Adatbázisok elmélete 25. előadás

Katona Gyula Y.
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 137/b
kiskat@cs.bme.hu
http://www.cs.bme.hu/˜kiskat

2005

Feladatok

1. Tegyük fel, hogy az alábbi műveletsorozatban minden egyes olvasás- és írásműveletet közvetlenül megelőzi az RLOCK ill. a WLOCK igénylése. Tegyük továbbá fel, hogy a zárak feloldása a tranzakció utolsó művelete után történik meg. Adjuk meg azokat a műveleteket, melyek végrehajtását az ütemező megtagadja, és mondjuk meg, hogy létrejön-e holtpont. Hogyan alakul a műveletek végrehajtása során a várakozási gráf? Ha létrejön holtpont, ABORT-áljuk az egyik tranzakciót, és mutassuk meg, hogyan folytatódik a műveletsorozat!

$$r_1(A), r_2(B), w_1(C), r_3(D), r_4(E), w_3(B), w_2(C), w_4(A), w_1(D)$$

Megoldás:
Először sorban kérünk zárakat A, B, C, D, E-re.

$$rl_1(A), r_1(A), rl_2(B), r_2(B), wl_1(C), w_1(C), rl_3(D), r_3(D), rl_4(E)$$

Az első probléma a $wl_3(B)$, hiszen ekkor van zár még $lr_2(B)$. $wl_3(B)$ megtagadva, várakozási gráfba (T_3, T_2) él, T_3 vár
$wl_2(C)$ megtagadva, várakozási gráfba (T_2, T_1) él, T_2 vár
$wl_4(A)$ megtagadva, várakozási gráfba (T_4, T_1) él, T_4 vár
$wl_1(D)$ megtagadva, várakozási gráfba (T_1, T_3) él, kört kapunk, holtpont alakul ki.
ABORT T_1, ekkor eltűnik (T_2, T_1) él és a (T_4, T_1) él a várakozási gráfból, ami így DAG lesz.
Ezért pl. T_2, T_3, T_4 sorrendben lefuthat a többi tranzakció:

$$ul_1(A), ul_1(C), wl_2(C), w_2(C), ul_2(C), ul_2(B), wl_3(B), w_3(B), ul_3(B), wl_1(A), w_4(A), ul_4(A), ul_4(E)$$

2. Tekintsük az alábbi (csak olvasásokból és írásokból álló) ütemezést:

$$r_2(A),\ w_3(B),\ r_1(A),\ w_2(B),\ w_1(C)$$

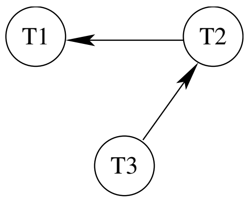
(Itt $r_2(A)$ jelentése: a második tranzakció olvassa A-t, $w_3(B)$ jelentése: a harmadik tranzakció írja B-t.)
Az egyszerű tranzakciómodellt használva illessz be zárkéréseket a fenti ütemezésbe oly módon, hogy legális zárolást kapjunk és
(a) ne kövesse mindegyik tranzakció a 2PL-t, de (a zárkérések alapján döntve) az ütemezés sorosítható legyen,
Megoldás:

$$l_2(A),\ r_2(A),\ u_2(A),$$
$$l_3(B),\ w_3(B),\ u_3(B),$$
$$l_1(A),\ r_1(A),\ u_1(A),$$
$$l_2(B),\ w_2(B),\ u_2(B),$$
$$l_1(C),\ w_1(C),\ u_1(C)$$

Ha felrajzoljuk a sorosítási gráfot: $T_3 \to T_2 \to T_1$, tehát sorosítható.
(b) mindegyik tranzakció kövesse a 2PL-t, de (a zárkérések alapján döntve) ne legyen sorosítható az ütemezés,
Megoldás:

Ilyet nem lehet adni, mert tanultuk azt a tételt, hogy ha minden tranzakció követi a 2PL-t, akkor sorosítható lesz az ütemezés.
(c) mindegyik tranzakció kövesse a 2PL-t, és (a zárkérések alapján döntve) legyen sorosítható az ütemezés.
Megoldás:
Az ötlet az, hogy az $l_2(B)$-t előbbre lehet hozni és így előbb fel lehet oldani a zárat A-n.

$$l_2(A),\ r_2(A),$$
$$l_3(B),\ w_3(B),\ u_3(B),$$
$$l_2(B),\ u_2(A),\ l_1(A),\ r_1(A),$$
$$w_2(B),\ u_2(B),$$
$$l_1(C),\ w_1(C),\ u_1(A),\ u_1(C)$$

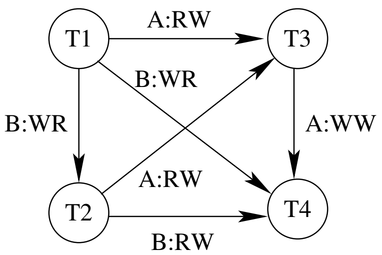


(Mivel ez 2PL, a tétel szerint sorosítható.)

