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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS16/17

Harald Lang, Linnea Passing (gdb@in.tum.de)
http://www-db.in.tum.de/teaching/ws1617/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 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 ¬*steal* ausgeschlossen ist.

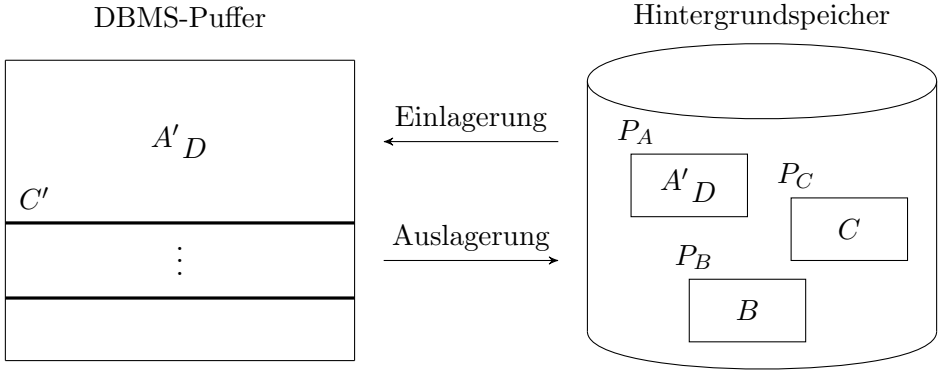

Abbildung 1: Schematische Darstellung der (zweistufigen) Speicherhierarchie

Hausaufgabe 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?

Hausaufgabe 3

Zeigen Sie, dass es für die Erzielung der Idempotenz der *Redo*-Phase notwendig ist, die – und nur die – LSN einer tatsächlich durchgeführten *Redo*-Operation in der betreffenden Seite zu vermerken.

Was würde passieren, wenn man in der *Redo*-Phase gar keine LSN-Einträge in die Datenseiten schriebe?

Was wäre, wenn man auch LSN-Einträge von Log-Records, für die die *Redo*-Operation nicht ausgeführt wird, in die Datenseiten übertragen würde?

Was passiert, wenn der Kompensationseintrag geschrieben wurde, und dann noch vor der Ausführung des *Undo* das Datenbanksystem abstürzt?

Schritt	T_1	T_2	Log [LSN,TA,PageID,Redo,Undo,PrevLSN]
1.	**BOT**		$[\#1, T_1, \textbf{BOT}, 0]$
2.	$r(A, a_1)$		
3.		**BOT**	$[\#2, T_2, \textbf{BOT}, 0]$
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**		$[\#6, T_1, \textbf{commit}, \#5]$
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**	$[\#8, T_2, \textbf{commit}, \#7\ ]$

Abbildung 2: Verzahnte Ausführung zweier Transaktionen und das erstellte Log

Hausaufgabe 4

You are using a database management system that implements the ARIES protocol for logging and recovery. The system uses strict two-phase locking, and the "no-force" and steal strategies. The database has just two items in it, X with starting value 10, and Y with starting value 100. You start three transactions at the same time (TA, TB, and TC):

```
TA:
  BEGIN TRANSACTION
  X = X + 1
  Y = Y * 3
  COMMIT TRANSACTION

TB:
  BEGIN TRANSACTION
  Y = Y * 2
  X = X + 5
  COMMIT TRANSACTION

TC:
  BEGIN TRANSACTION
  X = X * 10
  COMMIT TRANSACTION
```

These three transactions are the only activity in the system. The system crashes due to a power failure soon after you start the transactions. You are not sure whether or not any of them completed. You look at the disk while the system is down and see that, Y has the value 200. You restart the system and let the database recovery procedure complete. You query the database for the value of X, and it returns the value 110.

Please write down a log, as it would have appeared on the disk while the system was down, that is compatible with the above story. You only need to include Update (U), Commit (C), and Abort (A) records. Note that the database system described above (ARIES, 2PL) may abort a transaction. Specify the transaction (TA, TB, or TC) and record type (U/C/A) for each record. For Updates specify additionally the item being written (X or Y) and the new value being written.

TID (TA,TB,TC)	Type (U/C/A)	Item (X,Y)	New Value