IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-9, Sept. 2018 E-ISSN: 2347-2693

Comparative Study of Optimization of data query for SPARQL for Distributed Queries

Rakesh Kumar Pandey[1*] , Sachindra Kumar Azad[2]

[1,2]University Department of Statistics and Computer Applications
T. M. Bhagalpur University, Bhagalpur-812007, India

[*]*Corresponding Author: skazad@rediffmail.com*

Available online at: www.ijcseonline.org

Accepted: 20/Sept/2018, Published: 30/Sept/20

***Abstract*—** Semantic search tool is a user-friendly tool which helps to improve search accuracy by understanding what the user wants to search in the search space on the web or a closed system. But there are large number of challenges for translating the data query to SPARQL for better readability and visual ability. SPARQL is a RDF query language i.e. a semantic query language for database, which is able to retrieve and manipulate data stored in Resource Description Framework(RDF) format. Present work provides the optimum result of running queries over different SPARQL end points. This paper presents the comparative study of different algorithms for optimization and also discusses the two aspects of the result optimization like ranking and readability and it concludes the result for the user data.

***Keywords*—**Relational Database, SPARQL , RDF, Basic Graph pattern

I. INTRODUCTION

The term SPARQL is a semantic query language for databases which is able to retrieve and manipulate data stored in RDF format[1,2]. Due to the better features of SPARQL, the semantic research communities focus on this. As a result of these efforts, the SPARQL is the core technology for querying the semantic web. SPARQL provides the facility to generate the query on the RDBMS, Distributed databases with the help of RDF via a middleware. For example, a RDBMS can queried with SPARQL through RDB2RDF mapping software. However the Linked Open Data is utilized by the programmers and semantic technology experts only. The problem of friendly user interfaces for accessing Linked Data is still largely unsolved[3]. The main feature for the user friendly interfaces is the better visualization Result ranking. For searching , There are various tools utilized like Swoogle [4], Sindice[5] and Watson[6]. Increase in size of data adds a need for procedures to rank results and display the results in a user friendly way. Semantic search tool is a user friendly tool which helps to improve search accuracy by understanding what the user wants to search in the search space on the web or a closed system. But there are large number of challenges for translating the data query to SPARQL for better readability and visual ability. SPARQL is a RDF query language i.e a semantic query language for database, which is able to retrieve and manipulate data stored in Resource Description Framework(RDF) format. Present work provides the optimum result of running queries over different SPARQL end points. The paper presents the comparative study of different algorithms for optimization and also discusses the two aspects of the result optimization like ranking and readability and it concludes the result template for the user data. This work focuses on two aspects: (i) result position and (ii) compare results of queries.

II. OPTIMIZATION ALGORITHM

SPARQL is the query language [8] for query on the RDF data. Actually SPARQL generates queries in the form of triples patterns i.e. the pattern of SPARQL dependent on the basic graph pattern (BGP). The patterns of BGP and RDF are same. RDF triples [7], that may contain the values of the variables at the Subject, Predicate and Object locations of the variables. We know that the structure of web, it's in distributed in nature. So that, the query generation from all the sources are not possible due to the bad network speed, server problems, unauthorized access etc. The query optimization for SPARQL fully dependent on the rewriting of query and rearranging of triple patterns on their selection pattern. The roles of local relational databases are on the optimization of query quite high. In this paper, we implement the three different algorithms for finding the optimize solution with the help of various joining operations, can be adapted to SPARQL. While processing distributed SPARQL queries, most network traffic and processing time are affected due to execution of triple patterns, and query optimization aims to find out the optimal execution order and access plans of triple patterns. In a query plan all triple patterns are executed sequentially using plain access plans, any order of execution produces the same amount of network traffic and has the same processing time. Therefore, the choice that which triple patterns are executed using dependent access plans and whose bindings are required by these dependent access plans i.e. the number and dependency of dependent access plans in a query plan. This involves two steps. First we find the query plan with minimum response time which provides the information that how triple patterns are joined. Then we determine the actual order to execute all triple patterns in a parallel fashion. It should be noticed that whether a join is executed as hash join or bind join it will not affect the number of results of join. The methods to execute operations can also be determined during query execution using real-time statistics. For finding the optimum solution, we can use two algorithms and find the outcomes and finally we can design a hybrid method for optimization. This hybrid algorithm is the combination of both algorithms with some modification. The main goal of these implementations are to find the minimum response-time query plan, thus all possible execution orders are examined and for a specific order the best method to execute each operation is determined. At first scans all triple patterns that will be executed using simple access plans, and uses algorithms to decide access plans and order of join for rest of the triple patterns. Once the optimal query plans is generated, determines the execution order of the query plan. The query plan is generated by our algorithm.

III. ENVIRONMENT OF EXPERIMENT

For experimental study we use a machine with 64bit Oracle JDK Virtual machine running on Intel Core 2 Duo T6600 @2.2 GHz, 4 GB RAM and windows 8.1 Enterprise Operating System. Specific algorithm Greedy, Genetic and Hybrid algorithms are used for execution of queries. Execution time is combination of optimization time. We set the query evolution timeout to 300 seconds.

