery

9. In welcher Reihenfolge werden Log-Operationen beim UNDO-Logging ausgeführt?
- Änderungs-Eintrag, Write-Operation, Commit-Eintrag
 - Änderungs-Eintrag, Commit-Eintrag, Write-Operation
 - Abort-Eintrag, Write-Operation, Commit-Eintrag
10. Beim UNDO/REDO-Logging darf nach einem $\langle \text{START CKPT } (T_1, \dots, T_n) \rangle$ -Eintrag das zugehörige $\langle \text{END CKPT} \rangle$ erst geschrieben werden, wenn ...
- alle T_1, \dots, T_n ihren COMMIT- oder ABORT-Eintrag geschrieben haben.
 - alle T_1, \dots, T_n ihre Daten aus dem Puffer auf Festplatte geschrieben haben.
 - alle Daten, die beim $\langle \text{START CKPT} \rangle$ -Eintrag im Puffer waren, auf Festplatte geschrieben wurden.
 - alle Daten committeter Transaktionen, die beim $\langle \text{START CKPT} \rangle$ -Eintrag im Puffer waren, auf Festplatte geschrieben wurden.

Aufgabe 3: B+ Bäume

Gegeben sei ein B⁺-Baum mit maximal $n = 4$ Schlüsseln pro Knoten, der als Primärindex dient.

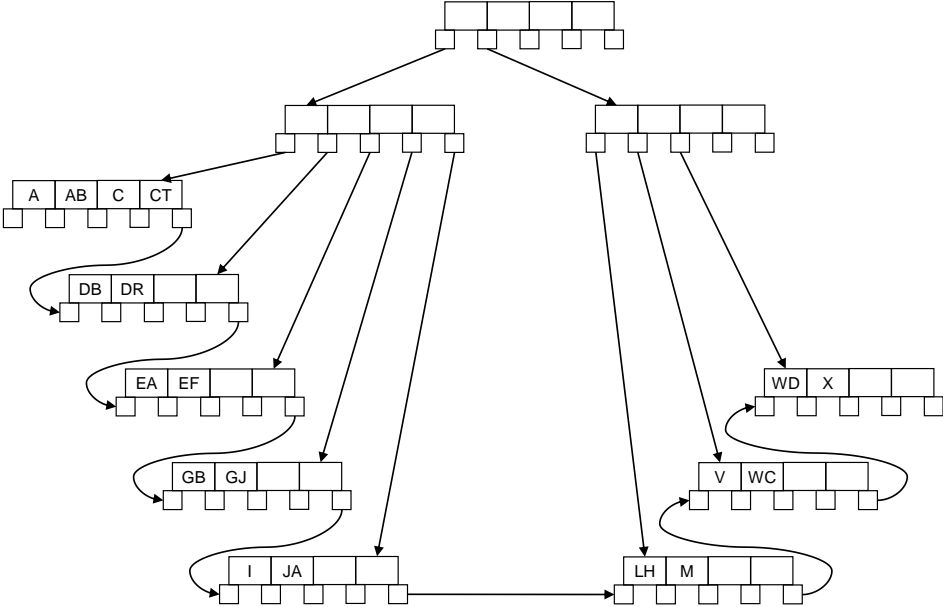
- Tragen Sie in die Nicht-Blattknoten des dargestellten B⁺-Baums die fehlenden Schlüsselwerte ein. Nehmen Sie die Eintragungen direkt in der Abbildung vor. **2 Punkte**



- Wie viele Datensätze können in den B⁺-Baum aus Teilaufgabe 1 noch maximal *zusätzlich* eingefügt werden, ohne dass die Höhe des B⁺-Baums wächst? Das Erzeugen neuer Knoten ist erlaubt. Stellen Sie den Rechenweg nachvollziehbar dar. **2 Punkte**

