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

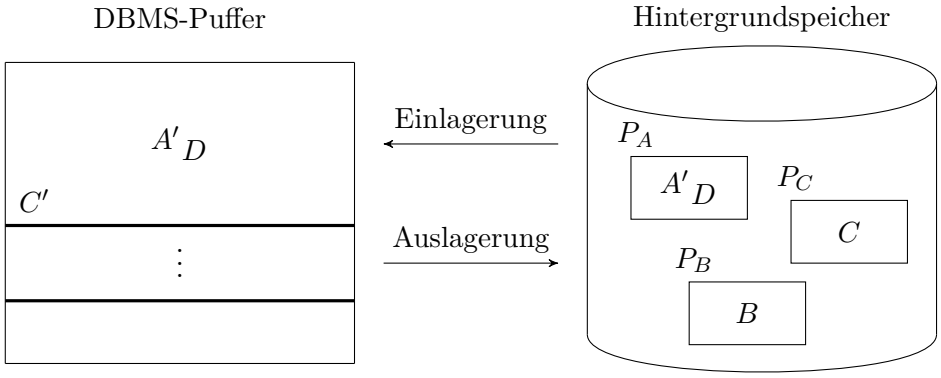
Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe24

Alice Rey, Maximilian Bandle, Michael Jungmair (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss24/impldb/

Blatt Nr. 01

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

Überlegen Sie sich, bei welcher Seitenersetzungsstrategie bei einem Wiederanlauf eine *Redo*- bzw. eine *Undo*-Phase notwendig ist. Verwenden Sie in diesem Zusammenhang den Begriff *dirty*. Welche der beiden Phasen entfällt bei einer Hauptspeicherdatenbank?

Hausaufgabe 3

1. 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?
2. Leider erhalten wir einen Fehler mit Hauptspeicherverlust der in Abbildung 1 gezeigten Ausführung nach Schritt 13. Welche Transaktion ist ein *Winner*, welche ein *Loser*? Geben Sie alle nötigen Kompensations-Rekorde (CLR) an.

Hausaufgabe 4

Sie verwenden ein Datenbanksystem mit Write-Ahead-Logging und der Strategie ¬*force* und *steal*. Die Datenbank verwaltet zwei Datenobjekte A und B mit einem Anfangswert von jeweils 1 ($A = 1$ und $B = 1$). Sie starten zwei Transaktionen T_1 und T_2 zeitgleich:

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.	$w(A, a_1)$		$[\#3, T_1, P_A, A\text{–}{=}50, A\text{+}{=}50, \#1]$
7.		$c_2 := c_2 + 100$	
8.		$w(C, c_2)$	$[\#4, T_2, P_C, C\text{+}{=}100, C\text{–}{=}100, \#2]$
9.	$r(B, b_1)$		
10.	$b_1 := b_1 + 50$		
11.	$w(B, b_1)$		$[\#5, T_1, P_B, B\text{+}{=}50, B\text{–}{=}50, \#3]$
12.	**commit**		$[\#6, T_1,$ **commit**$, \#5]$
13.		$r(A, a_2)$	
14.		$a_2 := a_2 - 100$	
15.		$w(A, a_2)$	$[\#7, T_2, P_A, A\text{–}{=}100, A\text{+}{=}100, \#4]$
16.		**commit**	$[\#8, T_2,$ **commit**$, \#7\ ]$

Abbildung 1: Verzahnte Ausführung zweier Transaktionen und das erstellte Log

T_1	T_2
BOT	**BOT**
$r(A, a_1)$	$r(B, b_2)$
$r(B, b_1)$	$r(A, a_2)$
$a_1 := a_1 + 1$	$b_2 := b_2 \cdot 10$
$b_1 := b_1 + 1$	$a_2 := a_2 \cdot 5$
$w(A, a_1)$	$w(B, b_2)$
$w(B, b_1)$	$w(A, a_2)$
COMMIT	**COMMIT**

Während der Ausführung stürzt Ihre Datenbank ab. Sie wissen nicht, welche Transaktionen erfolgreich ausgeführt worden sind. Nehmen Sie an, die Datenbank erzeugt ausschließlich *serielle Historien*. Bevor Sie die Datenbank neu starten, durchsuchen Sie die Festplatte und stellen fest, dass B dort den Wert 20 hat, A den Wert 2. Auch nach einem Neustart mit erfolgreichem Wiederherstellungsprozess liefert die Datenbank für A den Wert 2.

- Welche der Transaktionen hat erfolgreich committed (*Winner*), welche nicht (*Loser*)?
- Geben Sie das Log in *logischer* Protokollierung an, wie es zum Zeitpunkt des Absturzes auf der Platte stand (A liegt auf Seite `P_A` und B auf Seite `P_B`).
- Geben Sie die im Rahmen des Wiederanlaufs erzeugten CLRs (*compensation log records*) auf Basis logischer Protokollierung an.