Student ID: 201447766

COMP207 Assignment 1 – Transaction Management

Issue Date: Thursday, 31 October 2019

Submission Deadline: Thursday, 7 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, 7 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, 7 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)

Ans:

S_1: $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; $r_2(Y)$; $w_2(Y)$; $r_2(X)$; $w_2(X)$;

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

(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)

Ans:

In S_1, X=6 and Y=12;

In S_2, X=6 and Y=12.

(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)

Ans:

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

Explanation: S_1 and S_2 in (a) has the same effect as S_3 on every initial database state. Since S_1 is a serial schedule, S_3 is serializable.

(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)

Ans:

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

Explanation:

After executing the schedule S_4 on database, X=X+6 but Y=3*Y. Therefore, it means that S_4 does not have the same effect as serial schedules S_1 or S_2. Thus, 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)

Ans:

Conflict pairs in these schedules:

$S_1 : r_2(X)\text{-}w_1(X)\ ,\ r_3(Y)\text{-}w_2(Y)\ ,\ r_2(Z)\text{-}\ w_1(Z)$

$S_2 : w_1(Y)\text{-}r_2(Y)\ ,\ w_1(Y)\text{-}r_3(Y)\ ,\ w_3(U)\text{-}w_4(U)\ ,\ w_3(U)\text{-}r_4(U)\ ,\ w_2(Z)\text{-}r_4(Z)$

$S_3 : w_3(X)\text{-}r_1(X)\ ,\ w_1(Y)\text{-}r_2(Y)\ ,\ w_2(Z)\text{-}r_3(Z)$

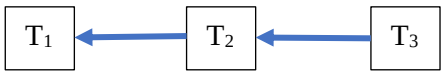
$S_4 : r_3(Y)\text{-}w_4(Y)\ ,\ w_4(Y)\text{-}r_1(Y)\ ,\ w_4(Y)\text{-}r_2(Y)\ ,\ w_4(Y)\text{-}\ w_2(Y)\ ,\ r_4(X)\text{-}\ w_1(X)\ ,\ r_3(X)\text{-}\ w_1(X)\ ,\ r_3(Y)\text{-}\ w_2(Y)\ ,\ r_1(Y)\text{-}\ w_2(Y)\ ,\ r_4(Y)\text{-}w_2(Y)$

$S_5 : w_1(Y)\text{-}\ r_2(Y)\ ,\ w_2(Z)\text{-}r_3(Z)\ ,\ r_1(X)\text{-}w_3(X)$

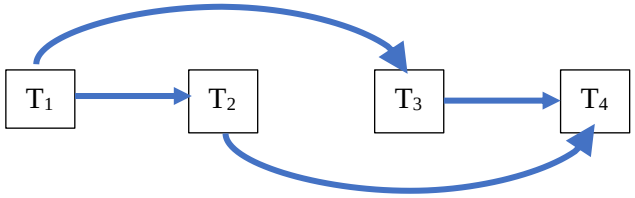
(b) What is the precedence graph of the schedule? (2 marks per schedule)

Ans:

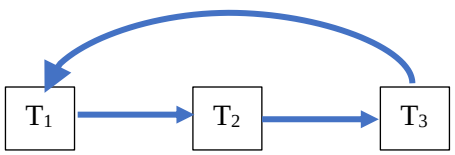
S_1 :

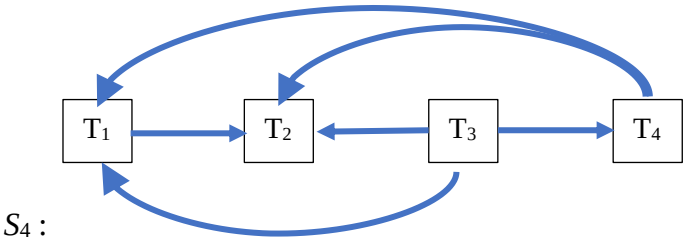


S_2 :

