

**Übung zur Vorlesung  
„Einsatz und Realisierung von Datenbanksystemen“  
im Sommersemester 2007**

Richard Kuntschke (richard.kuntschke@in.tum.de)

**Blatt 1**

**Aufgabe 1**

Demonstrieren Sie anhand eines Beispiels, dass man die Strategien *force* und  $\neg$ *steal* nicht kombinieren kann, wenn parallele Transaktionen gleichzeitig Änderungen an Datenobjekten innerhalb einer Seite durchführen. Betrachten Sie dazu z.B. die in Abbildung 1 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  $\neg$ *steal* ausgeschlossen ist. Aufgabennummer im Buch: 10.1

**Aufgabe 2**

In Abbildung 2 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?

Aufgabennummer im Buch: 10.2

**Aufgabe 3**

Betrachten Sie Abbildung 3. In Teil (a) ist das *Log* bis LSN #7 skizziert. Was passiert, wenn die auf der Platte stehende *Log*-Datei nur die Einträge bis LSN #6 enthielte? Demonstrieren Sie den Wiederanlauf des Systems unter diesem Gesichtspunkt.

Könnte auch der Eintrag #5 in der temporären *Log*-Datei fehlen, obwohl der Absturz erst nach Schritt 15 in Abbildung 2 stattfand? Welches Prinzip wäre dadurch verletzt?

Aufgabennummer im Buch: 10.5

**Aufgabe 4**

Warum muss beim Anlegen eines transaktionskonsistenten Sicherungspunkts der gesamte *Log*-Ringpuffer ausgeschrieben werden – wo man doch nach Fertigstellung des Sicherungspunkts wieder mit einer „leeren“ *Log*-Datei anfangen kann?

Aufgabennummer im Buch: 10.8

**Aufgabe 5**

Wie weit in die Vergangenheit müssen die Einträge des *Log*-Archives gehen, wenn man einen aktionskonsistenten Zustand der Datenbasis archiviert? Wie sieht in diesem Fall die Wiederherstellung des jüngsten konsistenten DB-Zustands nach Verlust des Hintergrundspeichers aus?

Aufgabennummer im Buch: 10.9

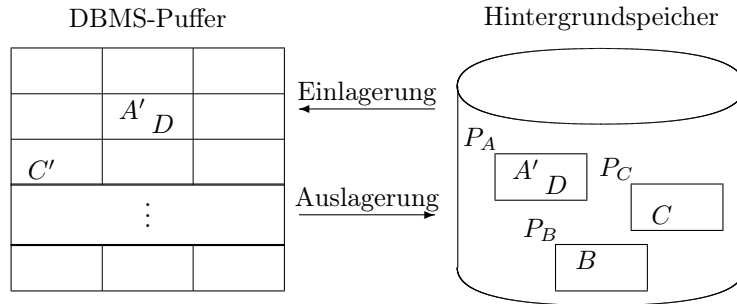


Abbildung 1: Schematische Darstellung der (zweistufigen) Speicherhierarchie

Schritt	$T_1$	$T_2$	Log
			[LSN, TA, PageID, Redo, Undo, PrevLSN]
1.	<b>BOT</b>		[#1, $T_1$ , <b>BOT</b> , 0]
2.	$r(A, a_1)$		
3.		<b>BOT</b>	[#2, $T_2$ , <b>BOT</b> , 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.	<b>commit</b>		[#6, $T_1$ , <b>commit</b> , #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.		<b>commit</b>	[#8, $T_2$ , <b>commit</b> , #7]

Abbildung 2: Verzahnte Ausführung zweier Transaktionen und das erstellte Log

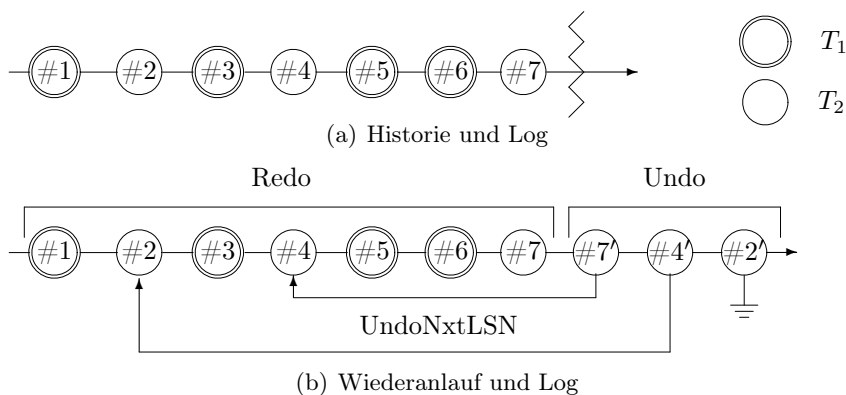


Abbildung 3: Wiederanlauf nach einem Absturz: (a) das vorgefundene Log und (b) die Fortschreibung der Log-Datei aufgrund der *Undo*-Aktionen