Free University of Bozen
Faculty of Computer Science
Alessandro Artale

Databases 2
Revision Exercise
Recovery Failure and Concurrency

1 Transactions and Log Files

You have been given the database elements X, Y, Z with starting values: $X = 4, Y = 5, Z = 1$, and two transactions T_1, T_2:

$\mathbf{T_1}$. $X := 2 * X; Y := Y + X$
$\mathbf{T_2}$. $Z := 2 * Z; X := X - Z$

Assume that transaction T_2 starts after T_1 finishes, and that an UNDO logging technique is used to recover from a system failure. Answer the following questions:

1.a. Give a table containing: a) the primitive actions for both T_1 and T_2 in the appropriate sequential order (when giving the actions use the primitive actions READ, WRITE, OUTPUT, FLUSH LOG, together with the required assignment statements for the local variables.); b) the values of the local variables; c) the values for X, Y, Z both in main memory and on disk; d) the status of the log in main memory.

1.b. Give a table like in part 1.a assuming that the recovery technique is a REDO logging technique, and the execution order is inverted, i.e., transaction T_1 starts after T_2 finishes.

2 Recovery from System Failure

Consider the following log file (to be read from left to right):

$< \texttt{START T}_1 > \ < \texttt{T}_1, \texttt{X}, 0 > \ < \texttt{T}_1, \texttt{Y}, 0 > \ < \texttt{START T}_2 > \ < \texttt{T}_2, \texttt{Z}, 0 > \ < \texttt{T}_1, \texttt{W}, 5 > \ < \texttt{COMMIT T}_1 >$
$< \texttt{START T}_3 > \ < \texttt{T}_3, \texttt{Y}, 10 > \ < \texttt{START T}_4 > \ < \texttt{T}_4, \texttt{X}, 5 > \ < \texttt{COMMIT T}_2 > \ < \texttt{T}_4, \texttt{W}, 10 >$
$< \texttt{T}_4, \texttt{V}, 5 > \ < \texttt{COMMIT T}_4 > \ < \texttt{T3}, \texttt{V}, 10 > \ < \texttt{COMMIT T}_3 >$

Suppose that we start a Nonquiescent Checkpoint under an UNDO logging technique immediately after one of the following log records has been written:

i. $< \texttt{T}_1, \texttt{Y}, 0 >$

ii. $< \texttt{T}_2, \texttt{Z}, 0 >$

iii. $< \texttt{START T}_3 >$

iv. $< \texttt{COMMIT T}_2 >$

v. $< \texttt{T}_4, \texttt{V}, 5 >$

2.a. For each of the above points, complete the log file with the insertion of START CKPT and END CKPT log records.

Consider now the log file as after the above completion under point ii. Suppose there is a crash. Using the primitive actions: WRITE, OUTPUT, ABORT, give the actions that need to be done to recover from the crash in each of the following situations:

2.b. The last record in the log file is $< \texttt{COMMIT T}_4 >$.

2.c. The last record in the log file is $< \texttt{T}_4, \texttt{X}, 5 >$.

Furthermore, for each of the above points, give the new log file after deleting useless portions.

Suppose now that we start a Nonquiescent Checkpoint under a REDO logging technique and the log file is as follows:

$< \texttt{START T}_1 > \ < \texttt{T}_1, \texttt{X}, 3 > \ < \texttt{T}_1, \texttt{Y}, 5 > \ < \texttt{START T}_2 > \ < \texttt{T}_2, \texttt{Z}, 12 > \ < \texttt{T}_1, \texttt{W}, 7 > \ < \texttt{COMMIT T}_1 >$
$< \texttt{START T}_3 > \ < \texttt{T}_3, \texttt{Y}, 16 > \ < \texttt{START CKPT}(\texttt{T}_2, \texttt{T}_3) > \ < \texttt{START T}_4 > \ < \texttt{T}_4, \texttt{X}, 32 > \ < \texttt{COMMIT T}_2 >$
$< \texttt{T}_4, \texttt{W}, 10 > \ < \texttt{END CKPT} >< \texttt{T}_4, \texttt{V}, 28 > \ < \texttt{COMMIT T}_4 > \ < \texttt{T3}, \texttt{V}, 14 > \ < \texttt{COMMIT T}_3 >$

Using the primitive actions: WRITE, OUTPUT, ABORT, give the actions that need to be done to recover from a crash in each of the following situations:

2.d. The last record in the log file is $< \texttt{COMMIT T}_4 >$.

