Contents

1 Basic Test Results

```
Extracting Archive:
Archive:  /tmp/bodek.N5LEo7/db/ex6/mohammadgh/presubmission/submission
  inflating: ex6.pdf
  inflating: README

************************************************************
** Testing that all necessary files were submitted:
README:
    SUBMITTED
ex6.pdf:
    SUBMITTED

************************************************************
** Checking for correct README format:
```

2 README

```
mohammadgh,muaz.abdeen
```

Transaction Management : תרגיל 6

Muaz Abdeen 300575297
Mohammad Ghanayem 208653220

תאריך הגשה : 12.06.22 ,23: 55.

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!

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).

שאלה 1: (36 נקודות)

נתון התזמון :

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

ענה על השאלות הבאות, ונמק בקצרה את תשובתך.

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

Explanation: The Dirty Read occurs when there is a transaction that reads another transaction's write, in our schedule above **the only Dirty Read is between T1 and T2**, because T2 has written Z without commit in line 8 and T1 has read it in line 10, and that's it, there is no other Dirty Read here.

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

Explanation: The Nonrepeatable Reads occur when there is a transaction that reads data and re-reads it, but it found out that the data has been changed but it didn't change it, in our schedule each transaction has just read a data at most once, so we **don't have** any Nonrepeatable Reads.

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

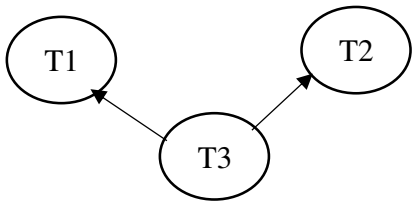
Explanation: The definition of Cascading Aborts says "a schedule S is Cascading Aborts if transactions read just the changes of transaction that commit changes", in our case T2 has made a change on Z but it didn't commit changes then T1 has read Z, this made our schedule as **not** preventing from Cascading Aborts.

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

Explanation: Our schedule is not Recoverable because T1 reads a change that T2 has made but T1 has commit before T2, that made our schedule **not** Recoverable.
(if T2 made commit before T1 then the schedule will be Recoverable)

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

Explanation: By using the "Checking for Conflict Serializability" process we can see that there is not directed cycle in the precedence graph.



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

Explanation: In order for a schedule to be achievable by 2PL Protocol, we need to find an order for locks and unlocks that make the schedule runs smoothly, so we try to find an order like this, this is the schedule after the new order of locks and unlocks:

	T1	T2	T3
1		S(Z)	
2		R(Z)	
3		S(Y)	
4		R(Y)	
5			S(X)
6			R(X)
7	S(X)		
8	R(X)		
9			S(Z)
10			R(Z)
11			S(Y)
12			R(Y)
13			U(X)
14			U(Z)
15			U(Y)
16			Commit
17		X(Z)	
18		W(Z)	
19		X(Y)	
20		W(Y)	
21		S(X)	
22		U(Z)	
23		U(Y)	
24	X(Z)		
25	R(Z)		
26	W(Z)		
27	U(Z)		
28	U(X)		
29	Commit		
30		R(X)	
31		U(X)	
32		Commit	

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

Explanation: In order for a schedule to be achievable by Strict 2PL Protocol, we need to make the transactions to unlock the locks just when they abort or commit, but in our situation we can't, because if we just unlock the locks of T2 with the commit process we got a problem with T1, because in line 24 we want to acquire the lock of Z to give T1 the ability to read and write Z, but if we didn't unlock the Z lock we will not be able to read and write Z in T1, so the schedule is not achievable by Strict 2PL Protocol.

ח. 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? 8

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

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

הקיף את התשובה הנכונה: כן לא

אם ענית לא, באיזה שורה הפרוטוקול ייכשל? X

שאלה 2 (30 נקודות)

בתזמון הבא ציינו את זמני ההתחלה של הטרנזקציות, בקשות למנעול משותף (למשל T1 מבקש מנעול משותף על A) ובקשות למנעול אקסקלוסיבי (למשל T2 מבקש מנעול אקסקלוסיבי על B).
שימו לב שהטרזקציות אינם מבקשות לשחרר מנעולים .

	$TS(T_1) = 4$	$TS(T_2) = 2$	$TS(T_3) = 3$	$TS(T_4) = 1$
No.	T1	T2	T3	T4
1				BEGIN
2		BEGIN		
3			BEGIN	
4				S(A)
5			X(B)	
6	BEGIN			
7		S(A)		
8		S(B)		
9	X(C)			
10	X(B)			
11				X(C)

הערה :אין צורך להתחשב בריצה מחודשת של טרנזקציות שנופלות .

Locks Requests:
Line 4: T_4 requests **shared** lock on A.
Line 5: T_3 requests **exclusive** lock on B.
Line 7: T_2 requests **shared** lock on A.
Line 8: T_2 requests **shared** lock on B.
Line 9: T_1 requests **exclusive** lock on C.
Line 10: T_1 requests **exclusive** lock on B.
Line 11: T_4 requests **exclusive** lock on C.

א. נניח שמנהל המנעולים משתמש בשיטת **wait-die.** בחר את כל התשובות הנכונות .

1. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T1 (כלומר יעשה לו abort)
2. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T1 (כלומר יגרום לו לחכות)
3. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T2 (כלומר יעשה לו abort)
4. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T2 (כלומר יגרום לו לחכות)
5. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T3 (כלומר יעשה לו abort)
6. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T3 (כלומר יגרום לו לחכות)
7. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T4 (כלומר יעשה לו abort)
8. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T4 (כלומר יגרום לו לחכות)
9. אף אחת מהתשובות האחרות איננה נכונה

ב. נניח שמנהל המנעולים משתמש בשיטת **wound-wait.** בחר את כל התשובות הנכונות .

1. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T1 (כלומר יעשה לו abort)
2. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T1 (כלומר יגרום לו לחכות)
3. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T2 (כלומר יעשה לו abort)
4. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T2 (כלומר יגרום לו לחכות)
5. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T3 (כלומר יעשה לו abort)
6. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T3 (כלומר יגרום לו לחכות)
7. במהלך ביצוע התזמון, מנהל המנעולים יפיל את T4 (כלומר יעשה לו abort)
8. במהלך ביצוע התזמון, מנהל המנעולים ישהה את T4 (כלומר יגרום לו לחכות)
9. אף אחת מהתשובות האחרות איננה נכונה

<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 grades(id integer primary key, name varchar, grade integer);
INSERT INTO grades VALUES(1, 'alice', 80), (2, 'bob', 90), (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> :

	T1	T2
1	Select * from grades;	
2		Select * from grades where id = 1;
3	update grades set grade = grade +10 where id = 1 returning *;	
4		Select * from grades where id = 1;
5	Commit;	
6		Select * from grades where id = 1;
7		Commit;

תזמון 2 :

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

תזמון 3 :

	T1	T2	T3
1	Select * from grades;		
2	insert into grades select 5, 'trans1', avg(grade) from grades returning *;		
3	Select * from grades;		
4		Select * from grades;	
5		insert into grades select 6, 'trans2', avg(grade) from grades returning *;	
6		Select * from grades;	
7	Commit;		
8		Commit;	
9			Select * from grades;

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’, 80)

- 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.
EMPTY

- 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?
EMPTY

תזמון 1, רמת בידוד repeatable read:

- מה מוחזר על ידי שורה 4?
80

- מה מוחזר על ידי שורה 6?
80

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

In schedule 1 we have just Nonrepeatable Read, so solve this problem we need to use an isolation level that help us to get rid of Nonrepeatable Reads, so in Repeatable Read isolation level this problem type will not occur.

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

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

EMPTY

תזמון 2, רמת בידוד read committed:

- מה מוחזר על ידי שורה 4?

(3, ‘claire’, 100)

- מה מוחזר על ידי שורה 6?

(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.

EMPTY

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

Phantom Read

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

EMPTY

תזמון 2, רמת בידוד repeatable read:

- מה מוחזר על ידי שורה 4?

(3, ‘claire’, 100)

- מה מוחזר על ידי שורה 6?

(3, ‘claire’, 100)

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

Repeatable Read isolation level prevent Phantom Read problem in postgres, if we use it in different sql server the problem may stay.

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

EMPTY

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

EMPTY

תזמון 3, רמת בידוד repeatable read:

- מה מוחזר על ידי שורה 3?

(1, 'alice', 80)
(2, 'bob', 90)
(3, 'claire', 100)
(5, 'trans1', 90)

- מה מוחזר על ידי שורה 6?

(1, 'alice', 80)
(2, 'bob', 90)
(3, 'claire', 100)
(6, 'trans2', 90)

- מה מוחזר על ידי שורה 9?

(1, 'alice', 80)
(2, 'bob', 90)
(3, 'claire', 100)
(5, 'trans1', 90)
(6, 'trans2', 90)

- 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?

EMPTY

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

EMPTY

תזמון 3, רמת בידוד serializable:

- מה מוחזר על ידי שורה 3?
  (1, ‘alice’, 80)
  (2, ‘bob’, 90)
  (3, ‘claire’, 100)
  (5, ‘trans1’, 90)

- מה מוחזר על ידי שורה 6?
  (1, ‘alice’, 80)
  (2, ‘bob’, 90)
  (3, ‘claire’, 100)
  (6, ‘trans2’, 90)

- מה מוחזר על ידי שורה 9?
  (1, ‘alice’, 80)
  (2, ‘bob’, 90)
  (3, ‘claire’, 100)
  (5, ‘trans1’, 90)

- האם התוצאות (כלומר מצב הטבלה בסיום הריצה) 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?

  EMPTY

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

This happens because Serializable isolation level make our schedule as if we run each transaction after the other (i.e., run all T1 first and all T2 after it) before committing the changes that T1 has made T2 has made another changes, but he didn’t saw what T1 did, after committing the changes of T1 we have 4 rows but T2 wanted to add a fourth row with different data, so when T2 wanted to commit its changes the server refuses because there is another changes has been made before and if we ran T2 before T1 we will got a different result so T2 can’t do commit in this situation.
