UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Prof. Dr.-Ing. G. Weikum
Dipl.-Inform. R. Schenkel

Datenbanksysteme WS 1999/2000

Musterlösung zur dritten Teilklausur

Aufgabe 1: Serialisierbarkeit (2 Punkte)

Gegeben ist der folgende Schedule der Transaktionen T1, ..., T5:

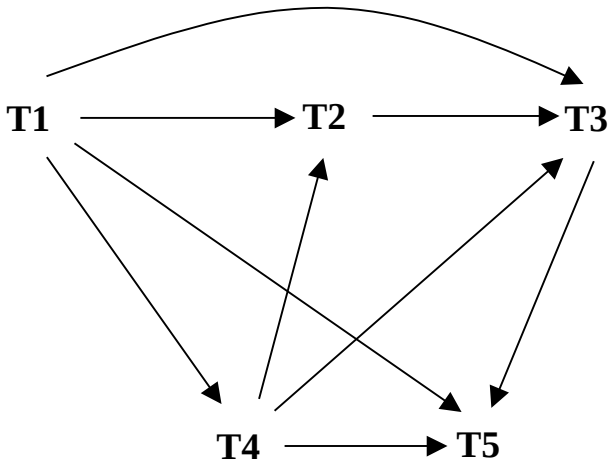
$r_1(z)\ w_1(z)\ r_2(x)\ r_3(x)\ w_3(x)\ w_1(y)\ r_4(y)\ w_4(y)\ r_3(q)\ w_3(q)\ r_2(y)\ r_5(y)\ w_4(z)\ r_3(z)\ w_5(z)\ w_5(q)$

Geben Sie den Abhängigkeitsgraphen für diesen Schedule an. Zeigen Sie durch Angabe eines (konflikt-)äquivalenten seriellen Schedules, dass der Schedule konflikt-serialisierbar ist.

Konfliktpaare:

<r2(x),w3(x)>, <w1(y),r4(y)>, <w1(y),w4(y)>, <w1(y),r2(y)>, <w1(y),r5(y)>, <w4(y),r2(y)>, <w4(y),r5(y)>, <r1(z),w4(z)>, <w1(z),w4(z)>, <w1(z),r3(z)>, <r1(z),w5(z)>, <w1(z),w5(z)>, <w4(z),r3(z)>, <w4(z),w5(z)>, <r3(q),w5(q)>, <w3(q),w5(q)>

Abhängigkeitsgraph:



Äquivalenter serieller Schedule: (Durch Topsort des Graphen)

T1 T4 T2 T3 T5

Aufgabe 2: Recovery (4 Punkte + 1 Bonuspunkt):

Gegeben sei der in der in Spalte 1 von Tabelle 1 auf dem Deckblatt aufgeführte chronologische Ablauf von Aktionen. Der Checkpoint, der in diesem Ablauf ausgeführt wird, ist ein asynchroner. Tragen Sie bitte Ihre Lösungen der Teilaufgaben a) bis c) direkt in die vorgegebenen Tabellen ein, und geben Sie die Tabellen mit ab.

a) Geben Sie die vom Log-Manager zu protokollierende Information durch Ausfüllen der Spalten 4 und 5 von Tabelle 1 sowie die Änderungen von Seitenheader-LSNs durch Aus-

füllen der Spalten 2 und 3 an. Verwenden Sie die Nummerierung der Aktionen für die LSNs.

Verwenden Sie bei dieser und allen folgenden Teilaufgaben bitte die folgende Notation: Tragen Sie in den Spalten 2 und 3 jeweils die betroffene(n) Seite(n) mit ihre(n) LSN ein, z.B. `p(17)` bedeutet Seite p mit LSN 17. Tragen Sie in den Spalten 4 die LSN des Logsatzes, die betroffene Seite und die Transaktion, die die Änderung ausgeführt hat, ein, also z.B. `17(p,T1)`. Commit-Logsätze haben keinen Seiteneintrag, z.B. `24(T1)` für das Commit von T1 mit LSN 24. In Spalte 5 genügt es, wenn Sie die LSN's der Logeinträge eintragen, die in die Logdatei geschrieben werden.

