Tranzakciókezelés gyakorlat

Írta: Szkupien Péter (szkupien.peter@db.bme.hu), 2021. január
Lektorálra: Dr. Gajdos Sándor (gajdos@db.bme.hu)

Jelen dokumentum a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Kar mérnökinformatikus képzés Adatbázisok (BMEVITMAB04) tárgyának tranzakciókezelés témájú gyakorlati feladatsorához tartalmaz részletes megoldásokat és magyarázatokat.

A magyarázatok során hivatkozunk a tárgy hivatalos jegyzetére (Gajdos Sándor: Adatbázisok), amely – a tárgyhoz kapcsolódó valamennyi további segédanyaghoz hasonlóan – elérhető a tárgy honlapján. Jelen gyakorlat témáját a jegyzet 10. fejezete fedi le.

Felhívjuk az olvasó figyelmét, hogy jelen dokumentum feldolgozása a kapcsolódó elmélet megértése után javasolt, önmagában nem elegendő a témakör elsajátításához, nem helyettesíti a tananyag megértését, az órák látogatását.

1. feladat

Legális-e az alábbi ütemezés? Ha nem, mit kellene módosítani rajta, hogy azzá váljon?

	T_1	T_2	T_3
(1)			RLOCK B
(2)			READ B
(3)		WLOCK B	
(4)			RLOCK A
(5)	RLOCK A		
(6)		WRITE B	
(7)			READ A
(8)	READ A		
(9)	UNLOCK A		
(10)			UNLOCK B
(11)			UNLOCK A

Elmélet

Egy *ütemezés legális*, ha i) a lockolt adategységeket unlock-kal fel is szabadítjuk, valamint ii) amíg egy adategység – egy másik tranzakció (nem megosztható) zárja által – foglalt, a tranzakció a zár felszabadulásáig várakozik (jegyzet 10.3. fejezet). Megjegyzés: ebből az is következik, hogy a legalitás szempontjából (is) csak a zárműveleteknek van jelentősége, az adatműveletek közömbösek, tehát az ütemezésből akár el is hagyhatók.

A definícióban szereplő „nem megosztható” kifejezés arra utal, hogy egymással kompatibilis zárból egyszerre több is lehet ugyanazon az adategységen. Például R/W zármodell esetén, ha egy adategységen egy tranzakció már fenntart egy olvasási zárat (RLOCK), egy másik tranzakció is elhelyezhet rajta egy ugyanilyet, vagyis az RLOCK egy megosztható zár – hiszen az nem okozhat problémát, ha egy adatot egyszerre két tranzakció is olvas. A zártípusok kompatibilitását az adott modell kompatibilitási mátrixa írja le (jegyzet 10.5. fejezet).

Megoldás

A táblázatban adott ütemezésről kell eldöntenünk, hogy legális-e. A feladatunk tehát az, hogy a legális ütemezés definíciójában szereplő két feltétel teljesülését ellenőrizzük. Amennyiben teljesülnek, az ütemezés legális, amennyiben nem, meg is kell adnunk, hogyan tudnánk legálissá tenni.

Az i) feltétel vizsgálata során minden tranzakciót ellenőrzünk, hogy minden általa zárolt adategységet fel is szabadít-e.

- T_1 tranzakció az (5) időegységben zárolja az A adategységet, majd a (9) időegységben fel is szabadítja azt. Ez a működés megfelel a legális ütemezés i) elvárásának.
- T_2 tranzakció a (3) időegységben zárolja a B adategységet, ám utána soha nem szabadítja azt fel. Ez a működés megsérti a legális ütemezés i) elvárását. Ez azonban könnyen kijavítható: T_2-nek fel kell szabadítania B-t, miután a (6) időegységben írta.
- T_3 tranzakció az (1) időegységben zárolja a B adategységet, a (4) időegységben pedig zárolja az A adategységet, majd (10)-ben felszabadítja B-t, (11)-ben pedig A-t. Ez a működés megfelel a legális ütemezés i) elvárásának.

A ii) feltétel vizsgálata során azt kell megvizsgálnunk, hogy a tranzakciók helyeznek-e el zárat olyan adategységre, amelyen abban az időpillanatban egy másik tranzakció már zárol (olyan zárral, amely nem kompatibilis).

- T_1 tranzakció (5)-ben helyez el egy olvasási zárat (RLOCK) A-n. Ekkor T_3-nak már van szintén olvasási zárja A-n (amit (4)-ben helyezett el rajta), de a két olvasási zár kompatibilis egymással, így ez a működés megfelel a legális ütemezés ii) elvárásának.
- T_2 tranzakció (3)-ban helyez el egy írási zárat (WLOCK) B-n. Ekkor T_3-nak már van olvasási zárja B-n (amit (1)-ben helyezett el rajta), ez a két zár azonban nem kompatibilis egymással, így ez a működés megsérti a legális ütemezés ii) elvárását. Ha azonban jobban megnézzük az ütemezést, láthatjuk, hogy a T_3 által B-n fenntartott RLOCK (2) után már teljesen felesleges, hiszen T_3 csak (2)-ben használja B-t. Vagyis ha T_3 a (2)-beli READ B után rögtön felszabadítja B-t (vagyis az eredetileg (10)-beli UNLOCK B-t előrébb hozza), T_2 már elhelyezheti a zárját B-n.
- T_3 tranzakció (1)-ben zárolja B-t, amikor még semelyik másik tranzakció nem tart fenn rajta zárat, majd (4)-ben zárolja A-t, amelyiken akkor szintén nem tart fenn zárat egyik tranzakció sem.

