Max-Planck-Institut Saarbrücken
AG5: Datenbanken und Informationssysteme
Dozenten: Prof. Dr.-Ing. G. Weikum, Dr. Ralf Schenkel
Betreuer: Dipl.-Inform. Sergej Sizov, Dipl.-Inform. Christian Zimmer

max planck institut informatik

Informationssysteme (SS 05)

Übungsblatt 10

Ausgabe: 28.Juni 2005

Abgabe: 5.Juli 2005 in der Vorlesung

Aufgabe 1: Serialisierbarkeit I

Gegeben ist der folgende Schedule der Transaktionen T1, T2, T3, T4, T5:

R1(a) R2(b) W1(b) W3(c) W2(c) W2(d) R4(a) W4(c) W5(c) W5(d) W4(a)

Ist der Schedule serialisierbar? Falls ja, geben Sie mindestens einen äquivalenten seriellen Schedule an. Falls nein, warum nicht?

Aufgabe 2: Serialisierbarkeit II

Gegeben ist der folgende Schedule der Transaktionen T1, T2, T3:

R1(a) R2(a) W2(a) R3(b) R2(c) W3(b) R1(c) R1(b)

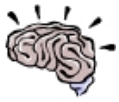
a) Ist dieser Schedule (konflikt-) serialisierbar? Falls ja, zu welchem seriellen Schedule ist er äquivalent? Begründen Sie Ihre Antwort!

b) Kann dieser Schedule bei Verwendung des Zweiphasen-Sperrprotokolls entstanden sein? Kann er bei Verwendung des strikten Zweiphasen-Sperrprotokolls entstanden sein? Begründen Sie Ihre Antwort!

Aufgabe 3: Sperrprotokoll

Betrachten Sie das folgende Sperrprotokoll:

- Für Transaktionen, die mindestens eine Änderungsoperation beinhalten, wird das strikte 2PL verwendet[1].
- Für Transaktionen, die nur Leseoperationen beinhalten, wird ein nicht zweiphasiges Sperrprotokoll wie folgt verwendet:
  - Vor dem Zugriff auf ein Objekt muss dieses gesperrt werden.
  - Alle Sperren einer Transaktion werden jeweils bei Beendigung einer SQL-Anweisung freigegeben. (Bei der nächsten SQL-Anweisung werden wieder neue Sperren angefordert.)



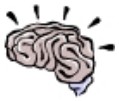
a) Zeigen Sie, dass bei Verwendung dieses Protokolls nichtserialisierbare Schedules entstehen können.

b) Kann bei Verwendung des angegebenen Sperrprotokolls eine Integritätsbedingung der Datenbank verletzt werden? Falls nein, ist der „Verlust“ der Serialisierbarkeit überhaupt gravierend? Falls ja, geben Sie ein Beispiel an.

c) Geben Sie ein Anwendungsbeispiel an, bei dem das angegebene Sperrprotokoll sinnvoll ist.

[1] Dieses Protokoll entspricht dem TRANSACTION ISOLATION LEVEL READ COMMITED des SQL-Standards.

Aufgabe 4: Concurrency Control



Betrachten Sie den folgenden Schedule mit den Operationen von vier Transaktionen T1 , T 2 , T 3 und T4 auf den beiden Zählerwerten (z.B. Kontoständen) *x* und y.

incr1 (x) *incr2* (y) *write3* (x) *read4* (y) *incr3* (y) *incr1* (y)

Die *incr*-Operation inkrementiert einen Zählerwert um Eins (ohne den Zählerwert zurückzuliefern). Eine *incr*-Operation steht nicht in Konflikt zu einer anderen *incr*-Operation, sie steht jedoch in Konflikt zu einer *read*- und einer *write*-Operation. Ist der Schedule konflikt-serialisierbar

(a) unter Ausnutzung der *incr*-Semantik ?

(b) ohne Ausnutzung der *incr*-Semantik, d.h. wenn jede *incr*-Operation durch eine *read*- und eine unmittelbar darauffolgende *write*-Operation ersetzt wird?

Geben Sie ggf. die Serialisierungsreihenfolge an.

Aufgabe 5: Recovery

Gegeben sei der in der in Spalte 1 von Tabelle 1 aufgeführte chronologische Ablauf von Aktionen. Der Checkpoint, der in diesem Ablauf ausgeführt wird, ist ein asynchroner. Tragen Sie bitte Ihre Lösungen der Teilaufgaben a) bis c) direkt in die vorgegebenen Tabellen ein, und geben Sie die Tabellen mit ab.

a) Geben Sie die vom Log-Manager zu protokollierende Information durch Ausfüllen der Spalten 4 und 5 von Tabelle 1 sowie die Änderungen von Seitenheader-LSNs durch Ausfüllen der Spalten 2 und 3 an. Verwenden Sie die Nummerierung der Aktionen für die LSNs.

