1) Concurrency control

a) Serializability

Consider the following schedule:

\[ r_1(A) ; w_1(A) ; r_2(A) ; w_2(A) ; r_1(B) ; w_1(B) ; r_2(B) ; w_2(B) ; \]

Is this schedule conflict-serializable? Demonstrate by trying to show a possible serial schedule.

Solution:

In each of the following step, we swap two executions which have no conflicts among themselves (hence each schedule has the same conflicts)

\[ r_1(A) ; w_1(A) ; r_2(A) ; w_2(A) ; r_1(B) ; w_1(B) ; r_2(B) ; w_2(B) ; \]

\[ r_1(A) ; w_1(A) ; r_2(A) ; w_2(A) ; w_1(B) ; r_2(B) ; w_2(B) ; \]

\[ r_1(A) ; w_1(A) ; r_1(B) ; r_2(A) ; w_2(A) ; w_1(B) ; r_2(B) ; w_2(B) ; \]

\[ r_1(A) ; w_1(A) ; r_1(B) ; r_2(A) ; w_2(A) ; w_1(B) ; r_2(B) ; w_2(B) ; \]

\[ r_1(A) ; w_1(A) ; r_1(B) ; w_1(B) ; r_2(A) ; w_2(A) ; r_2(B) ; w_2(B) ; \]

\[ => \text{the schedule is conflict serializable (can also be shown using the serializability graph).} \]

b) Graphs (serializability and waiting)

Consider the example from lecture (page 21):

\[
\begin{array}{ccc}
T1 & T2 & T3 \\
W1(x) & & \\
 & R2(x) & \\
W1(y) & & \\
W1(z) & & R3(z) \\
 & W2(y) & \\
 & & W3(y) \\
 & & W3(z) \\
\end{array}
\]

If you were to use 2PL, what would the waiting graph for this sequence of R/W look like? Draw your conclusions by comparing it with the serializability graph given in lecture.
c) Locks

In the figure given below, you see schedule of four transactions.

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slock A</td>
<td>Slock A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xlock B</td>
<td>Unlock A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlock A</td>
<td>Xlock A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlock B</td>
<td>Slock B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slock A</td>
<td>Unlock A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlock B</td>
<td>Unlock B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xlock C</td>
<td>Unlock A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlock A</td>
<td>Xlock A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlock B</td>
<td>Unlock C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

i) Assume that write-locks imply reading. Draw the serialization graph and determine whether the schedule is serializable.

Solution:

2) Recovery
After a system’s crash, the undo-log using non-quiescent checkpointing contains the following data:

\[
<\text{START T1}> \\
<T1, X1, 1> \\
<\text{START CKPT} \text{ ????>} \\
<\text{START T2}> \\
<T2, X2, 2> \\
<T1, X1, 3> \\
<\text{START T3}> \\
<\text{END T1}> \\
<\text{END CKPT}> \\
<\text{START CKPT} \text{ ????>} \\
<T2, X2, 4> \\
<T3, X3, 5> \\
<\text{START T4}> \\
<\text{END T2}> \\
<T4, X4, 6> \\
<\text{END T3}> \\
<\text{END CKPT}> \\
<\text{START T5}> \\
<T5, X5, 7> \\
<\text{START CKPT} \text{ ????>} \\
<T4 X4, 8> \\
\text{CRASH} !!!
\]

i) What are the correct values of the three \text{<START CKPT ????>} records? You have to provide three correct values for the three “????”s.

Solution:

T1
T2, T3
T4, T5

j) Assuming that the three \text{<START CKPT ????>} records are correctly stored in the log, according to your answer at a., show which elements are recovered by the undo recovery manager and compute their values after recovery.
Solution:

Recovery operations:
X4: 8
X5: 7
X4: 6

Final values of X4 = 6, X5 = 7

k) Indicate what fragment of the log the recovery manager needs to read.

Solution:

The portion of the log from second checkpoint (<start ckpt T2T3> ) till the end is used.