COMP207 Assignment 1 – Transaction Management

ID Number: 201447569

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)

Answer:

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

S2: r2(Y);w2(Y);r2(X);w2(X);c2;r1(X);w1(X);r1(Y);w1(Y);c1

S1:

Begin T1	
read item(X)	
X := X + 2	
write item(X)	
read_item(Y)	
Y := Y * 3	
write item(Y)	
commit	
End(T1)	
	Begin T2
	read item(Y)
	Y := Y * 2
	write item(Y)
	read item(X)
	X := X + 3

	write item(X)
	commit
	End(T2)

S2:

Begin T2	
read item(Y)	
Y := Y ∗ 2	
write item(Y)	
read item(X)	
X := X + 3	
write item(X)	
commit	
End(T2)	
	Begin T1
	read item(X)
	X := X + 2
	write item(X)
	read_item(Y)
	Y := Y ∗ 3
	write item(Y)
	commit
	End(T1)

(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)

Answer:

S1:

X = 6	Y = 12

S2:

X = 6	Y = 12

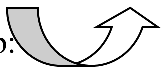
(c) Give a serialisable schedule S_3 for T_1 and T_2 that is not serial. Explain why your schedule S_3 is serialisable. (5 marks)

Answer:

Because serialisable is difficult to verify, we could find a conflict-serialisable to meet all requirement of serialisable.

So we use S1 to change it to the conflict-serialisable schedule: (We omit the non-database operation first)

Note:

Means to swap: 

S1: r1(X);w1(X);r1(Y);w1(Y);r2(Y);w2(Y);r2(X);w2(X)

Notice that the calculation of x and y only related to itself. We change the order of X and Y in T1, it is obviously that T1 is not changed result when we change the Y to X)

S1': r1(Y);w1(Y); r1(X);w1(X); r2(Y);w2(Y);r2(X);w2(X)

S1': r1(Y);w1(Y); r1(X); r2(Y); w1(X); w2(Y);r2(X);w2(X)

S1': r1(Y);w1(Y); r2(Y); r1(X); w1(X); w2(Y);r2(X);w2(X)

S1': r1(Y);w1(Y); r2(Y); r1(X); w2(Y);w1(X); r2(X);w2(X)

S3: r1(Y);w1(Y); r2(Y); w2(Y); r1(X); w1(X); r2(X);w2(X)

Because S3 is a conflict-serialisable schedule, so it must be serialisable. And it is not serial.

Adding non-database operation, we get this:

S3:

Begin T1	
read item(Y)	
Y := Y ∗ 2	
write item(Y)	
	Begin T2
	read_item(Y)
	Y := Y ∗ 3
	write item(Y)
read item(X)	
X := X + 2	
write item(X)	
End(T1)	
	read item(X)
	X := X + 3
	write item(X)
	End(T2)
Commit	
	Commit

(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)

Answer:

S4: r1(Y);r2(Y);w1(Y);w2(Y);r1(X);r2(X);w1(X);w2(X)

Assume that X = 1 and Y = 2 initially, the result is X = 1 + 3 = 4 ≠ 12; Y = 4 ≠ 6. So, result is different between serial schedule which is not serialisable. Or we can see the procedure diagram has circuit means not conflict- serializable.

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)

$S_1 : r_1(X);\ \boxed{r_2(X)};\ \boxed{r_3(Y)};\ \boxed{w_1(X)};\ \boxed{r_2(Z)};\ r_2(Y);\ \boxed{w_2(Y)};\ \boxed{w_1(Z)}$

r2(X)-w1(X);r3(Y)-w2(Y);r2(Z)-w1(Z)

$S_2 : r_1(X);\ r_1(Y);\ \boxed{w_1(Y)};\ \boxed{r_3(Y)};\ \boxed{r_2(Y)};\ r_2(Z);\ \boxed{w_3(U)};\ \boxed{w_2(Z)};\ \boxed{r_4(Z)};\ \boxed{r_4(U)};\ \boxed{w_4(U)}$

w1(Y)-r3(Y);w1(Y)-r2(Y);w3(U)-r4(U);w3(U)-w4(U);w2(Z)-r4(Z)

