Dr. rer. nat. Sven Groppe

Übungen zur Vorlesung

Mobile und Verteilte Datenbanken

WS 2009/2010

Blatt 4

Aufgabe 1:

Bestimmen Sie zu den folgenden Transaktions-Schedules, ob diese (konflikt-) serialisierbar sind. Falls ja, geben Sie ein äquivalentes serielles Schedule an. Falls nein, begründen Sie dies:

a)

T_1	T_2	T_3	T_4
		read(b)	
		write(b)	
			write(b)
	read(b)		
read(a)			
read(c)			
write(a)			
write(c)			
		read(a)	
		write(c)	
	read(a)		
	write(c)		

b)

T_1	T_2	T_3
read(a)		
read(b)		
write(a)		
		read(a)
	read(b)	
		write(c)
	read(c)	
	write(b)	
	read(a)	
		write(a)
	write(c)	
	write(a)	

Aufgabe 2:

Sowohl die Methodik der Validierung als auch die Methodik der Benutzung von Locks verhindern teilweise Parallelität, die nach dem Konzept der Serialisierbarkeit möglich wäre. Zeigen Sie an einem möglichst einfachen Beispiel, dass es parallele Abläufe von Transaktionen gibt, bei denen die 2-Phasigkeit unnötig Parallelität verhindert. Anders formuliert: Schreiben Sie einen möglichst einfachen Beispielablauf zweier 2-phasiger Transaktionen auf, bei dem eine Transaktion (unnötig) auf die andere wartet, weil die andere Transaktion 2-phasig sperrt, der aber auch serialisierbar wäre, wenn die andere Transaktion beim Sperren nicht 2-phasig vorginge.

Aufgabe 3:

Angenommen eine Transaktion T_1 macht zuerst ein *read(x)*, dann ein *read(y)*, dann ein *write(x)* und schließlich ein *write(y)*. Transaktion T_2 macht ein *read(y)* und dann ein *write(x)*. Geben Sie parallele Abläufe der beiden Transaktionen an, die serialisierbar sind und

a) wiederherstellbar (recoverable) aber nicht kaskadenfrei (cascadeless)
b) kaskadenfrei aber nicht strikt
c) strikt

Aufgabe 4:

Sperrende Transaktionen: um Deadlock-frei zu sperren können die Relationen R_1, R_2, ..., R_n global angeordnet werden und benötigte Sperren in der Reihenfolge dieser Ordnung angefordert werden. Man könnte durch einen Widerspruchbeweis zeigen, dass diese Anordnung der Betriebsmittel zyklische Wartegraphen verhindert, also Deadlock-Freiheit garantiert.

a) Wenn man zwischen Readlock(A) und Writelock(A) unterscheiden will und Deadlock-freies Sperren erweitern will auf Readlock- und Writelock-Operationen, in welcher Reihenfolge muss jede Transaktion dann ihre Sperranforderungen anordnen, um Deadlocks zu vermeiden?
b) Ergänzen Sie folgende Transaktion um Readlock- und Writelock-Befehle (Unlock braucht nicht extra eingetragen zu werden):

   read(R_2);
   read(R_1);
   write(R_3);
   read(R_4);

write(R_4);

Aufgabe 5:

Sowohl die Methodik der Validierung als auch die Methodik der Benutzung von Locks verhindern teilweise Parallelität, die nach dem Konzept der Serialisierbarkeit möglich wäre.
Geben Sie ein möglichst einfaches Beispiel für einen parallelen Ablauf zweier Transaktionen an, der serialisierbar wäre, bei dem aber trotzdem in der Validierungsphase eine Transaktion zurückgesetzt wird.