ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Dr. G. Alonso *C. Hagen, B. Kemme*

AR 1: Transaktionsverwaltung

Theoretische Übung 1 Beispiellösung

Aufgabe 1: Rücksetzbarkeit

Die Rücksetzbarkeit von Transaktionen wird zu jedem Zeitpunkt der Bearbeitung gefordert.

a) Warum wird diese Forderung aufgestellt? Begründen Sie Ihre Antwort!

Lösung:

Von einer Transaktion wird unter anderem gefordert, daß sie als atomare Operation erscheint. Die Frage ist nun, ob zu einem **beliebigen** *Zeitpunkt diese Operation abgebrochen werden kann, so daß bereits getätigte Teiloperationen rückgängig gemacht werden müssen. Dies ist aber sicherlich der Fall, da allein schon der (das) Anwender(-programm) jederzeit ein Abbruch auslösen kann.*

Um Transaktionen möglichst parallel ablaufen zu lassen, sollen Sperren so kurz wie möglich gesetzt bleiben.

Wir betrachten Transaktionen mit Lese- und Schreibaktionen, die Lesesperren (S) und Schreibsperren (X) anfordern. Wir wollen die beiden Fälle untersuchen, in welchen diese Sperren entweder i) bereits *vor Ende* oder ii) erst *am Schluß* der Transaktion freigegeben werden können.

b) Wann ergeben sich für Lese- oder Schreibsperren bei den Alternativen i) oder ii) Probleme bei der Rücksetzbarkeit? Beschreiben Sie die möglichen Fehlersituationen.

Lösung:
Mögliche Probleme bei der Freigabe von Sperren:

	Sperrfreigabe vor Ende	am Schluß *der Transaktion*
S-Sperre	*kein Problem*	*kein Problem*
X-Sperre	**Problem!**	*kein Problem*

Die einzige Situation, in der es Probleme geben kann, ist die vorzeitige Freigabe von X-Sperren. In diesem Fall würden geschriebene/veränderte Objekte vor dem Commit *einer Transaktion anderen Transaktionen zugänglich gemacht! Dies ist unannehmbar, sei es für den Fall, daß eine andere Transaktion diese Werte 'nur' liest (und aufgrund dessen Entscheidungen fällt) oder sogar verändert.*

c) In welchen Fällen führt das seitenweise Wiederherstellen des Anfangszustandes einer Transaktion zu Problemen?

Lösung:
Unabhängig von den in b) erwähnten Problemen bringt das seitenweise Wiederherstellen des Anfangszustandes einer Transaktion dann Probleme, wenn die Seite zwischenzeitlich von anderen Transaktionen ebenfalls verändert wurde. Dies kann dann passieren, wenn das Lock-Granulat (zu sperrende Objekte, hier bei Schreib-Operationen) kleiner als das Log-Granulat (z.B. Before-Images der bearbeiteten Seiten) ist.

Aufgabe 2: Lebendige Aktionen und Serialisierbarkeit

Die folgenden Transaktionen führen Aktionen auf vier Datenbankobjekten aus:

$$\begin{aligned} T_1 &= < r_1^1(d)\ r_2^1(a)\ r_3^1(b)\ w_4^1(a) > \\ T_2 &= < r_1^2(a)\ w_2^2(a)\ r_3^2(c)\ r_4^2(a) > \\ T_3 &= < r_1^3(b)\ r_2^3(a)\ w_3^3(b) > \end{aligned}$$

a) Prüfen Sie, ob der folgende Schedule tote Aktionen enthält, indem Sie zuerst seinen Abschluss bilden und dann alle nützlichen Aktionen herausfinden:

$$S = < r_1^2(a)\ r_1^1(d)\ w_2^2(a)\ r_1^3(b)\ r_2^3(a)\ r_3^2(c)\ r_2^1(a)\ w_3^3(b)\ r_4^2(a)\ r_3^1(b)\ w_4^1(a) >$$

Lösung:
Direkt nützlich:
$w_1^0(a) \rightarrow r_1^2(a),\ w_2^0(b) \rightarrow r_1^3(b),\ w_3^0(c) \rightarrow r_3^2(c),\ w_4^0(d) \rightarrow r_1^1(d),\ r_1^2(a) \rightarrow w_2^2(a),$
$r_1^1(d) \rightarrow w_4^1(a),\ w_2^2(a) \rightarrow r_2^3(a),\ w_2^2(a) \rightarrow r_2^1(a),\ w_2^2(a) \rightarrow r_4^2(a),\ r_1^3(b) \rightarrow w_3^3(b),$
$r_2^3(a) \rightarrow w_3^3(b),\ r_2^1(a) \rightarrow w_4^1(a),\ w_3^3(b) \rightarrow r_3^1(b),\ r_3^1(b) \rightarrow w_4^1(a),\ w_4^1(a) \rightarrow r_1^\omega(a),$
$w_3^3(b) \rightarrow r_2^\omega(b),\ w_3^0(c) \rightarrow r_3^\omega(c),\ w_4^0(b) \rightarrow r_4^\omega(d)$

Tot: $r_3^2(c),\ r_4^2(a)$

b) Ändern Sie die Aktion 4 von T_2 ($r_4^2(a)$) in einen Schreibvorgang ($w_4^2(a)$). Enthält der Schedule jetzt tote Aktionen?

