INM370 – Advanced Databases

Tutorial 5 – Relational Database Management Issues

Explain the purpose of the checkpoint mechanism. How often should checkpoints be performed? How does the frequency of checkpoints affect:
- System performance when no failure occurs?
- The time it takes to recover from a system crash?
- The time it takes to recover from a media (disk) failure?

Model Answers

1)

Checkpointing is done with log-based recovery schemes to reduce the time required for recovery after a crash. If there is no checkpointing, then the entire log must be searched after a crash, and all transactions must be undone/redone from the log. If the checkpointing is performed, then most of the log records prior to the checkpoint can be ignored at the time of recovery.

Another reason to perform the checkpoints is to clear log records from stable storage as it gets full.

Since checkpoints cause some loss in performance while they are being taken, their frequency should be reduced if fast recovery is not critical. If we need fast recovery, the checkpointing frequency should be increased. If the amount of stable storage available is limited, frequent checkpointing is unavoidable.

Checkpoints have no effect on recovery from a disk failure; archival dumps instead are the equivalent of checkpoints for recovery from disk crashes.

Suppose there is a transaction that has been running for a very long time, but has performed very few updates.
a) What effect would the transaction have on recovery time with the recovery algorithm covered in the lecture?
b) What effect would the transaction have on deletion of old log records?

2)

a) If a transaction has been running for a very long time, with few updates, it means that during recovery, the undo phase will have to scan the log backwards till the beginning of this transaction. This will increase the recovery time in the case of the recovery algorithm.
b) A long running transaction implies that no log records which are written after it started, can be deleted till it either commits or aborts. This might lead to a very large log file being generated, though most of the transactions in the log file have completed. This transaction becomes a bottleneck for deletion of old log records.

3)

Recovery would happen as follows:

Redo phase:

a. Undo-List = *T*0, *T*1
b. Start from the checkpoint entry and perform the redo operation.
c. C = 600
d. T1 is removed from the Undo-list as there is a commit record.
e. T2 is added to the Undo list on encountering the < T2 start > record.
f. A = 400
g. B = 2000

Undo phase:

a. Undo-List = T0, T2
b. Scan the log backwards from the end.
c. A = 500; output the redo-only record < T2, A, 500 >

d. output < T2 abort >
e. B = 2000; output the redo-only record < T0, B, 2000 >
f. output < T0 abort >

At the end of the recovery process, the state of the system is as follows:
A = 500
B = 2000
C = 600

The log records added during recovery are:
< T2, A, 500 >
< T2 abort >
< T0, B, 2000 >
< T0 abort >

Observe that B is set to 2000 by two log records, one created during normal rollback of T0, and the other created during recovery, when the abort of T0 is completed. Clearly the second one is redundant, although not incorrect. Optimisations are possible, e.g. using ARIES algorithm, but we do not cover it in the module.

4)
REDO T1 and T5, since these transactions have committed after the most recent checkpoint and before the failure time, and we cannot guarantee that the result of their operations have been permanently saved in the database (secondary memory).

UNDO T2 and T3, since they are still active during the failure time.

Do nothing with T4. T4 completed prior to the most recent checkpoint. For such a transaction, the $\langle T_i \text{ commit} \rangle$ record (or $\langle T_i \text{ abort} \rangle$ record) appears in the log before the <checkpoint> record. Any database modifications made by T4 must have been written to the database either prior to the checkpoint or as part of the checkpoint itself. Thus, at recovery time, there is no need to perform a redo or undo of T4.