Informatik 5 - Informationssysteme und Datenbanken
RWTH AACHEN UNIVERSITY

Prof. Dr. Stefan Decker, PD Dr. Ralf Klamma
K. Fidomski, M. Slupczynski, S. Welten

12.07.2021

Datenbanken und Informationssysteme (Sommersemester 2021)

Übung 10

Abgabe bis 19.07.2021 14:00 Uhr.

Zu spät eingereichte Übungen werden nicht berücksichtigt.

Wichtige Hinweise

- Bei Nichtbeachtung dieser Hinweise wird die Abgabe mit 0 Punkten bewertet!
- Bitte reichen Sie Ihre Lösung nur in Dreier- oder Vierergruppen ein.
- Achten Sie auch darauf, dass Ihre Gruppe im Moodle korrekt eingerichtet ist.
- **Bitte laden Sie Ihren schriftlichen Teil der Lösungen ins Moodle als ein zusammenhängendes PDF-Dokument hoch. Benutzen Sie dafür die entsprechend markierte Abgabefunktion.**
- Bitte geben Sie Namen, Matrikelnummern und Moodle-Gruppennummer auf der schriftlichen Lösung an.
- Wird offensichtlich die gleiche Lösung von zwei Gruppen abgegeben, dann erhalten beide Gruppen 0 Punkte.

Die Lösung zu diesem Übungsblatt wird in den Übungen am 19. Juli und 21. Juli 2021 vorgestellt. Bitte beachten Sie auch die aktuellen Ankündigungen im Moodle-Lernraum zur Vorlesung. * bezeichnet Bonusaufgaben.

Nummer der Abgabegruppe: [124]

Gruppenmitglieder: [Andrés Montoya, 405409], [Marc Ludevid, 405401], [Til Mohr, 405959]

Vergessen Sie nicht alle Gruppenmitglieder einzutragen!
Der Bearbeitungsmodus kann mit Doppelklick aktiviert und mit der Tastenkombination **Strg+Enter** beendet werden.

WICHTIG:

Das **gesamte** Übungsblatt ist **schriftlich** zu bearbeiten und wird manuell bewertet.
Die Lösungen der Übungsaufgaben müssen in einem zusammenhängenden .pdf Dokument abgegeben

werden.

Aufgabe 10.1 (Serialisierbarkeit) - Schriftlich

(7 Punkte)

Betrachten Sie die folgenden Schedules s_1, s_2, welche drei Transaktionen t_1, t_2 und t_3 enthalten, die auf den Datenelementen x, y, z arbeiten:

$s_1 = r_1(x)w_2(y)r_1(x)w_3(z)w_3(x)r_1(y)w_1(y)w_2(z)w_1(z)w_3(y)r_2(x)c_3r_2(y)c_2w_1(y)a_1$

$s_2 = r_1(x)w_2(y)r_1(x)w_3(z)w_3(x)r_1(y)w_1(y)w_2(z)w_1(z)w_3(y)c_3r_2(y)c_2w_1(y)c_1$

a) Bestimmen Sie für die Schedules s_1, s_2 die Konfliktmengen $conf(s_1)$, $conf(s_2)$.

$conf(s_1)\coloneqq\{(w_3(x), r_2(x)), (w_2(y), w_3(y)), (w_3(y), r_2(y)), (w_3(z), w_2(z))\}$

$conf(s_2)\coloneqq\{(r_1(x), w_3(x)), (w_2(y), r_1(y)), (w_2(y), w_1(y)), (w_2(y), w_3(y)), (r_1(y), w_3(y)), (w_1(y), u$

b) Entscheiden Sie, ob die Schedules konfliktserialisierbar sind. Begründen Sie Ihre Entscheidung.

Tipp: Konfliktserialisierbarkeit kann mit Konfliktgraphen bestimmt werden.

Falls Sie sich entscheiden sollten, die Aufgabe hier in diesem Notebook zu bearbeiten, können Sie folgendes Beispiel verwenden, um einen Konfliktgraphen zu zeichnen. Stellen Sie sicher, dass Ihre finale Abgabe auch diese visuellen Graphen beinhaltet. Code wird nicht bewertet!

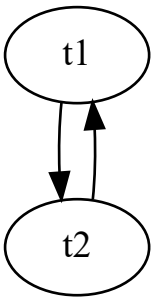
In [1]:

```
#Erstes Graph-Beispiel
from graphviz import Digraph # Importiere die Graph-Lib. Nicht verändern!
g = Digraph('G') # Definiere einen neuen Graphen G und speichere ihn in der Variable g

g.edge('t1', 't2') #Gerichtete Kante von Knoten t1 zu Knoten t2
g.edge('t2','t1') #Gerichtete Kante von Knoten t2 zu Knoten t1


g #Zeichne den Graphen.
```

Out[1]:



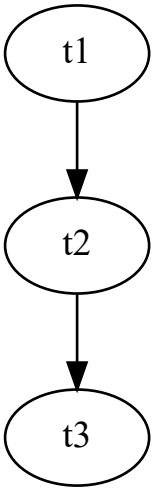
In [2]:

```
#Zweites Graph-Beispiel
from graphviz import Digraph # Importiere die Graph-Lib. Nicht verändern!
f = Digraph('F') # Definiere einen neuen Graphen F und speichere ihn in der Variable f

f.edge('t1', 't2') #Gerichtete Kante von Knoten 1 zu Knoten 2
f.edge('t2','t3') #Gerichtete Kante von Knoten 2 zu Knoten 3


f #Zeichne den Graphen.
```

Out[2]:



s_1, s_2 sind nicht konfliktserialisierbar, da $G(s_1), G(s_2)$ zyklisch sind.

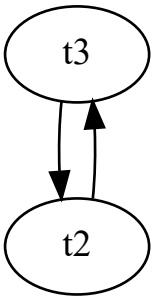
In [8]:

```
# Wenn du Python zum malen der Graphen verwendest, nutze diese Zelle für den Konfliktgr
from graphviz import Digraph
s1 = Digraph('S1')

s1.edge('t3', 't2')
s1.edge('t2', 't3')

s1 #Zeichne den Graphen.
```

Out[8]:



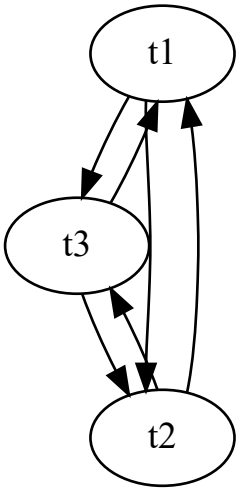
In [14]:

```
# Wenn du Python zum malen der Graphen verwendest, nutze diese Zelle für den Konfliktgr
from graphviz import Digraph
s2 = Digraph('S2')

s2.edge('t1', 't3')
s2.edge('t2', 't1')
s2.edge('t2', 't3')
s2.edge('t1', 't2')
s2.edge('t3', 't2')
s2.edge('t3', 't1')

s2 #Zeichne den Graphen.
```

Out[14]:



Aufgabe 10.2 (Recovery) - Schriftlich

(5+2* Punkte)

In der Vorlesung haben Sie die einzelnen Klassen RC, ACA und ST und die dazugehörigen Definitionen kennengelernt.

- **RC:** Ein Schedule s heißt recoverable, falls gilt: $(\forall t_i, t_j \in trans(s), i \neq j)$: t_i liest von t_j in $s \wedge c_i \in s \Rightarrow c_j <_s c_i$
- **ACA:** Ein Schedule s vermeidet cascading aborts, falls gilt: $(\forall t_i, t_j \in trans(s), i \neq j)$: t_i liest x von $t_j \in s \Rightarrow c_j <_s r_i(x)$
- **ST:** Ein Schedule s heißt strikt, falls gilt: $(\forall t_i \in trans(s))(\forall p_i(x) \in op(t_i), p \in \{r, w\})$: $w_j(x) <_s p_i(x), i \neq j \Rightarrow a_j <_s p_i(x) \vee c_j <_s p_i(x)$

Überprüfen Sie für jeden der folgenden Schedules, ob er jeweils (!) in den Klassen RC, ACA und ST liegt. Begründen Sie Ihre Entscheidung.

a)

$s_1 = w_1(x)r_2(y)r_1(x)r_2(x)c_1w_2(x)c_2$

$s_1 \in RC$, da nur t_2 von t_1 liest (und zwar x), und $c_1 <_{s_1} c_2$.
$s_1 \notin ACA$, da t_2 x aus t_1 liest, bevor t_1 commited.
$s_1 \notin ST$, da $w_1(x) <_{s_1} r_2(x)$, aber $c_1 \not<_{s_1} r_2(x)$.

b)

$s_2 = w_1(x)w_2(y)r_1(y)r_2(x)c_2w_1(y)c_1$

$s_2 \notin RC$, da $r_2(x) <_{s_2} w_1(x)$, aber $c_1 \not<_{s_2} c_2$.
$s_2 \notin ACA$, da t_1 y von t_2 liest, bevor t_2 commited.
$s_2 \notin ST$, da $r_2(x)$ auf $w_1(x)$ folgt, ohne dass vorher $c_1(x)$ auftritt.

c)

$s_3 = w_1(x)r_2(y)c_1w_2(x)r_2(x)w_2(y)c_2$

$s_3 \in RC$, da t_1 zwischen Schreiben ($w_1(x)$) und Commiten t_2 nur y liest.
$s_3 \in ACA$, da nur t_2 y liest, bevor t_1 commited, aber t_1 nie auf y schreibt.
$s_3 \in ST$, da nur t_2 y liest, bevor t_1 commited, aber t_1 nie auf y schreibt.

d) [2* Punkte]

$s_4 = w_2(x)r_1(y)r_2(x)r_1(x)c_2w_1(y)c_1$
$s_4 \in RC$, da nur t_1 von t_2 liest (x), und $c_2 <_{s_4} c_1$.
$s_4 \notin ACA$, da $c_2 \not<_{s_4} r_1(x)$.
$s_4 \notin ST$, da $c_2 \not<_{s_4} r_1(x)$.

