CHAPTER 21

Parallel and Distributed Storage

Practice Exercises

21.1 In a range selection on a range-partitioned attribute, it is possible that only one disk may need to be accessed. Describe the benefits and drawbacks of this property.

Answer:
If there are few tuples in the queried range, then each query can be processed quickly on a single disk. This allows parallel execution of queries with reduced overhead of initiating queries on multiple disks.

On the other hand, if there are many tuples in the queried range, each query takes a long time to execute as there is no parallelism within its execution. Also, some of the disks can become hot spots, further increasing response time.

Hybrid range partitioning, in which small ranges (a few blocks each) are partitioned in a round-robin fashion, provides the benefits of range partitioning without its drawbacks.

21.2 Recall that histograms are used for constructing load-balanced range partitions.

a. Suppose you have a histogram where values are between 1 and 100, and are partitioned into 10 ranges, 1 – 10, 11 – 20, … , 91 – 100, with frequencies 15, 5, 20, 10, 10, 5, 5, 20, 5, and 5, respectively. Give a load-balanced range partitioning function to divide the values into five partitions.

b. Write an algorithm for computing a balanced range partition with p partitions, given a histogram of frequency distributions containing n ranges.

Answer:

a. A partitioning vector which gives 5 partitions with 20 tuples in each partition is: [21, 31, 51, 76]. The 5 partitions obtained are $1-20$, $21-30$, $31-50$, $51-75$, and $76-100$. The assumption made in arriving at this partitioning vector is that within a histogram range, each value is equally likely.

b. Let the histogram ranges be called $h_1, h_2, \ldots, h_h$, and the partitions $p_1, p_2, \ldots, p_p$. Let the frequencies of the histogram ranges be $n_1, n_2, \ldots, n_h$. Each partition should contain N/p tuples, where $N = \Sigma_{i=1}^{h} n_i$.

To construct the load-balanced partitioning vector, we need to determine the value of the k_1^{th} tuple, the value of the k_2^{th} tuple, and so on, where $k_1 = N/p$, $k_2 = 2N/p$, etc., until k_{p-1}. The partitioning vector will then be $[k_1, k_2, \ldots, k_{p-1}]$. The value of the k_i^{th} tuple is determined as follows: First determine the histogram range h_j in which it falls. Assuming all values in a range are equally likely, the k_i^{th} value will be

$$s_j + (e_j - s_j) * \frac{k_{ij}}{n_j}$$

where

s_j	:	first value in h_j
e_j	:	last value in h_j
k_{ij}	:	$k_i - \Sigma_{l=1}^{j-1} n_l$

21.3 Histograms are traditionally constructed on the values of a specific attribute (or set of attributes) of a relation. Such histograms are good for avoiding data distribution skew but are not very useful for avoiding execution skew. Explain why.

Now suppose you have a workload of queries that perform point lookups. Explain how you can use the queries in the workload to come up with a partitioning scheme that avoids execution skew.

Answer:
FILL

21.4 Replication:

a. Give two reasons for replicating data across geographically distributed data centers.

b. Centralized databases support replication using log records. How is the replication in centralized databases different from that in parallel/distributed databases?

Answer:

a. By replicating across data centers, even if a data center fails, for example due to a power outage or a natural disaster, the data would still be available from another data center. By keeping the data centers geographically separated, the chances of a single natural disaster such as an earthquake or a storm affecting both the data centers at the same time are minimized.

b. Centralized databases typically support only full database replication using log records (although some support logical replication allowing replication to be restricted to some relations). However, they do not support partitioning, or the ability to replicate different parts of the database at different nodes; the latter helps minimize the load increase at a replica when a node fails by spreading the load across multiple nodes.

21.5 Parallel indices:

a. Secondary indices in a centralized database store the record identifier. A global secondary index too could potentially store a partition number holding the record, and a record identifier within the partition. Why would this be a bad idea?

b. Global secondary indices are implemented in a way similar to local secondary indices that are used when records are stored in a B^+-tree file organization. Explain the similarities between the two scenarios that result in a similar implementation of the secondary indices.