3. Löschen Sie aus dem *ursprünglichen* B⁺-Baum aus Teilaufgabe 1 die Schlüsselwerte DB und WD. Zeichnen Sie den resultierenden Baum nach der Löschoperation. Unveränderte Teile des Baumes können abgekürzt werden. **4 Punkte**

Wiederholung des B⁺-Baums aus Teilaufgabe 1:



Aufgabe 4: Anfrageausführung

Gegeben seien zwei Relationen $R(X, Y)$ und $S(Y, Z)$ mit $B(R) = 500$ und $B(S) = 400$. Gegeben sei die Anfrage $(R \bowtie S)$ mit einer nicht-leeren Ergebnismenge und es gilt: $V(R, X) = V(S, Z) = 25$.

Wie immer ignorieren wir die Kosten des Schreibens der Ergebnismenge auf Platte.

1. Nennen Sie zuerst zwei Dinge, um $R \bowtie S$ unter Verwendung des Sort-Merge-Joins durchführen zu können, wenn der Hauptspeicher nicht für einen one-pass-join reicht: Die minimal nötige Blockanzahl im Hauptspeicher und die I/O-Kosten.

4 Punkte

2. Es soll $\sigma_{(X=23) \wedge (Z=42)}(R \bowtie S)$ mit dem Sort-Merge-Join-Algorithmus berechnet werden. Berechnen Sie die minimalen I/O-Kosten für die gegebene Anfrage, wenn die Selektion vor dem Join durchgeführt wird und $M = 5$ Blöcke im Hauptspeicher zur Verfügung stehen. Nehmen Sie im weiteren an, dass Sie stets 1 Block für den Input bzw. Output benötigen, also 4 Blöcke effektiv als Arbeitsspeicher zur Verfügung stehen

4 Punkte

3. Angenommen, es steht nun beliebig großer Hauptspeicher zur Verfügung. Ermitteln Sie die minimalen I/O-Kosten für die Anfrage $\sigma_{(X=23)\wedge(Z=42)}(R \bowtie S)$, wenn die Selektion vor dem Join durchgeführt wird. **1 Punkt**

Aufgabe 5: Sortierung

1. Bestimmen Sie die I/O Kosten der Sortierung der Relation R mittels TPMMS falls $B(R) = 1000$ und $M = 201$. Die Kosten des Outputs nach der Sortierung werden im Weiteren nicht mitgezählt. **1 Punkt**

2. Gegeben sei nun $M = 31$. Ab welcher Blockzahl gelingt der TPMMS nicht mehr in zwei Phasen? Nehmen Sie im weiteren an, dass Sie in jeder Phase 1 Block für den Input bzw. Output benötigen, also 30 Blöcke effektiv als Arbeitsspeicher zur Verfügung stehen. **1 Punkt**

3. Beschreiben Sie für den 3-phasigen Algorithmus knapp die zu tätigen Operationen in jeder Phase (je ein-zwei Sätze). Benennen Sie für allgemeine $B(R)$ und $M + 1$ die maximale Anzahl und Größe der resultierenden (Teil-)listen nach jeder Phase. Hinweis: Der I/O Aufwand sollte bei $5 \times B(R)$ liegen. **3 Punkte**

4. Bestimmen Sie für $B(R) = 1000$ und $M = 31$ den konkreten I/O Aufwand in jeder der drei Phasen. **1 Punkt**
- I/O in Phase 1:
 - I/O in Phase 2:
 - I/O in Phase 3:
5. Mit einer einfachen Variation des Algorithmus sollte es gelingen, deutlich unter den genannten Kosten $5 \times 1000 = 5000$ zu liegen. Beschreiben Sie Ihre Idee und geben Sie wiederum für $B(R) = 1000$ und $M = 31$ für jede Phase die Kosten als ganze Zahl an. **4 Punkte**

