Grundlagen Datenbanken: Übung 14

Gruppe 20 & 21

Tanmay Deshpande

ge94vem@mytum.de

QR-Code für die Folien

Wiederholung

Woche 14

Mehrbenutzerbetrieb

- Es ist ineffizient, eine Transaktion nach der anderen durchzuführen (serielle Historie). Wir wollen Transaktionen parallel durchführen können
- Bei paralleler Durchführung kann es zu Problemen kommen, z.B. :

Lost Update

Schritt	T_1	T_2
1.	read(A,a_1)	
2.	$a_1 := a_1 - 300$	
3.		read(A,a_2)
4.		$a_2 := a_2 * 1.03$
5.		write(A,a_2)
6.	write(A,a_1)	
7.	read(B,b_1)	
8.	$b_1 := b_1 + 300$	
9.	write(B,b_1)	

Dirty Read

Schritt	T_1	T_2
1.	read(A,a_1)	
2.	$a_1 := a_1 - 300$	
3.	write(A,a_1)	
4.		read(A,a_2)
5.		$a_2 := a_2 * 1.03$
6.		write(A,a_2)
7.	read(B,b_1)	
8.	. . .	
9.	abort	

Mehrbenutzerbetrieb

- Es ist ineffizient, eine Transaktion nach der anderen durchzuführen (serielle Historie). Wir wollen Transaktionen parallel durchführen können
- Bei paralleler Durchführung kann es zu Problemen kommen, z.B. :

Phantom Problem

T_1	T_2
	select sum(KontoStand)
	from Konten
insert into Konten	
values (*C*,1000,...)	
	select sum(Kontostand)
	from Konten

Historie

- Historie: Legt die Reihenfolge von Operationen einer Menge von Transaktionen fest

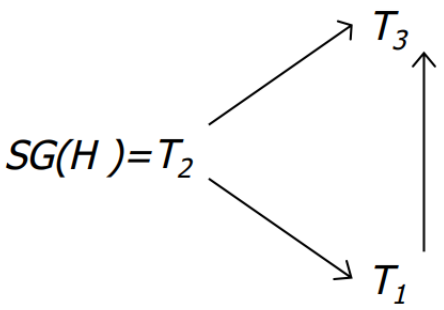
$$r_1(A) \rightarrow r_2(C) \rightarrow w_1(A) \rightarrow w_2(C) \rightarrow r_1(B) \rightarrow w_1(B) \rightarrow c_1 \rightarrow r_2(A) \rightarrow w_2(A) \rightarrow c_2$$

- Operationen einer Historie:
  $r_x(A)$ = read, $w_x(A)$ = write, c_x = commit ,a_x = abort
- Idee: serielle Historien sind sicher. Bringe die verzahnte Historie in einer Form, die äquivalent zu einer seriellen Historie ist
- Zwei Historien sind (konflikt-)äquivalent ($H_1 = H_2$), wenn die Konfliktoperationen der nicht abgebrochenen Transaktionen in der gleichen Reihenfolge angeordnet sind + sie die gleiche Menge von Transaktionen (inklusiv Operationen) haben
- Konfliktoperationen:
  Wenn zwei Transkationen die selben Daten lesen – Reihenfolge beliebig
  Wenn mindestens eine Transaktion die Daten verändert – Reihenfolge festlegen

	$r_j(A)$	$w_j(A)$
$r_i(A)$	-	×
$w_i(A)$	×	×

Serialisierbarkeit

- Eine Historie ist serialisierbar, wenn sie äquivalent zu einer seriellen Historie ist
- Das kann man mithilfe von einem Serialisierbarkeitsgraphen erkennen
- Falls der Serialisierbarkeitsgraph einer Historie azyklisch ist, ist sie serialisierbar



- Im SG(H), gibt es eine Kante von T_A nach T_B, falls es Konfliktoperationen o_A und o_B zwischen T_A und T_B gibt mit $o_A <_H o_B$
- Abgebrochene Transaktionen kann man von dem Serialisierbarkeitsgraphen weglassen

