

## Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken im SoSe25*

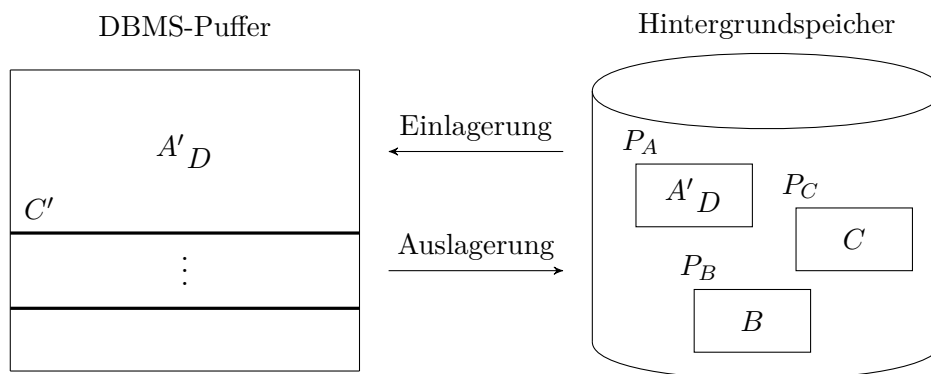
Alice Rey, Maximilian Reif, Tobias Goetz (i3erdb@in.tum.de)

<http://db.in.tum.de/teaching/ss25/impldb/>

### Blatt Nr. 01

#### Hausaufgabe 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 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  $\neg steal$  ausgeschlossen ist.



#### Lösung:

Folgendes Beispiel zeigt, warum man *force* und  $\neg steal$  nicht kombinieren kann:

| Schritt | $T_1$         | $T_2$      |
|---------|---------------|------------|
| 1.      | <b>BOT</b>    |            |
| 2.      |               | <b>BOT</b> |
| 3.      | read(A)       |            |
| 4.      |               | read(D)    |
| 5.      |               | write(D)   |
| 6.      | write(A)      |            |
| 7.      | <b>commit</b> |            |

In Schritt 7 führt  $T_1$  einen commit aus. Aufgrund der *force*-Strategie müssen nun alle von dieser Transaktion geänderten Seiten ausgelagert werden. Im Beispiel hat  $T_1$  nur  $P_A$  geändert, also muss diese ausgelagert werden. Gleichzeitig existiert aber noch eine laufende Transaktion  $T_2$ , die ebenfalls die Seite  $P_A$  verändert hat. Wegen der  $\neg steal$ -Strategie dürfen keine Seiten ausgelagert werden, die von noch nicht beendeten Transaktionen bearbeitet

wurden. Im Beispiel muss also  $P_A$  zwingend ausgelagert werden, da  $T_1$  einen commit ausführt, aber  $P_A$  darf nicht ausgelagert werden, da sie von der noch laufenden Transaktion  $T_2$  verändert wurde, was einen Widerspruch darstellt.

## 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?

**Lösung:**

- $\neg force$ : erfordert eine *Redo*-Phase, da nach einem erfolgreichen *commit* die Änderungen einer somit abgeschlossenen Transaktion (*Winner*) nicht in der Datenbasis materialisiert sein müssen.
- *steal*: erfordert eine *Undo*-Phase, da Änderungen einer noch nicht abgeschlossenen Transaktion (*Loser*) bereits auf der Datenbasis (im Hintergrundspeicher) materialisiert sein können, wenn die entsprechende Seite ausgelagert worden ist, da modifizierte Seiten (*dirty*) von einer Auslagerung nicht ausgeschlossen sind. Eine Seite ist *dirty*, sobald ihr Inhalt geändert worden ist und nicht mehr mit der materialisierten Datenbasis übereinstimmt.
- Bei einer Hauptspeicherdatenbank befindet sich die materialisierte Datenbasis ausschließlich im Hauptspeicher ( $\neg force$  und  $\neg steal$ ), ein Verlust des Hauptspeichers erfordert ein *Redo* der vollständigen Datenbasis, dafür ist eine *Undo*-Phase nicht nötig.

## 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?

**Lösung:**

$[ \#1, T_1, \mathbf{BOT}, 0 ]$   
 $[ \#2, T_2, \mathbf{BOT}, 0 ]$   
 $[ \#3, T_1, P_A, A=950, A=1000, \#1 ]$   
 $[ \#4, T_2, P_C, C=3100, C=3000, \#2 ]$   
 $[ \#5, T_1, P_B, B=2050, B=2000, \#3 ]$   
 $[ \#6, T_1, \mathbf{commit}, \#5 ]$   
 $[ \#7, T_2, P_A, A=850, A=950, \#4 ]$   
 $[ \#8, T_2, \mathbf{commit}, \#7 ]$

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.

**Lösung:**  $T_1$  konnte erfolgreich abschließen und ist somit ein *Winner*,  $T_2$  konnte kein *commit* ausführen und ist ein *Loser*. Daher müssen wir für alle Änderungen von  $T_2$  einen CLR anlegen. Es erfolgte eine Änderung von  $T_2$  in Schritt 8, diese müssen wir, nach einem erfolgten Wiederanlauf aller bis Schritt 13 geschriebenen Log-Dateien, zurücksetzen.

$\langle \#4', T_2, P_C, C-100, \#4, \#2 \rangle$   
 $\langle \#2', T_2, -, -, \#4', 0 \rangle$

| 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 1: Verzahnte Ausführung zweier Transaktionen und das erstellte Log

#### Hausaufgabe 4

Sie verwenden ein Datenbanksystem mit Write-Ahead-Logging und der Strategie  $\neg force$  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:

| $T_1$            | $T_2$                 |
|------------------|-----------------------|
| <b>BOT</b>       | <b>BOT</b>            |
| $r(A, a_1)$      | $r(B, b_2)$           |
| $r(B, b_1)$      | $r(A, a_2)$           |
| $a_1 := a_1 + 1$ | $b_2 := b_2 \cdot 10$ |
| $b_1 := b_1 + 1$ | $a_2 := a_2 \cdot 5$  |
| $w(A, a_1)$      | $w(B, b_2)$           |
| $w(B, b_1)$      | $w(A, a_2)$           |
| <b>COMMIT</b>    | <b>COMMIT</b>         |

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*)?

**Lösung:**  $T_1$  war erfolgreich,  $T_2$  nicht

- 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  $P_B$ ).

**Lösung:**

[#1,  $T_1$ , BOT, 0]

[#2,  $T_1$ ,  $P_A$ ,  $A+=1$ ,  $A-=1$ , #1]

[#3,  $T_1$ ,  $P_B$ ,  $B+=1$ ,  $B-=1$ , #2]

[#4,  $T_1$ , COMMIT, #3]

[#5,  $T_2$ , BOT, 0]

[#6,  $T_2$ ,  $P_B$ ,  $B*=10$ ,  $B/=10$ , #5]

- Geben Sie die im Rahmen des Wiederanlaufs erzeugten CLRs (*compensation log records*) auf Basis logischer Protokollierung an.

**Lösung:**

<#6',  $T_2$ ,  $P_B$ ,  $B/=10$ , #6, #5>

<#5',  $T_2$ , -, -, #6', 0>