Shared-Table for Textual Data Clustering in Distributed Relational Databases

Wael M.S. Yafooz , Siti Z.Z. Abidin, Nasiroh Omar *and* Rosenah A. Halim

Faculty of Computer and Mathematical Sciences, UiTM Shah Alam, Selagor, Malaysia

Waelmohammed1@hotmail.com,{zaleha,nasiroh,rosenah}
@tmsk.uitm.edu.my

Abstract. High-performance query processing is a significant requirement of database administrators that can be achieved by grouping data into continuous hard disk pages. Such performance can be achieved by using database partitioning techniques. Database partitioning techniques aid in splitting of the physical structure of database tables into small partitions. A distributed database management system is advantageous for many businesses because such a system aids in the achievement of high-performance processing. However, massive amount of data distributed over network nodes affect query processing when retrieving data from different nodes. This study proposes a novel technique based on a shared-table in a relational database under a distributed environment to achieve high-performance query processing by using data mining techniques. A shared-table is used as a guide to show where the data should be saved. Thus, the efficiency of query processing will improve when data is saved at the same location. The proposed method is suitable for news agencies and domains that rely on massive amount of textual data.

Keywords- database clustering, relational database, distributed environment

1 Introduction

A relational database management system (RDBMS) is the backbone of numerous businesses for its robust data structure in managing, organizing, and retrieving data. The digital work of any firm can be processed in a same place based on a centralized database or at different areas based on a distributed database. The workload or "database transaction" in a centralized database is based on one machine. By contrast, a distributed database consists of more than one machine, thus improving system performance because the workload is divided [1], which is the concern of this study. However, the increasing amount of massive data in different network nodes will affect the query processing efficiency. The efficiency of database query can be better when data resides in same node thus the transportation cost in the network is also reduce [2]. Transportation cost, which is the time it takes to retrieve data from different nodes, is an important factor that must be considered in distributed databases. Therefore, database partitioning, clustering and fragmentation is used exchangeable, which is employed to reduce the process of searching and retrieving for the required data by grouping data into continuous hard disk pages. In this way, the number of disk access is reduced, and the number of pages transfers from the secondary to primary storage, is minimized [3].

Database partitioning techniques aid in the splitting of the physical structure of database tables into small partitions based on the collected workload of database transactions. Thereafter, such workload is clustered based on the most accessed attributes or on the most similar records that may be accessed together. The physical structure of table is then divided in a process called splitting. Numerous attempts at database partitioning have been made. These attempts can be categorized into vertical [1, 4-8], horizontal [2, 9, 10], and mixed partitioning [11, 12].

Vertical partitioning is a method that groups the most frequent attributes (columns), which are accessed together into same network node. In vertical partitioning, response time will be reduced when searching for the required data because transportation cost decreases and most frequent attributes are stored in same network node or in a close network node. Users often do not need all the data on attributes; they only need specific tuples (records). Therefore, horizontal partitioning groups database records based on predicate or statistical analysis of the workload. Mix partitioning, which employs both vertical and horizontal clustering, is a good choice because records hold more data that are irrelevant to the requested information (more attributes).

All the aforementioned techniques can be static or dynamic. For static techniques, the data position never changes, depending on the base design. However, queries for any business change over time. All database partitioning methods employ meta-based statistical analysis (database workload information). Many studies focus on vertical clustering because it is more difficult to implement than horizontal clustering. However, all these techniques lack content similarity among data, which can be advantageous for many online news agencies or relational databases that store a large amount of textual data such as news articles, conference papers, and libraries.

This study proposes a novel technique based on shared-table. Shared-table holds frequent terms and named entities that can be found in any textual data that require storage in a relational database. In addition, shared-table conation columns hold the network node that stores such data. Thus, utilizing the shared-table conation column can be a guide for the next textual document to be stored if the documents bear any similarity with textual data in the network nodes. The similar textual data will then be stored in the same location (network node). Thus, the query will retrieve such data when requested from one network node or close network. Therefore, the efficiency of query processing will improve because data is clustered in same location or in a closed location.

The remainder of this paper is organized as follows: Section 2 presents some related works on database clustering. Section 3 explains the distributed database environment architecture. Section 4 discusses the idea of shared-table. Section 5 shows a comparison between existing methods and shared-table. Finally, Section 6 presents the conclusion of this paper.

2 Related Studies

This section presents several studies on database clustering techniques. These studies can be categorized into three: vertical, horizontal, and mix database clustering.

