COMP207 Assignment 1 – Transaction Management

Issue Date: Thursday, 31 October 2019

Submission Deadline: Thursday, 14 November 2019, 17:00

About This Assignment

This is the first of two assignments for COMP207. It is worth 10% of the total marks for this module. It consists of six questions, which you can find at the end of this document.

Submit your solutions to these questions in PDF format by the given submission deadline. Your solutions must be submitted on Vital (see the detailed submission instructions below).

Accuracy and relevance are more important in your answers, so don't write large volumes in your submission, but do ensure that what you write covers what is asked for and keeps to the problem statement.

Submission Details

Please submit **one PDF file with your solutions**. Name your file as follows:

<your student ID>-Assignment-1.pdf

If your student ID is 12345678, then your file should be named:

12345678-Assignment-1.pdf.

Please submit only this file (no archives).

To act as your 'signature' for the assignment, at the top of your PDF document put your Student ID number.

Your solutions must be submitted on Vital (see Vital for submission instructions).

The submission deadline for this assignment is **Thursday, 14 November 2019, 17:00**. Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties. Plagiarism and collusion guidelines will apply throughout the assignment submission. For details on late submissions, how to claim extenuating circumstances, etc., please see the undergraduate student handbook, which can be found at http://intranet.csc.liv.ac.uk/student/ug-handbook.pdf , or in Section 6 of the Code of Practice on Assessment.[1]

Assessment information at a glance

Assignment Number	1 (of 2)
Weighting	10% of the final module mark
Assignment Circulated	Thursday, 31 October 2019
Deadline	Thursday, 14 November 2019, 17:00
Submission Mode	Electronically on Vital
Learning Outcome Assessed	LO1: Identify and apply the principles underpinning transaction management within DBMS and the main security issues involved in securing transactions
Purpose of Assessment	Assessment of knowledge of multi-user databases and the need for relevant control to ensure database integrity
Marking Criteria	See description of this assignment
Submission necessary in order to satisfy module requirements?	N/A
Late Submission Penalty	Standard UoL Policy

[1] https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/code_of_practice_on_assessment.pdf

Question 1 (16 marks)

Consider the following two transactions (we omit the final 'commit' operation):

Transaction T_1	**Transaction T_2**
read item(X);	read item(Y);
$X := X + 2$;	$Y := Y * 2$;
write_item(X);	write item(Y);
read_item(Y);	read_ item(X);
$Y := Y * 3$;	$X := X + 3$;
write_item(Y);	write_item(X);
commit;	commit;

Assume that transactions T_1 and T_2 use shared buffers (i.e., once a transaction writes X back to the buffer, the new value of X can be read by the other transaction, and similarly for Y).

(a) Give serial schedules S_1 for $T_1 \rightarrow T_2$ and S_2 for $T_2 \rightarrow T_1$. **(1 mark per schedule)**

(b) For each of the two schedules you gave in **(a)** S_1 and S_2, give the values of X and Y after executing the schedule on a database with items $X = 1$ and $Y = 2$?

(2 marks per schedule)

(c) Give a serialisable schedule S_3 for T_1 and T_2 that is not serial. Explain why your schedule S_3 is serialisable. **(5 marks)**

(d) Give a schedule S_4 for T_1 and T_2 that is not serialisable. Explain why S_4 is not serialisable. **(5 marks)**

Solutions

(a) $S1 : r1(X); w1(X); r1(Y); w1(Y); c1; r2(Y); w2(Y); r2(X); w2(X); c2$

$S_2 : r_2(Y); w_2(Y); r_2(X); w_2(X); c_2; r_1(X); w_1(X); r_1(Y); w_1(Y); c_1$

(b) Values of X and Y after executing S_1 and S_2:

	X	Y
S_1	$(1 + 2) + 3 = 6$	$(2 * 3) * 2 = 12$
S_2	$(1 + 3) + 2 = 6$	$(2 * 2) * 3 = 12$

(c) $S_3 : r_1(X); w_1(X); r_2(Y); w_2(Y); r_2(X); w_2(X); c_2; r_1(Y); w_1(Y); c_1$

This schedule is not serial. If we execute it on a database with items $X = a$ and $Y = b$, we obtain

- $X = a + 2 + 3$,
- $Y = b * 2 * 3$

which is the same result as executing S_1 on the same database. Hence, S_3 is serialisable.

(d) $S_4 : r_1(X); w_1(X); r_1(Y); r_2(Y); w_2(Y); w_1(Y); c_1; r_2(X); w_2(X); c_2$

Assuming a database with items $X = 1$ and $Y = 2$ as in (a), the values of X and Y after executing S_4 are:

- $X = (1 + 2) + 3 = 6$
- $Y = 2 * 3 = 6$ (the update made by T_2 is lost)

Hence, S_4 has a different effect than the two serial schedules S_1 and S_2, implying that S_4 is not serialisable.