3. Az alábbi legális ütemezés négy tranzakció zárjait tartalmazza az RLOCK/WLOCK modellben. Rajzoljuk fel a sorosítási gráfot! Sorosítható-e az ütemezés? Ha igen, milyen soros ütemezések ekvivalensek az eredeti ütemezéssel?

(0)	T_1	T_2	T_3	T_4
(1)		RLOCK A		
(2)	RLOCK A			
(3)	WLOCK C			
(4)	UNLOCK C			
(5)			RLOCK C	
(6)	WLOCK B			
(7)	UNLOCK B			
(8)				RLOCK B
(9)	UNLOCK A			
(10)		UNLOCK A		
(11)			WLOCK A	
(12)				RLOCK C
(13)		WLOCK D		
(14)				UNLOCK B
(15)			UNLOCK C	
(16)		RLOCK B		
(17)			UNLOCK A	
(18)				WLOCK A
(19)		UNLOCK B		
(20)				WLOCK B
(21)				UNLOCK B
(22)		UNLOCK D		
(23)				UNLOCK C
(24)				UNLOCK A

Megoldás:



Sorosítható, mert DAG. Egy ekvivalens soros ütemezés van: T_1, T_2, T_3, T_4

4. Az alább megadott tranzakciók mindegyikénél tételezzük fel, hogy beszúrjuk a LOCK és UNLOCK műveletet miden egyes adatbáziselemhez, amihez hozzáférünk: $r_1(A), w_1(B)$. Adjuk meg, hogy a zárolási, feloldási, olvasási és írási műveleteknek hány olyan sorrendje lehet, ha a zárolások megfelelőek és a zárolás i) kétfázisú, ii) nem kétfázisú.

Megoldás:
Ha a zárolás megfelelő, akkor csak ennek összefésülései jönnek szóba:
a) $l_1(A); r_1(A); u_1(A)$ b) $l_1(B); w_1(B); u_1(B)$
ii) csak akkor nem kétfázisú, ha az egyik megelőzi a másikat: kétféle ilyen van.
i) Hány összefésülés van összesen? Ismétléses kombináció, vagy a 6 pozícióból melyik 3 az elsőből: $\frac{6!}{3!\cdot 3!} = \binom{6}{3} = 20$, tehát i)=18.

5. Az alábbi legális ütemezés két olyan tranzakció utasításait tartalmazza, melyek betartják a figyelmeztető protokollt. Hogy nézhet ki az ütemezésben szereplő adategységek egymásba ágyazottságát reprezentáló fa, ha tudjuk, hogy a gyökérnek legfeljebb 3 gyereke van? (Ha több lehetséges eset van, akkor mindet add meg).
$WARN_1(E), WARN_1(H), WARN_2(E), LOCK_1(A), LOCK_1(C), UNLOCK_1(A), LOCK_2(F), UNLOCK_1(H), UNLOCK_2(F), UNLOCK_1(C), UNLOCK_2(E), UNLOCK_1(E)$

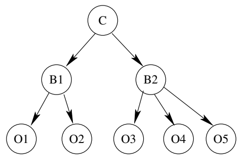
Megoldás:
T_2 miatt biztos, hogy E a gyökér és F ennek a fia.
T_1 miatt biztos, hogy H is E-nek a fia.
A és C helyzete a kérdéses még. Két eset lehetséges:

- Az A csúcs a H gyereke: C nem lehet se A, se H gyereke, mert később oldjuk fel C-n a zárat, mint A-n és H-n, így ekkor C csak E gyereke lehet és ez összhangban is van a zárolással. Ez egy lehetséges megoldás.
- Az A csúcs az E gyereke: C nem lehet se A, se H gyereke a zárfeloldások miatt, de E-é se lehet, mert a gyökérnek csak három gyereke lehet. Így ezen az ágon nem kapunk megoldást.

