Datenbanksysteme 2, 10. Übung
Transaktionsmanagement

Aufgabe 10.1: Sperrverfahren

In der Vorlesung wurde angesprochen, dass ein einfaches Sperrprotokoll nicht ausreicht, um Serialisierbarkeit zu gewährleisten. Dies wurde an folgendem Beispiel verdeutlicht:

T1	T2
R(a);	
W(a);	
	R(a);
	W(a);
	R(b);
	W(b);
R(b);	
W(b);	

T1		T2	
L(a);	R(a);		
	W(a);		
Ul(a);			
		L(a);	R(a);
			W(a);
		Ul(a);	
		L(b);	R(b);
			W(b);
		Ul(b);	
L(b);	R(b);		
	W(b);		
Ul(b);			

Wie in der VL besprochen, wird zur Lösung üblicherweise das 2-Phasen-Sperrprotokoll verwendet. Dies besagt, dass keine Sperre mehr gesetzt werden darf, sobald die erste Sperre freigegeben wurde. Oder andersherum: Jede Transaktion läuft in 2 Phasen ab, in der ersten Phase werden nur Sperren angefordert (lock), in der zweiten Phase werden nur Sperren freigegeben (daher der Name „2-Phasen-Sperrprotokoll“). Dieses Protokoll gibt es in verschiedenen Varianten. Die „klassische“ Variante beachtet nur die Zweiphasigkeit. Die „strikte“ Variante gibt alle Sperren erst zum Transaktionsende frei; die Variante „Preclaiming“ fordert alle benötigten Sperren bereits zum Start der Transaktion an.

a) Spielen Sie das obige Beispiel mit dem 2-Phasen-Sperrprotokoll durch. Entsteht hierbei ein serialisierbarer Schedule? Verwenden Sie in diesem Beispiel nur `lock` und `unlock` (die Lösung für diese Aufgabe finden Sie auch in den VL-Folien, versuchen Sie es aber zunächst ohne Blick auf die Folien).

T1	T2
L(a);	
R(a);	
W(a);	
	L(a);
	...

L(b);	
Ul(a);	
	R(a);
	W(a);
	L(b);
	...
R(b);	
W(b);	
Ul(b);	
	Ul(a);
	R(b);
	W(b);
	Ul(b);

b) Wie verändert sich der Ablauf für Preclaiming?

T1	T2
L(a);	
L(b);	
R(a);	
W(a);	
Ul(a);	
	L(a);
	L(b);
R(b);	...
W(b);	
Ul(b);	
	R(a);
	W(a);
	Ul(a);
	R(b);
	W(b);
	Ul(b);

Aufgabe 10.2: Mehrfachmodussperren

Betrachten Sie folgende Schedules:

S_1		S_2		S_3	
T1	T2	T1	T2	T1	T2
R(a)		R(a)		R(a)	
a:=a-10			R(b)	a:=a-10	
W(a)		a:=a-10			R(b)
R(b)			b:=b-20	W(a)	
b:=b+10		W(a)			b:=b-20
W(b)			W(b)	R(b)	
	R(b)	R(b)			W(b)
	b:=b-20		R(c)	b:=b+10	
	W(b)	b:=b+10			R(c)
	R(c)		c:=c+20	W(b)	
	c:=c+20	W(b)			c:=c+20
	W(c)		W(c)		W(c)

Ergänzen Sie die Schedules um die Operationen `read_lock(X)`, `write_lock(X)` sowie `unlock(X)` zum Sperren/Entsperren eines Datenbankobjekts X. Dabei sollen Sie das **strikte 2-PL** anwenden. Gehen Sie davon aus, dass eine Sperrenverschärfung möglich ist. Welche der Schedules können mit dem 2-PL ausgeführt werden, welche nicht? Vergleichen Sie dieses Ergebnis mit der Serialisierbarkeit des jeweiligen Schedules.

Zunächst: S1 und S2 sind serialisierbar, S3 hingegen nicht.

