TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

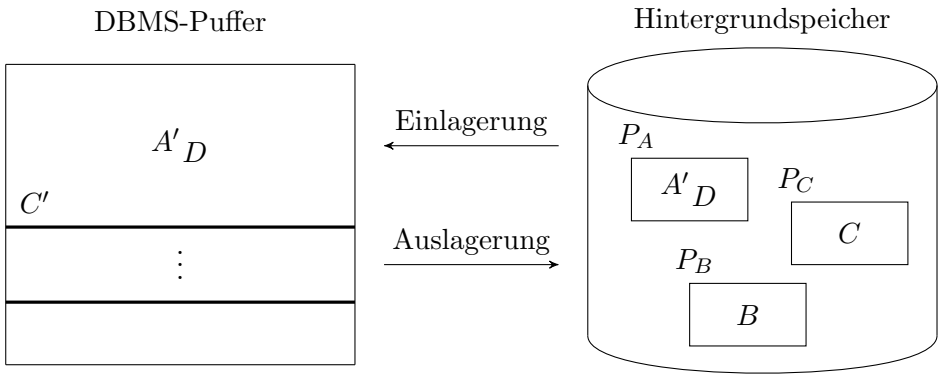
Übung zur Vorlesung *Grundlagen: Datenbanken* im WS18/19

Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws1819/grundlagen/

Blatt Nr. 14

Hausaufgabe 1

Demonstrieren Sie anhand eines Beispiels, dass man die Strategien *force* und ¬*steal* nicht kombinieren kann, wenn parallele Transaktionen gleichzeitig Änderungen an Datenobjekten innerhalb einer Seite durchführen. Betrachten Sie dazu z.B. die unten dargestellte Seitenbelegung, bei der die Seite P_A die beiden Datensätze A und D enthält. Entwerfen Sie eine verzahnte Ausführung zweier Transaktionen, bei der eine Kombination aus *force* und ¬*steal* ausgeschlossen ist.



Hausaufgabe 2

Warum ist es für die Erzielung der Idempotenz der Redo-Phase notwendig, die – und nur die – LSN einer tatsächlich durchgeführten Redo-Operation in der betreffenden Seite zu vermerken? Zeigen Sie für die folgenden Szenarien anhand von Beispielen mit logischer Protokollierung, dass die Idempotenz nicht sichergestellt werden kann.

a) LSN-Eintäge werden in der Redo-Phase nicht auf Datenseiten geschrieben.

b) LSN-Einträge von Log-Records, für die die Redo-Operation nicht ausgeführt wird, werden trotzdem in die Datenseiten übertragen.

Beantworten Sie außerdem folgende Frage:

c) Wie wird die Idempotenz der Undo-Phase sichergestellt, wenn ein Kompensationseintrag geschrieben wurde und dann noch vor der Ausführung des Undo das Datenbanksystem abstürzt?

Hausaufgabe 3

In Abbildung 1 ist die verzahnte Ausführung der beiden Transaktionen T_1 und T_2 und das zugehörige *Log* auf der Basis logischer Protokollierung gezeigt. Wie sähe das Log bei physischer Protokollierung aus, wenn die Datenobjekte A, B und C die Initialwerte 1000, 2000 und 3000 hätten?

Schritt	T_1	T_2	Log [LSN,TA,PageID,Redo,Undo,PrevLSN]
1.	**BOT**		[#1, T_1, **BOT**, 0]
2.	$r(A, a_1)$		
3.		**BOT**	[#2, T_2, **BOT**, 0]
4.		$r(C, c_2)$	
5.	$a_1 := a_1 - 50$		
6.	$w(A, a_1)$		[#3, T_1, P_A, A–=50, A+=50, #1]
7.		$c_2 := c_2 + 100$	
8.		$w(C, c_2)$	[#4, T_2, P_C, C+=100, C–=100, #2]
9.	$r(B, b_1)$		
10.	$b_1 := b_1 + 50$		
11.	$w(B, b_1)$		[#5, T_1, P_B, B+=50, B–=50, #3]
12.	**commit**		[#6, T_1, **commit**, #5]
13.		$r(A, a_2)$	
14.		$a_2 := a_2 - 100$	
15.		$w(A, a_2)$	[#7, T_2, P_A, A–=100, A+=100, #4]
16.		**commit**	[#8, T_2, **commit**, #7]

Abbildung 1: Verzahnte Ausführung zweier Transaktionen und das erstellte Log

Hausaufgabe 4

Sie verwenden ein Datenbanksystem mit Write-Ahead-Logging und der Strategie ¬*force* und *steal*. Die Datenbank verwaltet lediglich zwei Datenobjekte, X mit dem Anfangswert 10 und Y mit dem Anfangswert 100.

Sie starten die 3 Transaktionen T_1, T_2 und T_3 zum gleichen Zeitpunkt:

T_1	T_2	T_3
BOT	**BOT**	**BOT**
$r(X, x_1)$	$r(Y, y_2)$	$r(X, x_3)$
$x_1 := x_1 + 1$	$r(X, x_2)$	$x_3 := x_3 \cdot 10$
$w(X, x_1)$	$y_2 := y_2 \cdot 2$	$w(X, x_3)$
COMMIT	$x_2 := x_2 + 5$	**COMMIT**
	$w(Y, y_2)$	
	$w(X, x_2)$	
	COMMIT	

Während der Ausführung stürzt Ihre Datenbank ab. Sie wissen nicht, ob - und wenn ja, welche - Transaktionen festgeschrieben wurden. Sie wissen nur, dass die Datenbank ausschließlich *serielle Historien* erzeugt, also dass Transaktionen immer atomar ausgeführt werden und somit keine Verzahnung möglich ist. Bevor Sie die Datenbank neu starten, durchsuchen Sie die Festplatte und stellen fest, dass Y dort den Wert 200 hat. Nachdem die Datenbank neu gestartet wurde und der Recovery-Prozess abgeschlossen ist, liefert sie für X den Wert 110.

Sie wollen nun dem Fehler auf den Grund gehen:

a) Finden Sie zunächst anhand der Zwischenwerte für X und Y heraus, welche Transaktionen *winner* sind, und welche *loser*.

b) Geben Sie das Log an, wie es zum Zeitpunkt des Absturzes auf der Platte stand (verwenden Sie logische Protokollierung).

c) Geben Sie das Log nach Beendigung des Recovery-Prozesses an.