6. Vegyünk egy objektum orientált adatbázist. A C osztály objektumait két blokkban tároljuk a B_1-ben és a B_2-ben. A B_1 tartalmazza az O_1 és O_2 objektumokat, míg a B_2 az O_3, O_4, O_5 objektumokat. Adjuk meg a zárolási kérések sorozatát és a figyelmeztető protokoll alapú ütemező feladatát az alábbi kérési sorozatokhoz. Feltehetjük, hogy minden kérés éppen azelőtt fordul elő, mint amikor éppen szükség van rá, és minden zárfeloldás a tranzakció befejeztével történik. Használjuk az RLOCK/WLOCK modellt.

$$r_1(O_1), w_2(O_2), r_2(O_3), w_1(O_4)$$

Megoldás: A hierarchiát ábrázoló fa:



- Az első körben O_1-re kell majd zárat tenni: $RWARN_1(C), RWARN_1(B_1), RLOCK_1(O_1)$.
- A második körben O_2-re kell majd zárat tenni: $WWARN_2(C), WWARN_2(B_1), WLOCK_2(O_2)$.
- A harmadik körben O_3-ra kell majd zárat tenni: $RWARN_2(C), RWARN_2(B_2), RLOCK_2(O_3)$. Ezután T_2 felengedi a zárakat és figyelmeztetéseit: $UNLOCK_2(O_3), UNLOCK_2(B_2), UNLOCK_2(O_2), UNLOCK_2(B_1), UNLOCK_2(C)$

- A negyedik körben O_4-re kell majd zárat tenni: $WWARN_1(C), WWARN_1(B_2), WLOCK_1(O_4)$. Ezután T_1 felengedi a zárakat és figyelmeztetéseit: $UNLOCK_1(O_4), UNLOCK_1(B_4), UNLOCK_1(O_1), UNLOCK_1(B_1), UNLOCK_1(C)$

7. Tekintsük a T_1, T_2, T_3 és T_4 tranzakciók írási és olvasási kéréseiből álló

$$r_1(A),\ w_2(B),\ r_3(A),\ w_1(B),\ r_2(A),\ r_4(B),\ w_4(A),\ w_3(B)$$

sorozatot. Időbélyeges tranzakciókezeléssel akarjuk a sorosítható ütemezést kikényszeríteni, a tranzakciók időbélyegei: $t(T_1) = 1$, $t(T_2) = 2$, $t(T_3) = 3$, $t(T_4) = 4$. Írja le, hogy mi történik a fenti sorozat esetén, azaz mely kéréseket teljesíti az ütemező, melyeket nem, mely kérések vezetnek ABORT-hoz és adja meg azt is, hogy hogyan változnak az egyes műveletek után az adategységek írási és olvasási idejei (ezek kezdetben mind nullák).

Megoldás:

Kérés	r(A)	w(A)	r(B)	w(B)	Magyarázat
	0	0	0	0	kezdetben minden nulla
$r_1(A)$	1	0	0	0	mehet
$w_2(B)$	1	0	0	2	mehet
$r_3(A)$	3	0	0	2	mehet
$w_1(B)$	3	0	0	2	Nem lesz ABORT, de írás sem
$r_2(A)$	3	0	0	2	Nem lesz ABORT, $r(A)$ nem változik
$r_4(B)$	3	0	4	2	mehet
$w_4(A)$	3	4	4	2	mehet
$w_3(B)$	3	4	4	2	ABORT

8. Alább látható egy napló, melyet az UNDO/REDO protokoll szerint képeztünk. Állítsuk vissza a konzisztens helyzetet.

(T_1, BEGIN)
$(T_1, A, 4, 5)$
(T_2, BEGIN)
(T_1, COMMIT)
$(T_2, B, 9, 10)$
(START CHECKPOINT (T_2))
$(T_2, C, 14, 15)$
(T_3, BEGIN)
$(T_3, D, 19, 20)$
$(T_3, A, 5, 6)$
$(T_3, E, 10, 15)$
(T_4, BEGIN)
$(T_4, F, 15, 16)$
(END CHECKPOINT)
(T_2, COMMIT)
HIBA

Megoldás:
Mivel van END CHECKPOINT, elég a START checkpoint-tól előre haladva megállapítjuk, hogy befejeződött T_2, nem fejeződött be T_3 és T_4. (T_1-el nem kell foglalkozni, mert már minden adata kiíródott.)
A T_2 elejéről indulva $B := 10$, de ez már nem kell, hiszen lemezen van. Majd $C := 15$.
Utána végéről visszafelé $F := 15, E := 10, A := 5, D := 19$, ABORT T_3 és ABORT T_4, FLUSH LOG