Question 2 (35 marks)

Consider the following schedules:

- $S_1 : r_1(X); r_2(X); r_3(Y); w_1(X); r_2(Z); r_2(Y); w_2(Y); w_1(Z)$
- $S_2 : r_1(X); r_1(Y); w_1(Y); r_3(Y); r_2(Y); r_2(Z); w_3(U); w_2(Z); r_4(Z); r_4(U); w_4(U)$
- $S_3 : w_3(X); r_1(X); w_1(Y); r_2(Y); w_2(Z); r_3(Z)$
- $S_4 : r_1(X); r_4(X); r_3(Y); w_4(Y); r_1(Y); r_2(Y); r_3(X); r_4(Y); w_1(X); w_2(Y)$
- $S_5 : r_1(X); w_1(Y); r_2(Y); w_2(Z); r_3(Z); w_3(X)$

For each of these schedules, answer the following questions:

(a) What are the conflicts in the schedule? **(2 marks per schedule)**

(b) What is the precedence graph of the schedule? **(2 marks per schedule)**

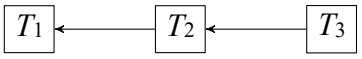
(c) Is the schedule conflict-serialisable? Why? If the schedule is conflict-serialisable, give a conflict-equivalent serial schedule. **(3 marks per schedule)**

Solutions

- S_1:

  (a) Conflicts: $r_2(X)$-$w_1(X)$, $r_3(Y)$-$w_2(Y)$, $r_2(Z)$-$w_1(Z)$

  (b) Precedence graph:

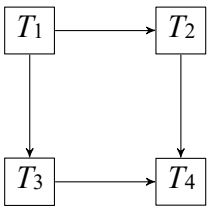
  

  (c) S_1 is conflict-serialisable, because its precedence graph is acyclic. The only conflict-equivalent serial schedule is T_3-T_2-T_1 (the one that executes T_3, then T_2, then T_1).

- S_2:

  (a) Conflicts: $w_1(Y)$-$r_2(Y)$, $w_1(Y)$-$r_3(Y)$, $w_2(Z)$-$r_4(Z)$, $w_3(U)$-$r_4(U)$, $w_3(U)$-$w_4(U)$

  (b) Precedence graph:

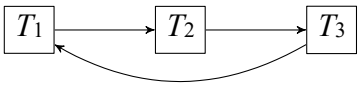
  

(c) S_2 is conflict-serialisable, because the precedence graph is acyclic. The only two conflict-equivalent serial schedules are T_1-T_2-T_3-T_4 and T_1-T_3-T_2-T_4.

- S_3:

  (a) Conflicts: $w_3(X)$-$r_1(X)$, $w_1(Y)$-$r_2(Y)$, $w_2(Z)$-$r_3(Z)$

  (b) Precedence graph:

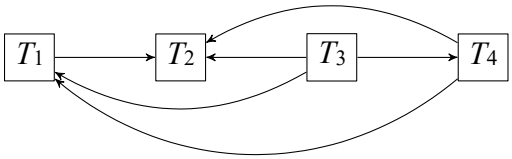


  (c) S_3 is not conflict-serialisable, because its precedence graph is not acyclic.

- S_4:

  (a) Conflicts: $r_3(Y)$-$w_4(Y)$, $w_4(Y)$-$r_1(Y)$, $w_4(Y)$-$r_2(Y)$, $w_4(Y)$-$w_2(Y)$, $r_4(X)$-$w_1(X)$, $r_3(X)$-$w_1(X)$, $r_3(Y)$-$w_2(Y)$, $r_1(Y)$-$w_2(Y)$, $r_4(Y)$-$w_2(Y)$

  (b) Precedence graph:

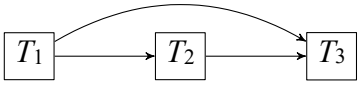


  (c) S_4 is conflict-serialisable, because its precedence graph is acyclic. The only conflict-equivalent serial schedule is T_3-T_4-T_1-T_2 (the one that executes T_3, then T_4, then T_1, and then T_2).

- S_5:

  (a) Conflicts: $r_1(X)$-$w_3(X)$, $w_1(Y)$-$r_2(Y)$, $w_2(Z)$-$r_3(Z)$

  (b) Precedence graph:



  (c) S_5 is conflict-serialisable, because its precedence graph is acyclic. The only conflict-equivalent serial schedule is S_5 itself.

