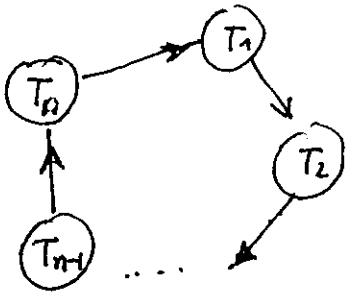
תרגיל 8 פתרונות

1. נניח שאפשר להמחיש את ה schedule הנתון ע"י cycle הכולל את טרנזקציות $(T_1, T_2 \dots T_n)$ (ע"פ סדרתן לאחר הגעתן T_n...T_2, T_1)



T_2 יכולה להשיג אחסון על פרט מסוים רק לאחר ש T_1 תשחרר אותו.
T_3 יכולה להשיג אחסון על פרט כלשהו רק לאחר ש T_2 תשחרר אותו וכו'.

במלים אחרות משמעות ה Schedule, אם כל טרנזקציה מקיימת את פרוטוקול Two-phase locking היא:

T_2 יכולה לסיים את שלב הצמיחה (ולקבל את ה lock) רק כאשר T_1 נכנסת לשלב ההתכווצות.
T_3 יכולה לסיים את שלב הצמיחה רק כאשר T_2 נכנסת לשלב ההתכווצות.
⋮
T_n יכולה לסיים את שלב הצמיחה רק כאשר T_{n-1} נכנסת לשלב ההתכווצות
T_1 " " " " " " " " T_n " " " .

משמעות הדבר היא, למשל ש T_1 יכולה לסיים את שלב הצמיחה רק כאשר T_1 נכנסת לשלב ההתכווצות! סתירה, משמעותה deadlock.
כלומר, לפחות אחד ה deadlock יגרום לאחת הטרנזקציות ל roll-back ויפר את ה cycle הנתון.

2. א. אפשרי הביצוע $\langle T_0, T_1 \rangle$ נקי
" " $\langle T_1, T_0 \rangle$ "

ב. נסתכל ב schedule הבא:

T_0	T_1
read(A)	
	read(B);
read(B)	
	read(A);
if A==0 then B=B+1;	
write(B);	
	if B==0 then A=A+1;
	write(A);

תוצאות אפשריות: $\begin{cases} A=0 & B=1 \\ A=1 & B=0 \end{cases}$ אין מתקיים

2/3

Schedule ב. הינו non-serializable כיון שלפי גרף "קדימויות" קדימות: read של T_1 "דורך" על write של T_0!

הינו מאחר שאין הבדל באלגוריתם לפי איזה סדר $A=1$ $B=1$.

ג. לפי קיום schedule בו-זמני של T_0 ; T_1 serializable הינו כיון שגרף ה-schedule הסדרתי: $\langle T_0, T_1 \rangle$ הקדימות: read(B) של T_1 תבוא אחרי write(B) של T_0 ולפי נתון לנקיים אחרת. כלומר אין ה-schedule הסדרתי $\langle T_1, T_0 \rangle$.

3. באופן עקרוני קיימת דרך שקולות לכתוב schedules הסדרתיים $\langle T_0, T_1, T_2, T_3, T_4 \rangle$ או $\langle T_0, T_1, T_3, T_2, T_4 \rangle$

(שהם שקולים ביניהם (conflict equivalent)

ש.ב.ל

T_0'

```
slock (A);
 read(A);
xlock (B);
 read(B);
 If A==0 then B=B+1;
 write(B);
 unlock(A);
 unlock(B);
```

T_1'

```
slock (B);
 read(B);
xlock (A);
 read(A);
 if B==0 then A=A+1;
 write(A)
 unlock(B);
 unlock(A);
```

ד: דוגמא כי נתן של טרנזקציות שלפי שיטה זו deadlock.

T_0' T_1'

```
slock (A);
 read(A);
                slock (B);
                 read(B);
xlock (B);
 WAIT           xlock (A);
                 WAIT
```

→ dead-lock

3. פרוטוקול 2PL

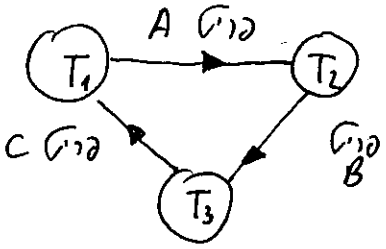
T_0	T_1
slock(A)	
read(A)	
xlock(B)	
read(B)	
	slock(B) ←
if A==0 then...	WAIT
write(B)	⋮
unlock(A)	⋮
unlock(B)	⋮
	read(B)
	xlock(A)
	read(A)
	⋮
	unlock(A)

נעילה מן סוג זה יכולה להתבצע כיוון ש T_0 מחזיקה נעילה בלעדית על B. T_1 תיכנס למצב המתנה עד T_0.

כאשר T_1 יכולה לנעול את B והמידע תואם ל T_1 (כיון ש T_0 סיימה).

התקבל תצמון סדרתי T_0 ואחרי T_1.

