An Analysis of Vertical Splitting Algorithms in Telecom Databases

Vinay Verma, Ruchika Bhaskar, Manish Kalra

Abstract— Distribution design includes making decisions on the allocation and fragmentation of data across the locations of a computer network. Vertical splitting is the process of subdividing the attributes of a relation to generate fragments. In this paper, we propose an analysis for vertical splitting algorithm using prototype approach. This approach starts from the attribute affinity matrix and generates initial clusters based on the affinity values between attributes (Cluster Affinity Matrix). Then, it uses the database according to optimal splitting solution to produce final groups that will represent the fragments. Then we use allocate these fragments in centralized distributed environment and find the analyzed result that shows improvement in the query response time.

Index Terms— Distributed Database, Centralized Database, Telecom Database, Vertical Fragmentation, Bond Energy Algorithm, Affinity Matrix.

1 Introduction

DISTRIBUTED and parallel processing on database management systems (DBMS) is a proficient way of improving performance of application that work on large volumes of data. This may be achieved by removing irrelevant data accessed during the execution of queries and by reducing the data exchange among sites, which are the two main goals of the design of Distributed databases [2].

The primary concern of distributed database systems is to design the fragmentation and allocation of the underlying database. The distribution design involves making decisions on the fragmentation and placement of data across the sites of a computer network. The first phase of the distribution design in a top-down approach is the fragmentation phase, which is the process of clustering into fragments the information accessed simultaneously by applications. The fragmentation phase is then followed by the allocation phase, which handles the physical storage of the generated fragments among the nodes of a computer network, and the replication of fragments.

2 Related Work

Most of the vertical splitting algorithm has started from constructing an attribute affinity matrix from the attribute usage matrix: the Attribute affinity matrix is an n x n matrix for the n-attribute problem whose (i, j) element equals the “between attributes” affinity which is the total number of accesses of transactions referencing both attributes i and j. An iterative binary partitioning method has been used in [8] and [5] based on first clustering the attributes and then applying empirical objective functions or mathematical cost functions to perform the fragmentation. The concept of using fragmentation of data as a means of improving the performance of a database management system has often appeared in the literature on file design and optimization. Attribute partitioning and attribute clustering have been studied earlier by [4], [3], [6], [8], [9] has discussed the implementation of a self-reorganizing database management system that carries out attribute clustering. They also show that in a database management system where storage cost is low compared to the cost of accessing the sub files, it is beneficial to cluster the attributes, since the increase in storage cost will be more than offset by the saving in access cost. Hoffer [6] developed a non-linear, zero-one program, which minimizes a linear combination of storage, retrieval and update costs, with capacity constraints for each file.

Navathe et al [8] used a two-step approach for vertical partitioning. In the first step, they used the given input parameters in the form of an attribute usage matrix to construct the attribute affinity matrix on which clustering is performed. After clustering, an empirical objective function is used to perform iterative binary partitioning. In the second step, estimated cost factors reflecting the physical environment of fragment storage are considered for further refinement of the partitioning scheme. Cornell and Yu [5] proposed an algorithm, as an extension of Navathe et al [8] approach, which decreases the number of disk accesses to obtain an optimal binary partitioning. This algorithm uses specific physical factors such as number of attributes, their length and selectivity, cardinality of the relation etc.

Navathe and Ra have developed a new algorithm based on a graphical technique [7]. This algorithm starts from the attribute affinity matrix by considering it as a complete graph called the “affinity graph” in which an edge value represents the affinity 1-4244-1364-8/07/$25.00 ©2007 IEEE between the two attributes, and then forms a linearly connected spanning tree. The algorithm generates all meaningful fragments in one iteration by considering a cycle as a fragment. A linearly connected tree has only two ends. By a “linearly connected tree” we imply a tree that is constructed by including one edge at a time such that only edges at the “first” and the “last” node of the tree would be considered for inclusion. We then form “af-

- *Vinay Verma, PG Scholar, Computer Science Department, Jagannath University, Jaipur, Rajasthan, India PH-00919413952047. E-mail: ervinayv@gmail.com*
- *Ruchika Bhaskar, Research Student, Computer Science Department, Rajasthan Technical University, Kota, Rajasthan, India PH-00919414238267. E-mail: ruchikabhaskar10@gmail.com*
- *Manish Kalra, Assistant Professor, Computer Science Department, JNU, Jaipur, Rajasthan, India*

finity cycles" in this spanning tree by including the edges of high affinity value around the nodes and "growing" these cycles as large as possible. After the cycles are formed, partitions are easily generated by cutting the cycles apart along "cut-edges". In this paper we will use an algorithm to cluster the database i.e. Bond Energy Algorithm (BEA). And use these cluster affinity as input to find final fragments using PARTITION algorithm. Then using prototypes we reach to the goal of reducing response time of query using fragmentation and show the mathematical result for proof.

