תרגיל 6 : Transaction Management

תאריך הגשה: 23:55, 17.01.21.

הוראות הגשה:
בתרגיל זה אתם נדרשים להגיש קובץ zip בודד שיכלול את הקבצים הבאים:

- ex6.pdf עם התשובות מפורטות לשאלות.
- README שמכיל שורה בודדת ובו ה-login של הסטודנט שמגיש את התרגיל. אם התרגיל מוגש בזוגות, על שורה זאת להכיל את שני ה-login מופרדים בפסיק.

שימו לב:

- נא לקרוא על הדרישות המנהליות של הקורס בלינק באתר הקורס כדי למלא אחר ההוראות להגשה של קבצים סרוקים!
- תרגיל מוקלד יזכה ב- 2 נקודות בונוס!

<u>על מנת להקל על הבדיקה של התרגיל הזה, אתם מתבקשים לענות עליו בגוף התרגיל עצמו. זאת גם הסיבה שהתרגיל נראה כל כך ארוך (למרות שמבחינת השאלות הוא אינו ארוך).</u>

<u>שאלה 1:</u> (36 נקודות)

נתון התזמון:

	T1	T2	T3
1	R(X)		
2	R(Y)		
3		R(X)	
4			R(Z)
5		R(Y)	
6		R(Z)	
7	W(Y)		
8	W(X)		
9		Commit	
10			R(X)
11	R(V)		
12	Commit		
13			W(Z)
14			Commit

ענה על השאלות הבאות, ונמק בקצרה את תשובתך.

1. כמה קריאות מלוכלכות (dirty reads) יש בתזמון? 1

נימוק:
נבחין כי dirty read מתקיים כאשר טרנזקציה קוראת ערך שנכתב על ידי טרנזקציה אחרת שעוד לא ביצעה commit. במקרה זה T1 כותבת ערך ל X בשורה 8. ולפני שT1 עשתה commit בשורה 12 , הטרנזקציה T3 קוראת את הערך של X בשורה 10, לכן יש dirty read יחיד.

2. כמה קריאות שלא ניתנות לשחזור (nonrepeatable reads) יש בתזמון? 0

נימוק: נבחין כי non repeatable read מתקיים כאשר טרנזקציה קוראת ערך שהיא כבר קראה קודם לכן, והיא מגלה שהערך השתנה, למרות שהיא עצמה לא שינתה אותו.
ראשית נבדוק האם יש קריאות חוזרות של טרנזקציות:

- T1 קוראת את את הערכים X,Y,V כל אחד פעם אחת
- T2 קוראת את את הערכים X,Y,Z כל אחד פעם אחת
- T3 קוראת את הערכים X,Z כל אחד פעם אחת

בס"הכ אף טרנזקציה לא קוראת ערך פעמיים ולכן אין non repeatable reads

3. האם התזמון נמנע מ-cascading aborts? הקיף את התשובה הנכונה: כן לא

נימוק: נבחין כי תזמון נמנע מ cascading aborts אם הטרנזקציות קוראות רק שינויים של טרנזקציות שעשו commit. אולם בשורה 10, T3 קוראת את הערך של X ששונה על ידי T1 בשורה 8, וזאת לפני שT1 עשתה commit בשורה 12.

4. האם התזמון הוא בר- התאוששות (recoverable)? כן לא

נימוק:

נאמר כי תיזמון הינו בר התאוששות אם טרנזקציות שביצעו שינויים עשו commit לפני שהטרנזקציות אשר קראו את השינויים , עשו commit. כעת, מכיוון שT1 ביצע שינוי ו T3 קרא אותו נטען כי T1 צריך לעשות commit לפני T3 - וזה אכן המצב. T1 עושה commit בשורה 12 ו T3 עושה commit אחריו בשורה 14.