Answer:

a. Any updated such as splitting or moving a partition, which is required to balance load, would require a large number of updates to secondary indices.

b. In both cases records may move (across nodes, or to a different location within the node) which would require a large number of updates to secondary indices if they stored direct pointers. The indirection through the clustering index key / partitioning key allows record movement without any updates to the secondary index.

21.6 Parallel database systems store replicas of each data item (or partition) on more than one node.

a. Why is it a good idea to distribute the copies of the data items allocated to a node across multiple other nodes, instead of storing all the copies in the same node (or set of nodes).

b. What are the benefits and drawbacks of using RAID storage instead of storing an extra copy of each data item?

Answer:

a. The copies of the data items at a node should be partitioned across multiple other nodes, rather than stored in a single node, for the following reasons:

   - To better distribute the work which should have been done by the failed node, among the remaining nodes.
   - Even when there is no failure, this technique can to some extent deal with hot-spots created by read-only transactions.

b. RAID level 0 itself stores an extra copy of each data item (mirroring). Thus this is similar to mirroring performed by the database itself, except that the database system does not have to bother about the details of performing the mirroring. It just issues the write to the RAID system, which automatically performs the mirroring.

   RAID level 5 is less expensive than mirroring in terms of disk space requirement, but writes are more expensive, and rebuilding a crashed disk is more expensive.

21.7 Partitioning and replication.

a. Explain why range-partitioning gives better control on tablet sizes than hash partitioning. List an analogy between this case and the case of B$^+$-tree indices versus hash indices.

b. Some systems first perform hashing on the key, and then use range partitioning on the hash values. What could be a motivation for this choice, and what are its drawbacks as compared to performing range partition direction on the key?

c. It is possible to horizontally partition data, and then perform vertical partitioning locally at each node. It is also possible to do the converse, where vertical partitioning is done first, and then each partition is then horizontally partitioned independently. What are are the benefits of the first option over the second one?

Answer:

a. Hash partitioning does not permit any control on individual tablet sizes, unlike range partitioning which allows overfull partitions to be split quite easily. B$^+$-tree indices use range partitioning, allowing a leaf node to be split if it is overfull. In contrast, it is not easy to split a hash bucket in a hash index if the bucket is overfull.

   Some approaches similar to those used for dynamic hashing (such as linear hashing or extendable hashing) have been proposed to allow overfull hash buckets to be split while leaving other hash buckets untouched, but range partitioning provides a simpler solution.

b. Hashing allows keys of various types to be mapped to a single data type, simplifying the job of partitioning the data. The drawback is that range queries cannot be supported using hashing (without performing a full table scan), whereas direct range-partitioning allows efficient support for range queries.

c. The first option allows reconstruction of records at a single node if a query only accesses records at that node. With the second option, the vertical fragments corresponding to one record may potentially be residing on different nodes, requiring extra communication to get the vertical fragments together.

21.8 In order to send a request to the master replica of a data item, a node must know which replica is the master for that data item.

a. Suppose that between the time the node identifies which node is the master replica for a data item, and the time the request reaches the identified node, the mastership has changed, and a different node is now the master. How can such a situation be dealt with?

b. While the master replica could be chosen on a per-partition basis, some systems support a *per-record master replica*, where the records of a partition (or tablet) are replicated at some set of nodes, but each record's master replica can be on any of the nodes from within this set of nodes, independent of the master replica of other records. List two benefits of keeping track of master on a per-record basis.

c. Suggest how to keep track of the master replica for each record, when there are a large number of records.

Answer:

a. If a node receives a request for a data item when it is not the master, it can send an error reply with the reason for the error to the requesting node. The requesting node can then find the current master and resend the request to the current master. Alternatively, the old master can forward the message to the new master, which can reply to the requesting node.

b. Tracking mastership on a per-record basis allows the master to be located in a geographical region where most requests for the data item occur, for example the region where the user resides. Reads can then be satisfied without any communication with other regions, which is generally much slower due to speed-of-light delays. Further, writes can also be done locally, and replicated asynchronously to the other replicas.

c. Each record can have an extra hidden field that stores the master replica of that record. In case the information is outdated, all the replicas of the data item can be accessed to find the nodes listed as masters for that data item; those nodes can be contacted to find the current master.