Vertical clustering approaches can be categorized into four: affinity-based [1, 5, 13-15], graph-based [15, 16], genetic algorithm-based [8, 12, 17], and transaction-based [3]. Most of these approaches are implemented in centralized [14, 15] or distributed environments [1, 2, 7, 11, 16, 18].The idea of vertical (attributes) clustering in an affinity-based approach begins with a constructed matrix called Attribute Usage Matrix (AUM) based on the Bond Energy Algorithm (BEA) [19]. BEA rearranges rows (attributes) and columns (transactions) into a two-dimensional matrix called AUM. Thus, the relationship between rows and columns can be informative. Most frequently accessed attributes are grouped together in one diagonal block. However, a clustering process can be evaluated based on user intervention to determine the similarity between attributes and the occurrence of overlapping [20]. AUM is extended to determine the degree of similarity between attributes based on the developed Affinity Attribute Matrix (AAM). However, AAM can only measure the similarity between two attributes in a column or in a row. Therefore, Navathe et al., [14] extended the work in [13] by proposing Navathe Vertical Clustering (NVP). NVP consists of two phases. The first phase involves the construction of the AAM, which is used as an input to perform clustering in the Clustered Affinity Matrix (CAM). Subsequently, CAM is used as an input in the second phase. The second phase performs iterative binary partitioning for existing clusters. However, time complexity is high because of the number of iterations in binary partitioning. In addition, overlapping between clusters is allowed. Therefore, S. B. Navathe & Ra [15] converts AAM into a graph-based approach to reduce time complexity. However, this method can only be applied in a small or centralized database.

Affinity-based methods lack the objective function [21]. Therefore, Muthuraj and Chakravarthy [20, 21] proposed an objective function called Partition Evaluator to assess clusters in a vertical database partitioning algorithm. In addition, the affinity concept only measures similarity between two attributes. Therefore, a transaction-based approach has been introduced by [3], where an efficient vertical clustering algorithm called Optimal Binary Partition Algorithm (OBP) was proposed. OBP clusters attributes based on a set of transactions. Important attributes often appear together in transactions. However, a large number of transactions in OBP affect the time execution of a query. Thus, time complexity increases. Therefore, Song and Gorla [8] proposed a genetic algorithm for database partitioning in a distributed database to generate more attribute partitions. However, genetic algorithms (GAs) are time-consuming. Several attempts have been made to achieve vertical partitioning.

All the aforementioned methods collect workloads, conduct statistical analyses, and partition tables based on the results of analysis. However, Abuelyaman [1] introduced “StatPart” based on a study of the environment and future tasks as a basis for attribute clustering at the initial stage of database design. Dynamic Vertical Partitioning Partitions (DYVEP) data based on the automatic monitoring of query frequencies introduced [7]. However, the communication cost within a distributed database is not considered. Therefore, Li and Gruenwald [4] proposed AutoClust, AutoClust as extended of [22] a previous version, AutoClust utilize a query optimizer to measure performance based on estimated cost in distributed database which lack in DYVEP.

Current studies focus on partitioning multimedia databases. Thus, Rodriguez & Li [6] introduced the Multimedia Adaptable Vertical Partitioning (MAVP).The MAVP static and database administrator shave to provide the information that MAVP needs to establish a vertical partitioning scheme. By contrast, [23] proved that Dynamic Multimedia ON line Distribution (DYMOND) does not need a database administrator because it automatically obtains all input information.

Predicate or affinity-based methods are the two main approaches in horizontal (record) clustering. In the predicate approach, [10] terms that are homogeneously accessible from different applications are mined and grouped together. However, this method suffer requires the completion of the set of terms to perform record partitioning. Therefore, Ozsu & Valduriez [18] introduced efficient iterative algorithms called COMMIN. COMMIN is used to identify a minimal and complete predicate min term set from a predicate set of transaction queries. However, the min term predicate approach can hardly produce a complete predicate set. The affinity-based method, which is used in vertical partitioning, was this introduced for horizontal clustering. [9] used two matrixes: Usage Predicate Matrix (UPM) and Predicate Affinity Matrix (PAM). Another research work utilizing PAM as input to GAs was introduced by [24], where a horizontal fragmentation was considered a traveling salesman problem. Khan & Hoque [2] recently introduced a new method for horizontal fragmentation and allocation in a distributed database. The method focused on the locality of the data recorded in a table called Attribute Locality Table precedence. This approach is implemented at the initial stage of distributed database system design by collecting information from a Create, Read, Update, and Delete matrix [25]. Mixed partitioning is the process of implementing vertical partitioning first, followed by horizontal partitioning, or vice versa. S. Navathe [11] Proposed mix database partitioning for a distributed database system. The idea is to construct grid representations based on data stored in attributes and records. The intersection between attributes and records is presented in grid form. Another approach utilizes GAs for mix partitioning in a relational database [12, 17].

