Prof. Dr. Felix Naumann
Fachgebiet Informationssysteme
Hasso-Plattner-Institut

08. Februar 2016

Klausur Datenbanksysteme II

Musterlösung

Wintersemester 2015/2016

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 Leseeinheit 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 könnnen?
   - ☐ 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 (ein dünnbesetzter Index setzt voraus, dass die Werte sortiert sind)

   ☐ Falsch

5. Ein dünnbesetzter Index kann nicht durch einen B^+-Baum implementiert werden.

   ☐ Wahr

   ☒ Falsch

6. Welche der folgenden Indexkonstruktionen ist *nicht* sinnvoll?

   ☐ Ein dünnbesetzer Level-1-Index mit einem dünnbesetzen Level-2-Index.

   ☐ Ein dichtbesetzer Level-1-Index mit einem dünnbesetzen Level-2-Index.

   ☒ Ein dichtbesetzer Level-1-Index mit einem dichtbesetzen 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?

   ☒ $V(R \bowtie S, b) = 40$

   ☐ $V(R \bowtie S, b) = 80$

   ☐ $V(R \bowtie S, b) = 100$

   ☐ $V(R \bowtie S, b) = 200$

   ☐ $V(R \bowtie S, b) = 10.000$

   ☐ $V(R \bowtie S, b) = 20.000$

   ☐ $V(R \bowtie S, b) = 25.000$

   ☐ $V(R \bowtie S, b) = 50.000$

   ☐ $V(R \bowtie S, b) = 2.000.000$

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 <START CKPT $(T_1, \ldots, T_n)$>-Eintrag das zugehörige <END CKPT> erst geschrieben werden, wenn ...

    ☐ alle $T_1, \ldots, T_n$ ihren COMMIT- oder ABORT-Eintrag geschrieben haben.

    ☐ alle $T_1, \ldots, T_n$ ihre Daten aus dem Puffer auf Festplatte geschrieben haben.

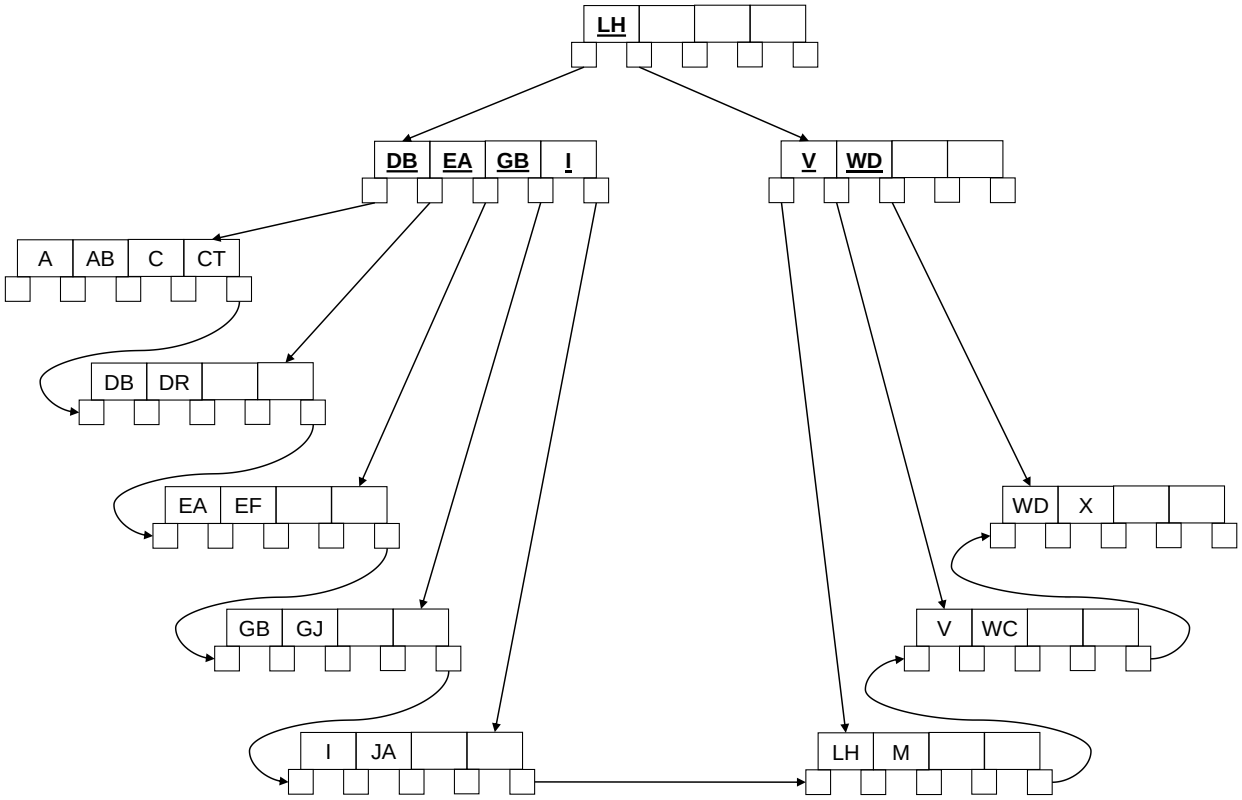
    ☒ alle Daten, die beim <START CKPT>-Eintrag im Puffer waren, auf Festplatte geschrieben wurden.

