UW-Madison – CompSci 764 fall 2023 – midterm exam

1. [2] In an implementation of external merge sort, what is "graceful degradation," e.g., with respect to internal and external sorting?
   Instead of a binary choice between execution modes, e.g., internal vs external sorting, graceful degradation enables a smooth and incremental transition. For example, if the sort input is merely 10% larger than the available memory allocation, only about 10% of the input spills to an external run; the remainder remains in memory.
2. [2] What is the "gold standard" for correctness in concurrency control?
   Equivalence to a serial execution, i.e., the same write-read relationships.
3. [2] Give an example of a "false conflict" due to lock granularity, i.e., the "scope" or "size" of a lock.
   In a b-tree, phantom protection requires concurrency control for a gap between key values, but traditional locking techniques always lock a gap and an index entry (or a key value) together.
   [1] Suggest or describe a remedy.
   Orthogonal locking of gap and index entry (or key value).
4. [2] Give an example of a "false conflict" due to lock timing, i.e., acquisition or release (or existence or enforcement) of a lock.
   Reader and writer do not conflict if the writer keeps the new value invisible to the reader, e.g., using a transaction-private update buffer or a new version.
   [1] Suggest or describe a remedy.
   Deferred lock enforcement – strict traditional X locks only during the commit logic.
5. [2] In deferred lock enforcement, which conflicts are detected immediately and which ones are deferred?
   write-write conflicts are detected immediately, read-write conflicts and write-read conflicts are deferred.
   [1] When are these deferred conflicts detected?
   During commit processing, e.g., writers must wait for readers to finish and new readers must while a writer attempts to commit.
6. [4] How do deferred lock enforcement and multi-version storage enable or assist one another? Or: how does each amplify the other's advantages?
   Deferred lock enforcement permits maximal concurrency enabled by multi-version storage; multi-version storage permits uncommitted new versions directly in the database (or its buffer pool) rather than a transaction-private update buffer.
7. [1] Describe system transactions…
   Hidden from users, they modify the database representation without changing the logical database contents – a typical example a splitting a b-tree node.
   [1] their needs for concurrency control and…
   Latching but not locks, because no changes in logical database contents.
   [1] their needs for logging and recovery.

Logging changes and committing, albeit often change + commit within the same log record.

8. [1] Describe ghost records (also known as pseudo-deleted records) and their role in transactional deletion of b-tree entries, with respect to both…
   Records and key values that are physically present but logically absent - they don't count in a "select count(*)…" query, for example. Logical deletion may simply turn a valid index entry into a ghost.
   [1] locking and…
   Ghosts can be locked – a deletion may retain a lock on the ghost only.
   [1] transaction rollback.
   Rollback requires turning a ghost back into a valid record or index entry – it does not require space allocation and therefore cannot fail due to a missing free space.
9. [2] When are transaction updates guaranteed to be persistent? (ignore distributed transactions)
   When the transaction's commit log record is in the recovery log on stable storage.
10. [2] How long can a user cancel a transaction, i.e., up to which event or activity?
    Until the user or application requests transaction commit.
11. [2] Within a distributed transaction, how long can a local participant transaction unilaterally abort, i.e., up to which event or activity?
    Until the participant pledges to implement the coordinator's global commit decision, whatever that may be and whenever it may come.
12. [2] When recovering from a system, media, or node failure, what piece of information forces that a distributed transaction must be preserved ("redo" in order to achieve "all" semantics) rather than rolled back ("undo" in order to achieve "nothing" semantics)?
    The coordinator's log record with its commit decision.
13. [2] What makes frequent server checkpoints desirable?
    Short log analysis and "redo" during restart after a system failure.
14. [2] What makes frequent server checkpoints undesirable?
    Overhead during transaction processing without system failures.
15. [2] If log analysis (during restart after a system failure) recovers the most relevant server state (in transaction manager, lock manager, and buffer pool manager), what is the resulting advantage in terms of mean time to failure or in terms of mean time to repair?
    New checkpoints and in particular new user transactions are possible after recovery of this server state; if log analysis can recover it, the apparent mean time to repair is much shorter and the availability MTTF/(MTTF+MTTR) is higher.
16. [2] If database backups and the log archive are sorted (on database page identifier) but not indexed, what advanced restore algorithm is possible and what more advanced restore algorithm is not possible?
    sorted ⇒ single-phase restore, indexed ⇒ on-demand incremental "instant" restore