German University in Cairo
Media Engineering and Technology
Dr. Wael Abouelsaadat

Data Bases II, Spring 2018 **Practice Assignment 9**

Part One: Locking + transactions

Exercise 9-1

For the following schedule:

r1(A); r2 (B); r3(C); r1 (B); r2 (C); r3 (D); w1A); w2 (B); w3 (C);

Do each of the following:

i. Insert shared and exclusive locks, and insert unlock actions. Place a shared lock immediately in front of each read action that is not followed by a write action of the same element by the same transaction. Place an exclusive lock in front of every other read or write action. Place the necessary unlocks at the end of every transaction.

ii. Tell what happens when each schedule is run by a scheduler that supports shared and exclusive locks.

iii. Insert shared and exclusive locks in a way that allows upgrading. Place a shared lock in front of every read, an exclusive lock in front of every write, and place the necessary unlocks at the ends of the transactions.

iv. Tell what happens when each schedule from (iii) is run by a scheduler that supports shared locks, exclusive locks, and upgrading.

v. Insert shared, exclusive, and update locks, along with unlock actions. Place a shared lock in front of every read action that is not going to be upgraded, place an update lock in front of every read action that will be upgraded, and place an exclusive lock in front of every write action. Place unlocks at the ends of transactions, as usual.

vi. Tell what happens when each schedule from (v) is run by a scheduler that supports shared, exclusive, and update locks.

Answer:

(i): xl1(A); r1(A); xl2(B); r2(B); xl3(C); r3(C); sl1(B); r1(B); sl2(C); r2(C); sl3(D); r3(D); w1(A); u1(A); u1(B); w2(B); u2(B); u2(C); w3(C); u3(C); u3(D)

(ii): The first three locks and reads are allowed. However, T1 cannot get a shared lock on B, nor can T2 get a shared lock on C. When T3 requests a shared lock on D it is granted. Thus, T3 can proceed and eventually unlocks C and D.

As soon as C is unlocked, T2 can get its shared lock on C. Thus, T2 completes and releases its locks. As soon as its lock on B is released, T1 can get its lock on A, so it completes.

(iii): sl1(A); r1(A); sl2(B); r2(B); sl3(C); r3(C); sl1(B); r1(B); sl2(C); r2(C); sl3(D); r3(D); xl1(A); w1(A); u1(A); u1(B); xl2(B); w2(B); u2(B); u2(C); xl3(C); w3(C); u3(C); u3(D)

(iv): First, all six shared-lock requests are granted. When T1 requests an exclusive lock on A, it gets it, because it is the only transaction holding a lock on A. T1 completes and releases its locks. Thus, when T2asks for an exclusive lock on B, it is the only transaction still holding a shared lock on B, so that lock too may be granted, and T2 completes. After T2 releases its locks, T3 is able to upgrade its shared lock on C to exclusive, and it too proceeds. The actions are thus executed in exactly the same order as they are requested; i.e., there are no delays by the scheduler.

(v): ul1(A); r1(A); ul2(B); r2(B); ul3(C); r3(C); sl1(B); r1(B); sl2(C); r2(C); sl3(D); r3(D); xl1(A); w1(A); u1(A); u1(B); xl2(B); w2(B); u2(B); u2(C); xl3(C); w3(C); u3(C); u3(D)

(vi): The update locks prevent the fourth and fifth shared-lock requests, sl1(B) and sl2(C), so T1 and T2 are delayed, while T3 is allowed to proceed. The situation is exactly as for part (ii), where the initial lock requests are for exclusive locks.

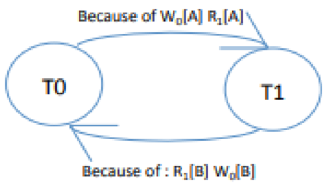
Exercise 9-2

1. **Schedules, Serializability, and Locking**

   (a) Consider the following two transactions and schedule (time goes from top to bottom). Is this schedule conflict-serializable? Explain why or why not.

Transaction T_0	Transaction T_1
$r_0[A]$	
$w_0[A]$	
	$r_1[A]$
	$r_1[B]$
	c_1
$r_0[B]$	
$w_0[B]$	
c_0	

Solution:
The schedule is not conflict serializable because the precedence graph contains a cycle. The graph has an edge $T_0 \rightarrow T_1$ because the schedule contains $w_0[A] \rightarrow r_1[A]$. The graph has an edge $T_1 \rightarrow T_0$ because the schedule contains $r_1[B] \rightarrow w_o[B]$.



(b) Show how 2PL can ensure a conflict-serializable schedule for the same transactions above. Use the notation $L_i[A]$ to indicate that transaction i acquires the lock on element A and $U_i[A]$ to indicate that transaction i releases its lock on A.

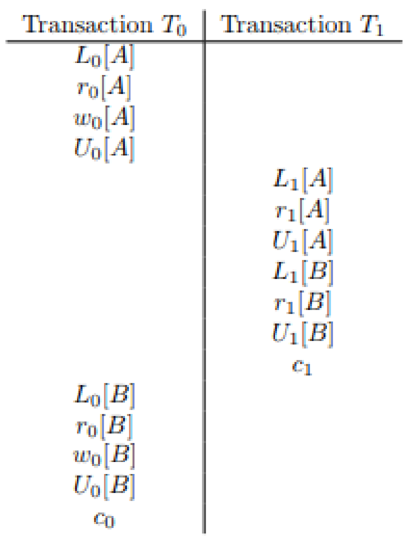
Solution:

Multiple solutions are possible.