2.e. The last record in the log file is $< \texttt{T}_4, \texttt{X}, 32 >$.

As in the UNDO case, for each of the above points, give also the new log file after deleting useless portions.

3 Concurrency Control

For each of the following two schedules:

$\mathbf{S}1.$ $\texttt{r}_3(\texttt{X}); \texttt{r}_1(\texttt{Y}); \texttt{r}_2(\texttt{Z}); \texttt{r}_2(\texttt{W}); \texttt{r}_3(\texttt{Y}); \texttt{r}_1(\texttt{Z}); \texttt{w}_1(\texttt{Y}); \texttt{r}_3(\texttt{W}); \texttt{w}_3(\texttt{W}); \texttt{w}_2(\texttt{Y}); \texttt{w}_2(\texttt{Z}); \texttt{w}_3(\texttt{X})$.
$\mathbf{S}2.$ $\texttt{r}_4(\texttt{Y}); \texttt{r}_2(\texttt{X}); \texttt{w}_2(\texttt{X}); \texttt{r}_1(\texttt{X}); \texttt{w}_4(\texttt{Y}); \texttt{r}_1(\texttt{Y}); \texttt{r}_2(\texttt{Z}); \texttt{w}_1(\texttt{Y}); \texttt{w}_2(\texttt{Z}); \texttt{r}_1(\texttt{Z}); \texttt{r}_3(\texttt{Y}); \texttt{w}_1(\texttt{Z}); \texttt{w}_3(\texttt{X})$.

Answer the following questions:

3.a. Give the precedence graph and check whether the schedule is conflict-serializable. Justify each precedence relation in the graph, i.e., for each arch in the obtained graph show a conflicting pair of actions which gives rise to such an arch.

3.b. If the schedule is conflict-serializable show all the admissible (given the precedence graph) serial schedules.

Given now the following schedule:

$\mathbf{S}.$ $\texttt{r}_2(\texttt{X}); \texttt{r}_4(\texttt{Y}); \texttt{r}_1(\texttt{X}); \texttt{w}_4(\texttt{Y}); \texttt{r}_2(\texttt{Z}); \texttt{r}_1(\texttt{Y}); \texttt{r}_3(\texttt{Y}); \texttt{w}_3(\texttt{Y}); \texttt{w}_2(\texttt{X})$.

3.c. Assuming that the scheduler deals with Two-Phase Locking with shared and exclusive locks and allowing upgrading (i.e., a shared lock can be upgraded to an exclusive lock), add shared and exclusive locks and unlocks to $\mathbf{S}$. Insert each shared and exclusive lock as delayed as possible, while unlock as soon as possible (always in accordance with the Two-Phase Locking strategy). Give $\mathbf{S}$ in a table with four columns T_1, T_2, T_3, T_4.

Given now the following schedule:

$\mathbf{S}.$ $\texttt{r}_4(\texttt{Y}); \texttt{r}_2(\texttt{X}); \texttt{r}_1(\texttt{X}); \texttt{w}_4(\texttt{Y}); \texttt{r}_1(\texttt{Y}); \texttt{r}_2(\texttt{Z}); \texttt{w}_2(\texttt{X}); \texttt{w}_1(\texttt{Y}); \texttt{r}_1(\texttt{Z}); \texttt{r}_3(\texttt{Z}); \texttt{r}_3(\texttt{X}); \texttt{w}_1(\texttt{Z}); \texttt{w}_3(\texttt{Z})$

3.d. Assume that the scheduler deals with Two-Phase Locking with shared and exclusive locks and allowing upgrading (i.e., a shared lock can be upgraded to an exclusive lock). Adding shared and exclusive locks and unlocks to $\mathbf{S}$, show that the schedule end with a deadlock and must be aborted.