3 Background Of Splitting Database

Today, mostly centralized databases are used to store and manage data [11]. They carry the advantages of high degree of security, concurrency and backup and recovery control. However, they also have disadvantages of high communication costs (when the client is far away and communication is very frequent), unavailability in case of system failure and a single source bottleneck [3].

Research conducted in 1991 for distributed databases predicted a huge shift from traditional databases to distributed databases in the coming arena primarily due to organizational needs to manage huge amounts of data [11].
The telecommunication sector also wants to embrace this technology of data distribution. But before distribution, fragmentation is a very important and critical task that needs to be done.

Most of the telecom industries are using centralized technique in storage of their database. Centralized database has its disadvantage of high communication cost. Some data is unavailable due to problem in server. To resolve these issues we are moving to distribution of the database.

4 Analysis of Algorithm Used For Splitting The Database

The vertical fragmentation proposed in this paper is executed in following manner. (1) When a query uses attribute from a relation its value is true (2) Information about databases and query are notified before fragmentation process. (3)Query consists of attributes. The use of attribute means accessing to the value of an existing attribute without any side-effect. Our fragmentation is composed of attribute fragmentation. A set of attributes defined in a relation is vertically partitioned into attribute fragments on basis of application queries like:

1. Normal
   Find subscriber via peer_id and area
   Find Narrative
2. Recharge
   Find subscriber via peer_id and area
   Find subscriber_id
   Find Narrative(units)
   Update Narrative(units)
   Update subscriber (Sub_Param1, 2, 3)
3. Balance Inquiry
   Find subscriber via peer_id and area
   Find Narrartives etc.

Figure 1: Sample Queries of experiment

After, the attribute fragments are generated, the attribute fragmentation is executed on basis of queries .Also, a query can access attributes of other relation on a database hierarchy. To reflect these characteristics of queries, we calculate QA(query access) matrix, FA(site access) matrix and AU(attribute usage) matrix. In FA matrix, QA matrix contains all description in of attribute used by Sites of different relations and to represent attributes usage by different queries.

4.1 QA matrix:

QA (Query Access) matrix represents the use of attributes in application queries. Figure 2 and Figure 3 is an example of the QA matrix for subscriber relation and Narrative relation used as database in this paper.

```
1 1 1 1 0 1 1 1 0 1 0
1 1 0 1 0 0 0 0 0 0 0
1 1 0 1 1 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0
1 1 1 0 1 0 0 0 0 0 0
1 1 0 1 0 0 0 0 1 0 0
1 1 1 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0
```

Figure 2: SUBSCR QA Matrix

```
1 1 1 1 1 1
1 1 1 1 1 0
1 1 1 0 1 0
1 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0
0 1 0 1 0 0
```

Figure 3: NARRATIVE QA Matrix

Each row means a query and each column means a attribute Q1-A1, Q1-A2, Q1-A3……Q1-A11 are queries about SUBSCR relation. And similarly in Figure 3 for NARRATIVE relation. The entry "1" indicates that the query uses the corresponding attributes. Attributes of other relation accessed by query can be represented in QA matrix. For example, Q1 accesses to not only sub_id, peer_id, status, language of its own relation SUBSCR but also narrative_id, sub_id, narrative_type, and unit_type of relation NARRATIVE.

Combined AQ matrix(CAQ)

1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1
1 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0
1 1 0 1 1 0 0 0 0 0 0 1 1 0 1 0
1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0
1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0

Figure 4: Combined AQ matrix (CAQ)

4.2 FA Matrix:

The access frequency represents the sum of number of accesses about query generated in one or more sites.

	S1	S2	S3
Q1	10	15	5
Q2	0	3	2
Q3	15	1	0
Q4	0	5	6
Q5	8	0	0
Q6	25	20	25
Q7	0	0	10
Q8	0	6	4
Q9	2	8	20

Figure 5: Access Frequency Matrix(FA)

Figure 5 shows the access frequency of different queries by different sites In QA matrix, A usage value of attribute Aj for a query qi is defined as:

use(qi, Aj) = 1 if query qi accesses to
Aj = 0 otherwise

The AA (attribute affinity) matrix is generated from the AQ matrix using the same technique as relational vertical fragmentation approach.