5. האם התזמון בר סידור קונפליקטים (conflict serializable)? כן לא

נסמן את הקונפליקטים בטבלה בצבעים תואמים :

	T1	T2	T3
1	(R(X		
2	(R(Y		
3		(R(X	
4			(R(Z
5		R(Y)	
6		(R(Z	
7	(W(Y		
8	(W(X		
9		Commit	
10			(R(X
11	(R(V		
12	Commit		
13			(W(Z
14			Commit

נימוק:

בהרצאה למדנו כי ניתן לבדוק האם תיזמון הוא בר סידור קונפליקטים באופן הבא :

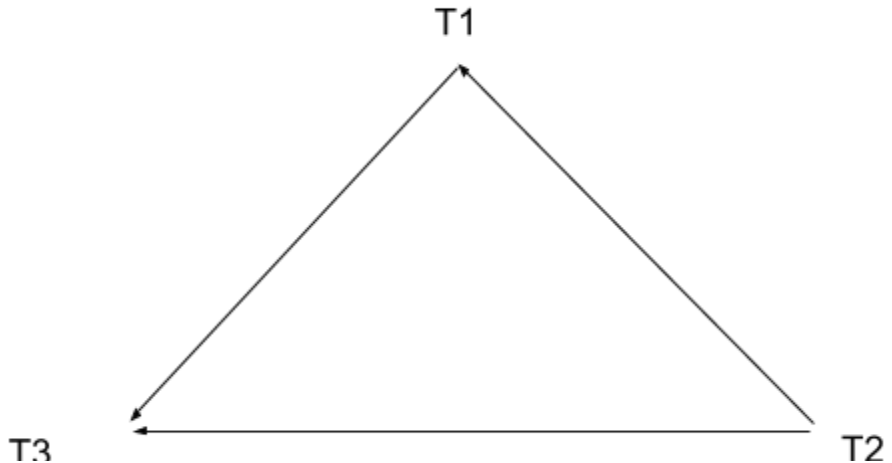
- יצירת גרף קדימויות עם קודקוד לכל טרנזקציה
- הוספת צלע מTi ל Tj אם קיים קונפליקט בינהם, כאשר Ti הוא הראשון
- אם לא קיים בגרף מעגל אז נטען כי התזמון הינו בר סידור קונפליקטים

כעת נחפש בגרף קונפליקטים מסוג WR , RW, WW. נבחין כי קיים קונפליקט מסוג RW בין T2 ל T1, כמו כן קיים קונפליקט מסוג RW בין T2 ל T3 , וקיים קונפליקט מסוג WR בין T1 ל T3.

בס"הכ העברנו צלע מ T2 ל T1 ו מT1 לT3 ולכן התיזמון conflict equivalent ל T2T1T3.

כמו כן, אין מעגל בגרף ומכאן שהתזמון הינו בר סידור קונפליקטים.

הערה : במייל שנשלח מהמרצה נאמר כי : "when determining if a conflict appears, we should not take into consideration whether or not a commit has occurred." ולכן קיים קונפליקט בין T2 ל T3 למרות ש T2 עשה commit.



6. האם התזמון יכול להיווצר על ידי פרוטוקול 2PL? כן לא

נימוק:

כפי שלמדנו בהרצאה תיזמון יכול להיווצר על ידי פרוטוקול 2PL אם יכולות להיות בקשות למנעולים, ושיחרורים של מנעולים, כך שהפעולות יתבצעו בסדר שבו הן כתובות בתיזמון.

נוסיף לטבלה הנתונה בקשות למנעולים באופן הבא :

- טרנזקציה שרוצה לקרוא אובייקט תבקש מנעול משותף עליו
- טרנזקציה שרוצה לכתוב אובייקט תבקש מנעול אקסקלוסיבי עליו
- טרנזקציה ששיחררה מנעול, לא תוכל לבקש מנעולים נוספים.

כאשר מתקיים כי

S(A) acquire a shared lock on A
X(A) acquire an exclusive lock on A
U(A) unlock A

ואכן על פי הטבלה שמוצגת מתחת, הפעולות מתבצעות באותו סדר שהן כתובות בתיזמון.