$S_3 : \boxed{w_3(X)};\ \boxed{r_1(X)};\ \boxed{w_1(Y)};\ \boxed{r_2(Y)};\ \boxed{w_2(Z)};\ \boxed{r_3(Z)}$

w3(X)-r1(X);w1(Y)-r2(Y);w2(Z)-r3(Z)

$S_4 : r_1(X);\ \boxed{r_4(X)};\ \boxed{r_3(Y)};\ \boxed{w_4(Y)};\ \boxed{r_1(Y)};\ \boxed{r_2(Y)};\ \boxed{r_3(X)};\ \boxed{r_4(Y)};\ \boxed{w_1(X)};\ \boxed{w_2(Y)}$

r4(X)-w1(X);r3(Y)-w4(Y);r3(Y)-w2(Y);w4(Y)-r1(Y);w4(Y)-r2(Y);w4(Y)-w2(Y);r1(Y)-w2(Y);r3(X)-w1(X);r4(Y)-w2(Y)

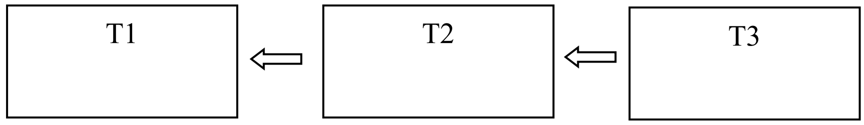
$S_5 : \boxed{r_1(X)};\ \boxed{w_1(Y)};\ \boxed{r_2(Y)};\ \boxed{w_2(Z)};\ \boxed{r_3(Z)};\ \boxed{w_3(X)}$

r1(X)-w3(X);w1(Y)-r2(Y);w2(Z)-r3(Z)

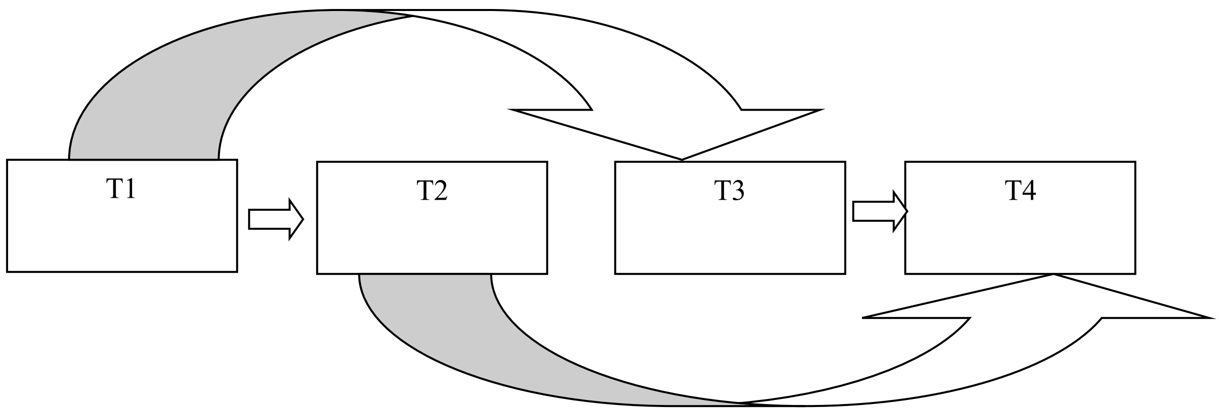
Note the line between each action is a pair of conflicts

