ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2013

Fuzzy Logic Based Query Optimization in Distributed Database

Abhijeet R Raipurkar[1],G.R.Bamnote[2]

M.E. Final Year, Department of Computer Science & Engineering, PRMIT&R, Badnera, India[1]

Head & Professor, Department Of Computer Science & Engineering, PRMIT&R, Badnera, India[2]

Abstract: Structural Query Language (SQL) is very restrictive in data extraction. Classical SQL queries have remarkable capabilities in terms of data extraction and answer formation from information stored at widely dispersed databases. Human queries are rarely crisp which poses challenges in efficient answer formation and data retrieval. These are based on human perception which is grossly inexact and imprecise based on world knowledge. Integration of query languages with fuzzy logic can increase their capability in data retrieval based on human perception. Query optimization is a difficult task in a distributed client/server environment as data location becomes a major factor. The integration of a query processing subsystem into a distributed database management system with fuzzy logic is used for analyzing query response time across fragmentations of global relations. Fuzzy logic based query optimization in distributed database have an important impact on the performance of distributed query processing.

Keywords: Fuzzy logic, Query optimization, fragmentation

I. INTRODUCTION

Distributed database adds to the conventional centralized Database system [3] some other types of processing expenses, because of the additional design (hardware & software) to handle the distribution. These expenses present as the cost of data transfer over the network. Data transferred could be, intermediate files resulting from local sites, or final results need to be sent back to the original site that issued the query. Therefore, database designers are concerned about query optimization, which target minimizing the cost of transferring data across the network.

Criteria for measuring the cost of a query evaluation strategy for centralized DBMSs number of disk accesses (# blocks read / written) and for distributed databases, additionally the cost of data transmission over the network and Potential gain in performance from having several sites processing parts of the query in parallel.

Join queries [4] in distributed database are ship whole, fetch as needed, semi joins [4] and bloom joins and each of these join strategies are having their own advantages and disadvantages. Main considerations of query processing in distributed databases are: communication cost, if there is several copies of a relation, decide which copy to use, amount of data being shipped, relative processing speed at each site and site selection

In a relational database [7] all information can be found in a series of tables. A query therefore consists of operations on tables. The most common queries are Select-Project-Join queries. Distributed query processing: Transform a high-level query (of relational calculus/SQL) on a distributed database (i.e., a set of global relations) into an equivalent and efficient lower-level query (of relational algebra) on relation fragments. Distributed query processing is more complex because of fragmentation/replication of relations, additional communication costs and parallel execution of operations.

Query optimization is the process of producing an optimal (close to optimal) query execution plan which represents an execution strategy for the query. The main task in query optimization is to consider different orderings of the operations and minimize total cost associated with execution of request. Query optimization is a crucial and difficult part of the overall query processing

II. RELATED WORK

In a relational database all information can be found in a series of tables. A query therefore consists of operations on tables. The most common queries are Select-Project-Join queries.

For a given query, the search space can be defined as the set of equivalent operator trees that can be produced using transformation rules. Distributed query optimization process consists of transforming global queries from control sites to Fragment queries in local site. The process of transformation is as shown below:

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2013

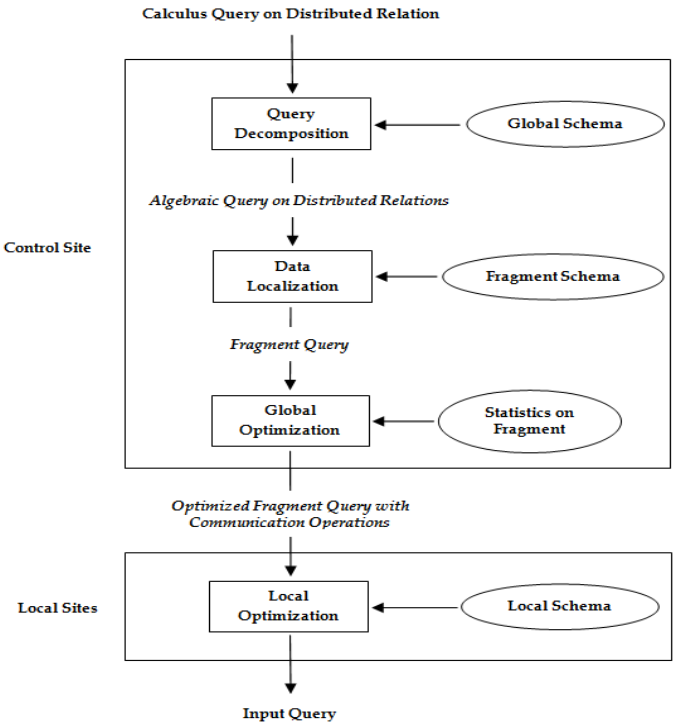

Fig.1 Distributed Query Optimization Process

The input is a query on distributed data expressed in relational calculus. Four main layers are involved to map the distributed query into an optimized sequence of local operations, each acting on a local database. These layers perform the functions of query decomposition, data localization, global query optimization, and local query optimization. Query decomposition and data localization correspond to query rewriting. The first three layers are performed by a central site and use global information. Local optimization is done by the local sites.