Az ütemezést tehát pl. a fent részletezett módosításokkal tudjuk legálissá tenni, ezeket mutatja az alábbi táblázat.

	T_1	T_2	T_3
(1)			RLOCK B
(2)			READ B
(3)		WLOCK B	**UNLOCK B**
(4)			RLOCK A
(5)	RLOCK A		
(6)		WRITE B	
(7)		**UNLOCK B**	READ A
(8)	READ A		
(9)	UNLOCK A		
(10)			~~UNLOCK B~~
(11)			UNLOCK A

A javított ütemezésünk táblázatos leírása akkor felel meg a megszokottaknak, ha minden sorban (időegységben) csak egy művelet történik. Ennek megfelelően a végleges javított ütemezést az alábbi táblázat mutatja.

	T_1	T_2	T_3
(1)			RLOCK B
(2)			READ B
(3)			UNLOCK B
(4)		WLOCK B	
(5)			RLOCK A
(6)	RLOCK A		
(7)		WRITE B	
(8)		UNLOCK B	
(9)			READ A
(10)	READ A		
(11)	UNLOCK A		
(12)			UNLOCK A

2. feladat

Ellenőrizd, hogy az alábbi ütemezés legális-e! Rajzold meg a sorosíthatósági gráfot, döntsd el, hogy sorosítható-e az ütemezés! Ha igen, adj egy soros ekvivalenst, ha nem, mutasd meg, miért nem! Hogyan nézne ki a gráf, ha egyszerű zármodellt használnánk?

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

(27)		RLOCK E		
(28)		UNLOCK E		
(29)				RLOCK E
(30)		UNLOCK B		
(31)	RLOCK F			
(32)		UNLOCK D		
(33)	UNLOCK F			
(34)				UNLOCK E

Elmélet

A legális ütemezés fogalmát már ismertettük az 1. feladatnál.

A *sorosíthatósági gráf* definíciója a jegyzet 10.5. fejezetéből: „Olyan irányított gráf, amelynek a csomópontjai a tranzakciók, egy élt pedig akkor rajzolunk a T_i csomópontból a T_j csomópont felé, ha van olyan A adategység, amelyen egy adott S ütemezésben a T_i tranzakció zárat helyezett el, majd a zár felszabadítása után először a T_j tranzakció helyez el zárat A-n."

Egy ütemezés pontosan akkor *sorosítható*, ha a sorosíthatósági gráf DAG (vagyis nincs benne irányított kör). Amennyiben a gráf DAG, létezik topologikus rendezése, amely egyben egy *soros ekvivalens* is.

Megoldás

Az ütemezés legalitásának megállapításához és a sorosíthatósági gráf megrajzolásához egyaránt segít, ha adategységenként időrendben kigyűjtjük, hogy melyik tranzakció milyen zárat, mikor helyez el rajtuk, és mikor szabadítja azt fel.

Adategység	Lock	Zártípus	Tranzakció	Unlock
A	(2)	R	T_1	(5)
	(8)	R	T_4	(10)
	(12)	W	T_4	(15)
	(17)	W	T_3	(26)
B	(6)	W	T_3	(11)
	(18)	W	T_2	(30)
C	(14)	R	T_1	(20)
	(23)	W	T_3	(24)
D	(3)	R	T_3	(4)
	(7)	R	T_1	(13)
	(21)	R	T_4	(22)
	(25)	W	T_2	(32)
E	(9)	W	T_1	(19)
	(27)	R	T_2	(28)
	(29)	R	T_4	(34)
F	(1)	R	T_2	(16)
	(31)	R	T_1	(33)

A fenti táblázat egy sora tehát egy adott tranzakció egy adott zárhasználatát írja le egy adott adategységen. Az első sor pl. azt írja le, hogy a T_1 tranzakció a (2) időegységben olvasásra zárolja az A adategységet, majd ezt a zárat az (5) időegységben felszabadítja.

Először vizsgáljuk meg, hogy az ütemezés legális-e. Mivel a táblázat minden sora teljes, vagyis minden sorban szerepel időegység a Lock és Unlock oszlopban is, minden lockhoz tartozik unlock, így a legalitás i) feltétele teljesül. A ii) feltétel lényegében azt jelenti, hogy egymással nem kompatibilis zárok időben

nem fedik át egymást, vagyis a [Lock, Unlock] intervallumaik diszjunktak. Ennek ellenőrzéséhez jelöljük az egymást követő, ugyanazon adategységre vonatkozó, egymással kompatibilis (vagyis R típusú) zárokat (továbbiakban zárcsoport) közös piros befoglaló körvonallal.

Adategység	Lock	Zártípus	Tranzakció	Unlock
A	(2)	R	T_1	(5)
	(8)	R	T_4	(10)
	(12)	W	T_4	(15)
	(17)	W	T_3	(26)
B	(6)	W	T_3	(11)
	(18)	W	T_2	(30)
C	(14)	R	T_1	(20)
	(23)	W	T_3	(24)
D	(3)	R	T_3	(4)
	(7)	R	T_1	(13)
	(21)	R	T_4	(22)
	(25)	W	T_2	(32)
E	(9)	W	T_1	(19)
	(27)	R	T_2	(28)
	(29)	R	T_4	(34)
F	(1)	R	T_2	(16)
	(31)	R	T_1	(33)

A legalitás ii) feltételét úgy vizsgálhatjuk, hogy adategységenként megvizsgáljuk, hogy az egymást követő zárok/zárcsoportok időintervallumai átfedik-e egymást. Pl. az A adategység esetében az első zárcsoport lockja (2), unlockja (10), az ezt követő zár lockja (12), unlockja (15), majd a következő zár lockja (17), unlockja (26). Ezek az értékek szigorúan monoton növekednek, vagyis a [Lock, Unlock] intervallumok ([2, 10], [12, 15], [17, 26]) diszjunktak. Ugyanezt megvizsgálva minden adategységre belátjuk, hogy a legális ütemezés ii) feltétele is teljesül, így az ütemezés legális.