Attribute Affinity Matrix aff(Ai, Aj)

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
A1	190	150	99	132	24	30	30	30	70	30	0
A2	150	150	59	132	24	30	30	30	70	30	0
A3	99	59	99	41	8	30	30	30	0	30	0
A4	132	132	41	132	16	30	30	30	70	30	0
A5	24	24	8	16	24	0	0	0	0	0	0
A6	30	30	30	30	0	30	30	30	0	30	0
A7	30	30	30	30	0	30	30	30	0	30	0
A8	30	30	30	30	0	30	30	30	0	30	0
A9	70	70	0	70	0	0	0	0	70	0	0
A10	30	30	30	30	0	30	30	30	0	30	0
A11	0	0	0	0	0	0	0	0	0	0	0

Figure 6: Attribute Affinity Matrix (SUBSCR)

	A12	A13	A14	A15	A16	A17
A12	62	62	62	46	51	30
A13	62	102	72	86	61	30
A14	62	72	72	56	61	30
A15	46	86	56	86	45	30
A16	51	61	61	45	61	30
A17	30	30	30	30	30	30

Figure 7: Attribute Affinity Matrix (NARRATIVE)

The attribute affinity represents the strength of bond between the two attributes. The attribute affinity for two attributes Ai and Aj defined as

aff(Ai, Aj)=k | use(qk, Ai) =1^ use(qk,Aj) =1acc (qk)

Where, aff(Ai, Aj) is affinity value between Ai and Aj, acc(qk) is the total number of access of query qk generated in multiple sites. Figure 6 and Figure 7 shows an example of AA matrix.

The AA matrix is needed to be clustered, and then becomes to divide into attribute fragments. Bond Energy Algorithm (BEA) [ll] is used to cluster the AA matrix. The purpose of clustering is to combine large affinity value of AA matrix with large affinity values, and the small one with small ones. The result of clustering AA matrix in Figure 6 and Figure 7 is a CA (C1uster Affinity) matrix shown in Figure 8. The partitioning of generated along the main diagonal of the CA matrix.

5 Implementation And Comparison

We have implemented the vertical class fragmentation proposed in this paper using JAVA programming language on an IBM-PC. The implementation have executed to the following procedures. First, establish the example structure of class schema and example queries with the access frequency. Second, generate the UQA, UAU matrix for all tables. Third, for each table, generate AA and CA matrixes.

Fourth, partition the CA matrixes and make the attribute fragments. When we apply both BEA algorithm[16] and vertical Partition algorithm[16]according to the Attribute usage matrix and Attribute Affinity matrix we conclude to the fragment results and selected one of the optimal solutions available. So following fragments has been selected with primary key [41].

Fragments of Subscr
S_fragment1 (Sub_id, Levy_Structure, Area, stage, Sub_param1, Sub_param2)
S_fragment2 (Sub_id, Peer_id, Sub_group, sub_param3, Disclosure_date , Ascent)
Fragments of Narrative
N_Fragment1(Narrative_id, Sub_id, Narrative_type, Units, Un it_Type)
N_Fragment2 (Narrative_id, Lower_Exceed_Limit

5.1 Testing Query Response Time Using Centralized and Distribued Model

All models were developed using Java Programming Language. We used MySQL to store the data in experiment. The third party tools (WEKA, Rapid Miner etc.) were also used.

5.2 Comparison of Centralized and Distributed Databases Response Time Figures

In this section we have compared the results from centralized and distributed databases and discussed the performance of the response time.

