Lehrstuhl für Praktische Informatik III
Prof. Dr. Guido Moerkotte
Email: moer@db.informatik.uni-mannheim.de

Datenbanksysteme 1 — 11. Übungsblatt
Herbst-/Wintersemester 2016

Aufgabe 1

Welche Eigenschaften haben die folgenden Historien jeweils (seriell, serialisierbar, rücksetzbar, vermeidet kaskadierendes Rücksetzen, strikt)? Geben Sie *alle* zutreffenden Eigenschaften an.

Aufgabe 1 a)

$w_1[x]w_1[y]c1r_2[x]w_2[z]c_2r_3[y]a_3$

Lösung

- seriell
- serialisierbar
- rücksetzbar
- ACA
- strikt

Anmerkung zu Serialisierbarkeit: Genau dann wenn der Serialisierbarkeitsgraph azyklisch ist, dann ist die Historie serialisierbar. Der Serialisierbarkeitsgraph wird nur aus den abgeschlossenen (committed) Transaktionen der Historie konstruiert. (Im Datenbanksysteme Buch Kapitel 11.3.5 Kriterien für Serialisierbarkeit.)

Aufgabe 1 b)

$w_1[x]w_2[x]w_2[y]w_1[y]w_1[z]c_1c_2$

Lösung

- rücksetzbar
- ACA

Aufgabe 1 c)

$w_1[x]w_2[z]r_2[y]r_1[y]c_1w_2[x]r_2[z]c_2$

Lösung

- serialisierbar
- rücksetzbar
- ACA
- strikt

Aufgabe 1 d)

$r_1[x]w_1[x]r_2[x]w_2[y]r_1[y]w_1[z]c_1c_2r_3[z]w_3[x]c_3$

Lösung

- keine der obigen Eigenschaften

Aufgabe 1 e)

$r_1[x]w_2[x]w_1[y]c_1r_3[y]c_3w_4[y]c_2a_4$

Lösung

- serialisierbar
- rücksetzbar
- ACA
- strikt

Aufgabe 1 f)

$r_1[x]r_2[z]w_2[x]w_2[z]c_2w_1[z]c_1$

Lösung

- rücksetzbar
- ACA
- strikt

Aufgabe 1 g)

$r_1[x]w_1[y]r_2[y]w_1[z]w_2[z]c_1w_2[x]c_2$

Lösung

- serialisierbar
- rücksetzbar

Aufgabe 1 h)

$w_3[z]r_1[z]w_1[y]w_1[x]w_2[x]r_3[y]c_1w_3[y]c_2a_3$

Lösung

- serialisierbar

Aufgabe 2

Abbildung 1 zeigt die Beziehungen von Historienklassen zueinander. Zeigen Sie, dass alle Mengen in diesem Diagramm nicht leer sind, d.h. geben Sie Beispiele für die Historien H_1 bis H_9 an.

Lösung

- H1: $w_1[x]r_2[x]w_2[y]r_1[y]c_1c_2$
- H2: $w_1[x]r_2[x]w_2[y]w_1[y]c_1c_2$
- H3: $w_1[x]w_2[y]w_1[y]c_1r_2[x]c_2$
- H4: $r_2[x]w_1[x]c_1w_2[x]c_2$
- H5: $r_1[x]w_1[x]c_1r_2[x]w_2[x]c2$
- H6: $w_1[x]w_2[z]w_1[y]c_1w_2[y]r_2[x]c_2$
- H7: $w_1[x]w_2[y]w_1[y]c_1c_2$
- H8: $w_1[x]r_2[x]c_1c_2$
- H9: $w_1[x]r_2[x]c_2c_1$

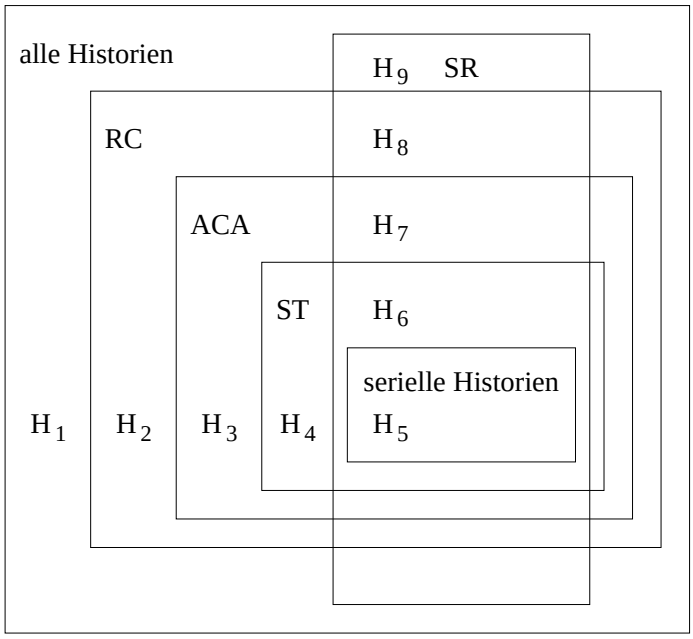


Abbildung 1: Beziehungen von Historienklassen zueinander

Aufgabe 3

Aufgabe 3 a)

