359109, Michelle Milde
356148, Philipp Hochmann
356092, Daniel Schleiz

1	2	3	$\sum$
/7	/7	/6	/20

Korrigiert am:_____

Aufgabe 10.1 (Punkte: /7)

(a)

- $conf(s_1) = \{(w_3(x), r_2(x)), (w_2(y), w_3(y)), (w_3(y), r_2(y)), (w_3(z), w_2(z))\}$
- $conf(s_2) = \{(r_3(x), w_1(x)), (r_2(y), w_3(y)), (r_2(y), w_1(y)), (w_3(y), w_2(y)), (w_3(y), w_1(y)), (w_2(y), w_1(y)), (r_2(z), w_3(z)), (r_2(z), w_1(z)), (r_3(z), w_2(z)), (r_3(z), w_1(z)), (w_3(z), w_2(z)), (w_3(z), w_1(z)), (w_2(z), w_1(z))\}$

(b)

- $commit(s_1) = \{t_2, t_3\}$. Somit besitzt der Konfliktgraph G_1 die Knoten t_2 und t_3. Da $(w_3(x), r_2(x)) \in conf(s_1)$ und $(w_2(y), w_3(y)) \in conf(s_1)$, existiert in G_1 eine Kante von t_2 zu t_3 und umgekehrt. Da somit G_1 einen Kreis besitzt, ist s_1 *nicht konfliktserialisierbar*.
- $commit(s_2) = \{t_1, t_2, t_3\}$. Somit besitzt der Konfliktgraph G_2 die Knoten t_1, t_2 und t_3. Da $(r_2(y), w_3(y)) \in conf(s_2)$ und $(w_3(y), w_2(y)) \in conf(s_2)$, existiert in G_2 eine Kante von t_2 zu t_3 und umgekehrt. Da somit G_2 einen Kreis enthält, ist s_2 *nicht konfliktserialisierbar*.

Aufgabe 10.2 (Punkte: /7)

s_i	RC	ACA	ST
1	X	X	X
2	√	√	√
3	√	√	X
4	√	X	X

(a)

s_1 ist nicht in ACA, da t_2 von t_1 liest, bevor t_1 committed wird.
s_1 ist nicht in ST, da er nicht in ACA ist.
s_1 ist nicht in RC, da t_2 von t_1 liest, t_2 wird aber vor t_2 committed.

(b)

s_2 ist in ACA. Zwar liest t_2 von t_1, aber t_1 wird vorher committed.
s_2 ist in ST, da s_2 in ACA ist und auf kein Objekt zweimal geschrieben wird.
s_2 ist in RC, da er in ACA ist.

(c)

s_3 ist aus dem selben Grund wie S_2 in ACA.
s_3 ist nicht in ST, da t_2 den von t_1 in x geschriebenen Inhalt überschreibt.
s_3 ist in RC, da er in ACA ist.

(d)

s_4 ist nicht in ACA, da t_1 von t_2 liest, bevor t_2 committed wird.
s_4 ist nicht in ST, da er nicht in ACA ist.
s_4 ist in RC, da t_2 vor t_1 committed wird.

Aufgabe 10.3 (Punkte: /6)

(a)

Ausgabe für s_1:
$wl_3(x)wl_2(y)wl_3(z)w_3(z)wu_3(z)w_3(x)wu_3(x)c_3rl_2(x)r_2(x)ru_2(x)w_2(y)wu_2(y)c_2rl_1(y)wl_1(z)$
$w_1(z)wu_1(z)r_1(y)ru_1(y)c_1$

(b)

Ausgabe für s2: Der Scheduler produziert einen Deadlock, weil zuerst eine Schreibsperre des Datenobjektes z fürt_1, eine Lesesperre des Datenobjektes x für t_2 und eine Schreibsperre des Datenobjektes y für t_3 gesetzt wird. Danach möchte t_2 jedoch ebenfalls schreibend auf y zugreifen, t_2 wartet also auf die Freigabe von y durch t_3. t_3 möchte lesend auf z zugreifen, t_3 muss also auf die Freigabe des Datenobjektes z durch t_1 warten. t_1 möchte schreibend auf x zugreifen, was jedoch auch nicht möglich ist, t_1 wartet auf die Freigabe von x durch t_2. Jede der Transaktionen wartet also auf die Freigabe eines bestimmten Datenobjektes durch eine der anderen Transaktionen (gegenseitiges Warten), jedoch kann auch keine der Transaktionen die jeweils erwartete Sperre lösen.

(c)

Ausgabe für s_3:
$rl_3(z)r_3(z)wl_1(y)w_1(y)wu_1(y)c_1rl_3(x)r_3(x)wl_2(y)w_2(y)wl_3(z)w_3(z)wu_3(z)ru_3(z)ru_3(x)$
$c_3wl_2(x)w_2(x)wu_2(x)wu_2(y)c_2$