	T1	T2	T3
	S(X)		
1	(R(X		
2	S(Y)		
3	(R(Y		
4		S(X)	
5		(R(X	
6			S(Z)
7			(R(Z
8		S(Y)	
9		(R(Y	
10		S(Z)	
11		(R(Z	
12		U(X)	
13		U(Y)	
14		U(Z)	
15	X(Y)		
16	(W(Y		
17	X(X)		
18	(W(X		
19	S(V)		
20	U(Y)		
21	U(X)		

22		Commit	
23			S(X)
24			(R(X
25	(R(V		
26	U(V)		
27	Commit		
28			X(Z)
29			(W(Z
30			U(X)
31			U(Z)
32			Commit

7. האם התזמון יכול להיווצר על ידי פרוטוקול strict 2PL? כן **לא**

<u>נימוק:</u> למדנו בהרצאה שההבדל בין 2PL ל strict 2PL מתבטא בכך שטרנזקציה תשחרר את כל המנעולים כאשר היא תסיים.
במקרה זה בשורה 20 ,T1 צריך לשחרר את המנעול האקסקלוסיבי על X - וזה קורה לפני ש T1 מסיים.
הסיבה לכך ש T1 צריך לשחרר את המנעול היא שלא ניתן להחזיק במנעול אקסקלוסיבי על אובייקט אם טרנזקציה אחרת מקבלת מנעול על אותו אובייקט. במקרה זה בשורה 21 T3 מבקשת מנעול משותף על אובייקט X, ולכן T1 נאלצת לשחרר את המנעול האקסקלוסיבי על X.
ניתן לראות זאת בטבלה שמוצגת למטה.

	T1	T2	T3
	S(X)		
1	(R(X		
2	S(Y)		
3	(R(Y		
4		S(X)	
5		(R(X	
6			S(Z)
7			(R(Z
8		S(Y)	
9		(R(Y	
10		S(Z)	
11		(R(Z	
12		U(X)	
13		U(Y)	
14		U(Z) , Commit	
15	X(Y)		
16	(W(Y		
17	X(X)		

18	(W(X		
19	S(V)		
20	U(X)		
21			S(X)
22			(R(X
23	(R(V		
25	U(V)		
26	U(Y) , commit		
27			X(Z)
28			(W(Z
29			U(X)
30			U(Z),commit

8. האם התזמון יכול להיווצר על ידי פרוטוקול חותמות הזמן כאשר

?TS(T1) = 1, TS(T2) = 2, TS(T3) = 3

הקיף את התשובה הנכונה: כן לא

אם ענית לא, באיזה שורה הפרוטוקול ייכשל? 7

9. האם התזמון יכול להיווצר על ידי פרוטוקול חותמות הזמן כאשר

?TS(T1) = 2, TS(T2) = 1, TS(T3) = 3

הקיף את התשובה הנכונה: כן לא

אם ענית לא, באיזה שורה הפרוטוקול ייכשל?

שאלה 2 (30 נקודות)

1. תן דוגמה *קצרה* של תזמון שהוא בר סידור קונפליקטים אך אינו ניתן להשגה על ידי 2PL.

רמז: אפשר למצוא תזמון שמקיים את הדרישה שיש בו 3 טרנזקציות, 2 פעולות קריאה ו2 פעולות כתיבה בסה"כ. מלאו את הטבלה הבאה עם הפתרון שלכם.

