CSCI5408 – ASSIGNMENT 2

Problem 1

Dhrumil Amish Shah (B00857606)
dh416386@dal.ca

Analysis of Joins and Semi-joins in Centralized and Distributed Database Queries

The central idea behind this research paper is to provide a detailed analysis of the performance and computing of joins and semi joins in centralized and distributed databases across various metrics such as cost of query execution, memory consumption, CPU utilization cost, I/O cost, total time taken for executing the query, and responding with the query results once the query execution completes. Centralized and distributed database management approaches are the two primary techniques for organizing the data. The former focuses on data placement at one site while the latter distributes the databases at multiple locations. Joins are primitive operations and, the prime idea behind the concept of joins is to extract information from several tables. Various database management systems offer different types of joins but, this paper concentrates on the most prevalent join, the equijoin. Join queries are not always performance-friendly but, research suggests that they can be optimized using semi joins by reducing the amount of data transfer for distributed databases.

The study conducts an experiment to compare the operation of joins and semi-joins in both centralized and distributed database management systems. The problem focuses on querying employees and departments details from employee (EMP) and department (DEPT) tables. The centralized approach host both the tables at one central location whereas, the distributed approach has tables at two different sites. In the centralized database, using the semi-join strategy, it can be comprehended that the query execution cost, bytes accessed and, I/O query cost is higher for semi-joins than joins. The experiment details the comparisons across the stated metrics using figures like tables and graphs. A database developer tool by Oracle - TOAD is used in the study to get alternative query plans and gauges the execution of queries for the total cost, logical reads made, the total number of rows scanned, and memory utilized.

The same approach is repeated to evaluate the performance of joins and semi-joins in a distributed database management system. The main focus of comparison of joins and semi-joins for distributed database, however, is the total time (TT) and response time(RT). Three scenarios are explored for the join approach, and the time taken for all three cases is compared. In the first case, both EMP and DEPT tables are transferred to the destination site (site 3) and joined there. In the second case, the EMP table on the first site (site 1) is transferred to the second site (site 2) and joined with DEPT there and the result set is transferred to the destination (site 3). In the last case, the DEPT table on the second site (site 2) is transferred to the first site (site 1) and joined with EMP there and the result is passed to the destination site (site 3). The data transferred in bytes were 830, 1204 and 606 respectively. For the semi-join approach, the first data transmission from site 2 to site 1 took 8 bytes whereas the second data transmission from site 1 to site 2 took 490 bytes after joining. In total, 498 bytes were transferred. By comparison, it shows that semi-joins reduces the data transmission amount.

In conclusion, the centralized approach shows that semi joins have higher local processing cost as compared to joins which make joins a preferred alternative. However, in distributed approach, joins are useful when data transmission is done from a table having a lower cardinality to a table having higher numbers of records. Semi joins, in distributed databases, decrease the data transmission cost significantly and thus can be chosen if transmission cost is crucial.

Topic of Interest

An interesting topic in the research paper is how comprehensively the authors have compared the performance of joins and semi-join on both centralized and distributed databases. The paper also details the working of semi-joins elaborately and shows how it best suits the distributed database management system.

A Survey on Distributed Deadlock and Distributed Algorithms to Detect and Resolve Deadlock

The primary focus of this research paper is to provide a comparison of various distributed algorithms for detecting and resolving deadlocks in a distributed database system (DDS). The chances of deadlock occurrence are higher in DDS as compared to an operating system(OS). It is because the sites have incomplete knowledge of the system state. In DDS, data placement is across multiple interrelated sites accessed by users as a single site. A typical scenario where a deadlock can occur in a distributed database is when a transaction, trying to access a shared resource is blocked because another transaction is accessing it and thus waits endlessly. This paper discusses different distributed algorithms such as B.M. Alom algorithm and Edge-Chasing algorithm for finding a deadlock in the system and solving it.

A transaction consists of a sequence of operations executed from any site as a single unit that includes read, write, lock and unlock commands. When transactions operate simultaneously, the concurrency control mechanism prevents them from coming into each other's way. Such a concurrency mechanism results in deadlock as it leaves the transactions in a never-ending waiting state. Because of this reason, it is crucial to detect and resolve deadlocks. This paper also covers the locking techniques used for maintaining the integrity of data in the database. Shared and Exclusive locks are two types of locks that implement locking in a DDS. The former applies a shared lock on data allowing other transactions to read data, while the latter uses an exclusive lock on data preventing other transactions from reading or writing data. Only an exclusive lock can result in a deadlock.

Deadlock detection is done using wait-for-graph (WFG) - a set of vertices and directed edges. Vertices represent a set of transactions, while edges represent a waiting state. A cycle in WFG depicts a deadlock and, it can be local or global. When multiple transactions of the same site are part of a deadlock cycle, it is considered a local WFG. A deadlock cycle formed by transactions from various locations is considered a global WFG. To resolve the deadlock, the approach is to terminate one transaction. Three strategies suggested in the research paper for handling deadlock in DDS are Deadlock Avoidance, Deadlock Prevention and Deadlock Detection and Recovery. Deadlock Avoidance deals with deadlock before it even occurs by ensuring that granted resources result in a stable state. Deadlock Prevention allows transactions to acquire resources before the execution by ensuring that at least one condition responsible for deadlock never occurs. Deadlock Detection and Recovery allows deadlock formation but solves it using WFG. Deadlock Detection and Recovery is the widely used approach for dealing with deadlocks.

The Deadlock recovery algorithms described in the research paper are B.M. Alom and Edge-Chasing algorithms. The B.M. Alom algorithm uses the idea of Linear Transaction Structure (LTS), Distributed Transaction Structure (DTS), and priorities. It maintains two tables - the first table is LTS (for deadlock cycle detected at a single site) or DTS (for deadlock cycles detected at multiple sites), and the second table records the transactions with their priorities. If a deadlock cycle is detected, then the transaction with the lowest priority (fetched from the second table) is terminated. It frees the transactions from the deadlock situation.
The Edge-Chasing algorithm makes use of a unique message called probe message. This message is circulated only for blocked transactions. If a transaction is in a blocked state, then all the

transactions on which the blocked transaction depends are sent probe messages. If the transaction that initiated the probe messages receives the probe message back, then a deadlock is said to have been detected. To resolve the deadlock, transactions are aborted based on priority or timestamps. In conclusion, the study explains the shortcomings of both the deadlock recovery algorithms. The B.M. Alom algorithm has a priority problem, and the Edge-Chasing algorithm is unable to detect a deadlock if the transaction initiator is not in the deadlock.

Topic of Interest

The two things I found interesting in this research paper are the different deadlock handling strategies used in DDS and the B.M. Alom Algorithm used for deadlock resolution in DDS. The concept of LTS, DTS and priorities along with WFG for deadlock detection and recovery is fascinating. Also, the example provided is simple to understand.

References

M. Sharma and G. Singh, "Analysis of Joins and Semi-joins in Centralized and Distributed Database Queries," 2012 International Conference on Computing Sciences, Phagwara, 2012, pp. 15-20, doi: 10.1109/ICCS.2012.15. [Accessed: 17- Jun- 2021].

V. Kate, A. Jaiswal and A. Gehlot, "A survey on distributed deadlock and distributed algorithms to detect and resolve deadlock," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, 2016, pp. 1-6, doi: 10.1109/CDAN.2016.7570873. [Accessed: 17- Jun- 2021].