    ☐ alle Daten committeter Transationen, die beim <START CKPT>-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.

1. 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

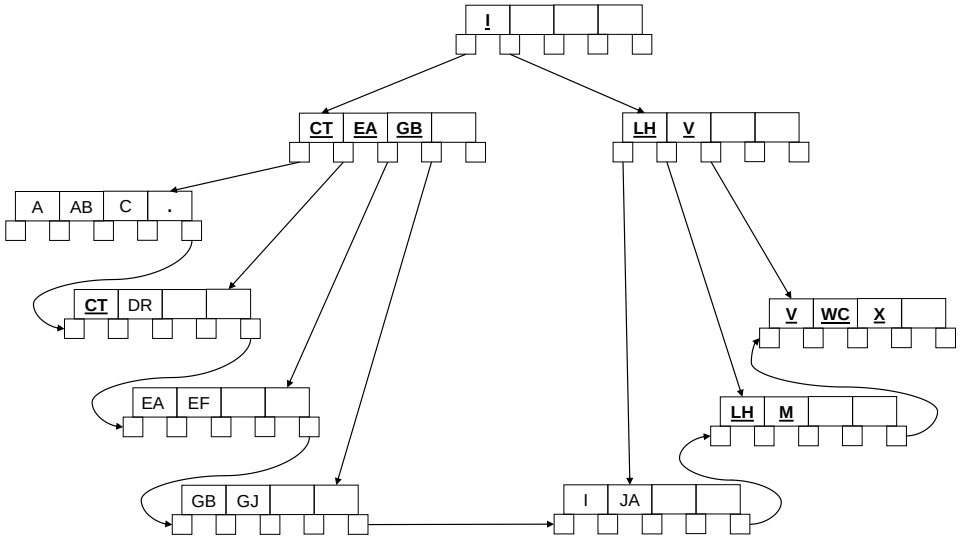


2. Wie viele Datensätze können in den B^+-Baum aus Teilaufgabe 1 noch maximal *zusätzlich* eingefügt werden, ohne dass die Höhe des B^+-Baums wächst? Das Erzeugen neuer Knoten ist erlaubt. Stellen Sie den Rechenweg nachvollziehbar dar.
2 Punkte

Musterlösung

5 Kinder der Wurzel $\times$ 5 Kinder innerer Knoten $\times$ 4 Werte pro Knoten $-$ 18 vorhandene Keys $= 82$

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**



Bewertung 2 Pkt. für Löschen von WD; 2 Pkt. für Löschen und Umverteilen nach Löschen von DB

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**

   Musterlösung

   $B < M^2 \rightarrow M = \sqrt{B(R) + B(S)} = 30$
   $3 \times (B(R) + B(S)) = 2700$

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 Überarbeiten, momentan sind 3 Phasen nötig!**

   Musterlösung

   R + S einlesen (900), relevante Tupel als sortierte Teillisten (5 Listen für R, 4 Liste für S, jeweils Größe 4) schreiben. Anschließend muessen die Teillisten verkleinert werden (Platz für nur 4 Blöcke, für einen direkten Merge-Join werden 9 benötigt). Dafür muessen erneut alle Blöcke gelesen und geschrieben werden. Anschliessend noch einmal einlesen und joinen (insgesamt $4 \times (20 + 16)$).