3 Distributed Database Architecture

A distributed environment consists of numerous connected machines. The machines can either be in the same or different geographical areas [1]. The workload in a distributed environment is divided among all machines to enhance system performance. Thus, all machines run a parallel mechanism.

Fragmentation and allocation are two main elements of distributed database management systems. Fragmentation is a method by which to divide the physical structure of database tables into sub-tables. Figure 1 shows the environment structure of a distributed database.

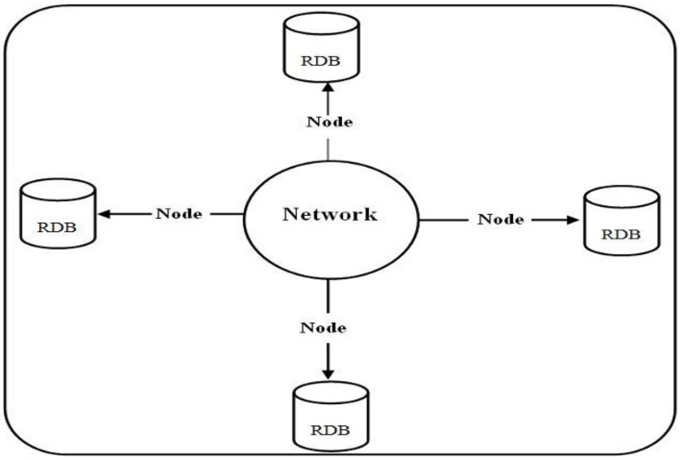

Fig. 1. Structure of the distributed database.

Allocation is a method by which to transfer sub-tables to one site or more over the network. In this way, if the user makes a query, the requested data will be retrieved from one network node. Thus, data retrieval improves when data are retrieved from more than one network node. Therefore, the transportation cost of data from the network node to the user is less than that of retrieving data from different nodes, which increases transportation cost.

4 Shared-Table

The proposed technique is designed for a relational distributed database. The technique, which deals with textual data, is called the Shared-Table Technique (STT). STT can be implemented in all network nodes that have a terminal for input data. STT will keep the database scheme as it is without partitioning. STT consists of a table and two main procedures: Mining Term and Allocation. The table referred to is a database table consisting of four columns: ID, Extracted_Data, Target_Node, and Time-stamp_ID as shown in Table 1. The ID column provides the identifier number, whereas the Extracted_Data column provides the extracted information comprising named entities [26-28] and frequent terms [29, 30]. The node address is stored in the Target_Node column, whereas the time of transaction is stored in the Time-stamp_ID column . Thus, when the user enters textual data directly, the mining term procedure runs.

The mining term procedure is the process of extracting the named entities and frequent terms, which are then stored in the Extracted_Data column. Such information will serve as a guide on where the subsequent textual data should be stored (in which node). Thus, the storing process matches the mining terms of the new textual data in the existing shared-table. Figure 2 shows a general view of the SST.

Table 1. Structure and simple information of STT

ID	Extracted_Data	Target_Node	Timestamp_ID
01	Malaysia,Najeeb, Ahmed, cairo, performance, electricity,	N1	12/5/20013 01:20
02	TNG,Malaysia, Najeeb, electricity	N1	12/5/20013 02:20
03	John,UK, Microsoft,Oracle	N4	12/5/20013 02:25
04	IBM,Oracle, USA	N4	15/5/20013 21:20
05	Football, Japan and China, team, Player	N5	16/5/20013 15:20

Table 1 demonstrates the structure of the STT and simple information about textual data and networks nodes that hold such data. In ID column, the identifier number of textual document is "1". Identifier "1" hold information of specific textual data such as the most frequent terms and named entities of textual documents which stored in network node "N1", In the next record, identifier "2" , its hold information about textual data which are similar to textual documents that hold identifier number "1" . Thus, both textual data stored in network node "1". Each network node holds textual documents that have similarity between its content. Such content is represented by the named entities and most frequent terms of each textual document. This method can be used as a guide on where data should be stored for easier retrieval. Therefore, retrieving process such data can be retrieved very fast rather than access another network nodes over distributed environment. Furthermore, more other manipulation for such data can be done very efficiently. For instance, deleting data that belong to specific action or specific month (s), it can be remove, backup or update without any additional transportation cost over such environment. In case the data are fully or partially similar based on content between two network nodes, the re-allocation procedure will transform the small-sized data from network node to another network node that has large-sized data

