תרגיל 6 : Transaction Management

תאריך הגשה: 23:55, 17.01.21.

מגישים:

מיכל לוי – 313573743

רועי בן יוסף - 307920116

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:

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

ענה על השאלות הבאות, ונמק בקצרה את תשובתך.

1

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

**נימוק:** הגדרת dirty read :

A transaction reads data written by a concurrent uncommitted transaction.""

הקריאה היחידה שנעשתה על משתנה שנכתב ללא קומיט היא R(X) בשורה 10 ע"י T 3

0

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

**נימוק:** הגדרת nonrepeatable read :

A transaction re-reads data it has previously read and finds that it has been modified by another transaction.

במקרה זה אין קריאות חוזרות כלל, ולכן גם אין Nonrepeatable reads .

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

**נימוק:** מכיוון שT 3 מבצע R(X) בשורה 10 לאחר שנכתב על ידי T1 ולפני שT 1 ביצע קומיט.

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

**נימוק:** לפי ההגדרה: \_\_\_\_\_

המקרה היחיד בו טרנסאקציה אחת קוראת שינוי שביצעה טרנסאקציה אחרת היא בשורה 10 , כאשר T 3 קוראת שינוי שביצעה T 1 .

T 3 מבצע קומיט לאחר ש T 1 מבצע קומיט ולכן recoverable

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

**נימוק:** אם נבצע את T-2 ואז את T-1 ואז את T-3 נקבל לו"ז שהוא שקול קונפליקטים ללו"ז המקורי.

הקונפליקטים הקיימים לפי הסדר:

1. RW בין T-2 ל- T-1 על משתנה X
2. RW בין T-2 ל- T-1 על משתנה Y
3. WR בין T-1 ל- T-3 על משתנה X
4. RW בין T-2 ל- T-3 על משתנה Z

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

**נימוק:** מצורף כטבלה:

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

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

**נימוק:** בפרוטוקול strict 2PL אנו נדרשים לשחרר את כל המנעולים של טרנסאקציה מסוימת רק בסיום הטרנסאקציה. לכן נוכל לשחרר את המנעול האקס' על X של T-1 רק כשנסיים איתה, לאחר שT-3 מבקש לקרוא את אותו X

1. 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

7

אם ענית לא, באיזה שורה הפרוטוקול ייכשל?

1. 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.

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

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

כבר הוכחנו שכל תזמון שבר השגה ע"י PL2 הוא בר סידור קונפליקטים, ובפרט אם יש בו בסך הכל 2 טרנזאקציות.

נוכיח את הכיוון השני:

יהי תזמון בר סידור קונפליקטים בעל 2 טרנזאקציות. (T1 ו-T2 )

נוכל לחלק למקרים לפי הקונפליקטים:

* במידה ואין קונפליקטים כלל, אז אין פעולות W כלל. לכן כל סידור PL2 יעבוד
* במידה ויש קונפליקט, אז נניח בה"כ כי בגרף הקדימויות T-2 קודם לT1 . שוב נחלק לתת מקרים:
  + אם פעולת הכתיבה מתבצעת ב-T2 . נוכל להסיק מגרף הקדימויות כי **לפניה** לא מתבצעת אף פעולה כלל של T-1 .

ולכן יכול להתקיים 2PL .

* + אם פעולת הכתיבה מתבצעת ב-T1 . נוכל להסיק מגרף הקדימויות כי **אחריה** לא מתבצעת אף פעולה כלל של T-2 .

כלומר 2PL יוכל לבקש את כל המנעולים לT-2 בהתחלה ולשחרר אותם לפני פעולת הכתיבה של T-1, ויוכל לבקש מנעול אקסקלוסיבי לT-1 מיד לפני הכתיבה, ולאחר מכן לשחרר אותו.

לכן קיים סידור 2PL

**שאלה 3** (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.

תזמון 1:

|  |  |  |
| --- | --- | --- |
|  | T1 | T2 |
| 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?

טרנסאקציה 2 קראה פעמיים את אותו משתנה מבלי לשנותו וקיבלה ערכים שונים

* 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 שהרצנו ב-RR חסמנו את האפשרות לnonrepeatable read ולכן טרנסאקציה 2 קראה את אותו הערך פעמיים

* 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 2, רמת בידוד read committed:**

* 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.

הרצנו ב-RC , ולכן קריאות פנטום אפשריות.

* 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 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.

ברמת בידוד RR , פוסטגרס (וספציפית הוא) לא מאפשר קריאות פנטום.

* 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 כלשהי של הטרנזקציות שביצעו commit?

כן לא

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

מכיוון שטרנסאקציה 1 ביצעה קומיט לאחר שטרנסאקציה 2 הכניסה ערכים חדשים, קיבלנו תוצאות שלא יכולות להתקבל ע"י אף הרצה סדרתית.

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

**תזמון 3, רמת בידוד serializable:**

* 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?

מכיוון שהרצנו על רמת בידוד S , לא אפשרנו serialization anomaly , ולכן גרמנו לטרנסאקציה מס' 2 ל -abort