Tutorium 7
Codegenerierung, ACID, Serialisierbarkeitstheorie

Big Data Engineering

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

27./28. Juni 2022

Verbesserung Übungsblätter - Häufige Fehler

Aufgabe 1:

- keine Wiederverwendung der HashMap für B.
- Inkonsistente Nutzung der HashMap (z.B. beim Zugriff auf eine HashMap sowohl einzelne Tupel, als auch Listen erwartet).
- Vorteile einer HashMap nicht genutzt (z.B. über alle Schlüssel iteriert um die Existenz eines Schlüssels zu überprüfen).

Aufgabe 2:

- Gruppierung nicht materialisiert.

Aufgabe 3:

- Verletzung der Atomarität durch Sichtbarkeit der Schreiboperationen begründet.
- Verletzung der Atomarität nicht erkannt.
- Konsistenzverletzung falsch begründet (z.B. falsche Werte angegeben).

Wiederholung - Frage 1

Frage

Ordnen Sie die Beschreibungen den jeweiligen ACID-Eigenschaften zu.

1. Die Effekte aller Transaktionen, die committed wurden, werden dauerhaft in der Datenbasis gespeichert.
2. Die Weltsicht jeder einzelnen Transaktion ist: Ich bin die einzige Transaktion, die aktuell ausgeführt wird.
3. Jede Transaktion wird entweder ganz ausgeführt oder abgebrochen. Falls sie abgebrochen wird, werden keine Spuren in der Datenbank hinterlassen.
4. Bestimmte Eigenschaften der Datenbasis sind vor und nach jeder Transaktion erfüllt.

Wiederholung - Frage 1

Frage

Ordnen Sie die Beschreibungen den jeweiligen ACID-Eigenschaften zu.

1. Die Effekte aller Transaktionen, die committed wurden, werden dauerhaft in der Datenbasis gespeichert.
2. Die Weltsicht jeder einzelnen Transaktion ist: Ich bin die einzige Transaktion, die aktuell ausgeführt wird.
3. Jede Transaktion wird entweder ganz ausgeführt oder abgebrochen. Falls sie abgebrochen wird, werden keine Spuren in der Datenbank hinterlassen.
4. Bestimmte Eigenschaften der Datenbasis sind vor und nach jeder Transaktion erfüllt.

Lösung

1. Dauerhaftigkeit
2. Isolation
3. Atomarität
4. Konsistenz

Wiederholung - Frage 2

Frage

Welche Bedingungen müssen gelten, sodass zwei Lese-/Schreiboperationen als Konfliktoperation gelten?

Wiederholung - Frage 2

Frage

Welche Bedingungen müssen gelten, sodass zwei Lese-/Schreiboperationen als Konfliktoperation gelten?

Lösung

Zwei Lese-/Schreiboperationen eines Ausführungsplans heißen Konfliktoperationen, falls **alle** folgenden Bedingungen gelten:

1. Sie gehören zu unterschiedlichen Transaktionen.
2. Beide greifen auf dasselbe Datenobjekt zu.
3. Mindestens eine von ihnen ist eine Schreiboperation.

Wiederholung - Frage 3

Frage

Wie viele Paare von Konfliktoperationen sind in folgendem Ausführungsplan zu finden?

$$w_1(A) \rightarrow r_2(A) \rightarrow w_3(B) \rightarrow w_2(A) \rightarrow r_3(C)$$

(A): 1

(B): 2

(C): 3

(D): 4

Wiederholung - Frage 3

Frage

Wie viele Paare von Konfliktoperationen sind in folgendem Ausführungsplan zu finden?

$$w_1(A) \rightarrow r_2(A) \rightarrow w_3(B) \rightarrow w_2(A) \rightarrow r_3(C)$$

Lösung

Die richtige Antwort lautet (B):

Wir haben folgende Paare:

- $w_1(A), w_2(A)$
- $w_1(A), r_2(A)$

Wiederholung - Frage 4

Frage

Welcher der folgenden Ausführungspläne ist seriell?

(A): $r_1(B) \rightarrow r_2(B) \rightarrow w_1(A) \rightarrow w_2(A)$ (B): $r_1(B) \rightarrow w_1(A) \rightarrow r_2(B) \rightarrow w_2(A)$

Wiederholung - Frage 4

Frage

Welcher der folgenden Ausführungspläne ist seriell?