Lösung:
Ja, wiederum, da $w_4^2(a)$ ebenso tot ist, ist es auch wieder $r_3^2(c)$.

c) Sind die beiden Schedules aus a) und b) final-state-serialisierbar (FSSR), view-serialisierbar (VSR) oder konfliktserialisierbar?

Lösung:

Der Schedule aus Teilaufgabe a) ist konfliktserialisierbar und damit auch VSR und FSSR, wie sich sofort aus seinem Abhängigkeitsgraphen ergibt.

Der Schedule aus Teilaufgabe b) hingegen ist überhaupt nicht serialisierbar, da das LRF-Paar $(w_2^2(a), r_2^1(a))$ aufgrund von $w_4^2(a)$ in keiner seriellen Ausführung erzielt werden kann. Denn bei serieller Ausführung würde $w_4^2(a)$ zwischen jenen beiden Aktionen stehen oder aber $r_2^1(a)$ vor $w_2^2(a)$.

Aufgabe 3: Lebendige Aktionen und Serialisierbarkeit

Gegeben seien die folgenden drei Transaktionen mit einem möglichen Schedule.

$$
\begin{aligned}
T_1 &= < r_1^1(a)\ w_2^1(a) > \\
T_2 &= < r_1^2(a)\ w_2^2(a)\ r_3^2(b)\ w_4^2(b) > \\
T_3 &= < r_1^3(a)\ w_2^3(b)\ w_3^3(a) > \\
S &= < r_1^1(a)\ r_1^2(a)\ w_2^1(a)\ w_2^2(a)\ r_3^2(b)\ r_1^3(a)\ w_2^3(b)\ w_3^3(a)\ w_4^2(b) >
\end{aligned}
$$

a) Welche der Aktionen in diesem Schedule sind tot?

Lösung:
Von $w_2^1(a)$ und $w_2^3(b)$ liest niemand; sie sind daher tot. Da $w_2^1(a)$ tot ist, ist es auch $r_1^1(a)$.

b) Können Sie einen Schedule finden, der keine toten Aktionen enthält?

Lösung:

Nein, da eine $r(b)$-Aktion mehr nötig wäre. So ist immer ein $w(b)$ (evtl. auch $w^0(b)$) tot.

c) Sind die beiden Schedules aus a) und — falls vorhanden — b) FSSR, VSR oder konfliktserialisierbar?

Lösung:

Nein, da die LRF-Relationen aller seriellen Schedules nicht mit der des obigen Schedules übereinstimmen.

Aufgabe 4: Konflikt-Serialisierbarkeit

Gegeben seien die folgenden drei Transaktionen mit einem möglichen Schedule.

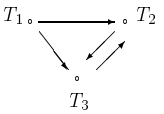
$$
\begin{aligned}
T_1 &= < r_1^1(a)\ r_2^1(b) > \\
T_2 &= < r_1^2(a)\ w_2^2(a)\ r_3^2(b)\ r_4^2(b) > \\
T_3 &= < r_1^3(a)\ w_2^3(b)\ w_3^3(a) > \\
S &= < r_1^1(a)\ r_1^2(a)\ r_2^1(b)\ w_2^2(a)\ r_3^2(b)\ r_1^3(a)\ w_2^3(b)\ w_3^3(a)\ r_4^2(b) >
\end{aligned}
$$

a) Ist der Schedule der drei Transaktionen konfliktserialisierbar?

Lösung:
Die Abhängigkeitsrelation lautet:

$$
\begin{aligned}
\rightarrow_S = \{&(r_1^1(a), w_2^2(a)), (r_1^1(a), w_3^3(a)), (r_1^2(a), w_3^3(a)), (r_2^1(b), w_2^3(b)), \\
&(w_2^2(a), r_1^3(a)), (w_2^2(a), w_3^3(a)), (r_3^2(b), w_2^3(b)), (w_2^3(b), r_4^2(b))\}
\end{aligned}
$$

Der Schedule ist nicht konfliktserialisierbar, da der dazugehörige Abhängigkeitsgraph einen Zyklus enthält (Z.B. durch $(w_2^2(a), r_1^3(a))$ und $(w_2^3(b), r_4^2(b))$).



b) Bei welchen Aktionen kann der Scheduler Probleme bei der Konfliktserialisierung erkennen?

Lösung:
Unter der Annahme, daß der Schedule nach jeder Aktion auf seine Serialisierbarkeit getestet wird, muß bei jeder neu bearbeiteten Aktion nach Konflikten mit bisherigen Aktionen gesucht werden. Konfliktpaare werden im Abhängigkeitsgraphen als Kanten eingetragen. Der Schedule wird dann *als nichtserialisierbar erkannt, wenn der erste Zyklus im Abhängigkeitsgraphen auftritt. In unserem Fall ist es das Paar (w_2^2, r_1^3), das mit (w_2^3, r_4^2) einen Zyklus im Abhängigkeitsgraphen erzeugt.*

c) Überprüfen Sie, ob der Schedule FSSR oder sogar VSR ist.

Lösung:
Der Schedule ist FSSR, da er die gleiche LRF-Relation

$$\{(T_0, a, T_2), (T_2, a, T_3), (T_3, a, T_\omega), (T_3, b, T_\omega)\}$$

hat wie der serielle Schedule $< T_1, T_2, T_3 >$. Er ist jedoch nicht viewserialisierbar, da T_3 von T_2 liest und umgekehrt ((w_2^2, r_1^3) und (w_2^3, r_4^2)).