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

Prof. H.-J. Schek *A. Feßler, U. Röhm, H. Schuldt, R. Weber*

Informationssysteme IS-G SS 1997

Übung 10 — Beispiellösung

Aufgabe 1: Serialisierbarkeit, Äquivalenz

a) Gegeben ist der folgende Schedule der Transaktionen T_1, T_2, T_3, T_4 und T_5:

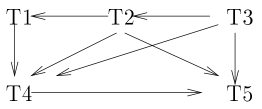
$$S_1 = < R_1(a), R_2(b), W_1(b), W_3(c), W_2(c), W_2(d), R_4(a), W_4(c), W_5(c), W_5(d), W_4(a) >$$

Ist der Schedule serialisierbar? Falls ja, geben Sie mindestens einen äquivalenten seriellen Schedule an. Falls nein, warum nicht?

Der Schedule ist serialisierbar. Die Konfliktpaare des Schedules sind:

$$\{(R_1(a), W_4(a)), (R_2(b), W_1(b)), (W_3(c), W_2(c)), (W_3(c), W_4(c)), (W_3(c), W_5(c)), (W_2(c), W_4(c)), (W_2(c), W_5(c)), (W_4(c), W_5(c)), (W_2(d), W_5(d))\}$$

Der dazugehörige Abhängigkeitsgraph ist azyklisch und sieht wie folgt aus:



Durch topologische Sortierung erhält man leicht den äquivalenten seriellen Schedule: $T_3\ T_2\ T_1\ T_4\ T_5$.

b) Betrachten Sie den folgenden Schedule der Transaktionen T_1, T_2 und T_3:

$$S_2 = < R_1(a), R_2(b), W_1(a), R_3(b), R_2(c), W_3(d), W_2(c), W_3(e) >$$

Wie sieht der dazugehörige Abhängigkeitsgraph aus? Ist dieser Schedule serialisierbar? Falls ja, geben Sie mindestens einen äquivalenten seriellen Schedule an. Falls nein, warum nicht?

Der Abhängigkeitsgraph dieses Schedules enthält keine Kanten, weil es keine Konfliktpaare gibt. Man kann also T_1, T_2 *und* T_3 *beliebig topologisch sortieren, und erhält immer einen äquivalenten seriellen Schedule, z.B.:* $T_2\ T_3\ T_1$.

c) Gegeben ist der folgende Schedule der Transaktionen T_1, T_2, T_3:

$$S_3 = < R_1(a), R_2(a), W_1(a), R_3(b), W_3(b), R_2(b) >$$

Ist dieser Schedule serialisierbar und korrekt? Was fällt Ihnen bei diesem Schedule auf?

Dieser Schedule ist zwar serialisierbar, aber er ist nicht ordnungserhaltend konfliktserialisierbar. Der dazugehörige Abhängigkeitsgraph zeigt, dass der einzige äquivalente serielle Schedule $T_3\ T_2\ T_1$ *ist, wobei* T_1 *erst nach Abschliessen der Transaktionen* T_3 *und* T_2 *ausgeführt werden darf. In dem ursprünglichen Schedule ist die Transaktion* T_1 *aber schon vollständig beendet, bevor die Transabktion* T_3 *startet.*

$$\text{T3} \longrightarrow \text{T2} \longrightarrow \text{T1}$$

d) Gegeben seien die folgenden zwei Schedules der Transaktionen T_1, T_2, T_3 und T_4:

$$S_4 = < R_1(a), R_4(b), W_1(a), W_3(e), W_2(c), W_2(b), R_3(d), R_2(a), W_3(c), R_4(e), W_4(d) >$$
$$S_5 = < W_3(e), R_1(a), W_1(a), W_2(c), R_4(b), R_4(e), W_2(b), W_4(d), R_2(a), R_3(d), W_3(c) >$$

Sind die beiden Schedules äquivalent? Begründen Sie Ihre Antwort!

