12.1 Suppose that there is a database system that never fails. Is a recovery manager required for this system?

12.2 Consider a file system such as the one on your favorite operating system.

a. What are the steps involved in creation and deletion of files, and in writing data to a file?

b. Explain how the issues of atomicity and durability are relevant to the creation and deletion of files and to writing data to files.

12.3 Database-system implementers have paid much more attention to the ACID properties than have file-system implementers. Why might this be the case?

12.4 Justify the following statement: Concurrent execution of transactions is more important when data must be fetched from (slow) disk or when transactions are long, and is less important when data are in memory and transactions are very short.

12.5 What is a cascadeless schedule? Why is cascadelessness of schedules desirable? Are there any circumstances under which it would be desirable to allow noncascadeless schedules? Explain your answer.

12.6 The lost update anomaly is said to occur if a transaction T_j reads a data item, then another transaction T_k writes the data item (possibly based on a previous read), after which T_j writes the data item. The update performed by T_k has been lost, since the update done by T_j ignored the value written by T_k.

a. Give an example of a schedule showing the lost update anomaly.

b. Give an example schedule to show that the lost update anomaly is possible with the **read committed** isolation level.

c. Explain why the lost update anomaly is not possible with the **repeatable read** isolation level.

12.7 Show that the two-phase locking protocol ensures conflict serializability, and that transactions can be serialized according to their lock points.

12.8 Consider the following two transactions:

T_{34}: **read**(A);
read(B);
if $A = 0$ **then** $B := B + 1$;
write(B).

T_{35}: **read**(B);
read(A);
if $B = 0$ **then** $A := A + 1$;
write(A).

扫描全能王 创建

Add lock and unlock instructions to transactions T_{31} and T_{32}, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock?

12.9 What benefit does rigorous two-phase locking provide? How does it compare with other forms of two-phase locking?

12.10 In timestamp ordering, **W-timestamp**(Q) denotes the largest timestamp of any transaction that executed write(Q) successfully. Suppose that, instead, we defined it to be the timestamp of the most recent transaction to execute write(Q) successfully. Would this change in wording make any difference? Explain your answer.

12.11 Use of multiple-granularity locking may require more or fewer locks than an equivalent system with a single lock granularity. Provide examples of both situations, and compare the relative amount of concurrency allowed.

12.12 For each of the following protocols, describe aspects of practical applications that would lead you to suggest using the protocol, and aspects that would suggest not using the protocol:

- Two-phase locking.
- Two-phase locking with multiple-granularity locking.
- Timestamp ordering.
- Validation.

12.13 Explain why the following technique for transaction execution may provide better performance than just using strict two-phase locking: First execute the transaction without acquiring any locks and without performing any writes to the database as in the validation-based techniques, but unlike the validation techniques do not perform either validation or writes on the database. Instead, rerun the transaction using strict two-phase locking. (Hint: Consider waits for disk I/O.)

12.14 Explain why log records for transactions on the undo-list must be processed in reverse order, whereas redo is performed in a forward direction.

12.15 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?

Exercises

12.16 List the ACID properties. Explain the usefulness of each.

扫描全能王 创建

12.17 During its execution, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Explain why each state transition may occur.

12.18 Explain the distinction between the terms *serial schedule* and *serializable schedule*.

12.19 Consider the following two transactions:

T_{13}: read(A);
read(B);
if $A = 0$ **then** $B := B + 1$;
write(B).

T_{14}: read(B);
read(A);
if $B = 0$ **then** $A := A + 1$;
write(A).

Let the consistency requirement be $A = 0 \vee B = 0$, with $A = B = 0$ the initial values.

a. Show that every serial execution involving these two transactions preserves the consistency of the database.

b. Show a concurrent execution of T_{13} and T_{14} that produces a nonserializable schedule.

c. Is there a concurrent execution of T_{13} and T_{14} that produces a serializable schedule?

12.20 Give an example of a serializable schedule with two transactions such that the order in which the transactions commit is different from the serialization order.

12.21 What is a recoverable schedule? Why is recoverability of schedules desirable? Are there any circumstances under which it would be desirable to allow nonrecoverable schedules? Explain your answer.

12.22 Why do database systems support concurrent execution of transactions, ... programming effort needed to ensure that concurrent

扫描全能王 创建