Eigenschaften von Historien

- Wir sagen, dass eine Transaktion von einer anderen Transaktion liest, wenn die folgenden Bedingungen gelten:
  - $w_j(A) <_H r_i(A)$
  - $a_j \neg <_H r_i(A)$
  - Falls $\exists\ w_k(A)$ mit $w_j(A) <_H w_k(A) <_H r_i(A)$, dann $a_k <_H r_i(A)$
- **Serialisierbar (SR):** SG(H) ist azyklisch
- **Rücksetzbar (RC):** Wenn die schreibende Transaktion (T_j) vor der lesenden Transaktion (T_i) ihr commit durchführt, also $c_j <_H c_i$

Eigenschaften von Historien

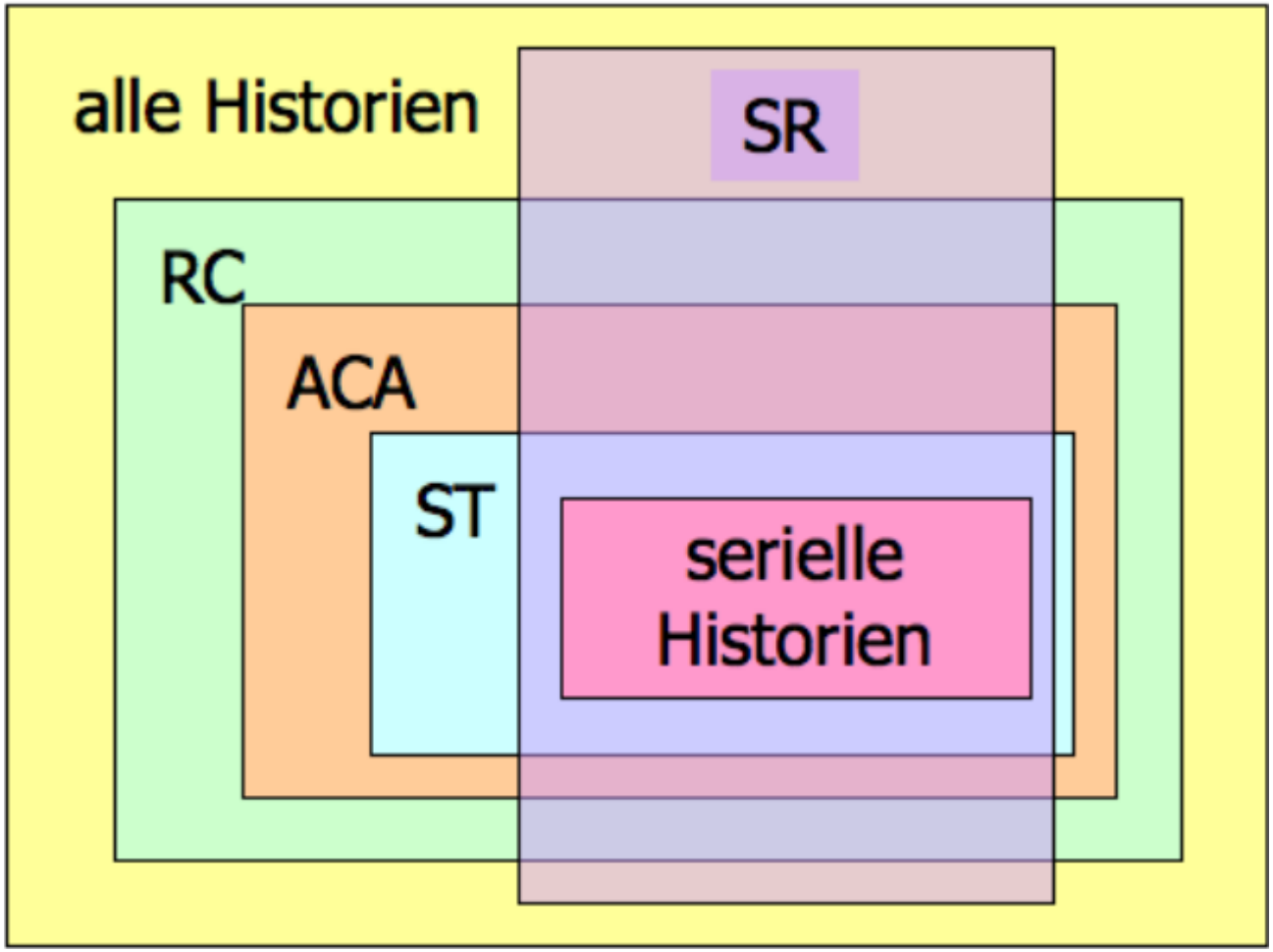
- **Ohne kaskadierendes Rücksetzen (ACA):** Für je zwei Transaktionen T_i und T_j sollte gelten, $c_j <_H r_i(A)$ falls T_i ein Datum A von T_j liest