A sorosíthatósági gráf megrajzolása következik, amelynek csúcsai tehát a tranzakciók, élt pedig (egyszerű zármodell esetén) akkor húzunk két tranzakció közé, ha az egyik zárjának feloldását követően a másik helyez el legközelebb zárat ugyanazon az adategységen. Vegyük észre, hogy ez megfelel annak, mintha a fenti táblázatban adategységenként az egymást követő sorok tranzakciói közé húznánk élt.

R/W modell esetén azonban az egy zárcsoportba tartozó RLOCK zárak tranzakciói közé nem kell élt húznunk a sorosíthatósági gráfban. Ennek az oka az, hogy valójában ezek az olvasási műveletek kezelhetők együtt, az egymáshoz viszonyított sorrendjük irreleváns: elegendő, ha a zárcsoportot megelőző utolsó zár és a zárcsoportot követő első zár között helyezkedik el a zárcsoport minden eleme. Ennek megfelelően kell az éleket is behúznunk: a zárcsoportot megelőző utolsó zár tranzakciójából a zárcsoport valamennyi tranzakciójába, valamint a zárcsoport valamennyi tranzakciójából a zárcsoportot követő első zár tranzakciójába.

Az alábbi táblázat mutatja a sorosíthatósági gráfba behúzandó éleket.

Adategység	Lock	Zártípus	Tranzakció	Unlock
A	(2)	R	T_1	(5)
	(8)	R	T_4	(10)
	(12)	W	T_4	(15)
	(17)	W	T_3	(26)
B	(6)	W	T_3	(11)
	(18)	W	T_2	(30)
C	(14)	R	T_1	(20)
	(23)	W	T_3	(24)
D	(3)	R	T_3	(4)
	(7)	R	T_1	(13)
	(21)	R	T_4	(22)
	(25)	W	T_2	(32)
E	(9)	W	T_1	(19)
	(27)	R	T_2	(28)
	(29)	R	T_4	(34)
F	(1)	R	T_2	(16)
	(31)	R	T_1	(33)

Az A adategység első zárcsoportja a T_1 és T_4 tranzakciók zárjából áll, ezt követi a T_4 tranzakció zárja. A zárcsoport tranzakcióiból kell a zárcsoportot követő zár tranzakciójába élt húzni, vagyis a $T_1 \rightarrow T_4$ és $T_4 \rightarrow T_4$ éleket kellene behúznunk, de utóbbinak nem lenne értelme, így azt természetesen nem húzzuk be. T_4 zárját T_3 zárja követi, így a $T_4 \rightarrow T_3$ élt is behúzzuk a sorosíthatósági gráfba.

A B adategységen mindössze egy-egy zárat helyez el T_3, majd T_2, így a $T_3 \rightarrow T_2$ élt húzzuk be a gráfba.

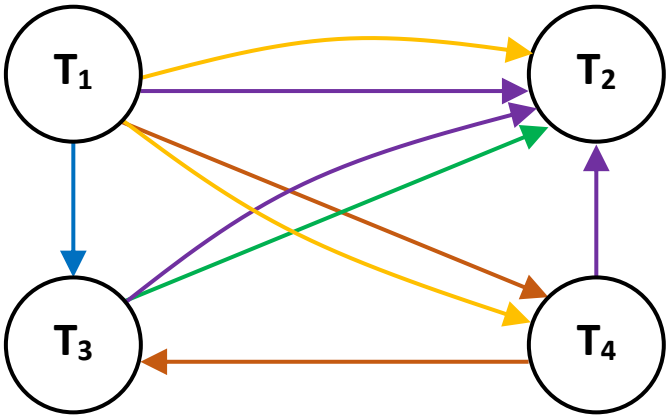
A C adategységen hasonlóan egy-egy zárat helyez el T_1, majd T_3, így a $T_1 \rightarrow T_3$ élt húzzuk be.

A D adategység első zárcsoportja a T_3, T_1 és T_4 tranzakciók zárjából áll, majd ezt követi a T_2 tranzakció zárja. Ennek megfelelően a $T_3 \rightarrow T_2$, $T_1 \rightarrow T_2$ és $T_4 \rightarrow T_2$ éleket húzzuk be.