(A): $r_1(B) \rightarrow r_2(B) \rightarrow w_1(A) \rightarrow w_2(A)$ (B): $r_1(B) \rightarrow w_1(A) \rightarrow r_2(B) \rightarrow w_2(A)$

Lösung

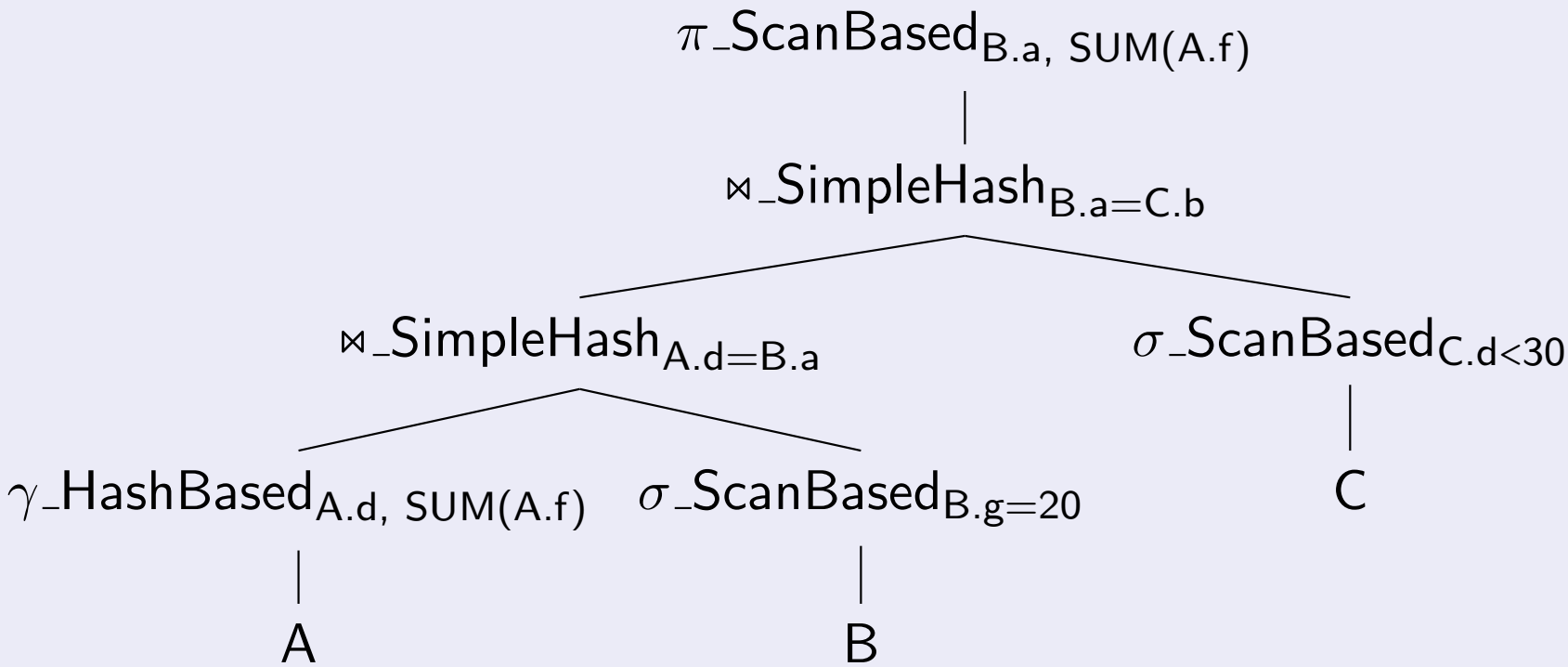
Die richtige Antwort lautet (B):

$r_1(B) \rightarrow w_1(A) \rightarrow r_2(B) \rightarrow w_2(A)$

Aufgabe 1

Frage

Setzen Sie unten stehenden physischen Plan in Pseudocode um.
Produzieren Sie dabei so wenig Zwischenergebnisse wie möglich.



Aufgabe 1

Lösung

```
HashMap hm1, hm2
for each a in A:
    if a.d in hm1:
        hm1.update(a.d, hm1[a.d] + a.f)
    else:
        hm1.insert(a.d, a.f)

for each b in B:
    if b.g = 20 and b.a not in hm2:
        e = hm1.probe(b.a)
        if e exists:
            hm2.insert(b.a, e)

for each c in C:
    if c.d < 30:
        e = hm2.probe(c.b)
        if e exists:
            yield(c.b, e)
```

Aufgabe 2

Frage

In dieser Aufgabe werden Sie mit Ausführungsplänen von Transaktionen arbeiten, die die ACID-Eigenschaften verletzten.
Gehen Sie jeweils davon aus, dass Schreiboperation sofort für alle Transaktionen sichtbar sind, jedoch nicht automatisch auf die Festplatte geschrieben werden.

Aufgabe 2.1

Frage

Welche der ACID-Eigenschaften werden durch die Ausführung der unteren Transaktionen verletzt? Geben Sie jeweils den Grund an.