Beispiel-Historie mit kaskadierendem Rücksetzen

Schritt	T_1	T_2	T_3	T_4	T_5
0.	. . .				
1.	$w_1(A)$				
2.		$r_2(A)$			
3.		$w_2(B)$			
4.			$r_3(B)$		
5.			$w_3(C)$		
6.				$r_4(C)$	
7.				$w_4(D)$	
8.					$r_5(D)$
9.	a_1(**abort**)				

- **Strikt (ST):** Wenn $w_j(A) <_H o_i(A)$, dann $a_j <_H o_i(A)$ oder $c_j <_H o_i(A)$

Eigenschaften von Historien



Sperrbasierte Synchronisation

Zwei Sperrmodi

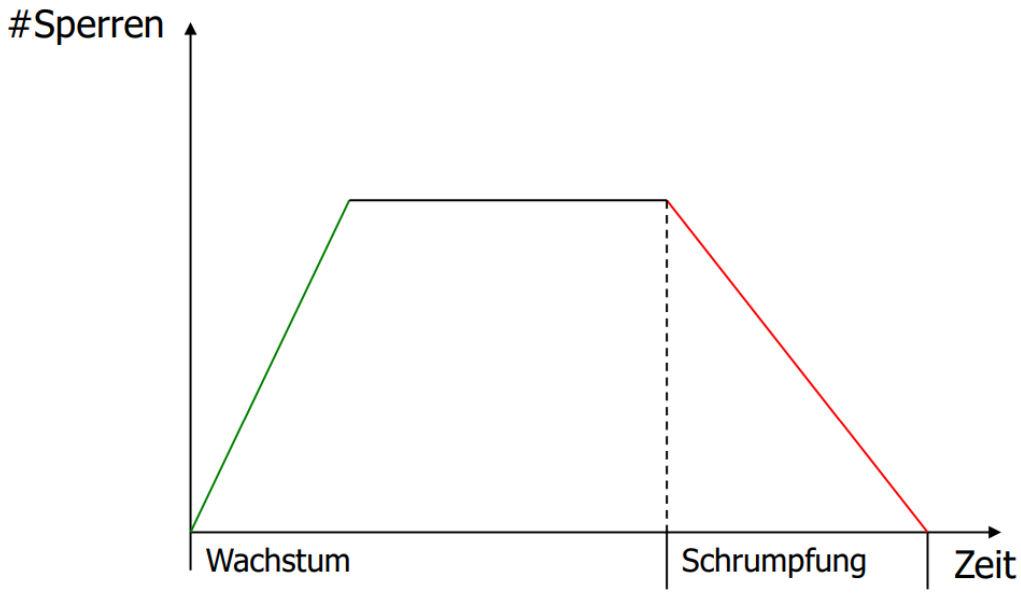
- *S* (shared, read lock, Lesesperre):
- *X* (exclusive, write lock, Schreibsperre):

- *Verträglichkeitsmatrix* (auch *Kompatibilitätsmatrix* genannt)

