DES3UE WS 2024 Übung 6

Abgabe: elektronisch, Abgabetermin siehe e-Learning

☒ **DES3UEG1: Glock** **Name** Elias Leonhardsberger **Aufwand in h** 4

☐ **DES3UEG2: Werth** **Punkte** _____ **Kurzzeichen Tutor** _____

Hinweis: Für die Ausarbeitung von Bsp 5.2 benötigen Sie eine(n) Partner(in). Geben Sie diese(n) in der Lösung an.

1. Fehlersicherheit von Transaktionen (9 Punkte)

Stellen Sie fest, welcher Schedule strikt (ST), kaskadenlos (ACA) und/oder rücksetzbar (RC) ist. **Begründen** Sie Ihre Antwort für jeden Schedule.

- S_1 = rT2(x) rT2(y) **cT1** wT2(x) rT2(y) **cT3** rT2(x) c**T2**
- S_2 = wT2(y) wT1(z) wT2(z) rT1(z) wT3(z) wT1(z) rT1(y) wT1(z) wT2(y) **cT3 cT2 cT1**
- S_3 = wT3(x) wT1(x) rT3(y) **cT1** wT3(x) c**T3** wT2(x) **cT2**

	RC	ACA	ST
S_1			
S_2			
S_3			

2. Konfliktgraph (6 Punkte)

Analysieren Sie folgende Schedules entsprechend der Angabe unten.

- S1 = wT3(z) rT1(y) rT3(z) rT2(z) wT1(x) wT1(x) rT2(z) rT1(y) rT3(z) rT3(z) rT3(z) rT2(y)
- S2 = rT1(x) rT1(x) wT3(y) wT3(y) wT1(y) wT2(x) rT1(x) rT1(y) wT1(z) rT1(z) rT3(x) rT3(y)
- S3 = rT1(y) wT3(x) rT1(z) rT3(z) rT3(x) wT2(y) wT3(x) wT2(y) rT3(y) rT1(x) wT1(z) rT2(y)

Für jeden Schedule (Ausführungsplan):

- Geben Sie die Konfliktrelationen an (C(S) = {}).
- Zeichnen Sie jeweils den (gesamten) Konfliktgraphen (Serialisierbarkeitsgraph)
- Identifizieren Sie, ob der Ausführungsplan serialisierbar ist. Je nachdem notieren Sie einen äquivalenten seriellen Ausführungsplan oder die Konflikte.

3. Zeitstempelverfahren (3 Punkte)

Überprüfen Sie die Serialisierbarkeit des angegebenen Schedules der Transaktionen T1, T2 und T3 mit Hilfe des (*nicht* optimierten) Zeitstempel-Verfahrens.

wT1(c) wT2(a) rT2(d) rT2(d) wT2(b) wT2(a) rT1(d) wT3(d) rT2(a) wT2(b) wT3(c) wT3(a) rT3(b) wT1(b) rT2(b)

Welche Transaktion wird abgebrochen, wenn den Transaktionen T1, T2 und T3 die Zeitstempel 75, 85 und 100 zugewiesen werden? Stellen Sie eine entsprechende Tabelle auf.

4. COMMIT, SAVEPOINT und ROLLBACK (0,5+0,5+1+0,5+0,5 = 3 Punkte)

1. Erstellen Sie eine Tabelle TEMP_EMPLOYEES als Kopie der Tabelle employees. Verwenden Sie dazu einen CREATE … AS SELECT … Befehl.

2. Erhöhen Sie das Gehalt aller Manager um 5 % (MGR und MAN). **Schreiben Sie die veränderten Daten dauerhaft fest.**

3. Ändern Sie das Gehalt aller Angestellten, deren Gehalt unter $ 5000 liegt, in $ 6100. **Markieren Sie einen Zwischenpunkt in der Verarbeitung der Transaktion.**

4. Leeren Sie die gesamte Tabelle, **ohne die Tabelle selbst zu löschen**, und prüfen Sie, ob die Tabelle leer ist.

5. Verwerfen Sie die letzte DELETE-Operation, ohne die vorherige UPDATE-Operationen zu verwerfen; prüfen Sie, ob die Änderungen aus Punkt 4.4 weiterhin vorhanden sind und schreiben Sie die Änderungen fest.