S1		S2	
T1	T2	T1	T2
Read_lock(a)		Read_lock(a)	
R(a)		R(a)	
a:=a-10			Read_lock(b)
Write_lock(a)			R(b)
W(a)		a:=a-10	
Read_lock(b)			b:=b-20
R(b)		Write_lock(a)	
b:=b+10		W(a)	
Write_lock(b)			Write_lock(b)
W(b)			W(b)
Unlock(a,b)		Read_lock(b)	
	Read_lock(b)	…	Read_lock(c)
	R(b)		R(c)
	b:=b-20		c:=c+20
	Write_lock(b)		Write_lock(c)
	W(b)		W(c)
	Read_lock(c)		Unlock(b,c)
	R(c)	R(b)	
	c:=c+20	b:=b+10	
	Write_lock(c)	Write_lock(b)	
	W(c)	W(b)	
	Unlock(b,c)	Unlock(a,b)	

S3	
T1	T2
Read_lock(a)	
R(a)	
a:=a-10	
	Read_lock(b)
	R(b)
Write_lock(a)	
W(a)	
	b:=b-20
Read_lock(b)	
R(b)	
	Write_lock(b)
b:=b+10	…
Write_lock(b)	
…	
DEADLOCK	DEADLOCK

Aufgabe 10.3: Zeitstempelverfahren

Betrachten Sie folgende Schedules. Wie würden diese Schedules unter einem Zeitstempelverfahren ablaufen? Nehmen Sie jeweils an, dass T1 den Zeitstempel 1 und T2 den Zeitstempel 2 bekommt. Notieren Sie jeweils die Werte von TSR und TSW für die einzelnen Objekte nach den Lese- und Schreibzugriffen. Wann muss ggf. ein Abbruch einer Transaktion stattfinden?
Wie erklären Sie sich die Ergebnisse? Überlegen Sie dazu, welche Schedules Konflikt-serialisierbar sind.

S_1		S_2		S_3	
T1	T2	T1	T2	T1	T2
R(a)		R(a)		R(a)	
	R(a)		R(b)	a:=a-10	
	a:=a+10	a:=a-10			R(b)
	W(a)		b:=b-20	W(a)	
	R(b)	W(a)			b:=b-20
	b:=b+10		W(b)	R(b)	
	W(b)	R(b)			W(b)
R(b)			R(c)	b:=b+10	
c:=a+b		b:=b+10			R(c)
W(c)			c:=c+20	W(b)	
		W(b)			c:=c+20
			W(c)		W(c)

S_1		S_2		S_3	
T1	T2	T1	T2	T1	T2
R(a) TSR(a)=1		R(a) TSR(a)=1		R(a) TSR(a)=1	
	R(a) TSR(a)=2		R(b) TSR(b)=2	a:=a-10	
	a:=a+10	a:=a-10			R(b) TSR(b)=2
	W(a) TSW(a)=2		b:=b-20	W(a) TSW(a)=1	
	R(b) TSR(b)=2	W(a) TSW(a)=1			b:=b-20
	b:=b+10		W(b) TSW(b)=2	R(b) TSR(b)=2 (keine Änderung!)	
	W(b) TSW(b)=2	R(b) TS(T) < TSW(b) -> abort!			W(b) TSW(b)=2
R(b) TS(T)< TSW(b) -> abort!			R(c) TSR(c)=2	b:=b+10	
					R(c) TSR(c)=2
			c:=c+20	W(b) TS(t) < Max(TSR,TSW) ➔ Abort!	
					c:=c+20
			W(c) TSW(c)=2		W(c) TSW(c)=2

Zunächst:

- Der Ablauf S_1 ist nicht konflikt-serialisierbar da R(a) aus T_1 nicht nach T_2 und R(b) aus T_1 nicht vor T_2 geschoben werden kann.
- Ablauf S_2 ist konflikt-serialisierbar mit der seriellen Ausführung T_2 vor T_1.
- Ablauf S_3 ist nicht konflikt-serialisierbar wegen der Reihenfolge R(b) in T_1 vor W(b) in T_2 in der Mitte.

Das Zeitstempelverfahren kann also auch konflikt-serialisierbare Schedules verhindern (siehe S_2). Das liegt daran, dass im Prinzip nur Schedules zugelassen werden, die der seriellen Ausführung in der Reihenfolge des Transaktionsbeginns genügen, hier also T_1 vor T_2. Zumindest werden aber nicht konflikt-serialisierbare Schedules auch zuverlässig verhindert.