Nr	**Aktion**	**Seitenpuffer [Seite (LSN)]**	**Datenbank [Seite (LSN)]**	**Logpuffer [LSN (Seite/Trans.)]**	**Logdatei [LSN's]**
1	BOT(T1)		*a(0),b(0),...*		
2	BOT(T2)				
3	W1(a)	*a(3)*		*3(a/T1)*	
4	BOT(T3)				
5	BOT(T4)				
6	W3(b)	b(6)		6(b/T3)	
7	W2(c)	c(7)		7(c/T2)	
8	W1(d)	d(8)		8(d/T1)	
9	EOT(T1)			9(T1)	3,6,7,8,9
10	Flush(d)		d(8)		
11	W3(d)	d(11)		11(d/T3)	
12	BOT(T5)				
13	W5(a)	a(13)		13(a/T5)	
14	Checkpoint	ActiveTrans [Trans(LastLSN)]: T2,T3,T4,T5 DirtyPages [Seite(RedoLSN)]: a(3), b(6), c(7), d(11)			11,13, 14:CP({T2,T3,T4,T5}, {a(3),b(6),c(7),d(11)})
15	EOT(T3)			15(T3)	15
16	Flush(d)		d(11)		
17	W4(d)	d(17)		17(d/T4)	
18	W2(e)	e(18)		18(e/T2)	
19	W5(b)	b(19)		19(b/T5)	
20	Flush(b)		b(19)		17,18,19
21	EOT(T4)			21(T4)	21
22	W5(f)	f(22)		22(f/T5)	
☠ 1. Systemfehler ☠					

b) Nehmen Sie an, der Datenbank-Server falle nach der letzten der angegebenen Aktionen aus. Geben Sie an, welche Aktionen zur Recovery ausgeführt werden müssen, indem Sie Tabelle 2 auf dem Deckblatt ausfüllen. Verwenden Sie dazu die Notation `redo(n)`, um die Aktion mit LSN n zu wiederholen, und `undo(m)`, um die Aktion mit LSN m rückgängig zu machen. Berücksichtigen Sie auch Idempotenztests für Redo- und Undo-Schritte; Schritte, die aufgrund des Idempotenztests unterdrückt werden, werden mit "`consider-redo(<LSN>)`" bzw. "`consider-undo(<LSN>)`" in der Tabelle vermerkt. Zur Verkürzung der Redo-Phase im Falle eines erneuten Absturzes schreibt das Datenbanksystem alle im Cache befindlichen Seiten nach Abschluss der Redo-Phase, aber vor Beginn der Undo-

Phase in die Datenbank; nehmen Sie die dazu notwendigen Aktionen in der „Flush-Phase“ in Tabelle 2 auf.

Nr	Aktion	Seitenpuffer [Seite (LSN)]	Datenbank [Seite (LSN)]	Logpuffer [LSN (Seite/Trans.)]	Logdatei [LSN's]
☠ 1. Systemfehler ☠					
	Analysephase	Verlierer [Trans(LastLSN)]: T2, T5		DirtyPages [Seite(RedoLSN)]: a(3),b(6),c(7),d(11),e(18)	
	Redo-Phase				
23	redo(3)	a(3)			
24	consider_redo(6)				
25	consider_redo(8)				
26	consider_redo(11)				
27	redo(17)	d(17)			
	Flush der DB				
28	flush(a)		a(3)		
29	flush(d)		d(17)		
	Undo-Phase				
30	undo(19)	b(6)			
	consider_undo(18)				
	consider_undo(13)				
	consider_undo(7)				
☠ 2. Systemfehler ☠					

c) Nehmen Sie an, dass während der Undo-Phase keinerlei Seiten mehr in die Datenbank zurückgeschrieben werden. Nach dem letzten Undo-Schritt falle der Datenbank-Server erneut aus, so dass nach dem Wiederanlauf eine zweite Recovery-Runde benötigt wird. Geben Sie durch Ausfüllen von Tabelle 3 an, welche Schritte bei dieser zweiten Recovery-Runde durchgeführt werden.

Nr	Aktion	Seitenpuffer [Seite (LSN)]	Datenbank [Seite (LSN)]	Logpuffer [LSN (Seite/Trans.)]	Logdatei [LSN's]
	☠ 2. Systemfehler ☠				
	Analysephase	Verlierer [Trans(LastLSN)]: T2, T5		DirtyPages [Seite(RedoLSN)]: a(3),b(6),c(7),d(11),e(18)	
	Redo-Phase				
	consider_redo(3)				
	consider_redo(6)				
	consider_redo(11)				
	consider_redo(17)				
	Flush der DB				
	Undo-Phase				
	undo(19)	b(6)			
	consider_undo(18)				
	consider_undo(13)				
	consider_undo(7)				