Transaction T_0	Transaction T_1
$L_0[A]$	
$r_0[A]$	
$w_0[A]$	
	$L_1[A] \rightarrow blocks$
$L_0[B]$	
$r_0[B]$	
$w_0[B]$	
$U_0[A]$	
$U_0[B]$	
c_0	
	$L_1[A] \rightarrow granted$
	$r_1[A]$
	$L_1[B]$
	$r_1[B]$
	$U_1[A]$
	$U_1[B]$
	c_1

(c) Show how the use of locks without 2PL can lead to a schedule that is NOT conflict serializable.

Solution:



Transaction T_0	Transaction T_1
$L_0[A]$	
$r_0[A]$	
$w_0[A]$	
$U_0[A]$	
	$L_1[A]$
	$r_1[A]$
	$U_1[A]$
	$L_1[B]$
	$r_1[B]$
	$U_1[B]$
	c_1
$L_0[B]$	
$r_0[B]$	
$w_0[B]$	
$U_0[B]$	
c_0	

Exercise 9-3

Consider two tables R(A,B) and S(C). Below are pairs of transactions. For each pair, decide whether it is possible for non-serializable behavior to be exhibited when executing the transactions concurrently, while respecting their specified isolation levels. Assume individual statements are executed atomically, and each transaction executes to completion.

(a) Transaction 1:

Set Transaction Isolation Level Read Committed;

Select count(*) From R;

Select count(*) From S;

Commit;

Transaction 2:

Set Transaction Isolation Level Serializable;

Insert Into R Values (1,2);

Insert Into S Values (3);

Commit;

(b) Transaction 1:

Set Transaction Isolation Level Read Committed;

Select count(*) From R;

Select count(*) From S;

Commit;

Transaction 2:

Set Transaction Isolation Level Serializable;

Insert Into R Values (1,2);

Insert Into R Values (3,4);

Commit;

(c) Transaction 1:

Set Transaction Isolation Level Repeatable Read;

Select count(*) From R;

Select count(*) From S;

Select count(*) From R;

Commit;

Transaction 2:

Set Transaction Isolation Level Serializable;

Insert Into R Values (1,2);

Commit;

Exercise 9-4

Consider table Item(name,price) where name is a key, and the following two concurrent transactions.

T1:

```
  Begin Transaction;

  S1: Insert Into Item Values ('scissors',40);

  S2: Update Item Set price = price + 30 Where name = 'pencil';

  Commit;
```

T2:

```
  Begin Transaction;

  S3: Select Avg(price) As a1 From Item;

  S4: Select Avg(price) As a2 From Item;

  Commit;
```

Assume that the individual statements S1, S2, S3, and S4 always execute atomically. Suppose initially there are two tuples in Item: (pencil,20) and (pen,30). Each transaction runs once and commits. Transaction T1 always executes with isolation level Serializable.

(a) If transaction T2 executes with isolation level Serializable, what possible pairs of values a1 and a2 are returned by T2?

(b) If transaction T2 executes with isolation level Repeatable-Read, what possible pairs of values a1 and a2 are returned by T2?

(c) If transaction T2 executes with isolation level Read-Committed, what possible pairs of values a1 and a2 are returned by T2?

(d) If transaction T2 executes with isolation level Read-Uncommitted, what possible pairs of values a1 and a2 are returned by T2?

Exercise 9-5

Consider table Worker(name,pay) where name is a key, and the following two concurrent transactions.

```
T1:
  Begin Transaction
  S1: update Worker set pay = 2*pay where name = 'Amy'
  S2: update Worker set pay = 3*pay where name = 'Amy'
  Commit
T2:
  Begin Transaction
  S3: update Worker set pay = pay-20 where name = 'Amy'
  S4: update Worker set pay = pay-10 where name = 'Amy'
  Commit
```

Assume that the individual statements S1, S2, S3, and S4 always execute atomically. Let Amy's pay be 50 before either transaction begins execution.

(a) Suppose both transactions T1 and T2 execute to completion with isolation level Serializable. What are the possible values for Amy's final pay?

(b) Suppose both transactions T1 and T2 execute to completion with isolation level Read-Committed. What are the possible values for Amy's final pay?

(c) Suppose transaction T1 executes with isolation level Read-Committed, transaction T2 executes with isolation level Read-Uncommitted, and both transactions execute to completion. What are the possible values for Amy's final pay?

(d) Suppose both transactions T1 and T2 execute to completion with isolation level Read-Uncommitted. What are the possible values for Amy's final pay?

(e) Suppose both transactions T1 and T2 execute with isolation level Serializable. Transaction T1 executes to completion, but transaction T2 rolls back after statement S3 and does not re-execute. What are the possible values for Amy's final pay?

Answers:

9-3

(a) Yes, nonserializable behavior is possible (two statements of Transaction 1 execute before and after Transaction 2, respectively)

(b) No, nonserializable behavior is not possible (state of S is same before and after Transaction 2)

(c) Yes, nonserializable behavior is possible (first and third statements of Transaction 1 execute before and after Transaction 2, respectively)

9-4

(a) (25,25) (40,40)

(b) (25,25) (40,40)

(c) (25,25) (25,40) (40,40)

(d) (25,25) (25,30) (25,40) (30,30) (30,40) (40,40)

9-5

(a) 120 270

(b) 120 270

(c) 120 210 270

(d) 120 150 170 210 230 270

(e) 300