Ordering of the operators of relational algebra is crucial for efficient query processing so expensive operators should be moved at the end of query processing. Various unary and binary operations can be pushed to the leaves of the tree as per time complexity associated with it.

Following are the various unary and binary operators used during query optimization process along with time complexity. In time complexity ‘n’ represents cardinality of relation.

Operations	Time Complexity
Select, Project (without duplicate elimination)	O(n)
Project (with duplicate elimination)	O(n log n)
Group Join Semi-join Division Set Operators	O(n log n)
Cartesian Product	O(n2)

Table 1: Comparison of Time Complexity For Various Relational Operations

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2013

III. PROPOSED SYSTEM

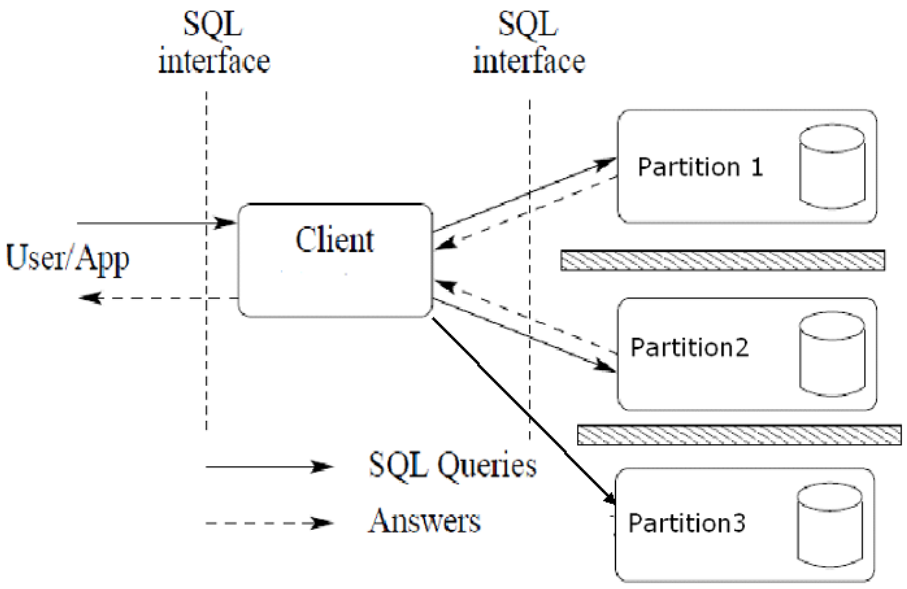

Fig. 2 Proposed System Architecture

The general architecture of a distributed secure database service, as illustrated in Fig 2, consists of a trusted client as well as two or more servers that provide a database service. Client will execute the queries on various fragments and proposed algorithm will find out the best minimized query processing time. Also client can execute fuzzy queries on the fragments which are created by partitioning algorithm. Fuzziness in the database can be handled by transforming fuzzy queries into normal SQL queries.

In the proposed system, database is partitioned into three partitions dbfpartition1, dbfpartition2 and dbfpartition3. In the database design, dbfpartition1 consists of two vertically fragmented relations on the basis of tupule Id (tid) from which original global relation can be recovered by applying simple join algorithm. Dbfpartition2 consists of three vertically fragmented relations on the basis of tupule id and dbfpartition3 consists of original global relation.

In the proposed query processing and query optimization process, query response time is compared against these two vertically fragmented relations with global relation and fragments with minimum response time for read only and update application is selected for development distributed database applications.

A. Fuzziness in database

A database is an ordered collection of related data elements intended to meet the information needs of an organization and designed to be shared by multiple users. If a regular or classical database is a structured collection of records or data stored in a computer, a fuzzy database is a database, which is able to deal with uncertain or incomplete information using fuzzy logic. Basically, a fuzzy database is a database with fuzzy attributes, which may be defined as attributes of an item, row, or object in a database, which allows storing fuzzy information.