Die Schedules S_4 *und* S_5 *sind genau dann äquivalent, wenn sie dieselben Transaktionsmengen und Aktionsmengen haben und wenn in beiden Schedules die Konfliktpaare jeweils gleich geordnet vorliegen. Die Transaktions- und Aktionsmengen von* S_4 *und* S_5 *sind gleich. Die Konfliktpaare von* S_4 *sind:*

$$\{(R_4(b), W_2(b)), (W_1(a), R_2(a)), (W_3(e), R_4(e)), (W_2(c), W_3(c)), (R_3(d), W_4(d))\}$$

Die Konfliktpaare von S_5 *sind:*

$$\{(W_2(c), W_3(c)), (W_4(d), R_3(d)), (R_4(b), W_2(b)), (W_3(e), R_4(e)), (W_1(a), R_2(a))\}$$

Die Konfliktpaare in beiden Schedules liegen nicht gleich geordnet vor ($(R_3(d), W_4(d))$ *in* S_4 *und* $(W_4(d), R_3(d))$ *in* S_5*). Damit sind die beiden Schedules nicht äquivalent.*

Aufgabe 2: Serialisierbarkeit

In der Vorlesung wurde der folgende Satz vorgestellt:

> Ein Schedule ist serialisierbar genau dann, wenn sein Abhängigkeitsgraph azyklisch ist.

Für jeden beliebigen Schedule, dessen Abhängigkeitsgraph azyklisch ist, wurde seine Zugehörigkeit zur Klasse der serialisierbaren Schedules durch topologische Sortierung des Abhängigkeitsgraphen nachgewiesen.

Beweisen Sie nun die umgekehrte Richtung des Satzes, d.h. dass jeder serialisierbare Schedule einen azyklischen Abhängigkeitsgraphen besitzt.

Beweis:

Sei S *ein beliebiger Schedule aus der Klasse der serialisierbaren Schedules. So existiert ein serieller Schedule* S'*, so dass beide Schedules* S *und* S' *die gleichen Konfliktpaare enthalten und entsprechende Konfliktpaare gleich geordnet sind.*

Angenommen, der Abhängigkeitsgraph des Schedules S *enthält einen Zyklus der Form* $T_1 \longrightarrow T_2 \longrightarrow \ldots \longrightarrow T_k \longrightarrow T_1$. *Der gleiche Zyklus existiert dann auch in* S'. *Da aber* S' *ein serieller Schedule ist, kann er keinen Zyklus enthalten. Da* S *zu* S' *äquivalent (d.h. serialisierbar) ist, kann also* S *auch keinen Zyklus enthalten.*

$\square$

Aufgabe 3: Transaktionsverwaltung

Gegeben sei der folgende Schedule:

$$S =< R_1(x), W_1(x), R_2(x), W_2(x), C_2, W_1(y), C_1 >$$

C_1 und C_2 markieren den erfolgreichen Abschluss der Transaktionen T_1 und T_2 (commit).

a) Ist der Schedule serialisierbar? Begründen Sie Ihre Antwort.

Ja, da der Abhängigkeitsgraph azyklisch ist ($T_1 \longrightarrow T_2$).

b) Kann der Schedule durch das strikte 2PL-Protokoll entstanden sein? Begründen Sie Ihre Antwort.

Nein, der Schedule kann nicht durch das strikte 2PL-Protokoll entstanden sein. Gemäss dieses Protokolls würde T_1 seine Sperren erst zum Commit-Zeitpunkt C_1 freigeben. Dadurch hätte T_2 die Sperre auf x nicht vorher bekommen können.

c) Kann dieser Schedule durch das allgemeine 2PL-Protokoll entstanden sein? Begründen Sie Ihre Antwort.

*Ja, der Schedule kann durch das 2PL-Protokoll entstanden sein. Die Transaktion T_1 kann nach $W_1(x)$ in die Unlock-Phase gehen (die Sperre auf y wird frühzeitig angefordert (*Preclaiming*)) und so seine Sperre auf x freigeben bevor T_2 die Sperre auf x möchte:*

$$< L_1(x), R_1(x), W_1(x), L_1(y), U_1(x), L_2(x), R_2(x), W_2(x), U_2(x), C_2, W_1(y), U_1(y), C_1 >$$

d) Welches Problem gibt es, wenn anstatt der Aktion C_1 nun die Abort-Aktion A_1 ausgeführt werden soll? Wie kann das Problem verhindert werden?

Wegen der Abort-Aktion und dem Rücksetzen des Objektes x müsste auch die Transaktion T_2 zurückgesetzt werden. Dies ist aber nicht mehr möglich, da T_2 bereits erfolgreich abgeschlossen wurde (Domino-Effekt).