   I/O: $900 + 4 \times (20 + 16) = 1044$

   (Annahme: Es steht genug Platz für Tupel mit gemeinsamen Y-Werten zur Verfügung (in unserer Rechnung: 1 Block).)

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**

   Musterlösung

   900

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**

   Musterlösung

   3 x B(R) = 3000

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**

   Musterlösung

   Maximal 30 Teillisten, jede Teilliste max 30 groß = 900 Blöcke

3. Beschreiben Sie für den 3-phasigen Algorithmus knapp die zu tätigenden 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 x B(R) liegen. **3 Punkte**

   Musterlösung

   a) Befuelle Hauptspeicher sukzessive mit M Blöcken. Sortiere im Hauptspeicher und speichere B(R)/M sortierte Teillisten der Größe M.

   b) Merge jeweils M Teillisten zu größeren sortierten Teillisten. Jeder grössere Teilliste ist bis zu M^2 groß. Es gibt maximal M Stück.

   c) Merge die M verbliebenen Listen. 1 Liste der Größe B(R).

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:

Musterlösung

- I/O in Phase 1: 2000
- I/O in Phase 2: 2000
- I/O in Phase 3: 1000

5. Mit einer einfachen Variation des Algorithmus sollte es gelingen, deutlich unter den genannten Kosten 5 x 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**

Musterlösung

Phase 1: 2 B(R). Es entstehen 1000 / 30 = 34 sortierte Teillisten (Tricks auch schon hier möglich!)

Phase 2 naiv: Erste 30 Teillisten mergen und dann restliche 4 Teillisten mergen. Kosten 2 x B(R) = 2000

Phase 2 schlau: Nur so viele Teillisten mergen wie nötig. Wenn wir 5 Teillisten mergen haben wir am Ende 29 kurze und eine lange Teilliste. Kosten 5 x 2 x 30 = 300 (Alternative: kurze Restliste (10 Blöcke) und 4 normale Tteillisten (je 30 Tupel) mergen = 2 x 10 + 2 x 4 x 30 = 260

Phase 3: 1x alles mergend lesen: 1 B(R)

Zusammen also 2000 + 300 + 1000 = 3300 (oder 3260)

Bewertung Nicht zu streng nur auf die korrekte Zahl am Ende achten. Lösungsweg honorieren.

Aufgabe 6: Optimierung

(Siehe Datei *Berechnung Kardinalitäten A6.xlsx* zur Anpassung der Werte und automatischer Berechnung der besten Pläne)

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

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 Kardinalitäen 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

   Musterlösung

   - Schritt 1: Kosten einzelner Joins:
     $(W \bowtie X) = 12.000; (X \bowtie Y) = 15.000;$
     $(Y \bowtie Z) = 3.750; (W \bowtie Z) = 600$
     Minimum: $W \bowtie Z$
   - Schritt 2: X bzw. Y jeweils ergänzen:
     $((W \bowtie Z) \bowtie X) = 12.000; ((W \bowtie Z) \bowtie Y) = 45.000$
     Minimum: $(W \bowtie Z) \bowtie X$
   - Schritt 3: Ergänzen um Y. Gesamtkosten 600 + 12.000 = 12.600

   Ergebnis: $(((W \bowtie Z) \bowtie X) \bowtie Y)$ mit Plankosten 12.600

   Es ist OK, die Kreuzproduktvarianten $(W \bowtie Y, X \bowtie Z)$ wegzulassen.

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**

Musterlösung

$(((Y \bowtie Z) \bowtie X) \bowtie W)$ mit Plankosten 11.250

1. Schritt (Kardinalität, Kosten, Plan)
   - W (600, 0, scan(W))
   - X (2.000, 0, scan(X))
   - Y (7.500, 0, scan(Y))
   - Z (50, 0, scan(Z))
2. Schritt (Kardinalität, Kosten, Plan)
   - WX (12.000, 0, W $\bowtie$ X)
   - WZ (600, 0, W $\bowtie$ Z)
   - XY (15.000, 0, X $\bowtie$ Y)
   - YZ (3.750, 0, Y $\bowtie$ Z)
3. Schritt (Kardinalität, Kosten, Plan)
   - WXY (90.000, 12.000, (W $\bowtie$ X) $\bowtie$ Y)
   - WXZ (12.000, 600, (W $\bowtie$ Z) $\bowtie$ X)
   - WYZ (45.000, 600, (W $\bowtie$ Z) $\bowtie$ Y)
   - XYZ (7.500, 3.750, (Y $\bowtie$ Z) $\bowtie$ X)
4. Schritt (Kosten, Plan)
   - WXYZ (7.500 + 3.750 = 11.250, ((Y $\bowtie$ Z) $\bowtie$ X) $\bowtie$ W)

Es ist OK, die Kreuzproduktvarianten ($W \bowtie Y, X \bowtie Z$) wegzulassen.