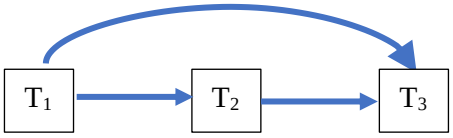


S_3 :





S_4 :



S_5 :

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

Ans:

S_1: S_1 is conflict-serialisable because it has no cycle in its precedence graph.

conflict-equivalent serial schedule of S_1:

$r_3(Y)$; $r_2(X)$; $r_2(Z)$; $r_2(Y)$; $w_2(Y)$; $r_1(X)$; $w_1(X)$; $w_1(Z)$

S_2: S_2 is conflict-serialisable because it has no cycle in its precedence graph.

conflict-equivalent serial schedule of S_2:

$r_1(X)$; $r_1(Y)$; $w_1(Y)$; $r_2(Y)$; $r_2(Z)$; $w_2(Z)$; $r_3(Y)$; $w_3(U)$; $r_4(Z)$; $r_4(U)$; $w_4(U)$

S_3: S_3 is not conflict-serialisable because it has a cycle in its precedence graph.

S_4: S_4 is conflict-serialisable because it has no cycle in its precedence graph.

conflict-equivalent serial schedule of S_4:

$r_3(Y)$; $r_3(X)$; $r_4(X)$; $w_4(Y)$; $r_4(Y)$; $r_1(X)$; $r_1(Y)$; $w_1(X)$; $r_2(Y)$; $w_2(Y)$

S_5: S_5 is conflict-serialisable because it has no cycle in its precedence graph.

conflict-equivalent serial schedule of S_5:

$r_1(X)$; $w_1(Y)$; $r_2(Y)$; $w_2(Z)$; $r_3(Z)$; $w_3(X)$

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)

Recoverable

Cascadeless

Not strict

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

Recoverable

Cascadeless

Not strict

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

Not recoverable

Not cascadeless

Not strict

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

Recoverable

Not cascadeless

Not strict

(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)

Recoverable

Cascadeless

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)

Time	T1	T2	Lock status	
			Locks owned by T1	Locks owned by T2
1	*s-lock(X)*		*s-lock(X)*	
2	*read_item(X)*		*s-lock(X)*	
3	*u-lock(Y)*		*s-lock(X)*, *u-lock(Y)*	
4	*read_item(Y)*		*s-lock(X)*, *u-lock(Y)*	
5		*s-lock(Y)* denied	*s-lock(X)*, *u-lock(Y)*	
6		*s-lock(X)*	*s-lock(X)*, *u-lock(Y)*	*s-lock(X)*
7		*read_item(X)*	*s-lock(X)*, *u-lock(Y)*	*s-lock(X)*
8		*unlock(X)*	*s-lock(X)*, *u-lock(Y)*	
9	*x-lock(Y)*		*s-lock(X)*, *u-lock(Y)*, *x-lock(Y)*	
10	*write_item(Y)*		*s-lock(X)*, *u-lock(Y)*, *x-lock(Y)*	
11	*unlock(Y)*		*s-lock(X)*	
12		*s-lock(Y)*	*s-lock(X)*	*s-lock(Y)*
13		*read_item(Y)*	*s-lock(X)*	*s-lock(Y)*
14		*unlock(Y)*	*s-lock(X)*	
15	*unlock(X)*			

(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)

Time	T1	T2	Lock status	
			Locks owned by T1	Locks owned by T2
1	*s-lock(X)*		*s-lock(X)*	
2	*read_item(X)*		*s-lock(X)*	
3		*s-lock(Y)*	*s-lock(X)*	*s-lock(Y)*
4		*read_item(Y)*	*s-lock(X)*	*s-lock(Y)*
5	*u-lock(Y)*		*s-lock(X)*, *u-lock(Y)*	*s-lock(Y)*
6	*read_item(Y)*		*s-lock(X)*, *u-lock(Y)*	*s-lock(Y)*