5. Vergabe und Entzug von Rechten (1+2=3 Punkte)

1. Fragen Sie die View `USER_TABLES` und die View `ALL_TABLES` im Data Dictionary ab, um Informationen über die Tabellen anzuzeigen, die Ihnen gehören und auf die Sie zugreifen können. Bei der Abfrage auf die View `ALL_TABLES` schließen Sie die Tabellen aus, die Ihnen gehören.

2. Gewähren Sie einer*m anderen Benutzer*in lesenden Zugriff auf Ihre Tabelle `DEPARTMENTS`. Lassen Sie sich von dieser*m Benutzer*in das Privileg zur Abfrage seiner*ihrer Tabelle `DEPARTMENTS` erteilen und testen Sie den Zugriff. Überprüfen Sie zusätzlich mit der Abfrage aus 5.1.

Zusatzaufgabe (+ 3 Punkte)

Z1. Fehler bei Mehrbenutzerbetrieb (3 Punkte)

Kreuzen Sie die richtige(n) Antwort(en) zu den gegebenen Fragen bezüglich Fehler im Mehrbenutzerbetrieb und zum Umgang von Datenbanken damit an.

a) Welche Aussagen treffen auf die angegebene Ausführung zu?

T1	T2
R(A)	
	R(A)
	A = A * 2
A = A + 50	
W(A)	
	W(A)
COMMIT	
	COMMIT

- □ Dieses Problem wird als *Non-Repeatable Read* beschrieben, da sich A bei einem erneuten Lesen von T1 verändert hat.
- □ Mit dem Schreiben von T2 gehen die Änderungen von T1 verloren, dies wird auch als *Lost Update* bezeichnet.
- □ Würde T1 nach dem Schreiben abgebrochen, so ist der vorangegangene Lesevorgang ungültig (*Dirty Read*).
- □ Hier erfolgt ein *Lesen von inkonsistenten Zuständen*, da A nie größer als 10 sein darf.

b) Welche Aussagen treffen auf die angegebene Ausführung zu?

T3	T4
	R(X)
R(X)	
	R(Y)
	X = Y * 2
R(Y)	
	W(X)
R(X)	
	Y = Y + 1
COMMIT	
	COMMIT

- □ Da T3 zum ersten Mal X liest, nachdem dieses bereits vorher von T4 gelesen wurde, entsteht ein *Dirty Read*-Problem.
- □ T4 verändert Y, dieses wird allerdings nicht geschrieben – es liegt somit ein *Lost Update* vor.
- □ Es handelt sich um ein *Non-Repeatable Read* Problem, da T3 beim erneuten Lesen von X andere Werte erhält.
- □ Da Y erst nach der Neuberechnung von X verändert wird, entsteht ein sogenanntes *Phantom*-Problem.

c) Welche Aussagen treffen auf die unten angegebene Ausführung zu?

T5	**T6**
R(C = COUNT(X))	
	R(X)
	INS(DATA,X)
	COMMIT
R(S = SUM(X))	
A = S / C	
W(A)	
COMMIT	

□ Die Berechnung von A enthält keine in sich stimmigen Daten, dies wird als *Lost Update* bezeichnet.

□ Dies ist ein *Phantom*-Problem, da Datensätze während einer Berechnung eingefügt werden und in T5 nicht sichtbar sind.

□ Beim *Lesen inkonsistenter Zustände* werden nicht zusammenhängende Daten bearbeitet.

□ Da sich die Daten zwischen den Leseoperationen von T5 verändert haben, spricht man von einem *Non-Repeatable Read*.

DES3UE Übung 6

Elias Leonhardsberger

23. Dezember 2024, Hagenberg

Inhaltsverzeichnis

1 Fehlersicherheit vor Transaktionen

	RC	ACA	ST
S_1	X	X	X
S_2			
S_3	X	X	

1.1 S_1

x	y
rT2	rT2
cT1	
wT2	rT2
cT3	
rT2	
cT2	

S_1 ist strikt seriell da nur T2 liest und schreibt. Da es strikt ist, ist es auch kaskadenlos und rücksetzbar.