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

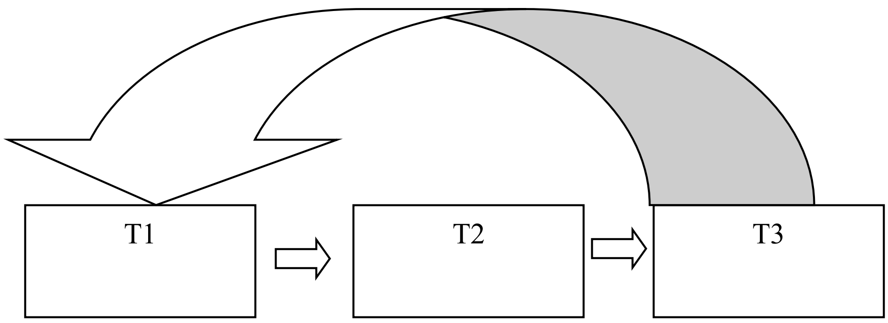
S1



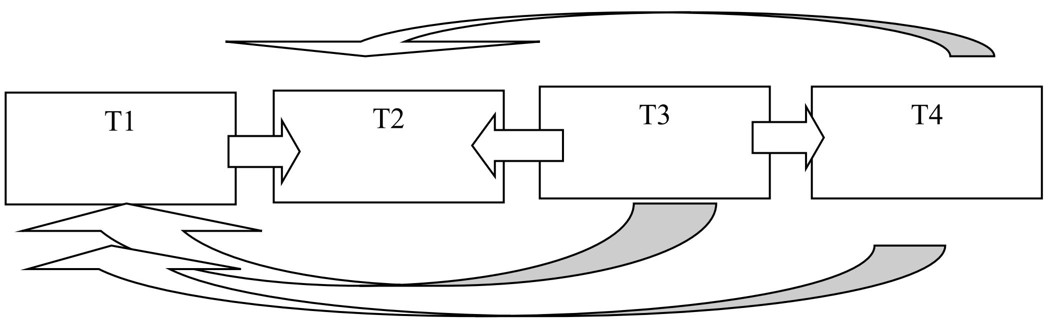
S2



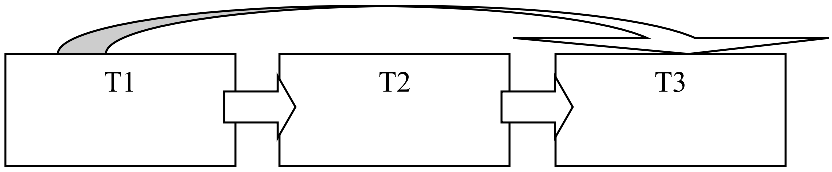
S3



S4



S5



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

S1: conflict-serialisable

S1' = r3(Y);r2(X);r2(Z);r2(Y);w2(Y);r1(X);w1(X);w1(Z)

S2: conflict-serialisable

S2' = r1(X);r1(Y);w1(Y);r2(Y);r2(Z);w2(Z);r3(Y);w3(U);r4(Z);r4(U);w4(U)

S3: Not conflict-serialisable because of circle

S4: conflict-serialisable

S4' = r3(Y);r3(X);r4(X);w4(Y);r4(X);r1(X);r1(Y);w1(X);r2(Y);w2(Y)

S5: conflict-serialisable

S5' = r1(X);w1(Y);r2(Y);w2(Z);r3(Z);w3(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) $S_1 : sl_1(X); r_1(X); \boxed{ul_1(Y);} r_1(Y); \boxed{sl_2(Y);} r_2(Y); sl_2(X); r_2(X); u_2(X); u_2(Y);$
$xl_1(Y); w_1(Y); u_1(Y); u_1(X)$ (6 marks)

Cannot proceed, when T1 use update lock, the T2 needs to wait T1 unlock, so T2 cannot add a sharelock.

$sl_1(X)$	T1 holds share lock of X

$r_1(X)$	T1 holds share lock of X
$ul_1(Y)$	T1 holds update lock of Y; T1 holds share lock of X
$r_1(Y)$	T1 holds update lock of Y; T1 holds share lock of X
$sl_2(Y)$	Cannot proceed, this must be refused
$r_2(Y)$	
$sl_2(X)$	
$r_2(X)$	
$u_2(X)$	
$u_2(Y)$	
$xl_1(Y)$	
$w_1(Y)$	
$u_1(Y)$	
$u_1(X)$	

