Exercise 12 “Implementation of Databases”

Due until 30 Jan, 2008

WS 07/08

Note: this exercise is optional.

**Exercise 12.1 [Concurrency Control & Recovery]**:

Judge whether the following statements are true or false:

1. A schedule of transactions T1 and T2 that produces the same output as a serial execution in the order \(< T1, T2 >\) is serializable.

2. Conflicting operations are operations from different transactions which operate on the same data item and one of them changes the data item.

3. Using strict 2PL, a serial schedule is always slower than any other schedule.

4. Two schedules are the same if they contain the same transactions and operations and if they order all conflicting operations of non-aborting transactions in the same way.

5. A schedule can be tested for serializability using a precedence graph. While creating that graph uncommitted transactions must be respected.

6. Using 2PL if a transaction requests a lock that cannot be granted (due to a lock conflict), that transaction is blocked until all the conflicting locks held by other transactions are released.

7. 2PL does not guarantee serializability.

8. The lock manager of a DBMS is responsible to implement 2PL and to deal with deadlock situations.

9. The concept of degrees of isolation or isolation levels has been developed to allow transactions to trade concurrency for consistency in a controlled manner.

10. We prefer deadlock detection to deadlock prevention, because the latter reduces the throughput of DBMS and deadlocks are infrequent in practice.

11. Recovery techniques are employed by DBMSs to guarantee the isolation and durability transactional properties.

12. In REDO logging, all actions of a transaction must be flushed to disk before the commit record is written to the log.

13. In UNDO/REDO logging, no modified data can be written to disk before the corresponding log records have been written to disk.

14. One possibility to perform a savepoint is to suspend execution of transactions, flush all modified memory buffers to disk, write a savepoint record to the log, and resume transaction execution.
15. The write-ahead logging principle (WAL) states that all log records for a given updated page should be written to stable storage before the page itself is written and that a transaction is only considered to be committed after all its log records have been written to stable storage.

16. When a no-force and no-steal strategy is used, one must implement schemes both for the redo of operations from committed transactions whose changes have not yet been written to the database and for the undo of operations from uncommitted transactions that have already updated the database.

17. Physical logging implies that only actions, not images, processed by the system are written to the log.

18. For correct behavior during recovery, redo and undo operations must be idempotent.

19. When action-consistent checkpoints are employed, the redo and undo phases of recovery only need to consider log records written after the last checkpoint.

20. Compared to consistent checkpointing, fuzzy checkpointing requires more scanning of logs during recovery, but exhibits better performance when performing checkpointing.

Exercise 12.2[ARIES] : (10 pt.)

Consider the recovery scenario described in the following. There are three transactions $T_1$, $T_2$, and $T_3$. $T_1$ updates page $C$, $T_2$ updates pages $B$ and $C$, and $T_3$ updates page $A$. At the time of crash, we have the following contents for the log:

<table>
<thead>
<tr>
<th>LSN</th>
<th>LAST_LSN</th>
<th>TRAN_ID</th>
<th>TYPE</th>
<th>PAGE_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>T1</td>
<td>update</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>T2</td>
<td>update</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>T1</td>
<td>commit</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>begin CKPT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>end CKPT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>T3</td>
<td>update</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>T2</td>
<td>update</td>
<td>C</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>T2</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

The transaction table and dirty page table for the checkpoint are:

<table>
<thead>
<tr>
<th>TRANSACTION_ID</th>
<th>LAST_LSN</th>
<th>STATUS</th>
<th>PAGE_ID</th>
<th>LSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>3</td>
<td>commit</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>T2</td>
<td>2</td>
<td>active</td>
<td>B</td>
<td>2</td>
</tr>
</tbody>
</table>

1. Write down the reconstructed transaction table and dirty page table at the end of analysis.

2. Write down the winner set and loser set.

3. Give the ranges of LSN’s of the REDO phase and the UNDO phase.

4. Give the set of log records causing pages to be rewritten to disk during the REDO phase.

5. Give the set of log records undone during the UNDO phase.