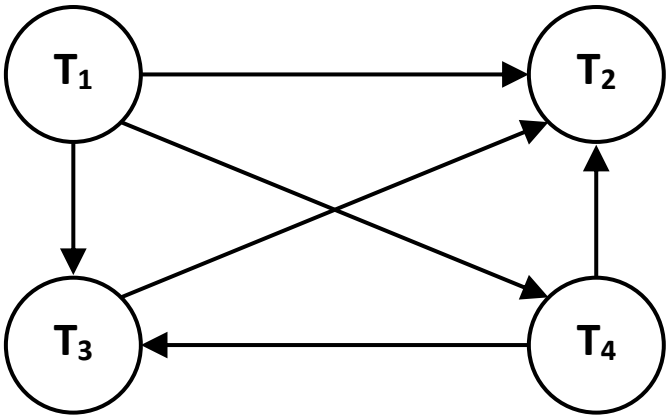
Az E adategység első zárját a T_1 tranzakció helyezi el, ezt követi a T_2 és T_4 tranzakciók zárcsoportja. A zárcsoportot megelőző utolsó zár tranzakciójából húzunk élt a zárcsoport összes tranzakciójába, így a $T_1 \rightarrow T_2$ és $T_1 \rightarrow T_4$ éleket húzzuk be.

Az F adategységnek mindössze egyetlen zárcsoportja van, így nem húzunk be új élt a gráfba.

Jelölje a sorosíthatósági gráfon az élek színe azt, hogy melyik adategység miatt húztuk be. Jelölje az A adategységet a barna, a B adategységet a zöld, a C adategységet a kék, a D adategységet a lila, az E adategységet a narancssárga, az F adategységet a magenta szín.



Az egyszerű sorosíthatósági gráfról könnyebben eldönthető, tartalmaz-e irányított kört.



Megállapítható, hogy a sorosíthatósági gráf nem tartalmaz irányított kört, vagyis az ütemezés sorosítható. Az ütemezés egy soros ekvivalense a sorosíthatósági gráf egy topologikus rendezése, aminek a jelen példában csak a $T_1 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2$ felel meg.

Kérdés még, hogy hogyan nézne ki a gráf, ha egyszerű zármodellt használnánk. Ebben az esetben a zárcsoportok megszűnnek, hiszen a zárakat nem különböztetjük meg, így egy adategységre minden egymást követő zár tranzakciója közé élt kell húzni. A sorosíthatósági gráf tehát a zárcsoportoknál változik. A táblázatban pontozott vonal jelöli azokat az éleket, amelyek kikerülnek a gráfból, vastag vonal pedig azokat, amelyek újonnan kerülnek bele.

Adategység	Lock	Zártípus	Tranzakció	Unlock
A	(2)	R	T_1	(5)
	(8)	R	T_4	(10)
	(12)	W	T_4	(15)
	(17)	W	T_3	(26)
B	(6)	W	T_3	(11)
	(18)	W	T_2	(30)
C	(14)	R	T_1	(20)
	(23)	W	T_3	(24)
D	(3)	R	T_3	(4)
	(7)	R	T_1	(13)
	(21)	R	T_4	(22)
	(25)	W	T_2	(32)
E	(9)	W	T_1	(19)
	(27)	R	T_2	(28)
	(29)	R	T_4	(34)
F	(1)	R	T_2	(16)
	(31)	R	T_1	(33)

Az A adategység RR ([2, 5], [8, 10]) zárcsoportja megszűnik, így a T1 és T4 tranzakciók benne lévő, közvetlenül egymást követő olvasási műveletei miatt új $T_1 \rightarrow T_4$ élt kell behúzni. A zárcsoportból induló (T_1, T_4) $\rightarrow T_4$ élek helyett pedig a zárcsoport utolsó eleme és az azt követő művelet miatt kellene élt húzni, ám ez egy $T_4 \rightarrow T_4$ él lenne, aminek továbbra sincs értelme. Összességében tehát egy $T_1 \rightarrow T_4$ él helyett húztunk be egy $T_1 \rightarrow T_4$ élt, vagyis a sorosíthatósági gráf nem változott, de a mögöttes logikát fontos érteni: a törölt $T_1 \rightarrow T_4$ él a [2, 5] olvasás és a [12, 15] írás közti viszony miatt szerepelt a gráfban, az új viszont a [2,5] és a [8, 10] olvasások közti viszony miatt került be.

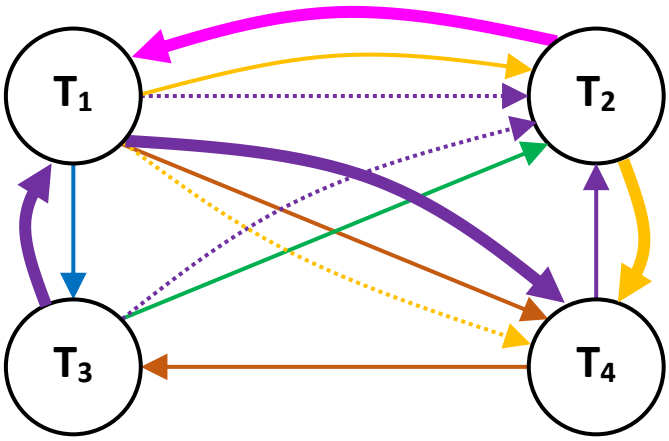
A B és C adategységek esetén eddig sem voltak zárcsoportok, így változás sincs.

A D adategység RRR ([3, 4], [7, 13], [21, 22]) zárcsoportja megszűnik, így a T_3, T_1 és T_4 tranzakciók közvetlenül egymást követő olvasási műveletei miatt új $T_3 \rightarrow T_1$ és $T_1 \rightarrow T_4$ élt kell behúznunk. A zárcsoportból induló (T_3, T_1, T_4) $\rightarrow T_2$ élek helyett pedig a zárcsoport utolsó eleme és az azt követő művelet miatt kell $T_4 \rightarrow T_2$ élt behúzni. Látható, hogy a $T_4 \rightarrow T_2$ él természetesen a (T_3, T_1, T_4) $\rightarrow T_2$ élek között is megtalálható. A három él törlése, majd az egyik visszahúzása természetesen ekvivalens azzal, mintha eleve csak kettőt törölnénk.