Welche Eigenschaften garantiert die Einhaltung des Zwei-Phasen-Sperrprotokolls (2PL) durch den Scheduler für die von ihm erzeugten Historien? Kann insbesondere die Rücksetzbarkeit der erzeugten Historien gewährleistet werden? Begründen Sie.

Lösung

Die von einem 2PL-Scheduler erzeugten Historien sind serialisierbar. Die Rücksetzbarkeit der erzeugten Historien ist nicht gewährleistet (und damit sind die Historien insbesondere auch nicht ACA, strikt oder seriell). Für die Rücksetzbarkeit muss gewährleistet sein, dass wenn eine Transaktion T2 von einer anderen Transaktion T1 liest, T1 vor T2 commiten muss. Für die 2PL-konforme Historie $w_1[x]r_2[x]w_2[y]c2c1$ gilt das nicht, da T1 nach der Schreiboperation die Sperre auf Datenelement x freigeben kann und sich die komplette Transaktion T2 vor das Commit von T1 "vordrängeln" kann.

Aufgabe 3 b)

Worin unterscheidet sich das strenge Zwei-Phasen-Sperrprotokoll vom Zwei-Phasen-Sperrprotokoll? Welche Auswirkungen hat der Einsatz des strengen 2PL-Sperrprotokolls

auf die vom Scheduler erzeugten Historien?

Lösung

Beim strengen Zwei-Phasen-Sperrprotokoll werden alle Sperren bis zum Abschluss (commit) der Transaktion gehalten. Erst dann werden alle Sperren (auf einmal) freigegeben. Hierdurch kann das in Aufgabenteil a) beschriebene vordrängeln vermieden werden. Zusätzlich vermeidet das strenge Zwei-Phasen-Sperrprotokoll kaskadiertes Rücksetzen und produziert strikte Historien (da eine Transaktion T2 erst auf ein von T1 geändertes Datenelement zugreifen kann nachdem T1 commitet hat). Die von einem Scheduler, der das strenge Zwei-Phasen-Sperrprotokoll befolgt, erzeugten Historien sind allerdings nicht seriell.

Aufgabe 3 c)

Ein Problem beim Einsatz von sperrbasierten Synchronisationsverfahren ist das Auftreten von Verklemmungen (sog. *Deadlocks*). Geben Sie eine Abfolge von Operationen an, die bei Einsatz des Zwei-Phasen-Sperrprotokolls zu einem Deadlock führt. Was kann das Datenbanksystem tun um den Deadlock aufzulösen?

Lösung

Wenn die Sperren jeweils direkt vor der Schreiboperation angefordert werdne, kann es bei der folgenden Historie zu einem Deadlock kommen: $w_1[x]w_2[y]w_1[y]w_2[x]$. Beispielsweise kann T1 nicht y schreiben, da dies noch von T2 gesperrt ist, T2 kann jedoch die Sperre nicht freigeben da sie noch Datenelement x schreiben und somit sperren muss.

Um den Deadlock aufzulösen kann das Datenbanksystem eine Transaktion, die am Deadlock (Zyklus im Wartegraphen) beteiligt ist zurücksetzen. Welche Transaktion ausgewählt wird, hängt dabei von der entsprechenden Strategie ab (z.B. jüngste TA, kürzeste TA, TA mit den wenigsten Änderungsoperationen, ...)

Aufgabe 3 d)

Wie können beim Einsatz des Zwei-Phasen-Sperrprotokoll Deadlocks vermieden werden?

Lösung

Fordert jede Transaktion *alle* von ihr benötigten Sperren *direkt zu Beginn* der Transaktion an (*Preclaiming*) können keine Verklemmungen auftreten. Alternativ können Zeitstempel zugewiesen werden, und darauf basierend Transaktionen zurückgesetzt werden, die benötigte Sperren halten (siehe wound-wait/wait-die).

Aufgabe 4

Gegeben seien die Transaktionen T_1, T_2, T_3, T_4 und T_5 sowie die Datenelemente a, b, c, d, e

und f. Es gelte das strenge Zwei-Phasensperrprotokoll.