פיתרון :

T1	T2	T3
		R(B)
	W(B)	
R(A)		
		W(A)

א. נטען כי התזמון אינו בר השגה על ידי 2PL :
נבנה טבלה באופן הבא :

- טרנזקציה T3 מעוניינת לקרוא את אובייקט B ולכן מבקשת עליו מנעול משותף.
- טרנזקציה T2 מעוניינת לכתוב את אובייקט B ולכן מבקשת עליו מנעול אקסקלוסיבי , כדי שתוכל לקבל אותו טרנזקציה T3 צריכה לשחרר את המנעול על B.
- לפני ש T3 משחררת את B היא מבקשת מפתח אקסקלוסיבי על A כדי שתוכל לקרוא אותו מאוחר יותר.
- טרנזקציה T2 מסיימת לכתוב את B ומשחררת את המנעול על B.
- טרנזקציה T1 מעוניינת לקרוא את A ולכן מבקשת מנעול משותף עליו, כדי שתוכל לקבל אותו טרנזקציה T3 צריכה לשחרר את המנעול על A.
- טרנזקציה T3 משחררת את המנעול על A.
- טרנזקציה T1 מסיימת לקרוא את A ומשחררת את המנעול על A.
- כעת טרנזקציה T3 מעוניינת לכתוב לA ולשם כך היא צריכה לבקש מפתח אקסקלוסיבי עליו, אולם היא שיחררה מפתח ולכן לא תוכל לעשות זאת.

T1	T2	T3
		S(B)
		R(B)
		X(A)
		U(B)
	X(B)	
	W(B)	
	U(B)	
		U(A)
S(A)		
R(A)		
U(A)		

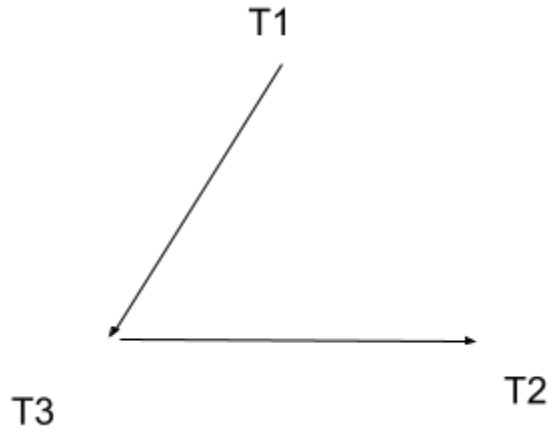
במידה ו T3 יכתוב את A ואז ישחרר אותו כך ש T1 תוכל לקרוא אותו, הסדר עם המנעולים לא יהיה תואם לתזמון המקורי , ולכן לא ניתן להשיג את התזמון על ידי 2PL. על פי הסדר המקורי של התזמון T1 היה אמור לקרוא את הערך של A לפני ש T3 שינה אותו, הטבלה הבאה ממחישה זאת :

T1	T2	T3
		S(B)
		R(B)
		X(A)
		U(B)
	X(B)	
	W(B)	
	U(B)	
		W(A)
		U(A)
S(A)		
R(A)		
U(A)		

ב. נטען כי התזמון הוא בר סידור קונפליקטים, שכן בגרף אין מעגל.

אופן בניית הגרף :

- קיים קונפליקט מסוג RW בין T3 ל T2 ולכן נמתח צלע בין T3 ל T2.
- קיים קונפליקט מסוג RW בין T1 ל T3 ולכן נמתח צלע בין T1 ל T3.



בס"הכ העברנו צלע מ T1 ל T3 ו מT3 לT2 ולכן התזמון conflict equivalent ל T1T3T2.
ואכן אם נריץ קודם את T1 , אחר כך את T3 ובסוף את T2 נקבל :

T1	T2	T3
R(A)		
		R(B)
		W(A)
	W(B)	

כך שכל זוג קונפליקטים מופיע בסדר שהיה בתזמון המקורי.

2. הוכח שכל תזמנון שיש בו 2 טרנזקציות בלבד הוא בר סידור קונפליקטים אם ורק אם ניתן להשיג את התזמון על ידי 2PL.

<u>פיתרון</u> :

◄ עבור הכיוון הראשון -
בהרצאה הראנו כי כל תיזמון שניתן להשיג על ידי 2PL הוא בר סידור קונפליקטים ובפרט תזמון שיש בו שתי טרנזקציות.

