CHAPTER 19

Distributed Databases

Exercises

19.3 Answer: Data transfer on a local-area network (LAN) is much faster than on a wide-area network (WAN). Thus replication and fragmentation will not increase throughput and speed-up on a LAN, as much as in a WAN. But even in a LAN, replication has its uses in increasing reliability and availability.

19.6 Answer:

a. The types of failure that can occur in a distributed system include
   i. Computer failure (site failure).
   ii. Disk failure.
   iii. Communication failure.

b. The first two failure types can also occur on centralized systems.

19.7 Answer: A proof that 2PC guarantees atomic commits/aborts inspite of site and link failures, follows. The main idea is that after all sites reply with a <**ready** *T*> message, only the co-ordinator of a transaction can make a commit or abort decision. Any subsequent commit or abort by a site can happen only after it ascertains the co-ordinator's decision, either directly from the co-ordinator, or indirectly from some other site. Let us enumerate the cases for a site aborting, and then for a site committing.

a. A site can abort a transaction T (by writing an <**abort** *T*> log record) only under the following circumstances:-
   i. It has not yet written a <**ready** *T*> log-record. In this case, the co-ordinator could not have got, and will not get a <**ready** *T*> or <**commit** *T*> message from this site. Therefore only an abort decision can be made by the co-ordinator.

ii. It has written the <**ready** *T*> log record, but on inquiry it found out that some other site has an <**abort** *T*> log record. In this case it is correct for it to abort, because that other site would have ascertained the co-ordinator's decision (either directly or indirectly) before actually aborting.

iii. It is itself the co-ordinator. In this case also no site could have committed, or will commit in the future, because commit decisions can be made only by the co-ordinator.

b. A site can commit a transaction T (by writing an <**commit** *T*> log record) only under the following circumstances:-

i. It has written the <**ready** *T*> log record, and on inquiry it found out that some other site has a <**commit** *T*> log record. In this case it is correct for it to commit, because that other site would have ascertained the co-ordinator's decision (either directly or indirectly) before actually committing.

ii. It is itself the co-ordinator. In this case no other participating site can abort/ would have aborted, because abort decisions are made only by the co-ordinator.

19.8 Answer: Site *A* cannot distinguish between the three cases until communication has resumed with site B. The action which it performs while *B* is inaccessible must be correct irrespective of which of these situations has actually occurred, and must be such that *B* can re-integrate consistently into the distributed system once communication is restored.

19.9 Answer: We can have a scheme based on sequence numbers similar to the scheme based on timestamps. We tag each message with a sequence number that is unique for the (sending site, receiving site) pair. The number is increased by 1 for each new message sent from the sending site to the receiving site.

The receiving site stores and acknowledges a received message only if it has received all lower numbered messages also; the message is stored in the *received-messages* relation.

The sending site retransmits a message until it has received an ack from the receiving site containing the sequence number of the transmitted message, or a higher sequence number. Once the acknowledgment is received, it can delete the message from its send queue.

The receiving site discards all messages it receives that have a lower sequence number than the latest stored message from the sending site. The receiving site discards from *received-messages* all but the (number of the) most recent message from each sending site (message can be discarded only after being processed locally).

Note that this scheme requires a fixed (and small) overhead at the receiving site for each sending site, regardless of the number of messages received. In contrast the timestamp scheme requires extra space for every message. The timestamp scheme would have lower storage overhead if the number of messages received within the timeout interval is small compared to the number

of sites, whereas the sequence number scheme would have lower overhead otherwise.

19.10 Answer: Consider the balance in an account, replicated at N sites. Let the current balance be \$100 – consistent across all sites. Consider two transactions T_1 and T_2 each depositing \$10 in the account. Thus the balance would be \$120 after both these transactions are executed. Let the transactions execute in sequence: T_1 first and then T_2. Let one of the sites, say s, be down when T_1 is executed and transaction t_2 reads the balance from site s. One can see that the balance at the primary site would be \$110 at the end.

19.12 Answer: In remote backup systems all transactions are performed at the primary site and the data is replicated at the remote backup site. The remote backup site is kept synchronized with the updates at the primary site by sending all log records. Whenever the primary site fails, the remote backup site takes over processing.

The distributed systems offer greater availability by having multiple copies of the data at different sites whereas the remote backup systems offer lesser availability at lower cost and execution overhead.

In a distributed system, transaction code runs at all the sites whereas in a remote backup system it runs only at the primary site. The distributed system transactions follow two-phase commit to have the data in consistent state whereas a remote backup system does not follow two-phase commit and avoids related overhead.

19.13 Answer: Consider the balance in an account, replicated at N sites. Let the current balance be \$100 – consistent across all sites. Consider two transactions T_1 and T_2 each depositing \$10 in the account. Thus the balance would be \$120 after both these transactions are executed. Let the transactions execute in sequence: T_1 first and then T_2. Suppose the copy of the balance at one of the sites, say s, is not consistent – due to lazy replication strategy – with the primary copy after transaction T_1 is executed and let transaction T_2 read this copy of the balance. One can see that the balance at the primary site would be \$110 at the end.

19.16 Answer: Let us say a cycle $T_i \rightarrow T_j \rightarrow \cdots \rightarrow T_m \rightarrow T_i$ exists in the graph built by the controller. The edges in the graph will either be local edges of the from (T_k, T_l) or distributed edges of the form (T_k, T_l, n). Each local edge (T_k, T_l) definitely implies that T_k is waiting for T_l. Since a distributed edge (T_k, T_l, n) is inserted into the graph only if T_k's request has reached T_l and T_l cannot immediately release the lock, T_k is indeed waiting for T_l. Therefore every edge in the cycle indeed represents a transaction waiting for another. For a detailed proof that this imlies a deadlock refer to Stuart et al. [1984].

We now prove the converse implication. As soon as it is discovered that T_k is waiting for T_l:-

a. a local edge (T_k, T_l) is added if both are on the same site.

b. The edge (T_k, T_l, n) is added in both the sites, if T_k and T_l are on different sites.

Therefore, if the algorithm were able to collect all the local wait-for graphs at the same instant, it would definitely discover a cycle in the constructed graph, in case there is a circular wait at that instant. If there is a circular wait at the instant when the algorithm began execution, none of the edges participating in that cycle can disappear until the algorithm finishes. Therefore, even though the algorithm cannot collect all the local graphs at the same instant, any cycle which existed just before it started will anyway be detected.

19.17 Answer:

a. i. Send the query $\Pi_{name}(employee)$ to the Boca plant.
ii. Have the Boca location send back the answer.

b. i. Compute average at New York.
ii. Send answer to San Jose.

c. i. Send the query to find the highest salaried employee to Toronto, Edmonton, Vancouver, and Montreal.
ii. Compute the queries at those sites.
iii. Return answers to San Jose.

d. i. Send the query to find the lowest salaried employee to New York.
ii. Compute the query at New York.
iii. Send answer to San Jose.

19.20 Answer: The result is as follows.

$r \ltimes s =$

A	B	C
1	2	3
5	3	2

19.22 Answer: The reasons are:

a. Directory access protocols are simplified protocols that cater to a limited type of access to data.

b. Directory systems provide a simple mechanism to name objects in a hierarchical fashion which can be used in a distributed directory system to specify what information is stored in each of the directory servers. The directory system can be set up to automatically forward queries made at one site to the other site, without user intervention.