	NL	*S*	*X*
S	✓	✓	-
X	✓	-	-

Zwei-Phasen Sperrprotokoll (2PL)

- Jedes Objekt, das von einer Transaktion genutzt wird, muss gesperrt werden
- Wachstumsphase - Sperren anfordern aber keine freigeben
- Falls eine Transaktion versucht, auf ein Objekt zuzugreifen, das schon gesperrt ist, geht es in einer Warteschlange (abhängig von Verträglichkeitsmatrix) bis diese Sperre freigegeben wird
- Schrumpfungsphase - Sperren freigeben aber keine anfordern
- Bis zum Ende der Transaktion müssen alle Sperren freigegeben werden

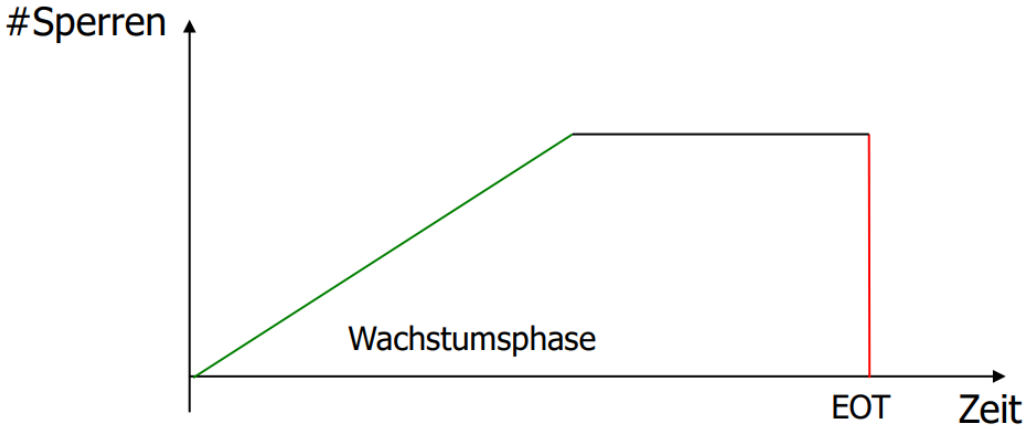


Beispiel 2PL

Schritt	T_1	T_2	Bemerkung
1.	**BOT**		
2.	**lockX**(*A*)		
3.	read(*A*)		
4.	write(*A*)		
5.		**BOT**	
6.		**lockS**(*A*)	T_2 muss warten
7.	**lockX**(*B*)		
8.	read(*B*)		
9.	**unlockX**(*A*)		T_2 wecken
10.		read(*A*)	
11.		**lockS**(*B*)	T_2 muss warten
12.	write(*B*)		
13.	**unlockX**(*B*)		T_2 wecken
14.		read(*B*)	
15.	**commit**		
16.		**unlockS**(*A*)	
17.		**unlockS**(*B*)	
18.		**commit**	

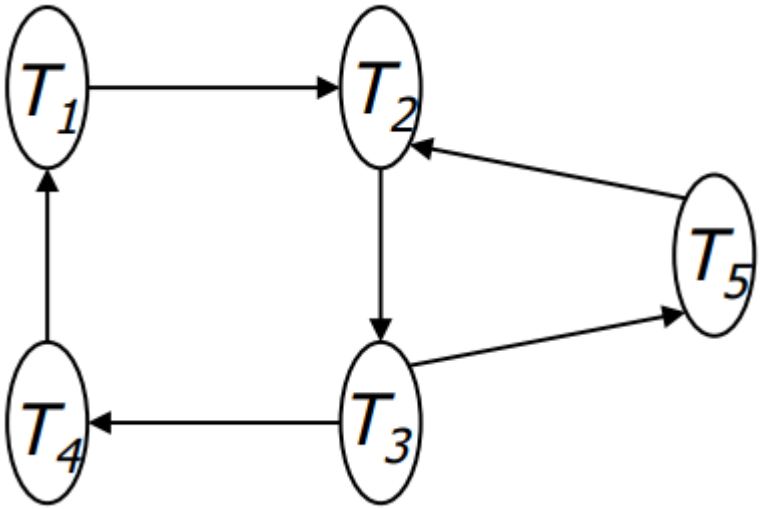
Strenges 2PL

- Normales 2PL schließt kaskadierendes Rücksetzen nicht aus
- Bei strengem 2PL werden Sperren bis EOT gehalten und gemeinsam freigegeben (keine Schrumpfungsphase)