Question 3 (15 marks)

For each of the following schedules, determine if the schedule is:

i) Recoverable
ii) Cascadeless
iii) Strict

(a) S_1 : $r_1(X)$; $r_2(X)$; $w_2(X)$; $w_1(X)$; a_2; c_1 **(3 marks)**

(b) S_2 : $r_1(X)$; $r_2(X)$; $w_2(X)$; $w_1(X)$; c_2; c_1 **(3 marks)**

(c) S_3 : $r_1(X)$; $w_1(X)$; $r_2(X)$; $w_1(X)$; c_2; c_1 **(3 marks)**

(d) S_4 : $r_1(X)$; $w_1(X)$; $r_2(X)$; $w_1(X)$; a_2; c_1 **(3 marks)**

(e) S_5 : $r_1(X)$; $r_2(X)$; $w_2(X)$; c_2; $w_1(X)$; c_1; $r_3(X)$; c_3 **(3 marks)**

Solutions

(a) S1 is recoverable since T1 writes and commits first and cascadeless since it is recoverable. However, it is not strict since T2 was first to write – so there should be no conflicting operations between this write and the T2 abort or commit (but T1 performs a write-x between these).

(b) S2 is recoverable since T2 is first to write and is first to commit and cascadeless since it is recoverable. However, it is not strict because T2 was first to write – so there should be no conflicting operations between this write and the T2 abort or commit (but T1 performs a write-x between these).

(c) S3 is not recoverable since T1 is first to write but not first to commit. It is not cascadeless since T2 has read x from an uncommitted transaction. If T1 were to abort, then T2 has an invalid read so must also be aborted. It is also not strict because T1 was first to write – so there should be no conflicting operations between this write and the T1 abort or commit (but T2 performs a read-x between these).

(d) S4 is recoverable since T1 is first to write and first to commit. The T2 abort does not count here as it is read only and it aborts. It is cascadeless as T2 is read only. It is strict as T1 is first to write. T2 has conflict between T1 write and T1 commit – but T2 aborts.

(e) S5 is recoverable, cascadeless and strict.

Question 4 (18 marks)

For each of the following sequences of operations, simulate its execution until it finishes or cannot proceed. If the execution cannot proceed, explain why.

For each step, indicate which of the two transactions T_1 and T_2 holds which type of lock on X and Y.

Note. Each schedule spans two lines of text.

(a) $S_1 : sl_1(X); r_1(X); ul_1(Y); r_1(Y); sl_2(Y); r_2(Y); sl_2(X); r_2(X); u_2(X); u_2(Y);$
$xl_1(Y); w_1(Y); u_1(Y); u_1(X)$ **(6 marks)**

(b) $S_2 : sl_1(X); r_1(X); sl_2(Y); r_2(Y); ul_1(Y); r_1(Y); sl_2(X); r_2(X); u_2(X); u_2(Y);$
$xl_1(Y); w_1(Y); u_1(Y); u_1(X)$ **(6 marks)**

(c) $S_3 : sl_2(Y); r_2(Y); sl_1(X); r_1(X); ul_1(Y); r_1(Y); xl_1(Y); w_1(Y); u_1(Y); u_1(X); sl_2(X); r_2(X); u_2(X); u_2(Y)$ **(6 marks)**

Solutions

(a) Simulation of S_1 (S=shared lock, X=exclusive lock, U=update lock):

Time	Operation	Locks on X	Locks on Y	
0				Initial state
1	$sl_1(X)$	T_1:S		
2	$r_1(X)$	T_1:S		
3	$ul_1(Y)$	T_1:S	T_1:U	
4	$r_1(Y)$	T_1:S	T_1:U	
5	$sl_2(Y)$	T_1:S	T_1:U	Lock request denied, stop

T_2's request for a shared lock on Y is denied, because T_1 already holds an update lock on Y at the time of the request.

(b) Simulation of S_2 (S=shared lock, X=exclusive lock, U=update lock):

Time	Operation	Locks on X	Locks on Y	
0				Initial state
1	$sl_1(X)$	T_1:S		
2	$r_1(X)$	T_1:S		
3	$sl_2(Y)$	T_1:S	T_2:S	
4	$r_2(Y)$	T_1:S	T_2:S	
5	$ul_1(Y)$	T_1:S	T_1:U, T_2:S	
6	$r_1(Y)$	T_1:S	T_1:U, T_2:S	
7	$sl_2(X)$	T_1:S, T_2:S	T_1:U, T_2:S	
8	$r_2(X)$	T_1:S, T_2:S	T_1:U, T_2:S	
9	$u_2(X)$	T_1:S	T_1:U, T_2:S	
10	$u_2(Y)$	T_1:S	T_1:U	
11	$xl_1(Y)$	T_1:S	T_1:U, T_1:X	
12	$w_1(Y)$	T_1:S	T_1:U, T_1:X	
13	$u_1(Y)$	T_1:S		
14	$u_1(X)$			Complete

(c) Simulation of S_3 (S=shared lock, X=exclusive lock, U=update lock):

Time	Operation	Locks on X	Locks on Y	
0				Initial state
1	$sl_2(Y)$		T_2:S	
2	$r_2(Y)$		T_2:S	
3	$sl_1(X)$	T_1:S	T_2:S	
4	$r_1(X)$	T_1:S	T_2:S	
5	$ul_1(Y)$	T_1:S	T_1:U, T_2:S	
6	$r_1(Y)$	T_1:S	T_1:U, T_2:S	
7	$xl_1(Y)$	T_1:S	T_1:U, T_2:S	Lock request denied, stop

T_1's request for an exclusive lock on Y is denied, because T_2 still holds a shared lock on Y at the time of the request.