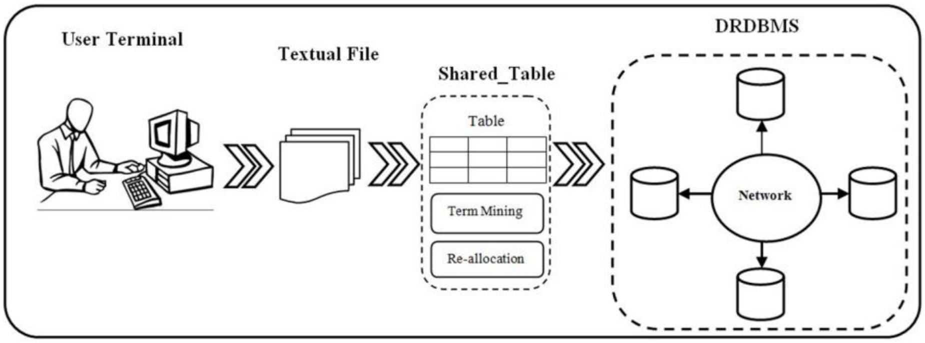

Fig. 2. General view of shared-table technique.

The re-allocation will consider and calculate the transportation cost to identify close network nodes that also have small data that are similar to that of the current node. This process is performed because the data that have to be transferred are often similar to that of another network node. In this case, the data that have to be transferred may be integrated with a far network node, thus affecting the transportation cost when retrieving data from these nodes. Therefore, the re-allocation procedure facilitates calculation and provides a better solution for the re-allocation of data between network nodes. In each database transaction, all tables will be replicated among all mining terms as new data are entered by the user.

Table 2. Transportation time between network nodes

Network Node	Node1	Node2	Node3	Node4
Node1	0	2	1	3
Node2	2	0	2	1
Node3	1	2	0	2
Node4	3	1	2	0

The example of query processing improvement is shown in Table. The measurement of transportation time between all network nodes is in seconds. If data is requested in the user query where query is issued from node 1 and data is allocated in different network nodes such as node number 2 and 3, *a transportation time* will occur in this case. There is also a *processing time* where it is a time taken to retrieve data from the secondary storage. Thus, the total *consuming time* (TC) to retrieve such data is the summation of transportation and processing time. TC= TC (Node1 to Node2) +TC (Node1 to Node3) + PT. Suppose that the *processing time* is 1 second, therefore, TC=2 +1+1=4. However, for data cluster that resides in the same network node, the total consuming time (TC) is equal 1, which is only the processing time.

5 Conclusion

The SST, which can reduce transportation cost over distributed relational databases, was introduced in this study. Query efficiency was improved by reducing the transport cost in a distributed database management system. Enhances efficiency can be achieved by grouping data that are likely to be accessed and retrieved together, specifically textual data that often have similar content. The SST serves as a guide on where data should be stored for easier retrieval. By utilizing the data mining concept, terms along with the location in which they are stored (in which network node) are contained in the shared-table.

Acknowledgment

The authors wish to thank Universiti Teknologi MARA (UiTM) for the financial support. This work was supported in part by a grant number 600-RMI-/DANA 5/3/RIF (498/2012).