Verklemmungen

- Entstehen wenn zwei Transaktionen auf Sperren warten, die die jeweils andere besitzt
- Wir können sie durch Wartegraphen erkennen



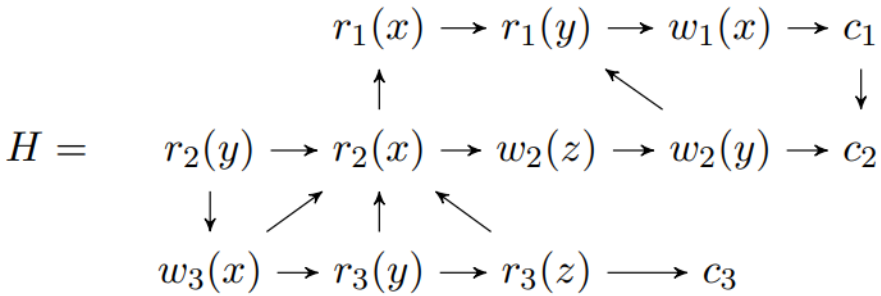
- Kante von T_A nach T_B bedeutet, T_A wartet auf Sperre von T_B
- Zyklen in Wartegraphen bedeuten, dass Verklemmungen existieren

Aufgaben

Woche 14

Aufgabe 01 a-c

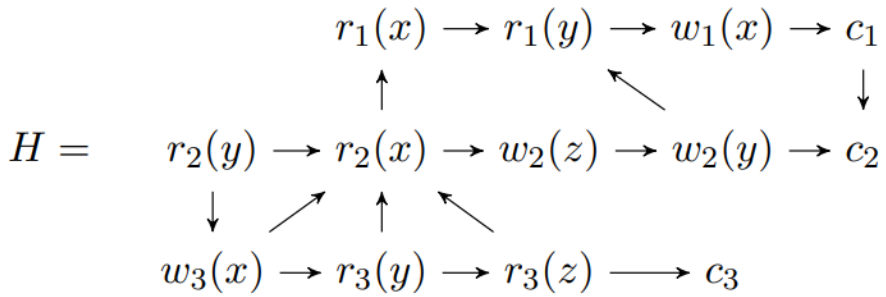
- Die Historie H für die Transaktionen T_1, T_2 und T_3 sei durch das folgende Diagramm gegeben:



- A) Geben Sie die Konfliktoperationen von H an
- B) Geben Sie eine total geordnete Historie H‘ an (also eine „lineare“ Abfolge von Operationen), die konfliktäquivalent zu H ist.
- C) Geben Sie an, welche Transaktionen voneinander lesen.

Lösungsvorschlag 01 a-c

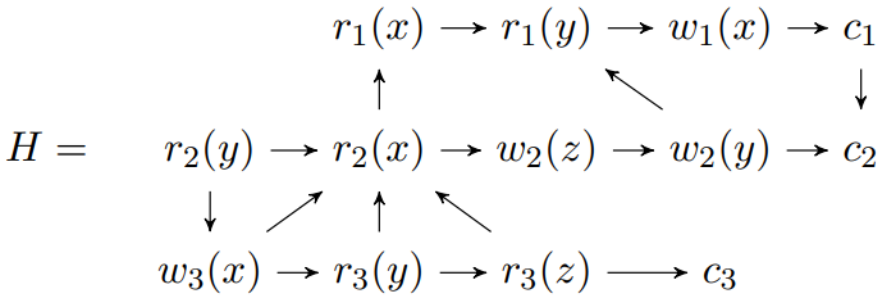
- Die Historie H für die Transaktionen T_1, T_2 und T_3 sei durch das folgende Diagramm gegeben:



- A) $r_2(x), w_1(x)$
  $w_3(x), r_2(x)$
  $w_3(x), r_1(x)$
  $w_3(x), w_1(x)$
  $w_2(y), r_1(y)$
  $r_3(y), w_2(y)$
  $r_3(z), w_2(z)$
- B) $r_2(y), w_3(x), r_3(y), r_3(z), c_3, r_2(x), r_1(x), w_2(z), w_2(y), r_1(y), w_1(x), c_1, c_2$
- C) T1 liest von T3 und T2, T2 liest von T3.

Aufgabe 01 d-e

- Die Historie H für die Transaktionen T_1, T_2 und T_3 sei durch das folgende Diagramm gegeben:



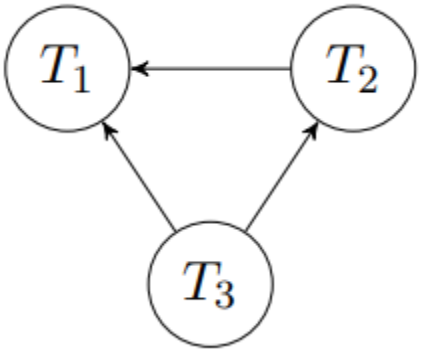
- D) Geben Sie den Serialisierbarkeitsgraphen von H an.
- E) Geben Sie eine serielle Historie H‘ an, die konfliktäquivalent zu H ist.

Lösungsvorschlag 01 d-e

- Die Historie H für die Transaktionen T_1, T_2 und T_3 sei durch das folgende Diagramm gegeben:

$$H = \begin{array}{ccccccccc} & & & & r_1(x) & \rightarrow & r_1(y) & \rightarrow & w_1(x) & \rightarrow & c_1 \\ & & & & \uparrow & & \nwarrow & & & & \downarrow \\ r_2(y) & \rightarrow & r_2(x) & \rightarrow & w_2(z) & \rightarrow & w_2(y) & \rightarrow & c_2 & & \\ \downarrow & \nearrow & \uparrow & \nwarrow & & & & & & & \\ w_3(x) & \rightarrow & r_3(y) & \rightarrow & r_3(z) & \longrightarrow & c_3 & & & & \end{array}$$

- D)



- E)

H'' muss die Transaktionen in der Reihenfolge T_3, T_2, T_1 enthalten:

$$H'' = T_3|T_2|T_1 = w_3(x), r_3(y), r_3(z), c_3, r_2(y), r_2(x), w_2(z), w_2(y), c_2, r_1(x), r_1(y), w_1(x), c_1$$

Aufgabe 02a

a) $H_1 =$

Schritt	T_1	T_2	T_3
1	$w(x)$		
2		$r(x)$	
3		$w(y)$	
4		c	
5			$r(y)$
6			$w(z)$
7			c
8	c		

richtig	**falsch**	**Aussage**
		Die Historie ist serialisierbar (SR)
		Die Historie ist rücksetzbar (RC)
		Die Historie ist vermeidet kaskadierendes Rücksetzen (ACA)
		Die Historie ist strikt (ST)

Lösungsvorschlag 02a

a) $H_1 =$

Schritt	T_1	T_2	T_3
1	$w(x)$		
2		$r(x)$	
3		$w(y)$	
4		c	
5			$r(y)$
6			$w(z)$
7			c
8	c		

richtig	**falsch**	**Aussage**
✔		Die Historie ist serialisierbar (SR)
	✔	Die Historie ist rücksetzbar (RC)
	✔	Die Historie ist vermeidet kaskadierendes Rücksetzen (ACA)
	✔	Die Historie ist strikt (ST)

