Lehrstuhl für Praktische Informatik III
Prof. Dr. Guido Moerkotte
Email: moer@db.informatik.uni-mannheim.de

Datenbanksysteme 1 — 10. Übungsblatt
Herbst-/Wintersemester 2016

Aufgabe 1

Aufgabe 1 a)

Die Eigenschaften des Transaktionskonzepts werden oft (wie in der Vorlesung) mit der Abkürzung ACID zusammengefasst. ACID steht hierbei für vier grundlegende Eigenschaften, die die Transaktionen in einem Datenbanksystem erfüllen müssen. Nennen Sie diese vier Eigenschaften und diskutieren Sie deren Anforderungen bzw. Bedeutung.

Lösung

Atomicity

Alle Operationen einer Transaktion müssen atomar, d.h. ganz oder gar nicht ausgeführt werden.

Consistency

Die von einer Transaktion durchgeführten Operationen müssen das Datenbanksystem von einem konsistenten Zustand wieder in einem konsistenten Zustand überführen.

Isolation

Mehrere nebenläufige Transaktionen müssen voneinander isoliert werden, d.h. die Transaktionen dürfen sich gegenseitig nicht beeinflussen.

Durability

Die von einer erfolgreich abgeschlossenen Transaktion (commit) durchgeführten Änderungen dürfen (auch im Fall von Systemfehlern) nicht verloren gehen, d.h. die Änderungen müssen dauerhaft erhalten bleiben.

Aufgabe 1 b)

In der Vorlesung wurden die Strategien force / ¬ force und steal / ¬ steal besprochen. Diese Strategien bestimmen das Verhalten eines Datenbanksystems in Bezug auf das Einbringen von Änderungsoperationen auf den Hintergrundspeicher bzw. das Ersetzen von Pufferseiten. Diskutieren Sie die Unterschiede zwischen den einzelnen Strategien bzw. deren Kombinationen. Worin liegen deren spezifische Vor- und Nachteile?

Lösung

- force: Geänderte Pufferseiten müssen beim Transaktionsabschluss (commit) auf den Hintergrundspeicher geschrieben werden. Hierdurch müssen keine Redo-Informationen gespeichert werden, da sichergestellt ist, dass alle Änderungsoperationen auch auf dem Hintergrundspeicher eingelagert sind. Die force-Strategie erzwingt, dass die von der jeweiligen Transaktion betroffenen Teile des Datenbankpuffers beim commit ausgeschrieben werden müssen. Dies kann die Leistung des Systems negativ beeinflussen.

- ¬ force: Bei der ¬ force Strategie müssen beim Transaktionsabschluss (commit) die geänderten Pufferseiten *nicht* sofort auf den Hintergrundspeicher geschrieben werden. Hierdurch kann es vorkommen, dass bei einem Systemabsturz die von einer bereits erfolgreich abgeschlossenen Transaktion durchgeführten Änderungen nicht auf dem Hintergrundspeicher vorhanden sind. Um die Dauerhaftigkeit (durability) der Transaktion zu gewährleisten, muss also ein Redo-Log gepflegt werden. Mit dessen Hilfe können beim Wiederanlauf (restart recovery) die Änderungen rekonstruiert werden.

- steal: Beim Einsatz der steal-Strategie können geänderte Seiten (sog. *dirty pages*) einer noch aktiven Transaktion aus dem Puffer auf den Hintergrundspeicher verdrängt werden. Hierdurch kann im Puffer Platz für andere Seiten geschaffen werden. Da deshalb möglicherweise Änderungsoperationen von noch nicht abgeschlossenen Transaktionen auf dem Hintergrundspeicher stehen, muss das System Undo-Informationen speichern, um diese Änderungen im Fall eines Transaktionsabbruchs (abort) rückgängig machen zu können.

- ¬ steal: Wird die ¬ steal-Strategie eingesetzt, werden die von noch aktiven Transaktionen geänderten Seiten nicht aus dem Puffer verdrängt. Es kann also kein Pufferplatz durch Verdrängung freigeräumt werden, es wird jedoch garantiert, dass keine Änderungsoperationen von noch aktiven Transaktionen auf dem Hintergrundspeicher materialisiert werden. Undo-Informationen müssen deshalb bei einer ¬-steal Strategie nicht gespeichert werden.

Aufgabe 2

Aufgabe 2 a)

Welche der ACID-Eigenschaften garantiert die Recovery?