The following is a brief definition of the characteristics of imperfect data:

- Uncertain data - The uncertainty is related to the degree of truth of its attribute value, and it means that we can apportion some, but not all, of our belief to a given value or a group of values.
- Vague data - Lack of definite or sharp distinctions. Imprecise data - The imprecision and vagueness are relevant to the content of an attribute value, and it means that a choice must be made from a given range (interval or set) of values but we do not know exactly which one to choose at present.
- Inapplicable data - There may be some entities for which a piece of data relating to one of its properties cannot be acquired due to a lack of the property.

In the real time situation, people express their ideas using the natural languages. Normally natural language has a lot of vagueness and ambiguity. However, while applying one's thoughts as a query in terms of natural languages into the database, a lot of problems are experienced due to the inefficiency of RDBMS to handle such queries.

Consider the query "Give names of the employees who have highest and lowest salary".

This query cannot be processed directly by the SQL, since it contains a lot of vagueness like highest and lowest salary. The best remedy for modelling the above situation is by the use of fuzzy Sets.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2013

IV. SYSTEM IMPLEMENTATION

The proposed scheme is implemented using Microsoft Visual Studio 2010 and Microsoft SQL server 2010. The Working scheme is,

1. Create a different partitions for the database (here three database are created with different schemas).
2. Apply workload i.e. queries on all databases and estimate the cost of each query in terms of response time.
3. The number of queries applied on the partitions is varied from 1 to 100.
4. Different types of query operations such as Insert, Update, delete, fetch and fuzzy value based searching are applied on all the partitions
5. Fuzzy range is defined for the salary attributed defined in all the partitions
6. If user is fetching records of particular fuzzy ranges then query is classified into fuzzy range1, 2 or 3 as per the user input.
7. Then the comparison is done on the basis of cost of queries i.e. response time of the query

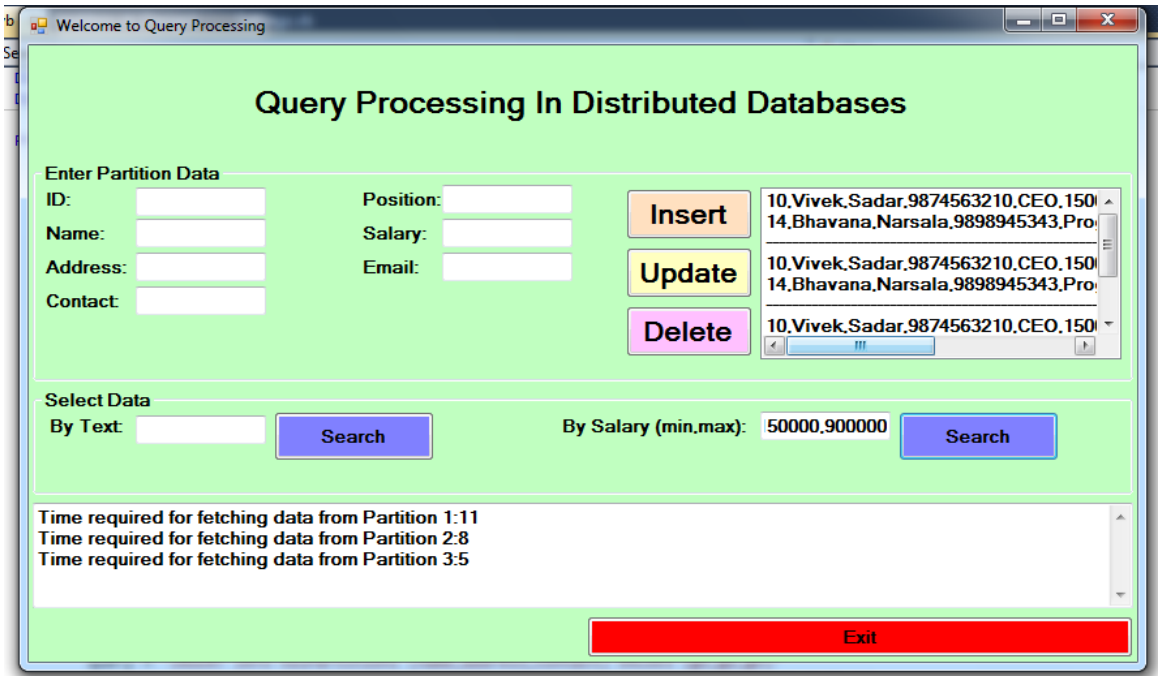

Fig. 3 Fuzzy Query for all partitions showing response time

For Fuzzy queries which we had simulated on salary attribute of the partitions, partion3 is best candidate as it results in least response time. Fuzzy ranges are defined in terms of minimum and maximum salary which will be entered by user in main window of project and according salary will be assigned proper fuzzy ranges as defined in system. Comparison of simulating normal query and fuzzy query on all partitions shows that fuzzy queries will always take less response time as compared to normal SQL query. Reason for the least response time is that query evaluation engine does not have to search all the records in database; it has to search records in fuzzy range values. Hence fuzzy queries are most optimized queries in distributed databases.