[Hier Lösung eingeben]

Aufgabe 10.3 (Scheduler) - Schriftlich

(6 Punkte)

In der Vorlesung haben Sie die einzelnen Scheduler und die dazugehörigen Definitionen kennengelernt.

- **2PL:** Alle Sperren halten bis alle Sperren aufgebaut sind (2-Phase).
- **C2PL:** Bevor eine Transaktion ihre Operationen beginnen kann, muss sie alle Sperren halten, die sie für die Transaktion benötigt.
- **S2PL:** Alle Schreibsperren bis nach letzter r/w Operation halten.
- **SS2PL:** Alle Sperren bis nach letzter r/w Operation halten.

In dieser Aufgabe sollen die Ausgaben von C2PL- S2PL- und SS2PL-Schedulern für eine gegebene Transaktion bestimmt werden. Geben Sie bei den Ausgaben der Scheduler das Setzen und Freigeben von Sperren mit an. Zusätzlich zu den C2PL, S2PL und SS2PL Sperrprotokollen, sollen die Scheduler Sperren erst anfordern, wenn sie benötigt werden, Sperren wieder freigeben, sobald dies möglich ist und Operationen nicht weiter verzögern als notwendig. Sollte ein Scheduler die Wahl haben, Sperren freizugeben oder weitere Operationen auszuführen, so soll er das Freigeben der Sperren vorziehen. **Falls der Scheduler einen Deadlock produziert, geben Sie dies an und begründen Sie.**

> Hinweis: Ein Deadlock ist zum Beispiel gegeben, wenn die Transaktion t_1 auf einer Sperre, die von t_2 gehalten wird, wartet und gleichzeitig die Transaktion t_2 auf eine Sperre, die von t_1 gehalten wird, wartet, wobei keine dieser Transaktionen ihre Sperren lösen kann.

a) Gegeben sei der Schedule s_1:

$s_1 = w_1(z)\ w_3(y)\ r_1(x)\ r_3(z)\ r_2(y)\ w_2(z)\ w_3(x)\ c_1\ c_2\ c_3$

Bestimmen Sie die Ausgabe eines C2PL-Schedulers für die Eingabe s_1.

Nebenrechnung:

- $wl_1(z)rl_1(x)w_1(z)wu_1(z)r_1(x)ru_1(x)c_1$
- $wl_3(y)rl_3(z)wl_3(x)w_3(y)wu_3(y)r_3(z)ru_3(z)w_3(x)wu_3(x)c_3$
- $rl_2(y)wl_3(x)r_2(y)ru_2(x)w_2(z)wu_2(z)c_2$

Ergebnis: $s_1 = w_1(z)r_1(x)w_3(y)r_3(z)w_3(x)r_2(y)w_2(z)c_1c_2c_3$

b) Gegeben sei der Schedule s_2:

$s_2 = r_1(x)\ w_3(z)\ r_2(x)\ r_1(z)\ r_3(y)\ w_1(z)\ w_2(y)\ c_1\ c_2\ c_3$

Bestimmen Sie die Ausgabe eines S2PL-Schedulers für die Eingabe s_2.

Nebenrechnung:

- $rl_1(x)r_1(x)rl_1(z)r_1(z)wl_1(z)ru_1(x)w_1(z)ru_1(z)wu_1(z)c_1$
- $wl_3(z)w_3(z)rl_3(y)wu_3(z)r_3(y)ru_3(y)c_3$

- $rl_2(x) r_2(x) wl_2(y) ru_2(x) w_2(y) wu_2(y) c_2$

Ergebnis: $s_2 = r_2(x) w_2(y) r_2(y) w_2(x) c_2 c_2$

c) Gegeben sei der Schedule s_3:

$s_3 = r_2(y)\ w_3(x)\ w_1(z)\ w_3(y)\ r_1(x)\ r_2(z)\ r_3(z)\ c_1\ c_2\ c_3$

Bestimmen Sie die Ausgabe eines SS2PL-Schedulers für die Eingabe s_3.

Nebenrechnung:

- $rl_2(y) rl_2(z) r_2(y) r_2(z) ru_2(y) ru_2(z) c_2$
- $wl_3(x) wl_3(y) rl_3(z) w_3(x) w_3(y) r_3(z) wu_3(x) wu_3(y) ru_3(z) c_3$
- $wl_1(z) rl_1(x) w_1(z) r_1(x) wu_1(z) ru_1(z) c_1$

Ergebnis: $s_3 = r_2(y) r_2(z) w_3(x) w_3(y) r_3(z) w_1(z) r_1(x) c_1 c_2 c_3$