Aufgabe 02b

b) $H_2 =$

Schritt	T_1	T_2
1	$w(x)$	
2	$w(z)$	
3		$w(z)$
4	c	
5		$w(x)$
6		c

richtig	**falsch**	**Aussage**
		Die Historie ist serialisierbar (SR)
		Die Historie ist rücksetzbar (RC)
		Die Historie ist vermeidet kaskadierendes Rücksetzen (ACA)
		Die Historie ist strikt (ST)

Lösungsvorschlag 02b

b) $H_2 =$

Schritt	T_1	T_2
1	$w(x)$	
2	$w(z)$	
3		$w(z)$
4	c	
5		$w(x)$
6		c

richtig	**falsch**	**Aussage**
✔		Die Historie ist serialisierbar (SR)
✔		Die Historie ist rücksetzbar (RC)
✔		Die Historie ist vermeidet kaskadierendes Rücksetzen (ACA)
	✔	Die Historie ist strikt (ST)

Aufgabe 3

In der Vorlesung haben Sie Serialisierbarkeitsgraphen und den Wartegraphen des (strikten) 2PL kennen gelernt.

a) Was bedeutet eine Kante $T_1 \rightarrow T_2$ im Serialisierbarkeitsgraphen einer Historie H?

b) Gehen Sie davon aus, dass die Datenbank die 2PL-Strategie verwendet. Was bedeutet eine Kante $T_1 \rightarrow T_2$ in einem Wartegraphen? Worin besteht der Unterschied zu Aufgabe a)?

c) Was bedeutet ein Kreis im Serialisierbarkeitsgraphen einer Historie H? Was im Wartegraphen? Wo liegt der Unterschied?

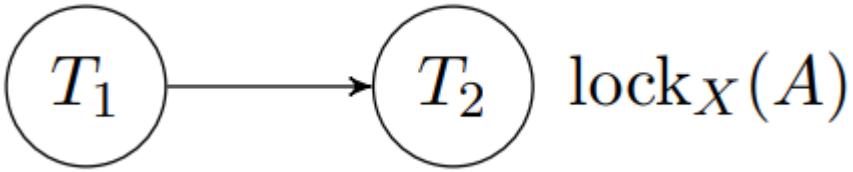
d) Wie viele neue Kanten werden dem Wartegraphen maximal hinzugefügt, wenn eine Transaktion eine S-Sperre anfordert? Wie viele bei einer X-Sperre?

Lösungsvorschlag 3a-c

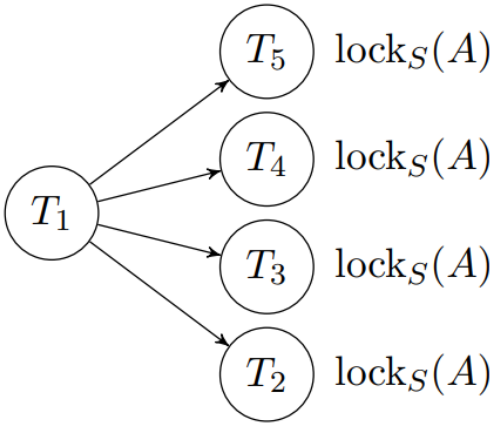
- A) In jeder zu H äquivalenten seriellen Historie wird T1 vor T2 ausgeführt, da es in der Historie H Konfliktoperationen zwischen T1 und T2 gibt, für die H die Reihenfolge T1 vor T2 festlegt.
- B) Die Transaktion T1 fordert eine Sperre auf mindestens ein Datenobjekt an, auf welches T2 bereits eine Sperre hat und muss daher warten. Die Kante verläuft hier also anders herum als in Teilaufgabe a), da sie als warten auf und nicht geschieht vor definiert ist.
- C)
  - Ein Kreis im Serialisierbarkeitsgraphen bedeutet, dass H nicht serialisierbar ist.
  - Ein Kreis im Wartegraphen hingegen bedeutet, dass es unter der strikten 2PL-Stratgie zu einem Deadlock gekommen ist.
  - Es bedeutet nicht, dass es keine äquivalente serielle Historie gibt. Es kann also sein, dass der Scheduler sie lediglich nicht gefunden hat.
  - Striktes 2PL kann also eigentlich serialisierbare Historien ablehnen. Es garantiert aber, dass es bei allen nicht serialisierbaren Historien irgendwann während der Ausführung zum Deadlock kommt.