7		*s-lock*(*X*)	*s-lock*(*X*), *u-lock*(*Y*)	*s-lock*(*Y*), *s-lock*(*X*)
8		*read_item(X)*	*s-lock*(*X*), *u-lock*(*Y*)	*s-lock*(*Y*), *s-lock*(*X*)
9		*unlock(X)*	*s-lock*(*X*), *u-lock*(*Y*)	*s-lock*(*Y*)
10		*unlock(Y)*	*s-lock*(*X*), *u-lock*(*Y*)	
11	*x-lock*(*Y*)		*s-lock*(*X*), *u-lock*(*Y*), *x-lock*(*Y*)	
12	*write_item(Y)*		*s-lock*(*X*), *u-lock*(*Y*), *x-lock*(*Y*)	
13	*unlock(Y)*		*s-lock*(*X*)	
14	*unlock(X)*			

(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)

Time	T1	T2	Lock status	
			Locks owned by T1	Locks owned by T2
1		*s-lock*(*Y*)		*s-lock*(*Y*)
2		*read_item(Y)*		*s-lock*(*Y*)
3	*s-lock*(*X*)		*s-lock*(*X*)	*s-lock*(*Y*)
4	*read_item(X)*		*s-lock*(*X*)	*s-lock*(*Y*)
5	*u-lock*(*Y*)		*s-lock*(*X*), *u-lock*(*Y*)	*s-lock*(*Y*)
6	*read_item(Y)*		*s-lock*(*X*), *u-lock*(*Y*)	*s-lock*(*Y*)
7	*x-lock*(*Y*) denied		*s-lock*(*X*), *u-lock*(*Y*)	*s-lock*(*Y*)
8	*unlock(X)*		*u-lock*(*Y*)	*s-lock*(*Y*)
9		*s-lock*(*X*)	*u-lock*(*Y*)	*s-lock*(*X*), *s-lock*(*Y*)
10		*read_item(X)*	*u-lock*(*Y*)	*s-lock*(*X*), *s-lock*(*Y*)
11		*unlock(X)*	*u-lock*(*Y*)	*s-lock*(*Y*)
12		*unlock(Y)*	*u-lock*(*Y*)	
13	*x-lock*(*Y*)		*u-lock*(*Y*), *x-lock*(*Y*)	
14	*write_item(Y)*		*u-lock*(*Y*), *x-lock*(*Y*)	
15	*unlock(Y)*			

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)

Ans:

S_1 is allowed by 2PL because in each transaction of S_1, all lock operations precede all unlocks.

(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)

Ans:

S_2 is allowed by 2PL because in each transaction of S_1, all lock operations precede all unlocks.

(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)

Ans:

S_3 is not allowed by 2PL because in transaction T_1, $l_1(Y)$ executes after $u_1(X)$.

(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)

Ans:

S_4 is not allowed by 2PL because in transaction T_1, $l_1(Y)$ executes after $u_1(X)$.

(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)

Ans:

S_5 is allowed by 2PL because in each transaction of S_1, all lock operations precede all unlocks.

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		

Ans:

Log record on the disk:
<START T_3>
<START T_1>
<T_3, Y, 2, 3>
<COMMIT T_3>
<START T_2>
<T_2, Y, 3, 4>
<COMMIT T_2>

The recovery manager processes the log in reverse order:

Step	Current log record	Recovery manager's action
1	<COMMIT T_2>	Remember for later steps that T_2 has finished.
2	<T_2, Y, 3, 4>	Do nothing, because T_2 has finished.
3	<START T_2>	Do nothing.
4	<COMMIT T_3>	Remember for later steps that T_3 has finished.
5	<T_3, Y, 2, 3>	Do nothing, because T_3 has finished.
6	<START T_1>	Undo T_1.
7	<START T_3>	Do nothing.

The final log on disk looks like this:
<START T_3>
<START T_1>
<T_3, Y, 2, 3>
<COMMIT T_3>
<START T_2>
<T_2, Y, 3, 4>
<COMMIT T_2>
<ABORT T_1>

Conclusion:
T_1: There is need to apply undo/redo recovery.
T_2: There is no need to apply undo/redo recovery.
T_3: There is no need to apply undo/redo recovery.