Aufgabe 7: Recovery

Betrachten Sie das folgende Undo/Redo-Logfile eines Banksystems, welches Kontoüberweisungen durchführt:

1	$<$ `START` $T_1 >$
2	$< T_1, A, 2500, 3000 >$
3	$<$ `START` $T_2 >$
4	$< T_2, B, 300, 50 >$
5	$<$ `START` $T_3 >$
6	$< T_3, C, 500, -50 >$
7	$<$ `COMMIT` $T_2 >$
8	$<$ `START CKPT` $(\dots) >$
9	$< T_1, D, 1000, 1500 >$
10	$< T_3, E, 200, 300 >$
11	$<$ `START` $T_4 >$
12	$< T_4, F, 5000, 3000 >$
13	$<$ `START` $T_5 >$
14	$< T_5, B, 50, 1200 >$
15	$<$ `END CKPT` $>$
16	$< T_4, G, -20, 100 >$
17	$< T_3, H, -200, 0 >$
18	$< T_1, J, 350, 50 >$
19	$<$ `COMMIT` $T_4 >$
20	$< T_5, K, 500, 200 >$
21	$<$ `COMMIT` $T_3 >$
	CRASH

1. Vervollständigen Sie den Log-Eintrag aus Zeile 8.

   1 Punkt

   $<$ `START CKPT(` $\quad\quad\quad\quad\quad\quad$ `)` $>$

   Musterlösung

   $<$ `START CKPT`$(T_1, T_3) >$
   ACHTUNG: Studenten könnten das auch ins Logfile direkt eintragen! **Bewertung** 1 P auf Nennung

2. Welche Länge hat die Logdatei nach dem Ausführen des Recovery?

   1 Punkte

   Musterlösung

   23, weil $<$ `ABORT` $T_1 >$, $<$ `ABORT` $T_5 >$ **Bewertung** 1 P auf Nennung

3. Welche Kontostände sind nach dem Ausführen des Recoverys in der Datenbank gespeichert? *Hinweis:* Ermitteln Sie zuerst, welche Transaktionen committed sind und welche aborted werden müssen, und führen Sie dann die entsprechenden UNDO- bzw. REDO-Schritte durch.

   5 Punkte

   Musterlösung

   Abort: T1,T5 (A,D,K,L); Rest REDO. (Muss nicht in Lösung stehen) **Bewertung** 0.5 P pro Variable

Musterlösung

A = 2500
B = 50
C = -50
D = 1000
E = 300
F = 3000
G = 100
H = 0
J = 350
K = 500

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**

   Musterlösung

   - Keine Consistency: Geld wird abgebucht aber nirgends zugebucht.
   - Keine Availability: Geld kann zurzeit am Automaten nicht abgehoben werden.

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

   Musterlösung

   - Keine Consistency: Tweet erscheint nur bei einigen Followern
   - Keine Availability: Tweet kann nicht abgesetzt werden.

2. Das Reiseunternehmen "Urlaubsreif" speichert alle aktuellen *Flüge* in einer folgender 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)` **Musterlösung**

```
{

if (value.Maximalgepäck >= 50) {
emit(value.Start + '_' + value.End,
{ 'Dauer' => value.Dauer, 'Preis' => value.Preis});
```

`reduce(key, values)` **Musterlösung**

```
{

Start = key.split('_')[0];
Ziel = key.split('_')[1];
Min_Dauer = values.first().Dauer;
Min_Preis = values.first().Preis;
for (value : values) {
Min_Dauer = Min(Min_Dauer, value.Dauer);
Min_Preis = Min(Min_Preis, value.Preis);
}
write(Start, Ziel, Min_Dauer, Min_Preis);
```

1. Zusatzblatt

2. Zusatzblatt

3. Zusatzblatt