◄ עבור הכיוון השני -

יהיו T1,T2 טרנזקציות של תזמון S
נתון כי S הוא בר סידור קונפליקטים ולכן בגרף הקדימויות שמכיל את הטרנזקציות T1,T2 אין מעגל.
כלומר רק אחת מהאפשרויות הבאות מתקיימת:

- ◆ קיימת צלע בגרף בין T1 לT2 והתזמון conflict equivalent ל T1T2.
- ◆ קיימת צלע בגרף בין T2 ל T1 והתזמון conflict equivalent ל T2T1.
- ◆ לא קיימות צלעות בין קודקודי הגרף

אם האופציה הראשונה מתקיימת אז קיים קונפליקט מסוג RW או WR או WW בין T1 ל T2.
אם האופציה השנייה מתקיימת אז קיים קונפליקט מסוג RW או WR או WW בין T2 ל T1.
אם האופציה השלישית מתקיימת אז אין קונפליקטים, ולכן התזמון בר השגה על ידי פרוטוקול 2PL.

כלומר נותר לנו לטפל בשתי האופציות הקודמות.
נניח ללא הגבלת הכללית כי קיים קונפליקט בין T1 ל T2.
כדי לקרוא אובייקט הטרנזקציות יבקש מנעול משותף, במידה וירצו לכתוב אובייקט יבקשו מנעל אקסקלוסיבי.

עבור קונפליקט מסוג WW - טרנזקציה T1 מעוניינת לכתוב לאובייקט A ואחרייה טרנזקציה T2 מעוניינת לכתוב לאובייקט A. לכן T1 תבקש מפתח אקסקלוסיבי ותשחרר אותו, ואז T2 תוכל לעשות זאת.
עבור קונפליקט מסוג WR - טרנזקציה T1 מעוניינת לכתוב לאובייקט A ואחרייה טרנזקציה T2 מעוניינת לקרוא את אובייקט A. לכן T1 תבקש מפתח אקסקלוסיבי ותשחרר אותו, ואז T2 תבקש מפתח משותף.
עבור קונפליקט מסוג RW - טרנזקציה T1 מעוניינת לקרוא אובייקט A ואחרייה טרנזקציה T2 מעוניינת לכתוב אובייקט A. לכן T1 תבקש מפתח משותף ותשחרר אותו, ואז T2 תבקש מפתח אקסקלוסיבי.

טרנזקציה T1 תבקש תחילה את כל המנעולים שהיא צריכה לקריאה ולכתיבה, לאחר סיום פעולות הכתיבה T1 תוכל לשחרר את המנעולים.
T2 תבקש מנעול כשתצטרך, ותשתחרר את כל המנעולים בסוף התזמון.

נתון כי התזמון הוא בר סידורים קונפליקטים ולכן הוא conflict equivalent לתזמון סדרתי כלשהו. שני תזמונים הם conflict equivalent מעל אותן טרנזקציות אם הם מסדרים כל זוג פעולות קונפליקטים של טרנזקציות שעשו commit באותה דרך. מכיוון שההנחה ללא הגבלת הכלליות היא שקיימת שקילות לתזמון הסדרתי T1T2, אז בעצם כל זוג קונפליקטים מופיע באותו סדר של התזמון הסדרתי T1T2 ואז בהכרח T1 הוא הראשון שקורא או כותב.

T1 לא תצטרך לבקש עוד מנעולים שכן בהתחלה כבר ביקשה את כולם. כמו כן, היא משחררת את כל המנעולים לאחר כל פעולות הכתיבה. לאחר הכתיבה לא תצטרך לבצע פעולת קריאה ולבקש מנעול שכן היא זו שכתבה והיא יודעת מה כתוב, ו T2 לא תוכל לשנות את מה שכתבה. נניח בשלילה כי T2 כותבת לאותו ערך לפני שT1 קוראת אותו אז קיים

קונפליקט מסוג WR כך ש T2 הראשונה, בסתירה לכך שנתון כי התזמון conflict equivalent לתזמון הסדרתי T1T2. מכאן שההנחה בשלילה שגויה.

