

Application Development and Administration

Solutions to Practice Exercises

23.1 No answer.

23.2 No answer.

23.3 No answer.

23.4 a. Let there be 100 transactions in the system. The given mix of transaction types would have 25 transactions each of type *A* and *B*, and 50 transactions of type *C*. Thus the time taken to execute transactions only of type *A* is 0.5 seconds and that for transactions only of type *B* or only of type *C* is 0.25 seconds. Given that the transactions do not interfere, the total time taken to execute the 100 transactions is $0.5 + 0.25 + 0.25 = 1$ second. i.e, the average overall transaction throughput is *100 transactions per second*.

b. One of the most important causes of transaction interference is lock contention. In the previous example, assume that transactions of type *A* and *B* are update transactions, and that those of type *C* are queries. Due to the speed mismatch between the processor and the disk, it is possible that a transaction of type *A* is holding a lock on a "hot" item of data and waiting for a disk write to complete, while another transaction (possibly of type *B* or *C*) is waiting for the lock to be released by *A*. In this scenario some CPU cycles are wasted. Hence, the observed throughput would be lower than the calculated throughput.

Conversely, if transactions of type *A* and type *B* are disk bound, and those of type *C* are CPU bound, and there is no lock contention, observed throughput may even be better than calculated.

Lock contention can also lead to deadlocks, in which case some transaction(s) will have to be aborted. Transaction aborts and restarts (which may

also be used by an optimistic concurrency control scheme) contribute to the observed throughput being lower than the calculated throughput.

Factors such as the limits on the sizes of data-structures and the variance in the time taken by book-keeping functions of the transaction manager may also cause a difference in the values of the observed and calculated throughput.

23.5 In the absence of an anticipatory standard it may be difficult to reconcile between the differences among products developed by various organizations. Thus it may be hard to formulate a reactionary standard without sacrificing any of the product development effort. This problem has been faced while standardizing pointer syntax and access mechanisms for the ODMG standard.

On the other hand, a reactionary standard is usually formed after extensive product usage, and hence has an advantage over an anticipatory standard - that of built-in pragmatic experience. In practice, it has been found that some anticipatory standards tend to be over-ambitious. SQL-3 is an example of a standard that is complex and has a very large number of features. Some of these features may not be implemented for a long time on any system, and some, no doubt, will be found to be inappropriate.