5.2.1 Min response time on sites

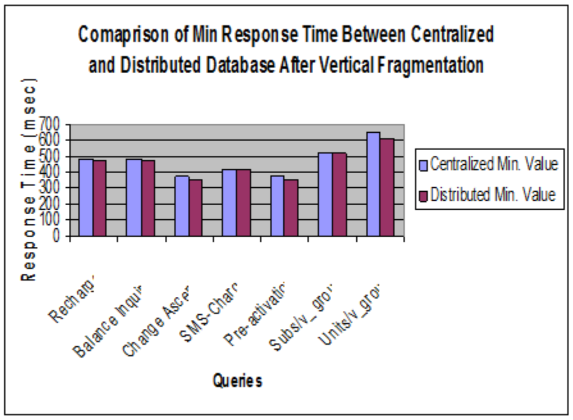

Figure 8: Comparison between Centralized and Distributed database

Graph in Figure 8 shows the contrast of minimum values of all queries in the centralized and distributed environment. We can see that the values for the distributed environment are less. The reason for this reduces is that in distributed databases data is local to the sites and index table is short as compared to the centralized databases. However, response time values have been increased for first queries (General call query excluded from the chart).Since these queries fetch data from all sites (fragments), so communication time contribute to increase their response time. One important thing to note is that the response time for sms charge and subs/v_group are almost the same for both environments. Since these queries fetch data from one table only and they update only one column of a small data set which is almost the same for both environments. So, there are no big variations in their response time in the two environments.

5.2.2 Max response time

Figure 9 is about the maximum values of all queries for both environments. Except for first query (General call) response time for queries in the centralized environment is greater than

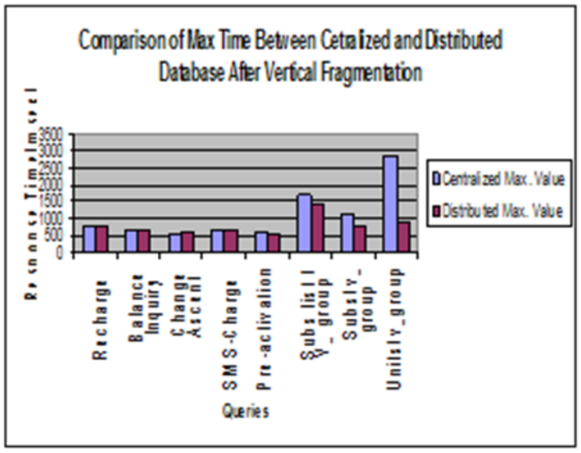

Figure 9: Comparison between Centralized and Distributed database (Max time)

distributed environment. General call queries generate reports by gathering data from all sites and they get data from all remote sites as well. So, their response time is better in distributed environment. However, in centralized environment all data is at one place so data probing is fast for this query and hence response time is less for these particular queries.

5.2.3 Avg. response time

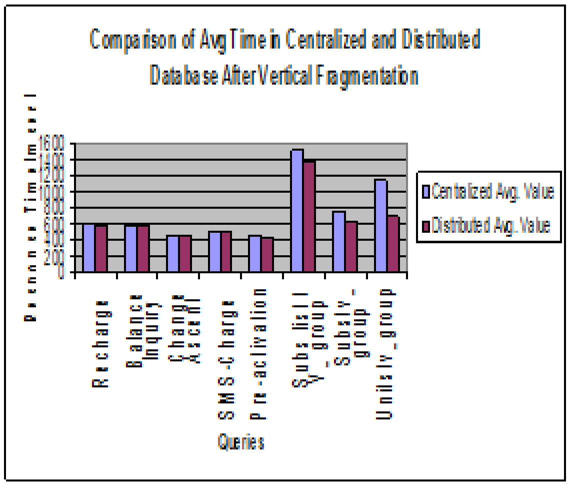

Figure 10: Comparison between Centralized and Distributed database (Avg time)

Graphs in Figure 10 shows that average values for distributed database are fewer than centralized environment. Since in distributed databases data is reserved local to the site where it is needed, so response time is very good. However, in case of remote access in distributed environment, response time has been augmented.

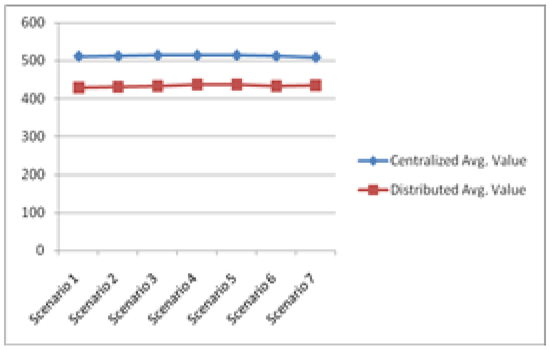

Figure 11: Occurrence Arrangement in Centralized and Distributed database

The graph in Figure 11 shows the average values of different scenarios used in centralized and distributed databases. Four queries (General call, Recharge, balance inquiry and SMS charge) are used in these scenarios. General call query performs both read and write operations. The purpose of these scenarios is to find the impact on the response time if we augment the ratio of general call to the total. The detail of these scenarios is given in Figure 12. The last column of Figure 12 is designed using average values of different sites for these four queries, then they were multiply with the ratio of each query used in the scenarios and average was designed.