T1 תוכל לקבל את המנעול האקסקלוסיבי כדי לכתוב, בהנחה של T2 לא מחזיקה מנעול על אובייקט זה. נניח בשלילה כי T2 מחזיקה במנעול על אובייקט כאשר T1 מבקשת מנעול אקסקלוסיבי כדי לכתוב על אותו אובייקט, אזיי T2 ביקשה לקרוא או לכתוב לפני T1 , כלומר קיים קונפליקט כך שT2 היא הראשונה שקוראת או כותבת. מכאן שאם נבנה גרף קדימויות קיימת צלע בין T2 ל T1 וזאת בסתירה לכך שנתון כי התזמון conflict equivalent לתזמון הסדרתי T1T2. מכאן שההנחה בשלילה שגויה.

T2 תוכל לקבל את המנעול האקסקלוסיבי כדי לכתוב, בהנחה ש T1 לא מחזיקה מנעול על אובייקט זה.
במידה ו T2 תבקש מנעול אקסקלוסיבי על אובייקט היא תוכל לקבל אותו מכיוון שמובטח שT1 תשחרר את כל המנעולים לאחר הכתיבה ו T2 תצטרך מנעול רק לאחר ש T1 סיימה לכתוב.
נניח בשלילה ש T2 מבקשת מנעול לפני ש T1 סיימה לכתוב, כלומר T2 צריכה לקרוא או לכתוב לפני שT1 כותבת.
אזיי קיים קונפליקט מסוג RW או WW כך שT2 היא הראשונה בסתירה לכך שנתון כי התזמון conflict equivalent לתזמון הסדרתי T1T2. מכאן שההנחה בשלילה שגויה.

בבניה זו אף טרנזקציה לא מבקשת מנעולים נוספים לאחר ששיחררה - T2 מבקשת כשהיא צריכה ומשחררת בסוף התזמון וT1 ביקשה את כל המנעולים בתחילת התזמון., ולא תבקש שוב לאחר השחרור.

לכן הבקשות למנעולים והשחרורים שלהם מאפשרים את ביצוע הפעולות בסדר שבו הן כתובות בתזמון, והתזמון הינו בר השגה על ידי פרוטוקול 2PL.

<u>שאלה 3</u> (34 נקודות)

למדנו שניתן להריץ טרנזקציות ברמות בידוד שונות, ובהתאם, התנהגות הטרנזקציות עלולה להיות שונה. הבנה טובה של רמות בידוד הוא קריטי באפליקציה אמתית. בחירת רמת הבידוד יכול להשפיע גם על נכונות הנתונים במסד, וגם על יעילות האפליקציה. בשאלה זו, אתם תתנסו בהרצה של אותו קוד ברמות בידוד שונות, ותדרשו לנמק את ההבדלים בתוצאות.

נתונים 3 תזמונים. לפני הרצת כל אחד מהתזמונים, מייצרים טבלה ומכניסים שורות:

```
CREATE TABLE accounts(id integer primary key, owner varchar, balance integer);
INSERT INTO accounts VALUES(1, 'alice', 100), (2, 'bob', 100), (3, 'claire', 100);
```

ולאחר הרצת כל אחד מהתזמונים, הטבלה נמחקת. **שימו לב:** פקודות עדכון (update או insert) שמסתיימים ב returning *, מחזירות למשתמש את השורות שהשתנו על ידי פעולת העדכון. כמו כן, שימו לב שאנחנו נתעניין בעיקר בתוצאות של השורות המודגשות בצהוב.