Lösung

Atomicity und Durability sind richtig.

Atomicity durch Redo- und Undo-Phasen der Recovery. Durability durch Redo im Falle eines Systemausfalls.

Consistency wird nur sichergestellt, wenn die Anwendung Datenbankzustände in konsistente Zustände überführt und in Mehrbenutzerumgebungen Isolation sichergestellt wird.

Aufgabe 2 b)

Welche der folgenden Regeln müssen beim Write-Ahead-Logging (WAL) befolgt werden?

1. Vor dem Ausschreiben der Logeinträge einer Seite muss die modifizierte Seite ausgelagert werden.
2. Vor dem Commit einer Transaktion müssen alle Log-Einträge der Transaktion ins Log geschrieben werden.
3. Vor dem Ausschreiben einer modifizierten Seite müssen alle Logeinträge, die zu dieser Seite gehören ins Log geschrieben werden.
4. Vor dem Commit einer Transaktion müssen alle modifizierten Seiten der Transaktion ausgeschrieben werden.

Lösung

1. falsch
2. richtig
3. richtig
4. falsch

Aufgabe 3

Aufgabe 3 a)

Nach einem Absturz führt ein Datenbanksystem die Wiederanlaufphase (*Recovery*) durch. Die Recovery kann hierbei konzeptuell in drei Phasen aufgeteilt werden. Nennen Sie diese drei Phasen und beschreiben Sie die im Rahmen jeder einzelnen Phase durchgeführten Operationen.

Lösung

Analyse

In der Analysephase werden die im Log protokollierten Transaktionen in Winner und Loser eingeteilt. Winner sind die Transaktionen, deren erfolgreicher Abschluss (commit) im Log vermerkt wurde. Alle anderen, noch aktiven Transaktionen sind Loser.

Redo

In der Redo-Phase werden die Operationen von sowohl Winner- als auch Loser- Transaktionen erneut durchgeführt. Nachdem die Redo-Phase abgeschlossen ist, spiegelt der Systemzustand des Datenbanksystems (grob) den Systemzustand zum Zeitpunkt des Absturzes wider.

Undo

Nach der Redo-Phase werden in der Undo-Phase die von den Loser-Transaktionen durchgeführten Operationen rückgängig gemacht.

Aufgabe 3 b)

Während der Recovery kann es zu einem erneuten Absturz des Datenbanksystems kommen. Warum kann dies ohne besondere Sicherungsmaßnahmen zu Problemen führen? Wie kann sichergestellt werden, dass auch bei (mehrmaligem) Absturz während der Recovery diese letztendlich erfolgreich abgeschlossen werden kann?

Lösung

Die Redo- und Undo-Phasen müssen auch bei mehrmaliger Ausführung hintereinander (z.B. durch erneuten Absturz während der Recovery) das selbe Ergebnis produzieren (*Idempotenz*).

Es muss sichergestellt werden, dass bei mehrfachem Redo kein Redo auf ein (bereits aktuelles) After-Image eines Datenelements ausgeführt wird (und hierdurch evtl. die ursprüngliche Aktion mehrfach durchgeführt wird). Dies geschieht mit Hilfe der PageLSN.

Um die Idempotenz der Undo-Phase sicherzustellen werden entsprechende Kompensationseinträge (*Compensation Log Records (CLR)*) in das Log geschrieben.

Aufgabe 3 c)

Warum müssen in den Kompensationseinträgen im Log keine Undo-Informationen gespeichert werden?

Lösung

Ein CLR beschreibt eine während eines vorherigen Wiederanlaufs durchgeführte Undo-Aktion. Die durchgeführte Aktion ist hierbei als Redo-Information protokolliert. Durch ein Redo der im CLR beschriebenen Aktion wird also (erneut) ein Undo der ursprünglichen Aktion durchgeführt. Durch eine entsprechende Verkettung der Logeinträge (*UndoNextLSN*) wird sichergestellt, dass die ursprüngliche Aktion nicht erneut während dem Undo berücksichtigt wird.

Aufgabe 4

In einem Datenbanksystem werden die drei Transaktionen T_1, T_2 und T_3 verzahnt ausgeführt. Die (schreibenden) Zugriffe der Transaktionen auf die Datenelemente B, C und E werden in einem Log-File mitprotokolliert (siehe Abbildung 1). Es existieren keine Sicherungspunkte, d.h. T_1, T_2 und T_3 sind die ersten Transaktionen, die auf dem neu gestarteten Datenbanksystem laufen.

