



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. 15

ZUSATZBLATT zu Mehrbenutzersynchronisation

Hausaufgabe 1 (optional)

- Erläutern Sie kurz die zwei Phasen des 2PL-Protokolls.
- Inwiefern unterscheidet sich das *strenge* 2PL?
- Welche Eigenschaften (SR,RC,ACA,ST) haben Historien, welche vom 2PL und vom strengen 2PL zugelassen werden?
- Wäre es beim strengen 2PL-Protokoll ausreichend, alle Schreibsperrern bis zum EOT (Transaktionsende) zu halten, aber Lesesperrern schon früher wieder freizugeben?

Lösung:

- Jede Transaktion durchläuft zwei Phasen:
 - Eine *Wachstumsphase*, in der sie Sperrern anfordern, aber keine freigeben darf und
 - eine *Schrumpfungsphase*, in der sie Sperrern freigibt, jedoch keine neuen Sperrern anfordern darf.
- Alle Sperrern werden bis zum Ende der Transaktion gehalten und gemeinsam freigegeben. Die Schrumpfungsphase entfällt somit.
- 2PL garantiert Historien aus SR. Das strenge 2PL garantiert Historien aus $SR \cap ST$.
- Es ist ausreichend, beim strengen 2PL-Protokoll nur die Schreibsperrern bis zum Ende der Transaktion zu halten. Lesesperrern können analog zum normalen 2PL-Protokoll in der Schrumpfungsphase (nach wie vor jedoch nicht in der Wachstumsphase) peu à peu freigegeben werden. Die generierten Schedules bleiben serialisierbar und strikt.

Begründung

- Schon das normale 2PL bietet Serialisierbarkeit; diese ist also auch hier gegeben.
- Das Halten der Schreibsperrern bis zum Ende der Transaktion stellt sicher, dass keine Transaktion von einer anderen lesen oder einen von ihr modifizierten Wert überschreiben kann, bevor diese nicht ihr **commit** durchgeführt hat.

Es gilt:

$$\forall T_i : \forall T_j : (i \neq j) \forall A : (w_i(A) <_H r_j(A)) \vee (w_i(A) <_H w_j(A)) \Rightarrow (c_i <_H r_j(A)) \text{ bzw. } (c_i <_H w_j(A))$$

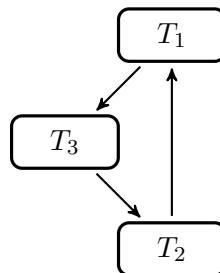
Hausaufgabe 2 (optional)

Ein inhärentes Problem der sperrbasierten Synchronisationsmethoden ist das Auftreten von Verklemmungen (Deadlocks). Zur Erkennung von Verklemmungen wurde der Wartegraph eingeführt. Dabei wird eine Kante $T_i \rightarrow T$ eingefügt, wenn T_i auf die Freigabe einer Sperre durch T wartet.

Skizzieren Sie einen Ablauf von Transaktionen, bei dem ein Deadlock auftritt, der einen Zyklus mit einer Länge von mindestens 3 Kanten im Wartegraphen erzeugt.

Schritt	T_1	T_2	T_3	Bemerkung
1.	BOT			
2.		BOT		
3.			BOT	
4.	lockX(A)			
5.		lockX(B)		
6.			lockX(C)	
7.	write(A)			
8.		write(B)		
9.			write(C)	
10.	lockS(C)			Will C lesen.
11.		lockS(A)		Will A lesen.
12.			lockS(B)	Will B lesen.

Der Wartegraph sieht dann wie folgt aus:



Hausaufgabe 3 (optional)

Kreuzen Sie alle Eigenschaften an, die von den Historien erfüllt werden.

a) $H_1 =$

Schritt	T_1	T_2	T_3
1	$w(x)$		
2		$r(x)$	
3		$w(y)$	
4		c	
5			$r(y)$
6			$w(z)$
7			c
8	c		

richtig	falsch	Aussage
✓		Die Historie ist serialisierbar (SR)
	✓	Die Historie ist rücksetzbar (RC)
	✓	Die Historie ist vermeidet kaskadierendes Rücksetzen (ACA)
	✓	Die Historie ist strikt (ST)

b) $H_2 =$

Schritt	T_1	T_2
1	$w(x)$	
2	$w(z)$	
3		$w(z)$
4	c	
5		$w(x)$
6		c

richtig	falsch	Aussage
✓		Die Historie ist serialisierbar (SR)
✓		Die Historie ist rücksetzbar (RC)
✓		Die Historie ist vermeidet kaskadierendes Rücksetzen (ACA)
	✓	Die Historie ist strikt (ST)