CHAPTER 23

Parallel and Distributed Transaction Processing

Practice Exercises

23.1 What are the key differences between a local-area network and a wide-area network, that affect the design of a distributed database?

23.2 To build a highly available distributed system, you must know what kinds of failures can occur.

a. List possible types of failure in a distributed system.

b. Which items in your list from part a are also applicable to a centralized system?

23.3 Consider a failure that occurs during 2PC for a transaction. For each possible failure that you listed in Exercise 23.2a, explain how 2PC ensures transaction atomicity despite the failure.

23.4 Consider a distributed system with two sites, *A* and *B*. Can site *A* distinguish among the following?

- *B* goes down.
- The link between *A* and *B* goes down.
- *B* is extremely overloaded and response time is 100 times longer than normal.

What implications does your answer have for recovery in distributed systems?

23.5 The persistent messaging scheme described in this chapter depends on timestamps. A drawback is that they can discard received messages only if they are too old, and may need to keep track of a large number of received messages. Suggest an alternative scheme based on sequence numbers instead of timestamps, that can discard messages more rapidly.

23.6 Explain the difference between data replication in a distributed system and the maintenance of a remote backup site.

23.7 Give an example where lazy replication can lead to an inconsistent database state even when updates get an exclusive lock on the primary (master) copy if data were read from a node other than the master.

23.8 Consider the following deadlock-detection algorithm. When transaction T_i, at site S_1, requests a resource from T_j, at site S_3, a request message with timestamp n is sent. The edge (T_i, T_j, n) is inserted in the local wait-for graph of S_1. The edge (T_i, T_j, n) is inserted in the local wait-for graph of S_3 only if T_j has received the request message and cannot immediately grant the requested resource. A request from T_i to T_j in the same site is handled in the usual manner; no timestamps are associated with the edge (T_i, T_j). A central coordinator invokes the detection algorithm by sending an initiating message to each site in the system.

On receiving this message, a site sends its local wait-for graph to the coordinator. Note that such a graph contains all the local information that the site has about the state of the real graph. The wait-for graph reflects an instantaneous state of the site, but it is not synchronized with respect to any other site.

When the controller has received a reply from each site, it constructs a graph as follows:

- The graph contains a vertex for every transaction in the system.
- The graph has an edge (T_i, T_j) if and only if:
  - There is an edge (T_i, T_j) in one of the wait-for graphs.
  - An edge (T_i, T_j, n) (for some n) appears in more than one wait-for graph.

Show that, if there is a cycle in the constructed graph, then the system is in a deadlock state, and that, if there is no cycle in the constructed graph, then the system was not in a deadlock state when the execution of the algorithm began.

23.9 Consider the chain-replication protocol, described in Section 23.4.3.2, which is a variant of the primary-copy protocol.

a. If locking is used for concurrency control, what is the earliest point when a process can release an exclusive lock after updating a data item?

b. While each data item could have its own chain, give two reasons it would be preferable to have a chain defined at a higher level, such as for each partition or tablet.

c. How can consensus protocols be used to ensure that the chain is uniquely determined at any point in time?

23.10 If the primary copy scheme is used for replication, and the primary gets disconnected from the rest of the system, a new node may get elected as primary. But the old primary may not realize it has got disconnected, and may get reconnected subsequently without realizing that there is a new primary.

a. What problems can arise if the old primary does not realize that a new one has taken over?

b. How can leases be used to avoid these problems?

c. Would such a situation, where a participant node gets disconnected and then reconnected without realizing it was disconnected, cause any problem with the majority or quorum protocols?

23.11 Consider a federated database system in which it is guaranteed that at most one global transaction is active at any time, and every local site ensures local serializability.

a. Suggest ways in which the federated database system can ensure that there is at most one active global transaction at any time.

b. Show by example that it is possible for a nonserializable global schedule to result despite the assumptions.

23.12 Consider a federated database system in which every local site ensures local serializability, and all global transactions are read only.

a. Show by example that nonserializable executions may result in such a system.

b. Show how you could use a ticket scheme to ensure global serializability.

23.13 Suppose you have a large relation $r(A, B, C)$ and a materialized view $v = {}_A\gamma_{\text{sum}(B)}(r)$. View maintenance can be performed as part of each transaction that updates r, on a parallel/distributed storage system that supports transactions across multiple nodes. Suppose the system uses two-phase commit along with a consensus protocol such as Paxos, across geographically distributed data centers.

a. Explain why it is not a good idea to perform view maintenance as part of the update transaction, if some values of attribute A are "hot" at certain points in time, that is, many updates pertain to those values of A.

b. Explain how operation locking (if supported) could solve this problem.

c. Explain the tradeoffs of using asynchronous view maintenance in this context.