Das Problem kann verhindert werden, indem das S2PL-Protokoll verwendet wird, da Sperren bis ans Ende der Transaktion gehalten werden. Die Transaktion T_2 hätte somit nicht auf das Objekt x zugreifen können, bevor T_1 erfolgreich oder nicht erfolgreich abgeschlossen worden wäre.

Aufgabe 4: Sperrprotokoll

Betrachten Sie das folgende Sperrprotokoll:

- Für Transaktionen, die mindestens eine Änderungsoperation beinhalten, wird das strikte 2PL verwendet.
- Für Transaktionen, die nur Leseoperationen beinhalten, wird ein nichtzweiphasiges Sperrprotokoll wie folgt verwendet[1]:
  - Vor dem Zugriff auf ein Objekt muss dieses gesperrt werden.
  - Alle Sperren einer Transaktion werden jeweils bei Beendigung einer SQL–Anweisung freigegeben. (Bei der nächsten SQL–Anweisung werden wieder neue Sperren angefordert.)

a) Zeigen Sie, dass bei Verwendung dieses Protokolls nichtserialisierbare Schedules entstehen können.

Der Schedule $S =< R_1(x), W_2(x), W_2(y), R_1(y) >$ kann durch das gegebene Protokoll entstanden sein, ist aber nicht serialisierbar.

[1]Dieses Protokoll entspricht dem TRANSACTION ISOLATION LEVEL READ COMMITTED des SQL–Standards.

L1(x)R1(x)U1(x) L1(y)R1(y)U1(y)

L2(x)W2(x)L2(y)W2(y)U2(x)U2(y) Zeit

b) Kann bei Verwendung des angegebenen Sperrprotokolls eine Integritätsbedingung der Datenbank verletzt werden? Falls nein, ist der „Verlust" der Serialisierbarkeit überhaupt gravierend? Falls ja, geben Sie ein Beispiel an.

Nein, da alle Änderungstransaktionen das strikte 2PL-Protokoll verwenden. Allerdings können Lesetransaktionen scheinbare Verletzungen der Integrität "beobachten" (inkonsistentes Lesen).

Betrachten wir die folgende Situation: In der Datenbank ist eine Integritätsbedingung gegeben, welche verlangt, dass bei Geld-Transfers zwischen Konto x und Konto y die Summe von x und y konstant bleibt, z.B. $x + y =1200$. Seien $x =1000$ und $y =200$. Schedule $S =< R_1(x), W_2(x), W_2(y), R_1(y) >$ beinhaltet zwei Transaktionen, wobei Transaktion T_2 eine bestimmte Geldmenge von Konto x nach Konto y transferiert und Transaktion T_1 die Summe der beiden Konten prüft:

T_1	T_2
$x = 1000$	$y = 200$
$R_1(x) \rightarrow x = 1000$	
	$W_2(x) \rightarrow x = x - 400 = 600$
	$W_2(y) \rightarrow y = y + 400 = 600$
$R_1(y) \rightarrow y = 600$	
$x + y = 1600 \neq Konstant\ 1200$	
$x = 600$	$y = 600$

Anhand dieses Ablaufs ist es leicht zu sehen, dass die Integritätsbedingung für T_1 scheinbar verletzt ist, da die Summe von x und y 1600 anstatt 1200 ist.

c) Geben Sie ein Beispiel an, bei dem das angegebene Sperrprotokoll sinnvoll ist.

Das angegebene Sperrprotokoll kann z.B. für statistische Auswertungen sinnvoll sein, die nicht unbedingt eine hohe Anforderung an die Genauigkeit stellen, jedoch sehr lange laufen.

Betrachten wir das folgende Anwendungsbeispiel: In einem grossen Lagerhaus wird eine wöchentliche Statistik erstellt. Diese Statistik gibt eine Übersicht, welche und wieviele Artikel in einer Woche verkauft worden sind, um den Markt-Trend festzustellen. Die Angaben über die einzelnen Artikel brauchen nicht sehr genau zu sein. Die Erstellung dieser Statistik dauert sehr lange, u.U. mehrere Stunden. Parallel zur Erstellung dieser Statistik laufen die normalen betrieblichen Buchungsprogramme und sollen möglichst wenig behindert werden.

Für das oben beschriebene Szenario ist das angegebene Protokoll sehr sinnvoll, da ansonsten für die Erstellung der Statistik alle Artikel gesperrt würden und erst nach Beenden der Erstellung freigegeben würden. Alle anderen Buchungsprogramme würden blockiert und müssten sehr lange warten.