References

1. Abuelyaman, E.S., An Optimized Scheme for Vertical Partitioning of a Distributed Database. IJCSNS International Journal of Computer Science and Network Security, 2008. VOL.8 No.1: p. 310-316.
2. Khan, S.I. and D.A.S.M.L. Hoque, A New Technique for Database Fragmentation in Distributed Systems. International Journal of Computer Applications, 2010. Volume 5– No.9: p. 0975 – 8887.
3. Chu, W.W. and I.T. Ieong, A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems. Software Engineering, IEEE Transactions on, 1993. VOL. 19, NO. 8.
4. Li, L. and L. Gruenwald, Autonomous Database Partitioning using Data Mining on Single Computers and Cluster Computers. Proceedings of the 16th International Database Engineering & Applications Sysmposium. ACM, 2012.
5. Ma, H., K.-D. Schewe, and M. Kirchberg, A Heuristic Approach to Vertical Fragmentation Incorporating Query Information. Databases and Information Systems, 2006. 7th International Baltic Conference on. IEEE: p. 69-76.
6. Rodriguez, L. and X. Li, A vertical partitioning algorithm for distributed multimedia databases. . In e. a. A Hameurlain, editor, Proceedings of DEXA, . Springer Verlag, 2011. Vol 6861 (544—558).
7. RodríguezA, L. and X. Li, A dynamic vertical partitioning approach for distributed database system. Systems, Man, and Cybernetics (SMC), IEEE International Conference on. IEEE, 2011.
8. Song, S. and N. Gorla, A genetic Algorithm for Vertical Fragmentation and Access Path Selection. The Computer Journal, 2000. vol. 45, no. 1: p. 81-93.
9. Zhang, Y., On horizontal fragmentation of distributed database design. in M. Orlowska & M. Papazoglou, eds, Advances in Database Re- search, 1993. World Scientific Publishing: p. 121-130.
10. Ceri, S., M. Negri, and G. Pelagatti, Horizontal data partitioning in database design. in Proc. ACM SIGMOD, 1982.
11. S. Navathe, K.K., Minyoung Ra, Amixed fragmentation methodology for initial distributed database design. Journal of Computer and Software Engineering 1995. 3.4 (1995): p. 395-426.
12. Gorla, N., V. Ng, and D.M. Law, Improving database performance with a mixed fragmentation design. J Intell Inf Syst (2012) 39, 2012. 39: p. 559–576.
13. Hoffer, H.A. and D.G. Severance, The Use of Cluster Analysis in Physical Database Design. Proceedings First Internutionul Conference on Vety Large Data Bases, 1975.
14. Navathe, S., et al., Vertical partitioning algorithms for database design. ACM Transactions on Database Systems (TODS) 9.4, 1984: p. 680-710.
15. Navathe, S.B. and M. Ra, Vertical Partitioning for Database Design: A Graphical Algorithm. ACM SIGMOD Record 18.2, 1989.
16. Ra, M., Horizontal partitioning for distributed database design. In Advances in Database Research, World Scientific Publishing, 1993: p. 101–120.

17. Ng, V., et al., Applying genetic algorithms in database partitioning. SAC '03 Proceedings of the ACM symposium on Applied computing, 2003: p. 544-549.
18. Ozsu, M.T. and P. Valduriez, Principles of Distributed Database Systems. 2nd ed., New Jersey: Prentice-Hall, 1999.
19. McCormick, W.T., P.J. Schweitzer, and T.W. White, Problem decomposition and data reorganization by a clustering technique. 1972. Operations Research 20.5: p. 993-1009.
20. Chakravarthy, S., et al., An objective function for vertically partitioning relations in distributed databases and its analysis. Distributed and parallel databases 2.2 1994. 183-207.
21. Muthuraj, J., et al., A formal approach to the vertical partitioning problem in distributed database design. Parallel and Distributed Information Systems, Proceedings of the Second International Conference on. IEEE, 1993.
22. Guinepain, S. and L. Gruenwald, Using Cluster Computing to Support Automatic and Dynamic Database Clustering. Cluster Computing, 2008 IEEE International Conference on. IEEE, 2008.
23. Rodríguez, L., et al., DYMOND: An Active System for Dynamic Vertical Partitioning of Multimedia Databases. Proceedings of the 16th International Database Engineering & Applications Sysmposium. ACM, 2012., 2012.
24. Cheng, C.-H., W.-K. Lee, and K.-F. Wong, A Genetic Algorithm-Based Clustering Approach for Database Partitioning. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 2002. VOL. 32, NO. 3: p. 215-230.
25. Surmsuk, P. and S. Thanawastien, The Integrated Strategic Information System Planning Methodology. 11th IEEE International Enterprise Distributed Object Computing Conference, 2007.
26. Montalvo, S., F. Víctor, and M. Raquel, NESM: a Named Entity based Proximity Measure for Multilingual News Clustering. Procesamiento de Lenguaje Natural, 2012. 48: p. 81-88.
27. Cao, T.H., T.M. Tang, and C.K. Chau, Data Mining: Foundations and Intelligent Paradigms Springer Berlin Heidelberg, 2012: p. 267-287.
28. YafoozB, W.M.S., S.Z. Abidin, and N. Omar, Challenges and issues on online news management. Control System, Computing and Engineering (ICCSCE),IEEE International Conference on., 2011.
29. Krishna, S.M. and S.D. Bhavani, An Efficient Approach for Text Clustering Based on Frequent Itemsets. European Journal of Scientific Research, 2010. ISSN 1450-216X Vol.42 No.3: p. 399-410.
30. Beil, F., M. Ester, and X. Xu, Frequent Term-Based Text Clustering. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2002.