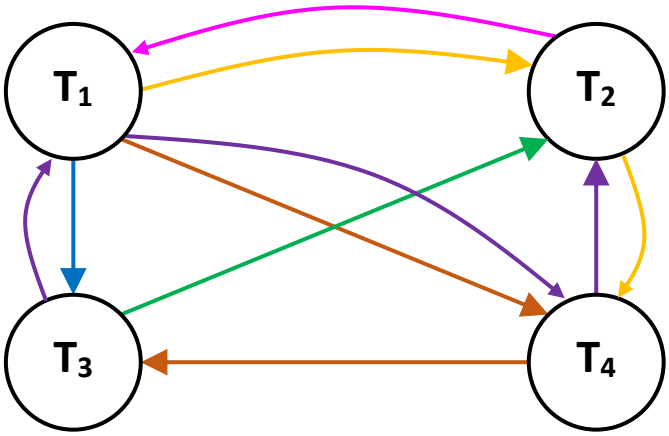
Az E adategység RR ([27, 28], [29, 34]) zárcsoportja megszűnik, így a T_2 és T_4 tranzakciók közvetlenül egymást követő olvasási műveletei miatt új $T_2 \rightarrow T_4$ élt kell behúznunk. A zárcsoportba vezető $T_1 \rightarrow$ (T_2, T_4) élek helyett pedig a zárcsoport első eleme és az azt megelőző művelet miatt kell $T_1 \rightarrow T_2$ élt behúzni. Az előző bekezdésben leírtakhoz hasonlóan természetesen a $T_1 \rightarrow T_2$ él itt is eleme a $T_1 \rightarrow$ (T_2, T_4) éleknek.

Az F adategység RR ([1, 16], [31, 33]) zárcsoportja megszűnik, így az egymást követő olvasási műveletei miatt új $T_2 \rightarrow T_1$ élt kell húzni a gráfba.

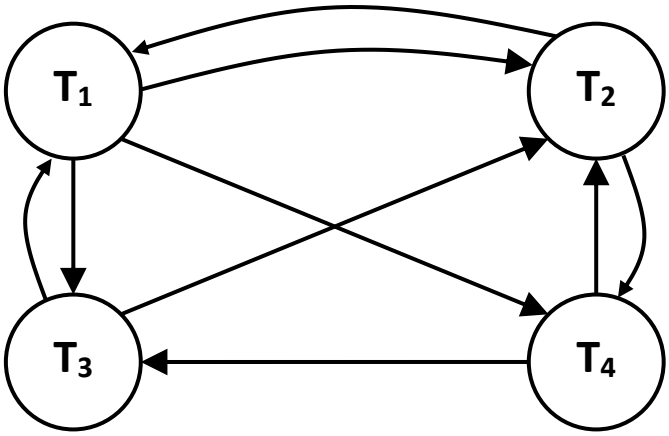
Az alábbi sorosíthatósági gráf mutatja a törlendő és újonnan beszúrandó éleket is, a táblázatban megismert jelöléssel (törlendő él pontozott, újonnan beszúrandó vastag). (Az A adategység okozta $T_1 \rightarrow T_4$ él törlendő, majd újra beszúrandó, ezért sima vonallal jelöljük.)



A módosításokat követően a sorosíthatósági gráf a következőképpen néz ki.



Az egyszerű sorosíthatósági gráfról még könnyebben eldönthető, hogy tartalmaz-e irányított kört.



Ez a sorosíthatósági gráf már láthatóan nem DAG (van benne irányított kör), vagyis az ütemezés nem sorosítható. Érezhető, hogy minél finomabb (precízebb) egy zármodell, annál valószínűbben lesz sorosítható az ütemezés (pl. ebben az esetben RW-vel sorosítható volt, de egyszerű zármodellel már nem).

3. feladat

Legális-e az alábbi táblázat szerinti ütemezés? A tranzakciók követik-e a 2PL-t? Hol van az alábbi tranzakciók zárpontja? Mi egy soros ekvivalens ütemezés?

	T_1	T_2	T_3
(1)	LOCK A		
(2)		LOCK B	
(3)			LOCK C
(4)			LOCK D
(5)	LOCK E		
(6)	UNLOCK A		
(7)			UNLOCK D
(8)		LOCK A	
(9)		LOCK D	
(10)	UNLOCK E		
(11)		UNLOCK B	
(12)			UNLOCK C
(13)		UNLOCK A	
(14)		UNLOCK D	

Elmélet

A *legalitást* már tárgyaltuk az 1. feladatnál.

A *2PL*-t (*kétfázisú protokollt*) követő tranzakciók működése két fázisra bontható: az első fázisban helyezik el a zárjaikat, a második fázisban pedig felszabadítják azokat, vagyis az utolsó zárolás megelőzi az első zárfeloldást. Egy 2PL tranzakció *zárpontjának* azt az időpillanatot hívjuk, amikor a tranzakció az utolsó zárját is megkapja (jegyzet 10.4.1. fejezet).

Tétel mondja ki, hogy ha egy legális ütemezés minden tranzakciója követi a 2PL-t, akkor az ütemezés sorosítható, a tranzakciók zárpont szerint időben növekvő rendezése pedig egy soros ekvivalenst ad.