<u>תזמון 1:</u>

<u>T1</u>	<u>T2</u>

1	Select * from accounts;	
2		Select * from accounts where id = 1;
3	update accounts set balance = balance – 10 where id = 1 returning *;	
4		Select * from accounts where id = 1;
5	Commit;	
6		Select * from accounts where id = 1;
7		Commit;

תזמון 2:

	T1	T2
1	Select * from accounts;	
2		Select * from accounts where balance = 100;
3	insert into accounts values(4, 'dan', 100) returning *;	
4		Select * from accounts where balance = 100;
5	Commit;	
6		Select * from accounts where balance = 100;
7		Commit;

תזמון 3:

	T1	T2	T3
1	Select * from accounts;		
2	insert into accounts select 5, 'trans1', sum(balance) from accounts returning *;		
3	Select * from accounts;		
4		Select * from accounts;	
5		insert into accounts	

		select 6, 'trans2', sum(balance) from accounts returning *;	
6		Select * from accounts;	
7	Commit;		
8		Commit;	
9			Select * from accounts;

עליכם להריץ את:

- תזמון 1 ברמות בידוד read committed ו repeatable read
- תזמון 2 ברמות בידוד read committed ו repeatable read
- תזמון 3 ברמות בידוד repeatable read ו serializable

יש 2 דרכים שונות להריץ את התזמונים, ותוכלו לבחור בדרך הנוחה לכם:

- הרצה ידנית: תפתחו חלון של postgres עבור כל טרנזקציה. בחלון הראשון, תרשמו את הפקודות של T1 בחלון הראשון ובחלון השני תרשמו את הפקודות של T2. שימו לב להפעיל את הפקודות לפי הסדר שרשום בתזמון, וכן להשתמש בפקודת BEGIN TRANSACTION ISOLATION LEVEL עם רמת הבידוד הדרושה. **הערה: השיטה הזאת פחות מומלצות, בגלל הקלות לטעות במהלך הכנסת הפקודות.**

- הרצה בעזרת תוכנית run-schedules.py: על מנת להקל עליכם, כתבנו תוכנית python שמתחבר למסד נתונים שלכם ומריץ את התזמונים. התוכנית רושמת את הפלט של כל אחד מהפקודות למסך. כדי להריץ את run-schedules.py, הורידו אותה מאתר הקורס לחשבון שלכם באוניברסיטה. התחברו לחשבון linux שלכם באוניברסיטה. בתיקיה שבו שמרתם את התוכנית, הריצו:

```
python run-schedules.py <user-name> <schedule-num> <isolation-level>
```

כאשר

- user-name הוא שם המשתמש שלכם ב linux,
- schedule-num הוא מספר 1, 2, או 3
- Isolation-level הוא RC (בשביל read committed), RR (בשביל repeatable read), או S (בשביל serializable)

להזכירכם, תצטרכו להריץ את התוכנית 6 פעמים, עם הפקודות:

- python run-schedules.py <user-name> 1 RC
- python run-schedules.py <user-name> 1 RR
- python run-schedules.py <user-name> 2 RC
- python run-schedules.py <user-name> 2 RR
- python run-schedules.py <user-name> 3 RR
- python run-schedules.py <user-name> 3 S

לאחר שתריצו את התזמונים, ענו על השאלות הבאות. בהסברים שלכם, עליכם להתייחס לרמת הבידוד ולמושגים כגון dirty write, dirty read, nonrepeatable read, phantom, serialization anomaly שנלמדו בהרצאות TM1 ו TM2.

תזמון 1, רמת בידוד read committed:

- מה מוחזר על ידי שורה 4?

(1, 'alice', 100)

- מה מוחזר על ידי שורה 6?

(1, 'alice', 90)

- האם 2 השורות החזירו את אותם תוצאות? כן **לא**
- אם כן, הסבר כיצד זה קשור לרמת הבידוד בו רץ השאילתה.