5. ציור גרף הקדימויות של התצמון: הגרף כולל מעגל ולכן סיגנו conflict-serializable.



6. T_1 שינה ב 2PL כיון שפעולת xlock(z) מתבצעת לאחר שקודמו פעולת unlock.
תצמון שאינו conflict serializable:

T_1	T_2	T_3
slock(X)		
⋮		
read(Y)		
⋮		
unlock(Y)		
	slock(X)	
	⋮	
	read(Z)	
	write(Y)	
	⋮	
	unlock(Z)	
xlock(Z)		
Z=t		
write(Z)		
unlock(Z)		
		xlock(W)
		⋮
		unlock(W)

א. יש להסביר את פעולת הנעילה xlock(Z) לפני שתי פעולות השחרור.

למשל:

T_1
slock(X)
⋮
t=X+Y
xlock(Z)
unlock(X)
unlock(Y)
⋮
unlock(Z)

דף תרגיל 8 - המשך פתרון

7. א. התנאי הנתון S_0 הוא view serializable כיוון שהוא שקול תצ"ס (view equivalent) לתצמ"ז הסדרתי S_1 הבא: $\overrightarrow{T_0, T_1, T_2}$.

צורת מכיוון שפריט מידע A (היתכן בו מטפל התצמ"ז) מק"מ:

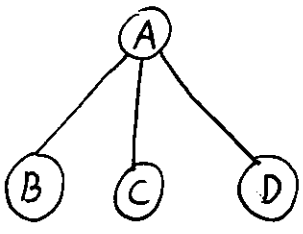
- אם ב S_0 ואם ב S_1 הטרנזקציה T_0 קוראת את ערכו ההתחלתי של A.
- אם ב S_0 ואם ב S_1 " T_2 קוראת ערך בינימי של A, שנכתב ע"י הטרנזקציה T_1.
- אם ב S_0 ואם ב S_1 הטרנזקציה T_2 כותבת את ערכו הסופי של A.

ב. 1. S_0 אינו יכול להתקבל ע"י 2PL כיוון שאינו conflict serializable (גרף הקדימויות מכיל מעגל מכיוון $T_0 \leftrightarrow T_1$ וכן מעגל $T_0 \leftrightarrow T_2$)

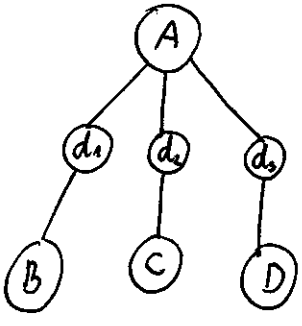
2. S_0 אינו יכול להתקבל ע"י פרוטוקול תצ"מ הבסיסי. בשורה 4 הטרנזקציה T_0 מנסה לבצע write(A) ע"י T_0 אינו יכולה להתבצע ו T_0 תתגלגל אחורנית.

3. כמו 2.

4. S_0 אינו יכול להתקבל ע"י פרוטוקול העץ כיוון שפרוטוקול העץ מבוסס נעילות ויכול ליצור רק תצמ"ז שהיא conflict serializable. שמא לך ש S_0 אינו conflict serializable (רמז: 1)



8. נסתכל לדוגמה על טרנזקציה T_1 הכותבת ל B ואח"כ ל C.
בצורה החסכונית ביותר T_1 תבצע נעילות בסדר הבא:
lock(A);lock(B);unlock(B);lock(C); unlock(A);unlock(C);
טרנזקציה אחרת T_2 הרוצה לכתוב ל A ול D חייבת לנעול ראשית את A אבל מכיוון שT_1 תשחרר אותו רק לקראת הסוף, ביצוע הטרנזקציות יהיה כמעט סדרתי (או סדרתי ממש אם השחרור של A יהיה אחרי השחרור של C.)



במבנה המוצע, הנעילות של T_1 יהיו
lock(A);lock(d1); lock(d2);unlock(A);lock(B);unlock(B);
;unlock(d1); lock(C); unlock(C); unlock(d2);
הטרנזקציה T2 תוכל להתבצע בו זמנית עם T1.

לסיכום, במקום להחזיק צומת אב, מספיק להחזיק את צומתי הדמה שמתחתיו, שאינם מייצגים פריטי מידע אמיתיים.

9. יהי S_0 סכ"ס הנתון שקיבץ את כל הפעולות של פרוטוקול נעילת-2ף הדף שהוצגה בכיתה (ובה סדר הקדמה ראה [illegible]).

הצמן S_1 ! S_0 שהשיטה כ 2PL אך לא פרוטוקול נעילת הדף.

S_0

T_0	T_1
write(C)	
	wrede(B)
write(A)	
	write(C)

S_1

T_0	T_1
read(A)	
	read(B)
read(B)	
	read(A)

שים לב: אמנם ב S_1 יש קדימות קריאה ולכן ב 2PL נשתמש בנעילה משתפת; אך בפרוטוקול נעילת-הדף שתוצג יש קדימות לנעילת הבלעדית!