- $T_i(s, a)$ bzw. $T_i(S, a)$ bedeute, dass die Transaktion T_i eine S-Sperre (Shared Lock) auf dem Datenelement a hält bzw. anfordert.
- $T_i(x, a)$ bzw. $T_i(X, a)$ bedeute, dass die Transaktion T_i eine X-Sperre (Exclusive Lock) auf dem Datenelement a hält bzw. anfordert.

Folgende Sperren werden aktuell durch die Transaktionen gehalten:

1. $T_1(x, a), T_1(s, b), T_1(s, e)$,
2. $T_2(s, b)$,
3. $T_3(s, c), T_3(s, e)$,
4. $T_4(s, c), T_4(x, d), T_4(s, e)$,
5. $T_5(x, f)$

Die einzelnen Transaktionen fordern jeweils folgende Sperren an, um mit der Bearbeitung fortfahren zu können:
$T_1(X, c)$, $T_2(S, f)$, $T_3(X, b)$, $T_4(S, a)$, $T_5(X, d)$
Zeichnen Sie den Wartegraphen dieser Transaktionen. Begründen Sie anhand Ihres Wartegraphen warum eine Verklemmung vorliegt. Lösen sie diese Verklemmung durch Zurücksetzen *einer* Transaktion auf.

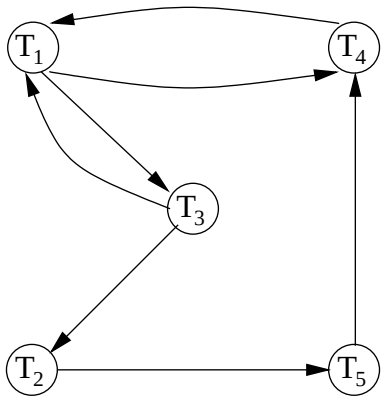
Lösung

Tabelle:

	a	b	c	d	e	f
T_1	x	s	X	-	s	-
T_2	-	s	-	-	-	S
T_3	-	X	s	-	s	-
T_4	S	-	s	x	s	-
T_5	-	-	-	X	-	x

- T_1 wartet auf T_3, T_4
- T_2 wartet auf T_5
- T_3 wartet auf T_1, T_2
- T_4 wartet auf T_1
- T_5 wartet auf T_4

Wartegraph:



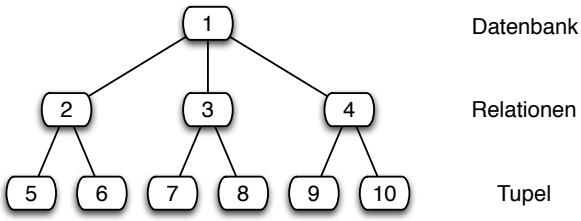
Es existieren Zyklen im Wartegraph. Daraus folgt, dass Verklemmungen vorliegen. Diese Verklemmungen können aber durch Zurücksetzen der Transaktion T_1 vollständig aufgelöst werden.

Aufgabe 5

Gegeben sei die Kompatibilitätsmatrix für das Multi-Granularity-Locking (MGL)

	S	X	IS	IX	SIX
S	√	-	√	-	-
X	-	-	-	-	-
IS	√	-	√	√	√
IX	-	-	√	√	-
SIX	-	-	√	-	-

Weiterhin sei die folgende logische Hierarchie in einem Datenbanksystem gegeben. Auf der höchsten Ebene (Knoten 1) befindet sich eine ganze Datenbank, darunter Relationen (Knoten 2-4) und auf der niedersten Ebene einzelne Tupel (Knoten 5-10).



Drei Transaktionen T1, T2 und T3 fordern folgende Sperren an:

- T1: Exklusive Sperre auf Knoten 5
- T2: Lese-Sperre auf Knoten 7
- T3: Lese-Sperre auf Knoten 4

Welche Sperren müssen in den darüberliegenden Hierarchieebenen bei Einsatz von MGL angefordert werden? Können alle erforderlichen Sperren parallel vergeben werden, d.h. sind sie miteinander kompatibel?

Anmerkung: In der Übung kam die Frage auf, warum SIX nicht kompatibel ist mit IX. SIX steht für "share and intention exclusive". Alle mit dieser Sperre belegten Knoten sind deshalb mit einem S-Lock und zusätzlich einem IX-Lock belegt. Die Sperre ist somit kompatibel mit allen Sperren, die sowohl mit S, als auch mit IX kompatibel sind.

Lösung

- T1: IX-Sperre auf Knoten 2 und Knoten 1
- T2: IS-Sperre auf Knoten 3 und Knoten 1
- T3: IS-Sperre auf Knoten 1

Die Sperren sind kompatibel.