Question 5 (10 marks)

For each of the following schedules, determine if the schedule is allowed by 2PL or not and explain your reasons for each schedule.

Note. Each schedule spans two lines of text.

(a) S_1: $l_1(X)$; $r_1(X)$; $w_1(X)$; $l_1(Y)$; $u_1(X)$; $l_2(X)$; $r_2(X)$; $r_1(X)$; $w_1(Y)$; $u_1(Y)$; $l_2(Y)$; $r_2(Y)$; $w_2(X)$; $w_2(Y)$; $u_2(X)$; $u_2(Y)$ **(2 marks)**

(b) S_2: $l_1(X)$; $l_1(Y)$; $r_1(X)$; $w_1(X)$; $u_1(X)$; $l_2(X)$; $r_2(X)$; $r_1(Y)$; $w_1(Y)$; $u_1(Y)$; $l_2(Y)$; $r_2(Y)$; $w_2(X)$; $u_2(X)$; $w_2(Y)$; $u_2(Y)$ **(2 marks)**

(c) S_3: $l_1(X)$; $r_1(X)$; $w_1(X)$; $u_1(X)$; $l_2(X)$; $r_2(X)$; $l_1(Y)$; $r_1(Y)$; $w_1(Y)$; $u_1(Y)$; $l_2(Y)$; $r_2(Y)$; $w_2(X)$; $w_2(Y)$; $u_2(X)$; $u_2(Y)$ **(2 marks)**

(d) S_4: $l_1(X)$; $r_1(X)$; $w_1(X)$; $u_1(X)$; $l_1(Y)$; $r_1(Y)$; $w_1(Y)$; $u_1(Y)$; $l_2(X)$; $r_2(X)$; $w_2(X)$; $u_2(X)$; $l_2(Y)$; $r_2(Y)$; $w_2(Y)$; $u_2(Y)$ **(2 marks)**

(e) S_5: $l_1(X)$; $r_1(X)$; $w_1(X)$; $l_1(Y)$; $u_1(X)$; $r_1(Y)$; $w_1(Y)$; $u_1(Y)$; $l_2(X)$; $r_2(X)$; $w_2(X)$; $l_2(Y)$; $u_2(X)$; $r_2(Y)$; $w_2(Y)$; $u_2(Y)$ **(2 marks)**

Solutions

To satisfy 2PL, all locks must precede unlocks. Also, note that in **(a)** this does not satisfy 2PL because T1 attempts to read X after it releases its lock on X, so the schedule breaks and so unlocks are incomplete. Therefore it doesn't satisfy 2PL conditions.

(a) , (c) and (d) S1, S3 and S4, respectively, are not allowed by 2PL.

(b) and (e) S2 and S5, respectively, are allowed by 2PL.

Question 6 (6 marks)

Examine the schedule given below. There are three transactions, T_1, T_2 and T_3. Again, assume that the transactions use shared buffers.

Initially, the value of $X = 1$ and $Y = 2$. The assignments happen within the local memory space of the transactions and the effects of these assignments are not reflected in the database until the commit operation.

Assume that the undo/redo algorithm with simple checkpoints is used and that the log records up to now are on disk.

Determine what recovery is needed for each of the transactions T_1, T_2 and T_3 if the system crashes with immediate effect at time $t = 13$ (at the end of line 13 but before the start of line 14).

(2 marks per transaction)

Time (t)	Transaction T_1	Transaction T_2	Transaction T_3
0			Start_transaction
1			read_item (Y)
2			$Y := Y + 1$
3	Start_transaction		
4	read_item (X)		
5	$X := X + 1$		
6			write_item (Y)
7			commit
8		Start_transaction	
9		read_item (Y)	
10		read_item (X)	
11		$Y := Y + X$	
12		write_item (Y)	
13		commit	
14	read_item (Y)		
15	$Y := Y + X$		
16	write_item (X)		
17	_____ checkpoint _____		
18	commit		

Solutions

T1 – UNDO, since it has started and committed when the crash happens. This is true even if T1 has not written anything yet on disk.

T2 – UNDO or REDO, ideal answer is to REDO (assume no commit at t13 (play it safe)).

T3 – REDO, since it had committed at the time of the crash.