Lösungsvorschlag 3d

- 1 für eine S-Sperre



- N-1 für eine X-Sperre



Aufgabe 4

a) Erläutern Sie kurz die zwei Phasen des 2PL-Protokolls.

b) Inwiefern unterscheidet sich das *strenge* 2PL?

c) Welche Eigenschaften (SR,RC,ACA,ST) haben Historien, welche vom 2PL und vom strengen 2PL zugelassen werden?

d) Wäre es beim strengen 2PL-Protokoll ausreichend, alle Schreibsperren bis zum EOT (Transaktionsende) zu halten, aber Lesesperren schon früher wieder freizugeben?

Lösungsvorschlag 4

- A) Jede Transaktion durchläuft zwei Phasen:
  - Eine Wachstumsphase, in der sie Sperren anfordern, aber keine freigeben darf und
  - eine Schrumpfungsphase, in der sie Sperren freigibt, jedoch keine neuen Sperren anfordern darf
- B) Alle Sperren werden bis zum Ende der Transaktion gehalten und gemeinsam freigegeben. Die Schrumpfungsphase entfällt somit.
- C) 2PL garantiert Historien aus SR. Das strenge 2PL garantiert Historien aus SR ∩ ST.
- D) Es ist ausreichend, beim strengen 2PL-Protokoll nur die Schreibsperren bis zum Ende der Transaktion zu halten. Lesesperren können analog zum normalen 2PL-Protokoll in der Schrumpfungsphase (nach wie vor jedoch nicht in der Wachstumsphase) peu à peu freigegeben werden. Die generierten Schedules bleiben serialisierbar und strikt.

  Begründung

  - Schon das normale 2PL bietet Serialisierbarkeit; diese ist also auch hier gegeben.
  - Das Halten der Schreibsperren bis zum Ende der Transaktion stellt sicher, dass keine Transaktion von einer anderen lesen oder einen von ihr modifizierten Wert überschreiben kann, bevor diese nicht ihr **commit** durchgeführt hat.

    Es gilt:

$$\forall T_i : \forall T_j : (i \neq j)\ \forall A : (w_i(A) <_H r_j(A)) \vee (w_i(A) <_H w_j(A)) \Rightarrow (c_i <_H r_j(A)) \text{ bzw. } (c_i <_H w_j(A))$$

Aufgabe 5

Bei der sperrbasierten Synchronisation hat jedes Datenobjekt eine zugehörige Sperre. Bevor eine Transaktion zugreifen darf, muss sie eine Sperre anfordern. Dabei unterscheiden wir zwei Sperrmodi: Lese- und Schreibsperre.

a) Erläutern Sie kurz die Unterschiede.

b) Geben Sie deren Verträglichkeiten an (wenn mehrere Transaktionen Sperren auf dem selben Datenobjekt anfordern).

Lösungsvorschlag 5

a) Eine Lesesperre (auch S-Sperre, shared lock) für ein Datum wird angefordert bevor eine Transaktion das Datum lesen möchte. Ein Schreibvorgang erfordert eine entsprechende Schreibsperre (auch X-Sperre, exclusive lock).

   Mehrere Transaktionen können gleichzeitig eine S-Sperre auf dem selben Datenobjekt besitzen, wohingegen maximal eine Transaktion eine X-Sperre für ein Datum besitzen kann.

b) Verträglichkeitsmatrix (auch Kompatibilitätsmatrix):

	gehaltene Sperre		
angeforderte Sperre	keine	S	X
S	$\surd$	$\surd$	–
X	$\surd$	–	–