	T_1	T_2	T_3	Log-Datei					
				[LSN,	TA,	PageId,	Redo,	Undo,	PrevLSN]
1.		BOT		[#1,	T_2,	-,	BOT,	-,	0]
2.		$r(B, b_1)$							
3.	BOT			[#2,	T_1,	-,	BOT,	-,	0]
4.		$b_1 = b_1 + 5$							
5.		$w(B, b_1)$		[#3,	T_2,	P_B,	$B = B + 5$,	$B = B - 5$,	#1]
Abbruchpunkt 1									
6.			BOT	[#4,	T_3,	-,	BOT,	-,	0]
7.	$r(B, b_2)$								
8.			$r(E, e_1)$						
9.		Commit		[#5,	T_2,	-,	Commit,	-,	#3]
10.	$b_2 = b_2 + 4$								
11.	$w(B, b_2)$			[#6,	T_1,	P_B,	$B = B + 4$,	$B = B - 4$,	#2]
12.	$r(C, c_1)$								
13.	$c_1 = c_1 - 3$								
14.	$w(C, c_1)$			[#7,	T_1,	P_C,	$C = C - 3$,	$C = C + 3$,	#6]
Abbruchpunkt 2									
15.			$r(C, c_2)$						
16.			$e_1 = e_1 + c_2$						
17.			$w(E, e_1)$	[#8,	T_3,	P_E,	$E = E + C$,	$E = E - C$,	#4]
18.	Commit			[#9,	T_1,	-,	Commit,	-,	#7]
19.			Commit	[#10,	T_1,	-,	Commit,	-,	#8]

Abbildung 1: Log-File

Aufgabe 4 a)

Angenommen an Abbruchpunkt 1 stürzt das System mit Verlust des Hauptspeicherinhalts ab. Geben Sie die einzelnen Schritte des Wiederanlaufs an.

Lösung

1. Analyse:
   Transaktionen T1 und T2 sind beide Loser, da sie nicht erfolgreich abgeschlossen wurden.

2. Redo:
   Falls LSN(P_B) $< \#3$, dann $B = B + 5$ (und LSN(P_B) ersetzen).

3. Undo:
   $B = B - 5$
   Folgende Compensation Log Records werden angelegt:
   $[\#3', T_2, P_B, B = B - 5, \#3, \#1]$
   $LSN(P_B) = \#3'$

   $[\#2', T_1, -, -, \#2, 0]$
   $[\#1', T_2, -, -, \#3', 0]$

Aufgabe 4 b)

Führen Sie den Wiederanlauf für einen Absturz an Abbruchpunkt 2 durch (Abbruchpunkt 1 wird ohne Absturz durchlaufen). Gehen Sie dabei wie in Teil (b) vor.

Lösung

1. Analyse:
   T_2 ist Winner, T_1 und T_3 sind Loser

2. Redo:
   Falls LSN(P_B) $< \#3$, dann $B = B + 5$ (und LSN(P_B) ersetzen).
   Falls LSN(P_B) $< \#6$, dann $B = B + 4$ (und LSN(P_B) ersetzen).
   Falls LSN(P_C) $< \#7$, dann $C = C - 3$ (und LSN(P_C) ersetzen).

3. Undo (nur T_1 und T_3):
   $C = C + 3$
   $[\#7', T_1, P_C, C = C + 3, \#7, \#6]$
   $LSN(P_C) = \#7'$

   $B = B - 4$
   $[\#6', T_1, P_B, B = B - 4, \#7', \#2]$
   $LSN(P_B) = \#6'$

   $[\#4', T_3, -, -, \#4, 0]$
   $[\#2', T_1, -, -, \#6', 0]$

Aufgabe 4 c)

Welche Art von Logeinträgen verwendet das Datenbanksystem (logisch oder physisch)?

Begründen Sie. Wie würden sich die erzeugten Logeinträge verändern, wenn eine andere Protokollierung eingesetzt werden würde?

Lösung

Es werden logische Logeinträge erzeugt, in denen die durchgeführte Änderungsoperation (bzw. die zugehörige, logische Undo-Operation) protokolliert wird. Bei physischem Logging würden direkt die Before-/After-Images der Daten gespeichert werden, d.h. die konkreten Werte (also z.B. [B = 42, B = 38] statt [B = B+4, B = B-4]).