- אם לא, איזה מהתופעות הבאות התרחשה dirty write, dirty read, nonrepeatable read, phantom, serialization anomaly?

nonrepeatable read

- האם שני הטרנזקציות הצליחו לבצע commit? **כן** לא
- אם לא, מדוע?

תזמון 1, רמת בידוד repeatable read:

- מה מוחזר על ידי שורה 4?

(1, 'alice', 100)

- מה מוחזר על ידי שורה 6?

(1, 'alice', 100)

- האם 2 השורות החזירו את אותם תוצאות? **כן** לא
- אם כן, הסבר כיצד זה קשור לרמת הבידוד בו רץ השאילתה.

מכיוון שברמת בידוד RR, לא אפשרי שיהיו nonrepeatable reads.

- אם לא, איזה מהתופעות הבאות התרחשה dirty write, dirty read, nonrepeatable read, phantom, serialization anomaly?

- האם שני הטרנזקציות הצליחו לבצע commit? **כן** לא
- אם לא, מדוע?

<u>תזמון 2, רמת בידוד read committed:</u>

- מה מוחזר על ידי שורה 4?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)

- מה מוחזר על ידי שורה 6?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)
(4, 'dan', 100)

- האם 2 השורות החזירו את אותם תוצאות? כן **לא**
- אם כן, הסבר כיצד זה קשור לרמת הבידוד בו רץ השאילתה.

- אם לא, איזה מהתופעות הבאות התרחשה dirty write, dirty read, nonrepeatable read, phantom, serialization anomaly?

phantom

- האם שני הטרנזקציות הצליחו לבצע commit? **כן** לא

- אם לא, מדוע?

תזמון 2, רמת בידוד repeatable read:

- מה מוחזר על ידי שורה 4?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)

- מה מוחזר על ידי שורה 6?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)

- האם 2 השורות החזירו את אותם תוצאות? **כן** לא
- אם כן, הסבר כיצד זה קשור לרמת הבידוד בו רץ השאילתה.

מכיוון שברמת בידוד RR, לא אפשרי שיהיו phanthom reads.

- אם לא, איזה מהתופעות הבאות התרחשה dirty write, dirty read, nonrepeatable read, phantom, serialization anomaly?

- האם שני הטרנזקציות הצליחו לבצע commit? **כן** לא
- אם לא, מדוע?

תזמון 3, רמת בידוד repeatable read:

- מה מוחזר על ידי שורה 3?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)
(5, 'trans1', 300)

- מה מוחזר על ידי שורה 6?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)
(6, 'trans2', 300)

- מה מוחזר על ידי שורה 9?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)
(5, 'trans1', 300)
(6, 'trans2', 300)

- האם התוצאות שקולות לריצה סדרתית כלשהו של הטרנזקציות שביצעו commit?
  כן **לא**
- אם לא, איזה מהתופעות הבאות התרחשה dirty write, dirty read, nonrepeatable read, phantom, serialization anomaly?
  serialization anomaly

- האם שני הטרנזקציות הצליחו לבצע commit? **כן** לא
- אם לא, מדוע?

<u>תזמון 3, רמת בידוד serializable:</u>

- מה מוחזר על ידי שורה 3?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)
(5, 'trans1', 300)

- מה מוחזר על ידי שורה 6?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)
(6, 'trans2', 300)

- מה מוחזר על ידי שורה 9?

(1, 'alice', 100)
(2, 'bob', 100)
(3, 'claire', 100)
(5, 'trans1', 300)

- האם התוצאות שקולות לריצה סדרתית כלשהו של הטרנזקציות שביצעו commit?
  כן לא
- אם לא, איזה מהתופעות הבאות התרחשה dirty write, dirty read, nonrepeatable read, phantom, serialization anomaly?

- האם שני הטרנזקציות הצליחו לבצע commit? כן **לא**
- אם לא, מדוע?

"could not serialize access due to read/write dependencies among transactions"
זה קורה מכיוון שסידור הפעולות מחדש, ישנה את התוצאות
(serialization anomaly)