Megoldás

Az ütemezés legalitását az 1. feladatnál megismert módon ellenőrizzük, és belátjuk, hogy mivel minden tranzakció felold minden általa elhelyezett zárat, valamint egyik adategységen sincsen egyszerre több zár, az ütemezés legális.

Minden tranzakcióról belátható, hogy követi a 2PL-t, hiszen az első UNLOCK utasítás után sehol sincsen már LOCK utasítás.

A tranzakciók zárpontja az utolsó LOCK utasításuk utáni időpillanat, vagyis T_1 esetén (5) után, T_2 esetén (9) után, T_3 esetén (4) után. Az alábbi táblázatban pirossal láthatók a zárpontok.

	T_1	**T_2**	**T_3**
(1)	LOCK A		
(2)		LOCK B	
(3)			LOCK C
(4)			LOCK D
(5)	LOCK E		
(6)	UNLOCK A		
(7)			UNLOCK D
(8)		LOCK A	
(9)		LOCK D	
(10)	UNLOCK E		
(11)		UNLOCK B	
(12)			UNLOCK C
(13)		UNLOCK A	
(14)		UNLOCK D	

Mivel az ütemezés legális, és minden tranzakció követi a 2PL-t, az ütemezés sorosítható. Egy soros ekvivalens ütemezés a zárpontok növekvő sorrendjéből adódik, vagyis $T_3 \rightarrow T_1 \rightarrow T_2$, hiszen a zárpontok (4) < (5) < (9). Megjegyzés: a sorosíthatósági gráf alapján megállapítható, hogy a $T_1 \rightarrow T_3 \rightarrow T_2$, is egy soros ekvivalens.

4. feladat

Időbélyeges tranzakciókezelést használunk R/W modellben. Jegyezd fel az alábbi sorozat minden művelete után az R(A), R(B), W(A), W(B) értékeit, ha kezdetben mindegyik 0. Mely tranzakciók abortálnak? r_i és w_i a T_i tranzakció olvasás (r) és írás műveleteit (w) jelöli, és $t(T_i) = i$.

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

Elmélet

Az *időbélyeg* olyan érték, amelyet minden tranzakcióhoz szigorú egyediséget biztosítva rendelünk hozzá, és amely arányos (legegyszerűbb esetben azonos) a tranzakció kezdőidejével (jegyzet 10.8. fejezet). Az ütemezés soros ekvivalense mindig a tranzakciók időbélyeg szerinti növekvő sorrendje, ami annak felel meg, mintha a tranzakciók a kezdőidejükben nulla idő alatt végre tudnának hajtódni. Ha egy művelet eredménye ezzel összhangban van, akkor végrehajtható, ha nincs, akkor az érintett tranzakció abortálandó.

R/W modellű időbélyeges tranzakciókezelés esetén minden X adategységről nyilvántartunk két időbélyeget: az adategység olvasási (R(X)) és írási (W(X)) idejét. Egy ilyen időbélyeg annak a tranzakciónak az időbélyege, amely legutóbb elvégezte az adott (olvasási vagy írási) műveletet az adategységen.

Az alábbi táblázat a jegyzet 10.9.1. fejezetében található, és tömören összefoglalja a működést T tranzakció és X adategység esetén, ahol t(T) a tranzakció időbélyegét, R(X) és W(X) pedig az adategység olvasási és írási időbélyegét jelölik.

		T olvasni akar	T írni akar
(1)	t(T) < R(X) t(T) < W(X)	**abort T** *(később indult tranzakció már írta)*	**abort T** *(később indult tranzakció már olvasta)*
(2)	t(T) < R(X) t(T) > W(X)	**T olvassa X-et, R(X) nem változik** *(később indult tranzakció már olvasta)*	**abort T** *(később indult tranzakció már olvasta)*
(3)	t(T) > R(X) t(T) < W(X)	**abort T** *(később indult tranzakció már írta)*	**abort T** *(később indult tranzakció már írta[1])*
(4)	t(T) > R(X) t(T) > W(X)	**T olvassa X-et** **R(A) := t(T)**	**T írja X-et** **W(X) := t(T)**

Megoldás

Foglaljuk táblázatba R(A), W(A), R(B) és W(B) értékeit, és figyeljük meg lépésenként a változásokat (a táblázatokban pirossal jelöljük az utolsó művelet okozta változásokat).

Kezdetben minden időbélyeg 0.

Művelet	R(A)	W(A)	R(B)	W(B)
–	**0**	**0**	**0**	**0**

Az első művelet $r_1(A)$, vagyis T_1 olvasná A-t. $t(T_1) = 1 > R(A)$. Mivel $W(A) = 0$, így $t(T_1) > W(A)$ ezért a táblázat (4) sorának olvasási oszlopa alapján T_1 olvassa A-t, R(A) pedig frissül 1-re.

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
$r_1(A)$	**1**	0	0	0

A következő művelet $r_2(B)$, vagyis T_2 olvasná B-t. $t(T_2) = 2 > R(B)$, $W(B) = 0$, ezért szintén a (4) alapján T_2 olvassa B-t, R(B) pedig frissül 2-re.

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
$r_1(A)$	1	0	0	0
$r_2(B)$	1	0	**2**	0

A következő művelet $r_1(B)$, vagyis T_1 olvasná B-t. $t(T_1) = 1 < R(B) = 2$, de $t(T_1) = 1 > W(B) = 0$, ezért a (2) olvasási oszlopa alapján T_1 olvassa B-t, R(B) értéke viszont változatlan marad.