Aufgabe 5

Gegeben seien die Transaktionen T_1, T_2 und T_3 mit folgenden Operationen (T_1,T_2 und T_3 bearbeiten die Datenelemente A und B):

- $T_1 : BOT_1; Read_1(B); Write_1(A); Read_1(B); Write_1(B); COMMIT_1$
- $T_2 : BOT_2; Read_2(B); Read_2(A); Write_2(B); COMMIT_2$
- $T_3 : BOT_3; Read_3(B); Write_3(B); ABORT_3$

Verzahnen Sie jeweils zwei der drei Transaktionen so, dass ein

Aufgabe 5 a)

"Lost Update" auftritt.

Lösung

T_1	T_2
BOT_1	
$\text{Read}_1(\text{B})$	
$\text{Write}_1(\text{A})$	
$\text{Read}_1(\text{B})$	
	BOT_2
	$\text{Read}_2(\text{B})$
$\text{Write}_1(\text{B})$	
Commit_1	
	$\text{Read}_2(\text{A})$
	$\text{Write}_2(\text{B})$
	Commit_2

Aufgabe 5 b)

"Dirty Read" auftritt.

Lösung

T_2	T_3
	BOT_3
	$\text{Read}_3(\text{B})$
	$\text{Write}_3(\text{B})$
BOT_2	
$\text{Read}_2(\text{B})$	
$\text{Read}_2(\text{A})$	
$\text{Write}_2(\text{B})$	
Commit_2	
	Abort_3

Aufgabe 5 c)

"Non-repeatable Read" auftritt.

Lösung

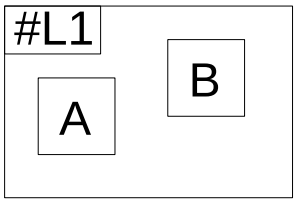
T_1	T_2
BOT_1	
$\text{Read}_1(\text{B})$	
	BOT_2
	$\text{Read}_2(\text{B})$
	$\text{Read}_2(\text{A})$
	$\text{Write}_2(\text{B})$
	Commit_2
$\text{Write}_1(\text{A})$	
$\text{Read}_1(\text{B})$	
$\text{Write}_1(\text{B})$	
Commit_1	

Aufgabe 6

Zusatzaufgabe: Warum müssen in der Redo-Phase der Recovery auch alle Operationen der Loser-Transaktionen erneut durchgeführt werden? Konstruieren Sie ein Beispiel, dass demonstriert, warum es zu Fehlern kommt, wenn nur die Winner erneut ausgeführt werden.

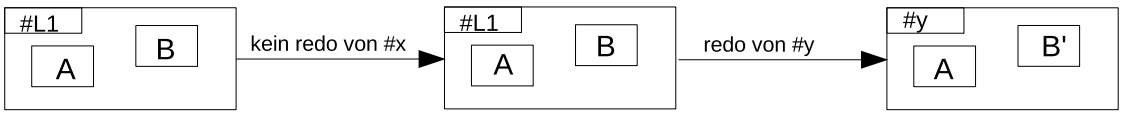
Lösung

Ausgangssituation:



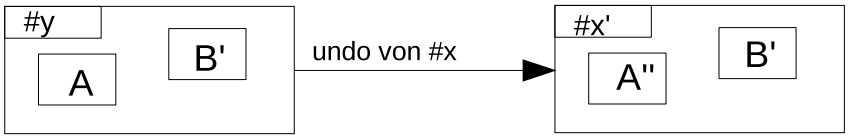
- Seite P mit Datenobjekten A und B, PageLSN = $\#L1$
- Zwei Transaktionen: T_L (Loser) und T_W (Winner)
- Logrecords:
  - $(\#x, T_L, P, redo : A \leftarrow A')$
  - $(\#y, T_W, P, redo : B \leftarrow B')$

Redo:



Es wird kein Redo der Loser-Transaktion ausgeführt. Beim Redo der Winner-Transaktion wird die LSN #y in die Seite geschrieben und B zu B' geändert.

Undo:



Die Loser-Transaktion wird rückgängig gemacht und dabei die Undo-Operation von Logeintrag #x auf dem Before-Image ausgeführt. Bei *logischen* Logeinträgen führt dies zu einem falschen Wert für A.

Anmerkung: In der Übung habe ich ein Beispiel gezeigt, bei dem der Wiederanlauf im ersten Versuch scheitert. Der Fehler tritt aber, wie oben gezeigt, auch bei einem erfolgreichen einmaligen Durchlauf der Recovery auf!