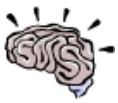
   Verwenden Sie bei dieser und allen folgenden Teilaufgaben bitte die folgende Notation: Tragen Sie in den Spalten 2 und 3 jeweils die betroffene(n) Seite(n) mit ihre(n) LSN ein, z.B. `p(17)` bedeutet Seite p mit LSN 17. Tragen Sie in den Spalten 4 die LSN des Logsatzes, die betroffene Seite und die Transaktion, die die Änderung ausgeführt hat, ein, also z.B. `17(p,T1)`. Commit-Logsätze haben keinen Seiteneintrag, z.B. `24(T1)` für das Commit von T1 mit LSN 24. In Spalte 5 genügt es, wenn Sie die LSN's der Logeinträge eintragen, die in die Logdatei geschrieben werden.

b) Nehmen Sie an, der Datenbank-Server falle nach der letzten der angegebenen Aktionen aus. Geben Sie an, welche Aktionen zur Recovery ausgeführt werden müssen, indem Sie Tabelle 2 auf dem Deckblatt ausfüllen. Verwenden Sie dazu die Notation `redo(n)`, um die Aktion mit LSN n zu wiederholen, und `undo(m)`, um die Aktion mit LSN m rückgängig zu machen. Berücksichtigen Sie auch Idempotenztests für Redo- und Undo-Schritte; Schritte, die aufgrund des Idempotenztests unterdrückt werden, werden mit "`consider-redo(<LSN>)`" bzw. "`consider-undo(<LSN>)`" in der Tabelle vermerkt. Zur Verkürzung der Redo-Phase im Falle eines erneuten Absturzes schreibt das Datenbanksystem alle im Cache befindlichen Seiten nach Abschluss der Redo-Phase, aber vor Beginn der Undo-Phase in die Datenbank; nehmen Sie die dazu notwendigen Aktionen in der „Flush-Phase" in Tabelle 2 auf.

c) Nehmen Sie an, dass während der Undo-Phase keinerlei Seiten mehr in die Datenbank zurückgeschrieben werden. Nach dem letzten Undo-Schritt falle der Datenbank-Server erneut aus, so dass nach dem Wiederanlauf eine zweite Recovery-Runde benötigt wird. Geben Sie

durch Ausfüllen von Tabelle 3 an, welche Schritte bei dieser zweiten Recovery-Runde durchgeführt werden.

d) Nehmen Sie an, der Log-Manager des Datenbank-Servers protokolliere zusätzlich auch das Zurückschreiben von Seiten aus dem Cache in die Datenbank in Form von "`flush(<Seitennummer>)`"-Logsätzen. Nehmen Sie ferner an, dass alle erzeugten flush-Logsätze vor dem ersten bzw. zweiten Crash vom Logpuffer in die Logdatei geschrieben seien. Während der jeweiligen Analysephase der beiden Recovery-Runden soll dann die Dirty-Pages-Liste so weit wie möglich aktualisiert werden. Wie sieht die Dirty-Pages-Liste nach der ersten bzw. zweiten Analysephase aus, und wie ändern sich dann die während der ersten bzw. zweiten Recovery-Runde durchzuführenden Aktionen?



Aufgabe 6: Recovery & Concurrency Control

Warum kann man bei Verwendung von Tupelsperren als Concurrency–Control–Protokoll die Recovery nicht einfach mittels Before- und After-Images von Seiten realisieren?
Könnte man wenigstens nur die Undo-Phase mittels Seiten-Before-Images durchführen oder nur die Redo-Phase mittels Seiten-After-Images?
Welche Vorteile hätten diese eingeschränkten Varianten gegenüber einer Recovery, die sowohl bezüglich Undo als auch Redo auf tupelorientierten Operationen basiert? Geben Sie ggf. Beispiele an, die Ihre Überlegungen veranschaulichen.

Tabelle 1 für Aufgabe 5 (mit abgeben)

Nr	Aktion	Seitenpuffer [Seite (LSN)]	Datenbank [Seite (LSN)]	Logpuffer [LSN (Seite/Trans.)]	Logdatei [LSN's]
1	BOT(T1)		*a(0),b(0),...*		
2	BOT(T2)				
3	W1(a)	*a(3)*		*3(a/T1)*	
4	BOT(T3)				
5	BOT(T4)				
6	W3(b)				
7	W2(c)				
8	W1(d)				
9	EOT(T1)				
10	Flush(d)				
11	W3(d)				
12	BOT(T5)				
13	W5(a)				
14	Checkpoint	ActiveTrans [Trans(LastLSN)]: DirtyPages [Seite(RedoLSN)]:			
15	EOT(T3)				
16	Flush(d)				
17	W4(d)				
18	W2(e)				
19	W5(b)				
20	Flush(b)				
21	EOT(T4)				
22	W5(f)				
☠ 1. Systemfehler ☠					

Tabelle 2 für Aufgabe 5 (mit abgeben)

Nr	Aktion	Seitenpuffer [Seite (LSN)]	Datenbank [Seite (LSN)]	Logpuffer [LSN (Seite/Trans.)]	Logdatei [LSN's]
☠ 1. Systemfehler ☠					
	Analysephase	Verlierer [Trans(LastLSN)]:		DirtyPages [Seite(RedoLSN)]:	
	Redo-Phase				
23					
	Flush der DB				
	Undo-Phase				
☠ 2. Systemfehler ☠					

Tabelle 3 für Aufgabe 5 (mit abgeben)

Nr	Aktion	Seitenpuffer [Seite (LSN)]	Datenbank [Seite (LSN)]	Logpuffer [LSN (Seite/Trans.)]	Logdatei [LSN's]
	☠ 2. Systemfehler ☠				
	Analysephase	Verlierer [Trans(LastLSN)]:		DirtyPages [Seite(RedoLSN)]:	
	Redo-Phase				
	Flush der DB				
	Undo-Phase				