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
$r_1(A)$	1	0	0	0
$r_2(B)$	1	0	2	0
$r_1(B)$	1	0	2	0

A következő művelet $w_3(B)$, vagyis T_3 írná B-t. $t(T_3) = 3 > R(B) = 2$, $W(B) = 0$, ezért a (4) írási oszlopa alapján T_3 írja B-t, W(B) pedig frissül 3-ra.

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0

[1] A konkurencia tovább növelhető a Thomas-féle írási szabállyal, lásd a jegyzet 67. feladatának megoldását.

$r_1(A)$	1	0	0	0
$r_2(B)$	1	0	2	0
$r_1(B)$	1	0	2	0
$w_3(B)$	1	0	2	**3**

A következő művelet $r_2(B)$, vagyis T_2 olvasná B-t. $t(T_2) = 2 < W(B) = 3$, vagyis egy később indult tranzakció (T_3) már írta B-t, így a (3) olvasási oszlopa alapján T_2-nek abortálnia kell. T_2 abortálása természetesen annak korábbi $r_2(B)$ műveletét is érinti. (T_2 abortálása után az R(B) időbélyeget visszaállíthatnánk a korábbi 0 értékre, de ez nem kötelező: azzal, hogy az időbélyeget nem csökkentjük, biztosan nem okozunk hibás működést, legfeljebb a feltétlenül szükségesnél több abortot kapunk.)

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
$r_1(A)$	1	0	0	0
~~**$r_2(B)$**~~	1	0	2	0
$r_1(B)$	1	0	2	0
$w_3(B)$	1	0	2	3
~~**$r_2(B)$**~~	1	0	2 **ABORT T_2**	3

A következő művelet $w_4(A)$, vagyis T_4 írná A-t. $t(T_4) = 4 > R(A) = 1$, $W(A) = 0$, ezért a (4) írási oszlopa alapján T_4 írja A-t, W(A) pedig frissül 4-ra.

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
$r_1(A)$	1	0	0	0
$r_2(B)$	1	0	2	0
$r_1(B)$	1	0	2	0
$w_3(B)$	1	0	2	3
$r_2(B)$	1	0	2 ABORT T_2	3
$w_4(A)$	1	**4**	2	3

A következő művelet $r_4(B)$, vagyis T_4 olvasná B-t. $t(T_4) = 4 > R(B) = 2$, $W(B) = 3$, ezért a (4) olvasási oszlopa alapján T_4 olvassa B-t, R(B) pedig frissül 4-re.

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
$r_1(A)$	1	0	0	0
$r_2(B)$	1	0	2	0
$r_1(B)$	1	0	2	0
$w_3(B)$	1	0	2	3
$r_2(B)$	1	0	2 ABORT T_2	3
$w_4(A)$	1	4	2	3
$r_4(B)$	1	4	**4**	3

A következő művelet $w_1(A)$, vagyis T_1 írná A-t. $t(T_1) = 1 < W(A) = 4$, vagyis egy később indult tranzakció (T_4) már írta A-t, így a (3) írási oszlopa miatt T_1-nek abortálnia kell.[2] T_1 abortálása természetesen annak korábbi $r_1(A)$ és $r_1(B)$ műveleteit is érinti. (Továbbá az R(A)=1 időbélyeget opcionálisan visszaállíthatjuk

[2] A Thomas-féle írási szabály használata esetén, amennyiben T_4 már commitolt, ez az abort elkerülhető.

0-ra – de az R(B) időbélyeget nem, hiszen az már nagyobb, mint t(T_1), vagyis már egy T_1-nél később indult tranzakció is olvasta B-t.)

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
~~$r_1(A)$~~	1	0	0	0
$r_2(B)$	1	0	2	0
~~$r_1(B)$~~	1	0	2	0
$w_3(B)$	1	0	2	3
$r_2(B)$	1	0	2 ABORT T_2	3
$w_4(A)$	1	4	2	3
$r_4(B)$	1	4	4	3
~~$w_1(A)$~~	1	4 **ABORT T_1**	4	3

Az utolsó művelet $w_3(B)$, vagyis T_3 írná B-t. t(T_3) = 3 < R(B) = 4, vagyis egy később indult tranzakció (T_4) már olvasta B-t, így a (2) írási oszlopa miatt T_3-nak abortálnia kell. T_3 abortálása természetesen annak korábbi $w_3(B)$ műveletét is érinti. Az abortálandó T_3 korábbi $w_3(B)$ művelete után azonban T_4 is kiolvasta B T_3 által módosított értékét, így T_4-nek is abortálnia kell (piszkos adat olvasása, lásd piros nyíl).

Művelet	R(A)	W(A)	R(B)	W(B)
–	0	0	0	0
$r_1(A)$	1	0	0	0
$r_2(B)$	1	0	2	0
$r_1(B)$	1	0	2	0
~~$w_3(B)$~~	1	0	2	3
$r_2(B)$	1	0	2 ABORT T_2	3
~~$w_4(A)$~~	1	4	2	3
~~$r_4(B)$~~	1	4	4	3
$w_1(A)$	1	4 ABORT T_1	4	3
~~$w_3(B)$~~	1	4	4	3 **ABORT T_3** **ABORT T_4**

Végül tehát mind a négy tranzakció abortál.

