UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Lehrstuhl Prof. Weikum

Datenbanksysteme WS 1999/2000

Übung 10 — Beispiellösung

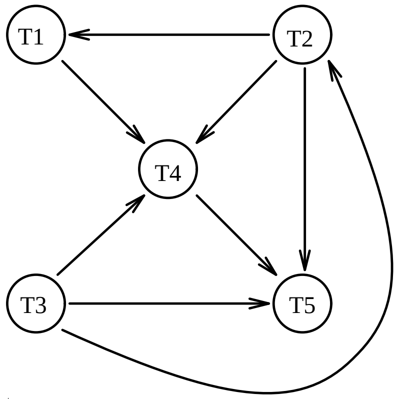
Aufgabe 1: Serialisierbarkeit I

Der Schedule ist serialisierbar.

Konfliktpaare des Schedules:

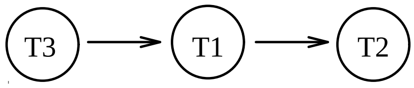
<R2(b), W1(b)>, <R1(a), W4(a)>, <W3(c), W2(c)>,
<W2(c), W4(c)>, <W4(c), W5(c)>, <W2(d), W5(d)>,
<W3(c), W4(c)>, <W3(c), W5(c)>, <W2(c), W5(c)>

Der dazugehörige Abhängigkeitsgraph des Schedules sieht folgendermaßen aus und ist azyklisch. Durch die topologische Sortierung erhält man den äquivalenten seriellen Schedule : T3 T2 T1 T4 T5.



Aufgabe 2: Serialisierbarkeit II

a) Der Schedule ist serialisierbar.
   Konfliktpaare des Schedules: <R1(a), W2(a)>, <W3(b), R1(b)>
   Der dazugehörige Abhängigkeitsgraph des Schedules sieht folgendermaßen aus und ist azyklisch. Durch die topologische Sortierung erhält man den äquivalenten seriellen Schedule : T3 T1 T2.



b) Der Schedule kann nicht durch das Zweiphase–Sperrprotokoll entstanden sein, da:

1	2	3	4	5	6	7	8
R1(a)						R1(c)	R1(b)
	R2(a)	W2(a)		R2(c)			
			R3(b)		W3(b)		

1. Transaktion T1 muß spätestens zum Zeitpunkt 1 die Seite a sperren (d.h. $L1(a) \leq 1$).
2. Transaktion T2 muß spätestens zum Zeitpunkt 2 die Seite a sperren (d.h. $L2(a) \leq 2 \Rightarrow U1(a) < 2$).
3. Transaktion T1 muß zum Zeitpunkt 8 eine Sperre auf der Seite b halten (d.h. $U1(b) > 8$).
4. Transaktion T3 muß die Sperre auf Seite b mindestens bis zum Zeitpunkt 6 halten (d.h. $U3(b) > 6 \Rightarrow L1(b) \geq 6$).

Aus (1) und (2) folgt, daß die Schrumpfungsphase von Transaktion T1 vor dem Zeitpunkt 2 beginnen muß, während aus (3) und (4) folgt, daß die Wachstumsphase über den Zeitpunkt 7 hinaus andauern muß.
$\Rightarrow$ Widerspruch

Aufgabe 3: Sperrprotokoll

a) Schedule S: R1(x) W2(x) W2(y) R1(y) ist nicht serialisierbar und kann jedoch bei Verwendung des gegebenen Protokolls entstanden sein:

b) Nein, da alle Änderungstransaktionen das strikte 2PL-Protokoll verwenden. Allerdings können Lesetransaktionen scheinbare Verletzungen der Integrität "beobachten" (inkonsistentes Lesen).

Betrachten wir die folgende Situation: In der Datenbank ist eine Integritätsbedingung gegeben, welche verlangt, daß bei Geld-Transfer zwischen Konto x und Konto y die Summe von x und y konstant bleibt, beispielsweise x+y=1200. Seien x=1000 DM und y=200 DM. Schedule S: R1(x) W2(x) W2(y) R1(y) beinhaltet zwei Transaktionen, wobei Transaktion T2 eine bestimmte Menge von Geld, beispielsweise 400 DM, von Konto x nach Konto y transferiert und Transaktion T1 die Summe von Konto x und Konto y prüft. Anhand des folgenden Ablaufs ist leicht zu sehen, daß die Integritätsbedingung für T1 verletzt ist, da die Summe von x und y 1600 anstatt 1200 ist.

T1	T2
x=1000	y=200
R1(x) $\rightarrow$ x=1000	
	W2(x) $\rightarrow$ x=x-400 =600
	W2(y) $\rightarrow$ y=y+400 =600
R1(y) $\rightarrow$ y=600	
x+y=1600$\neq$ Konstant 1200	
x=600	y=600

c) Das angegebene Sperrprotokoll kann z.B. für statistische Auswertungen sinnvoll sein, die nicht unbedingt eine hohe Anforderung an die Genauigkeit stellen, jedoch sehr lange laufen.

Betrachten wir das folgende Anwendungsbeispiel: In einem großen Lagerhaus wird eine wöchentliche Statistik erstellt. Diese Statistik gibt eine Übersicht, welche und wieviele Artikel in einer Woche verkauft worden sind, um den Markt-Trend festzustellen, weshalb die Angaben über die einzelnen Artikel nicht sehr genau zu sein brauchen. Die Erstellung dieser Statistik dauert sehr lange, u.U. mehrere Stunden. Parallel zur Erstellung dieser Statistik laufen die normalen betrieblichen Buchungsprogramme und sollen möglichst wenig behindert werden.

Für das oben beschriebene Szenario ist das angegebene Protokoll sehr sinnvoll, da ansonsten für die Erstellung der Statistik alle Artikel gesperrt würden und erst nach Beenden der Erstellung freigegeben würden. Alle anderen Buchungsprogramme würden blockiert und müßten sehr lange warten.