

Assignment 4

DUONG NGUYEN

1. (EX 16.2)16.2.1. $T_2 : R(Y), W(Y)$ without concurrency control, T_2 could interfere with T_1 $T_1 : R(X) W(X)$ $R(Y)$ $W(Y)$ $T_2 :$ $R(Y)$ $W(Y)$

} last update.

16.2.2.

with strict 2PL locking, either T_1 or T_2 will obtain exclusive lock on Y first.If T_1 obtains firstIf T_2 obtains first

T_1	T_2
$X(X)$	
$R(X)$	
$X(Y)$	
$R(Y)$	
$W(Y)$	
Commit	

$X(Y)$
 $R(Y)$
 $W(Y)$
 Commit

T_1	T_2
$X(X)$	
$R(X)$	
$W(X)$	
	$X(Y)$
	$R(Y)$
	$W(Y)$
	Commit
$X(Y)$	
$R(Y)$	
$W(Y)$	
Commit	

2. (EX. 17.2)

1, Serializable: unknown.

If both T_1 and T_2 commit $\Rightarrow$ No

If T_1 aborts, T_2 commit $\Rightarrow$ YES ($T_1(\text{abort}) \rightarrow T_2$)

T_1	T_2
RC(X)	RC(X)
WC(X)	WC(X)

Conflict Serializable: unknown.

If both commit $\Rightarrow$ precedence graph is



$\Rightarrow$ cycle $\Rightarrow$ No

If either transaction aborts $\Rightarrow$ YES.

View - Serializable: unknown

If both T_1 and T_2 commit $\Rightarrow$ No.

If T_1 abort, T_2 commit $\Rightarrow$ YES ($T_1(\text{abort}) \rightarrow T_2$)

recoverable: YES

Avoid-cascading abort: YES

Strict: No.

Since X written by T_1 is overwritten by T_2 before T_1 commits/aborts

2, Serializable: YES

($T_1 \rightarrow T_2$)

T_1	T_2
WC(X)	RC(Y)
RC(Y)	RC(X)

Conflict Serializable: YES

Even both

commit, graph is a cycle $T_1 \rightarrow T_2$

View Serializable: YES

($T_1 \rightarrow T_2$)

Recoverable: unknown

If T_2 commits after T_1 commits $\Rightarrow$ YES

If T_2 commits before T_1

$\Rightarrow$ No

Avoid Cascading Abort: No.

Since T_2 read X written by T_1 before T_1 commit/abort.

Strict: No

Since X written by T_1 is read by T_2 before T_1 commits.

5, Serializable: YES

($T_1(\text{abort}) \rightarrow T_2 \equiv T_2$)

T_1	T_2
RC(x)	W(x)
W(x)	Abort
Com.	

Conflict - Serializable: YES

graph has only 1 node $\Rightarrow$ acyclic

View - Serializable: YES

($T_1 \rightarrow T_2 \equiv T_2$)

recoverable: YES

Avoid cascading abort: YES

Strict: No

Since X written by T_2 is overwritten by T_1 before T_2 aborts.

9, Serializable: YES

($T_1(\text{abort}) \rightarrow T_2 \equiv T_2$)

T_1	T_2
W(x)	RC(x)
W(x)	Com.
Abort	

Conflict - Serializable: YES

graph has only one node $\Rightarrow$ acyclic

View Serializable: YES

($T_1(\text{abort}) \rightarrow T_2 \equiv T_2$)

recoverable: No

Since T_2 read X ~~for~~ written by T_1 while T_2 commits before T_1 abort.

Avoid cascading abort: No

Since T_2 commits before T_1 abort while T_2 read X written by T_1

Strict:

No

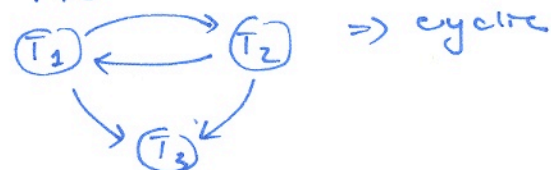
Since T_2 read X from T_1 before T_1 commits.

Serializable: No.

Due to conflict between T_1 and T_2

Conflict serializable: No

Graph



View serializable: No.

Since T_3 read x written by $T_1 \Rightarrow$

T_3 must follow T_1 in serial

Schedule $T_1 \rightarrow T_3$.

T_2 can't be at the middle since

T_3 will read x from T_2

T_2 can't be at the end since

T_2 will be last write of x

T_2 can't be at front since

T_1 will not read x from initial value

$\Rightarrow$ can't find equivalent serial schedule.

1 1,

T_1	T_2	T_3
R(x)		
	W(x)	
	Com.	
W(x)		
Com.		
		R(x)
		Com.

recoverable
avoid cascading
abort
Strict

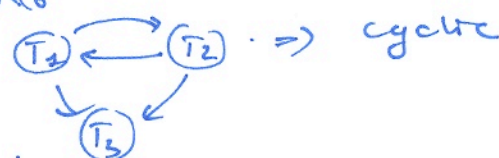
} YES

Serializable: No

Due to conflict between T_1 and T_2

Conflict serializable: No

graph



View serializable: No

Explain as the case of (11)

Recoverable: YES

T_3 read x from T_1, T_2 and commits after T_1, T_2 commits.

Avoid cascading abort: No

Since T_3 read x written by T_1 before T_1 commit.

Strict: No

Since x written by T_2 is overwritten by T_1 before T_2 commits.

1 2,

T_1	T_2	T_3
R(x)		
	W(x)	
W(x)		
Com		R(x)
	Com	
		Com

3. (EX. 17.8.1)

If the second query is $+ \text{Salary}$

Replace ($\text{Salary} = 100$) where $\text{EMP.ename} = \text{'Santa'}$;

Then it could conflict with the provided query.

For example, suppose table EMP has 1 'Santa' tuple

<u>eid</u>	<u>ename</u>	<u>age</u>	<u>salary</u>	<u>dnd</u>
15	'Santa'	30	20	50

If we execute query 1 and query 2 serially, the result will be

<u>Q₁</u>	<u>Q₂</u>					
R(EMP)		$\Rightarrow$				
W(EMP)		<table><tr><td><u>ename</u></td><td><u>salary</u></td></tr><tr><td>'Santa'</td><td>1002</td></tr></table>	<u>ename</u>	<u>salary</u>	'Santa'	1002
<u>ename</u>	<u>salary</u>					
'Santa'	1002					
	R(EMP)					
	W(EMP)					

<u>Q₁</u>	<u>Q₂</u>		<u>ename</u>	<u>Salary</u>
	R(EMP)	$\Rightarrow$	'Santa'	120
	W(EMP)			
R(EMP)				
W(EMP)				

If we execute the 2 queries concurrently, it may be

<u>Q₁</u>	<u>Q₂</u>					
R(EMP)		$\Rightarrow$				
	R(EMP)	<table> <tr> <th><u>ename</u></th> <th><u>Salary</u></th> </tr> <tr> <td>'Santa'</td> <td>120</td> </tr> </table>	<u>ename</u>	<u>Salary</u>	'Santa'	120
<u>ename</u>	<u>Salary</u>					
'Santa'	120					
W(EMP)		$\Rightarrow$ different from both serial				
	W(EMP)	Schedule!				

Using locking rules, then Q₂ can't read until Q₁ releases the exclusive lock or Q₂ will hold the exclusive lock until it is done and releases to Q₁. Thus the result will be similar to one of the 2 serial schedule.