	T_1	T_2
1		bal_{a2} = r(balA)
2	bal_{a1} = r(balA)	
3		bal_{a2} = bal_{a1} + 30
4	bal_{a1} = bal_{a1} - 10	
5	w(balA=bal_{a1})	
6	commit	
7		w(balA=bal_{a2})
8		commit
	Der aktuelle Zustand wird persistent auf die Festplatte geschrieben.	

Aufgabe 2.1

Lösung

- Isolation:
  Beide Transaktionen greifen auf das Datenobjekt A zu, T_2 liest allerdings den Wert in Zeile 1 aus, während T_1 dasselbe in Zeile 2 tut. Beide verändern anschließend den Wert lokal und schreiben ihn anschließend in die Datenbank, wobei T_2 jedoch in Zeile 7 die vorher eingebrachte Änderung durch T_1 in Zeile 5 überschreibt. So wird die Isolation verletzt, da sich damit beide Transaktionen beeinflusst haben.
- Dauerhaftigkeit:
  Erst nach dem Commit von T_2 werden die Änderungen persistiert, allerdings nicht direkt nach Commit von T_1, das heißt falls die Datenbank in Zeile 7 oder 8 crasht, sind die Änderungen von T_1 trotz des Commits verloren, was eine Verletzung der Dauerhaftigkeit ist.

Aufgabe 2.2

Frage

Welche der ACID-Eigenschaften werden durch die Ausführung der unteren Transaktionen verletzt? Geben Sie jeweils den Grund an.

	T_1	T_2
1		$bal_{b2} = \text{r(baIB)}$
2	$bal_{b1} = \text{r(baIB)}$	
3		$bal_{b2} = bal_{b2} - 300$
4		$bal_{b2} = \text{w(baIB=}bal_{b2}\text{)}$
	Der aktuelle Zustand wird persistent auf die Festplatte geschrieben.	
5	$bal_{b1} = \text{r(baIB)}$	
6		commit
7	commit	
	Der aktuelle Zustand wird persistent auf die Festplatte geschrieben.	

Aufgabe 2.2

Lösung

- Isolation:
  T_2 greift in Zeile 1 zunächst auf das Datenobjekt B zu, während T_1 in Zeile 2 ebenfalls auf B zugreift. Anschließend greift jedoch T_2 schreibend auf B zu, sodass T_1 mit ihrem zweitem Lesezugriff in Zeile 5 diese Änderungen unabhängig vom persistenten Schreiben bereits mitbekommt. Daher ist die Isolation verletzt, da der Wert von B bei beiden Lesezugriffen unterschiedlich ist, obwohl T_1 selbst keine Änderungen vorgenommen hat.
- Atomarität:
  Nach Zeile 4 wird bereits der aktuelle Zustand auf die Festplatte geschrieben, obwohl keine der Transaktionen zu diesem Zeitpunkt committet haben, was eine Verletzung der Atomarität darstellt.

Aufgabe 3

Frage

Gegeben seien folgende Ausführungspläne:

1. $r_3(B) \rightarrow w_1(A) \rightarrow w_2(B) \rightarrow r_2(A) \rightarrow w_4(B) \rightarrow w_2(A)$
2. $r_1(C) \rightarrow w_2(A) \rightarrow w_2(C) \rightarrow w_3(A) \rightarrow r_3(C) \rightarrow w_4(B) \rightarrow w_1(C) \rightarrow w_2(B)$
3. $r_2(A) \rightarrow r_3(B) \rightarrow w_1(A) \rightarrow w_1(B) \rightarrow r_4(C) \rightarrow w_4(C) \rightarrow w_2(C) \rightarrow r_1(A)$

Entscheiden Sie für jeden Ausführungsplan, ob dieser konfliktserialisierbar ist. Sofern der Ausführungsplan konfliktserialisierbar ist, sortieren Sie die Nicht-Konfliktoperationen so um, dass Sie einen seriellen, konfliktäquivalenten Ausführungsplan erhalten und geben Sie diesen an.

Aufgabe 3

Lösung

1. Der Ausführungsplan ist konfliktserialisierbar, da er konfliktäquvivalent zu dem Plan $T_3 \to T_1 \to T_2 \to T_4$ (oder $T_1 \to T_3 \to T_2 \to T_4$) ist.
2. Der Ausführungsplan ist nicht konfliktserialisierbar, da die Konfliktoperationen $r_1(C) \to w_2(C)$ und $w_2(C) \to w_1(C)$ es nicht möglich machen, die Nicht-Konfliktoperationen so umzustellen, dass man einen seriellen, konfliktäquvivalenten Ausführungsplan erhält.
3. Der Ausführungsplan ist konfliktserialisierbar, da er konfliktäquivalent zu dem Plan $T_3 \to T_4 \to T_2 \to T_1$ (oder $T_4 \to T_3 \to T_2 \to T_1$ bzw. $T_4 \to T_2 \to T_3 \to T_1$) ist.