(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)

it finishes

$sl_1(X)$	T1 holds share lock of X
$r_1(X)$	T1 holds share lock of X
$sl_2(Y)$	T1 holds share lock of X; T2 holds share lock of Y
$r_2(Y)$	T1 holds share lock of X; T2 holds share lock of Y
$ul_1(Y)$	T1 holds share lock of X and update lock of Y; T2 holds share lock of Y
$r_1(Y)$	T1 holds share lock of X and update lock of Y; T2 holds share lock of Y
$sl_2(X)$	T1 holds share lock of X and update lock of Y; T2 holds share lock of Y and share lock of X
$r_2(X)$	T1 holds share lock of X and update lock of Y; T2 holds share lock of Y and share lock of X
$u_2(X)$	T1 holds share lock of X and update lock of Y; T2 holds share lock of Y
$u_2(Y)$	T1 holds share lock of X and update lock of Y
$xl_1(Y)$	T1 holds share lock of X and Exclusive lock of Y
$w_1(Y)$	T1 holds share lock of X and Exclusive lock of Y
$u_1(Y)$	T1 holds share lock of X
$u_1(X)$	

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

(6 marks)

Cannot proceed, Y needs to be unlocked before T1 add an Exclusive lock

$sl_2(Y)$	T2 holds share lock of Y
$r_2(Y)$	T2 holds share lock of Y
$sl_1(X)$	T1 holds share lock of X; T2 holds share lock of Y
$r_1(Y)$	T1 holds share lock of X; T2 holds share lock of Y
$ul_1(Y)$	T1 holds share lock of X and update lock of Y; T2 holds share lock of Y
$r_1(Y)$	T1 holds share lock of X and update lock of Y; T2 holds share lock of Y
$xl_1(Y)$	Cannot proceed, this must be refused
$w_1(Y)$	
$u_1(Y)$	
$u_1(X)$	
$sl_2(X)$	
$r_2(X)$	
$u_2(X)$	
$u_2(Y)$	

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: $\boxed{l_1(X); r_1(X); w_1(X); l_1(Y); u_1(X)}; l_2(X); r_2(X); \boxed{r_1(X); w_1(Y); u_1(Y);} l_2(Y); r_2(Y);$
$w_2(X); w_2(Y); u_2(X); u_2(Y)$ (2 marks)

NOT 2PL, T2 holds a lock, r1(X) cannot process in simple locking mechanism

(b) S_2: $\boxed{l_1(X); l_1(Y); r_1(X); w_1(X); u_1(X)}; l_2(X); r_2(X); \boxed{r_1(Y); w_1(Y); u_1(Y);} l_2(Y); r_2(Y);$
$w_2(X); u_2(X); w_2(Y); u_2(Y)$ (2 marks)

2PL, two phrase are shown in the Figure

(c) S_3: $\boxed{l_1(X); r_1(X); w_1(X); u_1(X);} l_2(X); r_2(X); \boxed{l_1(Y); r_1(Y); w_1(Y); u_1(Y);} l_2(Y); r_2(Y);$
$w_2(X); w_2(Y); u_2(X); u_2(Y)$ (2 marks)

Not 2PL, it is lock -unlock and lock in T1- unlock rather than lock lock unlock unlock

(d) S_4: $\boxed{l_1(X); r_1(X); w_1(X); u_1(X); l_1(Y); r_1(Y); w_1(Y); u_1(Y);} l_2(X); r_2(X); w_2(X); u_2(X);$
$l_2(Y); r_2(Y); w_2(Y); u_2(Y)$ (2 marks)

Not 2PL, it is lock -unlock and lock in T1- unlock rather than lock lock unlock unlock

(e) S_5: $\boxed{l_1(X); r_1(X); w_1(X); l_1(Y); u_1(X); r_1(Y); w_1(Y); u_1(Y);} l_2(X); r_2(X); w_2(X); l_2(Y);$
$u_2(X); r_2(Y); w_2(Y); u_2(Y)$ (2 marks)

2PL, two phrase are shown in the Figure

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		

T1 is not commit and it does not write anything to the buffer or Database and the logging has not written in to the disk. So, it does not need to recovery;

T2 has commit means that T2 has done all expect work, so it doesn't need to recovery;

T3 has commit means that T3 has done all expect work, so it doesn't need to recovery;