Percentage of Arrangements				
Query	G. Call%	Re-charge%	B. In-quiry%	SMS %
Scenario 1	10	5	9	76
Scenario 2	22	5	6	67
Scenario 3	35	6	10	49
Scenario 4	40	11	9	40
Scenario 5	65	5	9	21
Scenario 6	74	7	11	8
Scenario 7	7	7	7	79

Figure 12: Percentages of Arrangements

Their increasing pattern is almost the same. However, average response time in centralized database is more than distributed database. The data is accumulated at one place in centralized database, so it takes long to process a query. Moreover, data index tables are large to search.

6 Conclusion

The purpose of conducting this study is to know the impact on the response time while moving from centralized to distributed databases

Distributed databases have many aspects and every organization has certain preferences. For the telecom sector, the response time is prioritized.

Our experiment showed that the average response time is decreased if we switch from centralized database to distributed database. In distribution we put the data to the site where it is used most frequently. This locality of data reduces the response time. In the distributed database, data is fragmented. These fragments are short compared to the full database (centralized database contains maximum columns). However, when we need data from multiple sites for a query (report queries), the response time is increased. Accessing data from multiple remote sites and then joining those takes long time. But in the centralized database since data is at one place so, it is easy and fast to search it. The purpose of conducting this study is to know the impact on the response time while moving from centralized to distributed databases using vertical fragmentation.

Experiment results showed that the response time is decreased in distributed databases. Due to fragmentation data set for single site contains less records than centralized database, so response time is less.

References

[1] Ceri, S. and Pelagatti, G. Distributed DatabasesPrinciples and Systems. NY, McGraw Hill, 1984.

[2] Ezeife, C. I. and Barker, K. Vertical Class Fragmentation in a Distributed Object Based Svstem. TR 94-03, Univ. of Manitoba DeRt. Of Computer Science, 1993.

[3] H.o ffer. 1. A.. and Severance. D. G. The Use of Cluster Analysis in Physical Database Design.In Proceedings of 1st VLDB Conference, Mass., 1975.

[4] Karlapalem, K. and Li, 8. Partitioning Schemes for Object Oriented Database. In 5th InternationalWorkshop on Research Issues on Data Engineering: Distributed Object Management, 1995.

[5] Karlapalem, K., Li, 8. and Vieweg,, S. Method Induced Partitioning Schemes in Object OrientedDatabases. In 16th intemational conference on Distributed Computing System, Hong Kong, 1996.

[6] Karlapalem, K., Navathe, S. B. and Morsi, M. M.A. Issues in Distribution design. of object-oriented databases, in Distributed Object Management,Morgan Kaufmann Publishers, 1994.

[7] Lee, S. and Lim, H., Extension of Vertical Technical Conference on Circuits/systems, Computers and Communications, Japan, 1997.

[8] Navathe, S. B., Ceri, S. Wiederhold, G. and Dou, J.Vertical partitioning algorithms for database design.in ACM TODS 9(4), 1984.

[9] Farhi Marir, Yahiya Najjar, Mahmoud Y. AlFaress, Hassan I. Abdalla, “An Enhanced Grouping Algorithm for Vertical Partitioning Problem in DDBs.

[10] Adrian Runceanu, Towards Vertical Fragmentation in Distributed Databases.

[11] Ashraf, Imran and Khokhar, A.S. 2010. Principles for Distributed Databases in Telecom Environment., Sweden.

[12] Huang , Y.-F. And Chen , J.-H. 2001. Fragment Allocation in Distributed Database Design.

[13] Mitchell, C. Components of a Distributed Database.

[14] Jonker, W. 2000. Databases in telecommunications: international workshop co-located with VLDB-99, Edinburgh, Scotland, UK, September 6th 1999: proceedings. Springer, Berlin.

[15] Hvasshovd, S.-O. 1995. the clustRa telecom database: high availability high throughput and real-time response Proceedings of 21st VLDB Conference.

[16] Wiederhold, G., and Dou, J.,"Vertical Partitioning Algorithms for Database Design," ACM Trans.on Database Systems, Vol. 9, No.4, Dec. 1984.