Datenbanksysteme WS 2016/17
Prof. Dr.-Ing. Sebastian Michel
BSc. Peter Brucker
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Übungsblatt 10: Ausgabe 17.01.2017, Abgabe 23.01.2017

Aufgabe 1: Serialisierbarkeit und Klassen von Schedules (1 P.)

(i) In welche Klasse, FSR, VSR, CSR, fällt der folgende Schedule?

$$s_1 := r_1(x)\ r_3(x)\ w_3(y)\ w_2(x)\ r_4(y)\ c_2\ w_4(x)\ c_4\ r_5(x)\ c_3\ w_5(z)\ c_5\ w_1(z)\ c_1$$

(ii) Zeigen Sie, dass folgender Schedule in $VSR - CSR$ liegt:

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

(iii) Gegeben der folgende Schedule

$$s_3 := r_3(z)\ r_3(y)\ r_4(y)\ r_2(x)\ r_4(x)\ w_3(y)\ w_2(x)\ r_1(x)\ w_1(x)\ r_1(y)\ c_1\ r_5(z)\ r_2(y)\ c_2\ w_5(z)\ c_5\ c_3\ c_4$$

Erstellen Sie den Konfliktgraphen von s_3 und begründen Sie, so ob $s_3 \in CSR$ gilt. Falls ja, ordnen Sie die Operationen entsprechend der Kommutativitätsregeln um. Gilt auch $s_3 \in OCSR$?

(iv) Bestimmen Sie für jeden der drei folgenden Schedules in welche der Klassen RC, ACA oder ST er fällt:

$$\begin{aligned} s_4 &:= w_1(x)\ r_2(y)\ r_1(x)\ c_1\ r_2(x)\ w_2(y)\ c_2 \\ s_5 &:= w_1(x)\ r_2(y)\ r_1(x)\ r_2(x)\ c_1\ w_2(y)\ c_2 \\ s_6 &:= w_1(x)\ r_2(y)\ r_2(x)\ r_1(x)\ c_2\ w_1(y)\ c_1 \end{aligned}$$

(v) In der Vorlesung wurde erwähnt, dass VSR nicht monoton ist. Geben Sie ein Beispiel an, das dies zeigt.

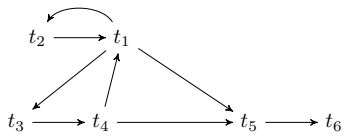
Aufgabe 2: Sperrbasierte Mehrbenutzersynchronisation und Deadlocksvermeidung (1 P.)

(i) Gegeben folgende Eingabeschedules:

$$\begin{aligned} s_1 &= w_1(x)\ r_2(y)\ r_1(x)\ c_1\ r_2(x)\ w_2(y)\ c_2 \\ s_2 &= r_1(x)\ r_2(x)\ w_3(x)\ w_4(x)\ w_1(x)\ c_1\ w_2(x)\ c_2\ c_3\ c_4 \end{aligned}$$

Geben Sie an, welcher Ausgabeschedule jeweils für 2PL produziert wird und beschreiben Sie parallel den Waits-for-Graph (WfG). Sobald ein Deadlock gefunden wird, wenden Sie die Strategie Immediate Restart an.

(ii) Geben Sie für folgenden WfG für und die Strategien Meiste Zyklen und Meiste Kanten an, welche Kanten ausgewählt werden.



(iii) Zeigen Sie, dass die wait-die und wound-wait Ansätze zur Deadlock-Vermeidung zu jeder Zeit einen azyklischen WfG garantieren.

Aufgabe 3: Zeitstempelverfahren (1 P.)

(i) Gegeben folgende Schedules:

$$s_1 = r_1(x)\ r_2(x)\ w_1(x)\ w_2(x)$$
$$s_2 = r_1(x)\ r_1(y)\ w_2(y)\ w_1(x)\ w_1(y)\ r_1(y)$$

Beschreiben Sie jeweils, wie ein Timestamp-Ordering (TO) basierter Scheduler die angeforderten Operationen ausführt. Benutzen Sie dazu eine Tabelle wie die folgende. Tragen Sie unter "Anmerkung" ein, welche Regel angewendet wurde. Die max-r bzw. max-w Spalten beinhalten den Wert von max-q-scheduled für das jeweilige Objekt, eine Zeile enthält jeweils den Eintrag nachdem die Regel angewendet wurde. Nehmen Sie an, dass $ts(t_1) < ts(t_2) < ts(t_3)$. Lassen Sie abgebrochene Transaktionen nicht nochmal anlaufen.

Operation	max-r(x)	max-w(x)	max-r(y)	max-w(y)	Anmerkung
BOT_1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$ts(t_1) = 0$

(ii) Beschreiben Sie warum gilt $Gen(TO) \subseteq CSR$

(iii) Beschreiben Sie die Thomas' Write Rule (TWR). Sind Schedules, die durch einen Scheduler mit der TWR erzeugt werden konfliktserialisierbar?

(iv) Eine vereinfachte Version des TO Schedulers benutzt nur einen Timestamp pro Objekt. Wenden Sie diese Version ebenfalls auf die obigen Schedules s_1 und s_2 an.