d) *Bonusaufgabe:* Nehmen Sie an, der Log-Manager des Datenbank-Servers protokolliere zusätzlich auch das Zurückschreiben von Seiten aus dem Cache in die Datenbank in Form von "`flush(<Seitennummer>)`"-Logsätzen. Nehmen Sie ferner an, dass alle erzeugten flush-Logsätze vor dem ersten bzw. zweiten Crash vom Logpuffer in die Logdatei geschrieben seien. Während der jeweiligen Analysephase der beiden Recovery-Runden soll dann die Dirty-Pages-Liste so weit wie möglich aktualisiert werden. Wie sieht die Dirty-Pages-Liste nach der ersten bzw. zweiten Analysephase aus, und wie ändern sich dann die während der ersten bzw. zweiten Recovery-Runde durchzuführenden Aktionen?

Nach der 1. Analysephase:

a(3),c(7),d(17),e(18)

Nach der 2. Analysephase:

c(7),e(18)

Aktionen in der ersten Recovery-Runde:

keine Änderung bei redo, keine Änderung bei undo

Aktionen in der zweiten Recovery-Runde:

kein Redo mehr erforderlich, keine Änderung bei undo

Aufgabe 3: Relationaler Entwurf (4 Punkte)

Betrachten Sie das folgende Schema einer relationalen Datenbank über alle Bankkonten Deutschlands (die z.B. für steuerliche Zwecke o.ä. aufgebaut werden könnte):

R(Bank, Konto, Direktor, Stand, Typ, Prozent, Zins, Inhaber, Adresse)

Die Attribute von R haben die folgende Bedeutung:

Bank	Name der Bank
Konto	Nummer des Bankkontos
Direktor	Name des Direktors der Bank
Stand	aktueller Kontostand in Euro
Typ	Typ des Kontos (z.B. Privatkonto, Sparkonto, Anlagekonto,...)
Prozent	Zinssatz in Prozent
Zins	Zinsen für den heutigen Tag (Absolutbetrag in Euro)
Inhaber	Name des Kontoinhabers
Adresse	Adresse des Kontoinhabers

Folgende Funktionalabhängigkeiten seien spezifiziert:

{
Bank → Direktor
Bank, Konto → Stand, Typ, Inhaber
Bank, Typ → Prozent
Prozent, Stand → Zins
Inhaber → Adresse
}

a) Bestimmen Sie alle Schlüsselkandidaten von R.

b) Ist R in 3NF? Ist R in BCNF? Begründen Sie Ihre Antwort.

c) Falls R nicht in BCNF ist, so führen Sie eine verlustfreie Zerlegung durch. Prüfen Sie, in welcher Normalform sich die durch die Zerlegung entstandenen Relationen befinden und zerlegen Sie gegebenenfalls weiter, bis die Ergebnisrelationen in 3NF oder BCNF sind. Sind bei Ihrer Zerlegung Abhängigkeiten verlorengegangen?

Hinweis: Sie können bei allen Teilaufgaben die Attributnamen mit dem jeweils eindeutigen Anfangsbuchstaben abkürzen (z.B. ‘B‘ statt ‘Bank‘).

a) In jedem Schlüsselkandidaten müssen offenbar *Bank* und *Konto* auftauchen, da sie aus den übrigen Attributen nicht (durch Funktionalabhängigkeiten) abgeleitet werden können. Gleichzeitig gilt aber auch, dass aus *Bank* und *Konto* alle übrigen Attribute abgeleitet werden können. Also ist die Menge *{Bank, Konto}* der einzige Schlüsselkandidat.

b) R ist nicht in BCNF, da z.B. in B→D die linke Seite keinen Schlüsselkandidaten enthält. R ist auch nicht in 3NF, da die rechte Seite außerdem kein Schlüsselattribut enthält.

c) Zerlegung von R entlang den Funktionalabhängigkeiten:

- Zerlegung von R entlang B→D:

  R1(B,D); R2(B,K,S,T,P,Z,I,A)

- Zerlegung von R2 entlang I→A:

  R21(I,A); R22(B,K,S,T,P,Z,I)

- Zerlegung von R22 entlang P,S→Z

  R221(P,S,Z); R222(B,K,S,T,P,I)

- Zerlegung von R222 entlang B,T→P

  R2221(B,T,P); R2222(B,K,S,T,I)

Diese Relationen sind jetzt alle in BCNF.

Die Zerlegung ist außerdem abhängigkeitsbewahrend.