1.2 S_2

y	z
wT2	wT1
rT1	wT2
	rT1
	wT3
	wT1
	wT1
	wT2
cT3	
cT2	
cT1	

S_2 hat keine der gefragten Eigenschaften, da T1 vor T2 schreibt und nach T2 liest.

1.3 S_3

x	y
wT3	rT3
wT1	
cT1	
wT3	
cT3	
wT2	
cT2	

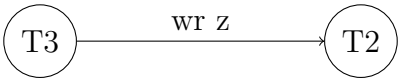
S_3 ist kaskadenlos da kein Lesevorgang nach einem uncommiteten Schreibvorgang stattfindet. Es ist auch rücksetzbar da es kaskadenlos ist. Es ist jedoch nicht fehlerfrei serialisierbar.

2 Konfliktgraph

2.1 S_1

x	y	Z
wT1	rT1	wT3
wT1	rT1	rT3
	rT2	rT2
		rT2
		rT3
		rT3
		rT3

$$C(S_1) = \{rw_z(T3, T2)\}$$

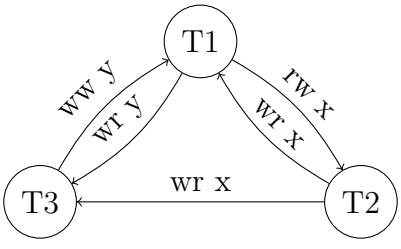


S_1 ist konfliktserialisierbar, da kein Zyklus im Graphen existiert.

2.2 S_2

x	y	Z
rT1	wT3	wT1
rT1	wT3	rT1
wT2	wT1	
rT1	rT1	
rT3	rT3	

$$C(S_2) = \{rw_x(T1, T2), wr_x(T2, T1), wr_x(T2, T3), ww_y(T3, T1), wr_y(T1, T3)\}$$

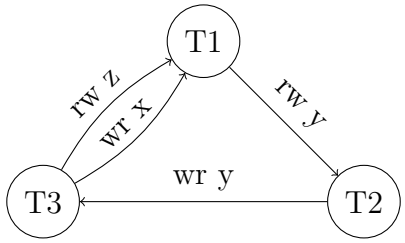


S_2 ist nicht konfliktserialisierbar, da ein Zyklus im Graphen existiert.

2.3 S_3

x	y	Z
wT3	rT1	rT1
rT3	wT2	rT3
wT3	wT2	wT1
rT1	rT3	
	rT2	

$$C(S_3) = \{wr_x(T3, T1), rw_y(T1, T2), wr_y(T2, T3), rw_z(T3, T1)\}$$



S_3 ist nicht konfliktserialisierbar, da ein Zyklus im Graphen existiert.

3 Zeitstempelverfahren

75	85	100	a		b		c		d	
T1	T2	T3	R(a)	W(a)	R(b)	W(b)	R(c)	W(c)	R(d)	W(d)
w(c)							75	75		
	w(a)		85	85			75	75		
	r(d)		85	85			75	75	85	
	r(d)		85	85			75	75	85	
	w(b)		85	85	85	85	75	75	85	
	w(a)		85	85	85	85	75	75	85	
r(d)			85	85	85	85	75	75	85	
		w(d)	85	85	85	85	75	75	85	100
	r(a)		85	85	85	85	75	75	85	100
	w(b)		85	85	85	85	75	75	85	100
		w(c)	85	85	85	85	100	100	85	100
		w(a)	100	100	85	85	100	100	85	100
		r(b)	100	100	85	85	100	100	85	100
w(b)			100	100	85	x	100	100	85	100
	r(b)		100	100	85	x	100	100	85	100

T1 versucht auf b zu schreiben nachdem T2 beteits auf b geschrieben hat. T1 wird deshalb abgebrochen.

4 COMMIT, SAVEPOINT und ROLLBACK