Aufgabe 6: Optimierung

$W(a, b)$	$X(b, c)$	$Y(c, d)$	$Z(d, a)$
$T(W) = 600$	$T(X) = 2.000$	$T(Y) = 7.500$	$T(Z) = 50$
$V(W, a) = 20$			$V(Z, a) = 50$
$V(W, b) = 50$	$V(X, b) = 100$		
	$V(X, c) = 1.000$	$V(Y, c) = 400$	
		$V(Y, d) = 100$	$V(Z, d) = 25$

Es ist der *natural join* zwischen den vier Relationen W, X, Y und Z zu berechnen (Join jeweils über alle gemeinsamen Attribute). Wie in der Vorlesung berechnen wir die Kosten eines Plans als die Summe der Kardinalitäten der nötigen Zwischenergebnisse. Einzelne Relationen sowie das Endergebnis stellen *kein* nötiges Zwischenergebnis dar.

In den folgenden Aufgaben sollen die Kosten zweier Pläne berechnet werden. Geben Sie dazu jeweils den *left-deep-tree* Anfrageplan als Ausdruck der relationalen Algebra an (z.B. $((Z \bowtie Y) \bowtie X) \bowtie W$), sowie dessen Gesamtkosten (z.B. 10,000). Bitte notieren Sie nachvollziehbar Ihre Rechnung.

1. Berechnen Sie den **Anfrageplan und dessen Kosten** mit dem *Greedy* Verfahren: In jedem Schritt wird der Teilplan mit dem kleinsten Zwischenergebnis gewählt. Die Größe der verbleibenden möglichen Zwischenergebnisse werden nach jedem Schritt neu berechnet. Geben Sie für alle Zwischenergebnisse die Kardinalität an.

5 Punkte

2. Berechnen Sie den **optimalen Anfrageplan und dessen Kosten** mit Hilfe des Algorithmus der dynamischen Programmierung. Geben Sie für alle Zwischenergebnisse Kardinalitäten und Kosten an. **7 Punkte**

Aufgabe 8: Verteilte Systeme

1. Das so-genannte CAP Theorem sagt etwas über die Beziehung der drei Eigenschaften *Consistency*, *Availability* und *Partition tolerance* verteilter Systeme aus: Ein verteiltes System kann immer nur zwei dieser Eigenschaften besitzen. Wir gehen im Weiteren davon aus, dass *Partition tolerance* als Eigenschaft fest vorgegeben ist.

a) Beschreiben Sie für die Domäne *Finanztransaktionen* jeweils ein beispielhaftes Problem falls *Consistency* bzw. *Availability* nicht gelten. **2 Punkte**

b) Beschreiben Sie für die Domäne *Twitter-Nachrichten* jeweils ein beispielhaftes Problem falls *Consistency* bzw. *Availability* nicht gelten. **2 Punkte**

2. Das Reiseunternehmen “Urlaubsreif” speichert alle aktuellen *Flüge* in einer folgenden Tabelle:

Flüge(Nummer, Start, Ziel, Datum, Uhrzeit, Dauer, Preis, Maximalgepäck)

Auf Anfrage eines Kunden, der mit seinem 50kg Koffer eine Weltreise plant, erstellt ein Mitarbeiter von “Urlaubsreif” die folgende Anfrage um alle möglichen Verbindungen aufzulisten:

```
SELECT Start, Ziel, MIN(Dauer) AS Min_Dauer, Min(Preis) AS Min_Preis
FROM Flüge
WHERE Maximalgepäck ≥ 50
GROUP BY Start, Ziel;
```

Entwerfen Sie einen Map/Reduce-Job um die gegebene SQL-Anfrage zu beantworten. Die `map`-Funktion erhält als Eingabe `key/value`-Paare, wobei `key` die Tupel-ID und `value` ein vollständiges Tupel aus der Relation *Flüge* ist. Senden Sie nur die notwendigen Daten vom `map`- zum `reduce`-Job um die Netzwerklast gering zu halten. **6 Punkte**

```
map(key, value)
```

```
reduce(key, values)
```

1. Zusatzblatt

2. Zusatzblatt

3. Zusatzblatt