Egy tranzakció abortálása után az időbélyegek visszaállításával további felesleges abortok elkerülhetők lehetnek, de ennek hiánya sem okoz hibás működést, hiszen azzal, hogy egy időbélyeg értékét a szükségesnél nagyobbon hagyjuk, biztosan nem engedünk át olyan tranzakciót, amit nem szabadna, legfeljebb abortálunk olyat, amit nem lenne muszáj.

5. feladat

Oldd meg a 4. feladatot verziókezeléssel kiegészítve! Most mi történik?

Elmélet

Időbélyegek melletti verziókezelés (jegyzet 10.8.4. fejezet) esetén azt feltételezzük, hogy minden adategység írásakor megőrizzük annak korábbi értékét is, így egy adott időpillanatban egy adategységnek nemcsak a legutóbbi (utoljára írt) értékét tudjuk olvasni, hanem a korábbiakat is, ami elkerülhetővé teszi az olvasás miatti abortokat.

Egy adategység írásakor tehát nem felülírjuk az előző értéket, hanem egy újat hozunk létre.

Megoldás

Foglaljuk táblázatba az adategységeket és időbélyegeket. Minden oszlop egy adategység-verziót jelöl (növekvő számozással), a cellákban pedig az adott sorú műveletet követő olvasási (R) és írási (W) időbélyegek szerepelnek R/W formátumban. A kihúzott (–) cellák olyan adategység-verzióhoz tartoznak, amelyek az adott műveletkor még nem léteznek. Pirossal jelöljük az adott művelet által módosított értékeket.

Művelet	**A0**	**A1**	**A2**	**B0**	**B1**
–	0/0	–	–	0/0	–
$r_1(A)$	**1**/0	–	–	0/0	–
$r_2(B)$	1/0	–	–	**2**/0	–
$r_1(B)$	1/0	–	–	2/0	–
~~$w_3(B)$~~	1/0	–	–	2/0	**2/3**
$r_2(B)$	1/0	–	–	2/0	2/3
~~$w_4(A)$~~	1/0	**1/4**	–	2/0	2/3
~~$r_4(B)$~~	1/0	1/4	–	2/0	**4**/3
$w_1(A)$	1/0	1/4	**1/1**	2/0	4/3
~~$w_3(B)$~~	**ABORT T_3** **ABORT T_4**				

A táblázatban látható, hogy minden írás művelet után létrejön egy új adategység-verzió.

Látható, hogy a 4. feladatban abortot okozó $r_2(B)$ utasítás itt nem okoz abortot, hiszen T_2 ki tudja olvasni B értékét B0-ból, amelynek olvasási időbélyege R(B0) = 2, írási időbélyege pedig W(B0) = 0, így $t(T_2) = 2$ esetén megfelelő.

Hasonlóan a $w_1(A)$ utasítás sem okoz abortot (bár ez az abort a Thomas-féle írási szabállyal is elkerülhető lett volna), hanem új adategység-verziót hoz létre.

Az utolsó $w_3(B)$ utasítás viszont itt is abortot okoz, hiszen T_4 már olvasta B-t (B1-et), így T_3 nem írhatja. A T_3 által a $w_3(B)$ utasítással írt B1-et olvasta ezután T_4 is az $r_4(B)$ utasítással, így T_4-et is abortálni kell (piszkos adat olvasása, lásd piros nyíl).

Verziókezelés esetén tehát csak a T_3 és T_4 tranzakciók abortálnak.

6. feladat

Egy rendszerleállás után a napló vége az 5. táblázat szerinti bejegyzéseket tartalmazza. Melyek a redo helyreállítás lépései? Mi lesz a helyreállítás után A, B és C értéke?

checkpoint
(T_1, begin)
(T_2, begin)
(T_2, A, 20)
(T_2, B, 10)
(T_1, A, 2)
(T_3, begin)
(T_1, C, 5)
(T_1, commit)
(T_3, C, 6)
(T_3, commit)

Elmélet

A *redo helyreállítás* (jegyzet 10.7.2.2. fejezet) az adatbázist a helyreállítás végére egy konzisztens állapotba viszi. Lépései:

1. Az összes zár felszabadítása
2. A napló vizsgálata visszafelé: (T_i, commit) bejegyzésű T_i tranzakciók feljegyzése
3. Haladás visszafelé a naplóban, amíg konzisztens állapotot (checkpoint) nem találunk
4. A megtalált (T_i, commit) bejegyzésű T_i tranzakciókra vonatkozó bejegyzések segítségével felülírjuk az adatbázisban található értékeket

Megoldás

A redo helyreállítás első lépéseként felszabadítjuk az összes zárat. Ezután visszafelé vizsgáljuk a naplót, amíg checkpoint bejegyzést nem találunk, és feljegyezzük a (T_i, commit) bejegyzésű tranzakciókat.

A checkpoint bejegyzésig megtalált (T_i, commit) bejegyzések visszafelé haladva (T_3, commit) és (T_1, commit), vagyis a feljegyzett tranzakciók T_3 és T_1.

A commitolt tranzakciók (tranzakció, adategység, érték) bejegyzései időrendben: (T_1, A, 2), (T_1, C, 5), (T_3, C, 6)

Vagyis a helyreállítás után A értéke 2 lesz (hiszen az egyetlen A-ra vonatkozó megfelelő bejegyzésben a 2 érték szerepel), B értékét nem tudjuk helyreállítani (hiszen nincs rá vonatkozó megfelelő bejegyzés, C értéke pedig 6 lesz (hiszen a (T_3, C, 6) bejegyzés újabb, mint a (T_1, C, 5)).