V. CONCLUSION

We have proposed method of incorporating fuzziness in distributed database for accommodating fuzzy queries which are approximations based on human knowledge. Fuzzy queries to relational database are proposed as one candidate model and fuzzy relational database is presented as another relational model. With the help of example of employee distributed database, we have successfully implemented fuzzy logic based queries using dot net environment for GUI and SQL server for databases. This establishes feasibility of the concepts proposed.

Distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur, and it provides local control of data for users. However, there is some complexity when attempting to manage and control distributed database systems. A distributed database allows faster local queries and can reduce network traffic. With these benefits comes the issue of maintaining data integrity.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2013

Proposed work for query processing handles two important issues in data distribution i.e. minimizing query response time through partitions and handling fuzziness in database by translating fuzzy queries into SQL.

REFERENCES

1. C. T. Yu, K. C. Guh, W. Zhang, M. Templeton, D. Brill, and A. L. P. Chen, "An Integrated Algorithm for Distributed Query Processing," in IFIP Conference on Distributed processingAmsterdam, 1987.
2. D. Chiang, L.R. Chow and N. Hsien, "Fuzzy information in extended fuzzy relational databases", Fuzzy Sets and Systems 92, pp.1-10. (1997).
3. G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani,U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In Conference on Innovative Data Systems Research,2005.
4. G.J. Klir and B. Yuan [2001], "Fuzzy Sets and Fuzzy Logic: Theory and Applications", Prentice Hall, Inc. Englewood Cliffs, N. J., U.S.A.
5. G. Pelagatti and F. A. Schreiber, "A Model of an access Strategy in Distributed Database System," in International Conference of Database Architecture, Venice, Italy, 1979.
6. J. Callan, "Distributed information retrieval". In Advances in Information Retrieval, W. B. Croft, Ed. Kluwer Academic Publishers, 2000, pp. 127–150.
7.] K. T. Huang, "Query Optimization in Distributed Databases," in Laboratory for information and Decision Systems: MIT, 1982.
8. Li, Victor O. K. "Query processing in distributed data bases", MIT. Lab.for Information and Decision Systems Series/Report no.: LIDS-P ; 1107,1981
9. M. Tamer Ozsu, Patrick Valduries, "Principles of Distributed Database System", Second Edition, Pearson Education, pp 169.
10. N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of max-csp problems. In J. Comput. Syst. Sci., pages 212–243,2003.
11. Nilarun Mukherjee, Synthesis of Non Replicated Dynamic Fragment Allocation Algorithm in Distributed Database System", Published in Proceeding of International conference on advances in Computer Science , 2010
12.] P. Valduriez and T. Ozsu, "Principle of Distributed Database Systems.," Prentice Hall,1999. P. A. Bernstein, N. Gooman, E. Wong, C.L. Reeve, and J.B. Rothie, "Query Processing in a system for Distributed Databases", ACM Transaction on Database Systems, Vol. 6, pp. 602- 625, (1981).
13. Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of Database System", Fifth Edition,Pearson Education, Second Impression, pp 894, 2009.
14. P. A. Bernstein and W. C. D-M, "Using semi-joins to solve relational queries," ACM vol. 28:1, pp. 25-40,1981.
15. M. S. Chen and P. S. Yu, "Combining Join and Semi Operations for Distributed Query Processing," IEEE Trans. of Knowledge & Data Engineering, vol. 5:3, pp. 534-542, 1993.
16. M. S. Chen and P. S. Yu, "A Graph Theoritical Approach to Determine a Join Reducer Sequences in Distributed Query Processing," IEEE Trans. Of Knowledge & Data Engineering, vol. 6:1, pp. 152-165, 1994.
17. M. S. Chen and P. S. Yu, "Interleaving a Join Sequence with Semijoins in distributed Query Processing," IEEE Trans. on Parallel and Distributed Systems, vol. 3:5, pp. 611-621, 1992.
18. T. Ibaraki and T. Kameda, "On the Optimal Nesting Order for N-Relatioanl Joins," ACM (TODS), vol. 9:3, pp. 482-502, 1984.
19. P. A. Bernstein, N. Goodman, E. Wong, C. L. Reeve, and J. B. Rothie, "Query Processing in a System for Distributed Databases (SDD-1)," ACM Transactions On Database Systems, , vol. 6: 4, pp. 602-625, 1981.