Tranzakciókezelés

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!

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

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,

(b) mindegyik tranzakció kövesse a 2PL-t, de (a zárkérések alapján döntve) ne legyen sorosítható az ütemezés,

(c) mindegyik tranzakció kövesse a 2PL-t, és (a zárkérések alapján döntve) legyen sorosítható az ütemezés.

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

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.

5. Egy adathozzáférési naplót szeretnénk ellenőrizni egy sorosíthatóságot ellenőrző programmal. (A napló az adatokhoz történő hozzáférések egy ütemezését tartalmazza: pl. $r_1(A), w_2(B), \ldots.$)

   A programunk sajnos nem tud elég hosszú naplókat elemezni, ezért kénytelenek vagyunk a naplót ketté vágni, és az elejét illetve a véget külön-külön ellenőrizni.

   Igaz-e, hogy ha a két naplórészben levő ütemezés – U_1 és U_2 – külön-külön sorosítható, akkor a naplóban szereplő teljes ütemezés is az, amennyiben a szétvágás során ügyeltünk arra, hogy:

   a) Úgy vágjuk szét a naplót, hogy ne legyen olyan tranzakció, amely mindkét naplófélben végez műveletet, azaz ha az U_1 ütemezésben tranzakciók egy S_1 halmaza végez műveleteket, az U_2 ütemezésben pedig a tranzakciók egy S_2 halmaza, akkor $S_1 \cap S_2 = \emptyset$?

   b) Úgy vágjuk szét a naplót, hogy ne legyen olyan adatelem, amihez mindkét naplófélben történik hozzáférés, azaz ha az U_1 ütemezésben a tranzakciók az adatelemek egy X_1 halmazán végeznek műveleteket, az U_2 ütemezésben levők pedig az adatelemek egy X_2 halmazán, akkor $X_1 \cap X_2 = \emptyset$?

6. 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).

7. Időbélyeges tranzakciókezelést alkalmazunk, három tranzakció van: T_1, T_2 és T_3, időbélyegeik rendre 10, 20 és 30. Az olvasások és írások iránti kérések így jönnek: $r_1(A)$, $r_2(A)$, $w_1(C)$, X, $w_3(C)$, $r_2(B)$ (az X kérés nem ismert).
   Mi lehetett X, ha tudjuk, hogy a tranzakciókezelő a kéréssorozat hatására

   (a) ABORT T_1-et rendel el?

   (b) ABORT T_2-et rendel el?

   (c) ABORT T_3-et rendel el?

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

(T_1, BEGIN)
($T_1, X, 5$)
(T_2, BEGIN)
($T_1, Y, 7$)
($T_2, X, 9$)
(T_3, BEGIN)
($T_3, Z, 11$)
(T_1, COMMIT)
(START CHECKPOINT (T_2, T_3))
($T_2, X, 13$)
($T_3, Y, 15$)
HIBA

9. Alább látható egy napló, melyet az REDO protokoll szerint képeztünk. Mit áll a ???? helyén? Állítsuk vissza a konzisztens helyzetet.

(T_1, BEGIN)
($T_1, A,$)
(T_2, BEGIN)
($T_2, B, 5$)
($T_1, C, 7$)
(T_3, BEGIN)
($T_3, D, 12$)
(T_1, COMMIT)
(START CHECKPOINT (????))
(T_4, BEGIN)
($T_2, E, 5$)
(T_2, COMMIT)
($T_3, F, 1$)
($T_4, G, 15$)
(END CHECKPOINT)
(T_3, COMMIT)
(T_5, BEGIN)
($T_5, H, 3$)
(START CHECKPOINT (????))
(T_5, COMMIT)
HIBA