IV. ALGORITHM IMPLEMENTATION

We run six queries, each and every queries are different (complex or simple) and evaluates the result, execution time, total number of servers and number of RDF graph.

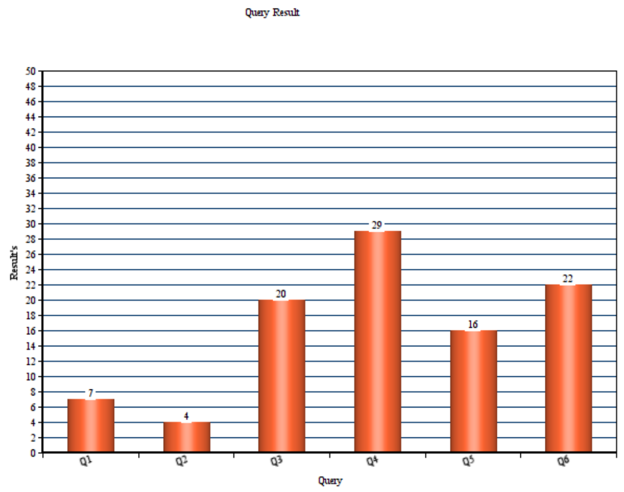
Greedy Algorithm

This algorithm is worked on the concept of bottom-up for building more complex sub plans from the given simple plans up to the complete plane generation [9]. In this algorithm at first bind the access plane for each and every table with the help of query. In the second phase this algorithm performs the simple and rigorous selection of join order with the every repetition of the loop. Algorithm applies a plan finding function[10], in respect to select the next best join. With the help of this algorithm we can find the result, execution time, total no of servers and number of RDF graph as:

Table 1.1: Applied Queries result.

Query	Q1	Q2	Q3	Q4	Q5	Q6
Result	7	4	20	29	16	22
Execution Time (Sec.)	200	120	250	195	188	210
Total No. of Servers accessed	22	36	65	69	45	12
No. of RDF graph	300	175	95	503	341	279

Figure 1.1: Query Result



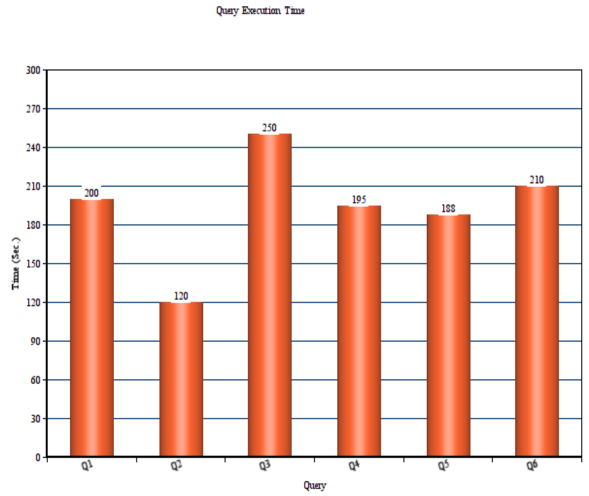


Figure 1.2: Query Execution Time

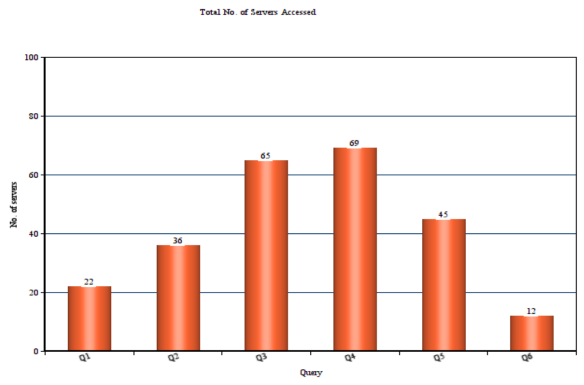


Figure 1.3: Total No. of Servers Accessed

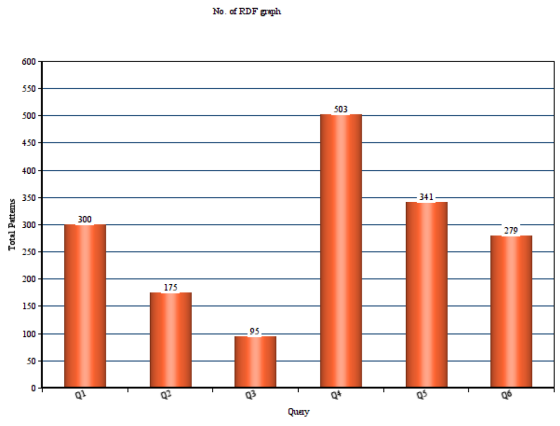


Figure 1.4: Total No. of RDF graph

Here Figure 1.1 represents the total result generated by the SPARQL with the help of greedy algorithm. Figure 1.2 represents the total time taken in the process of query generation. Figure 1.3 represents the total number of server accessed in the web for generation of the optimum result. Figure 1.4 represents the total number of RDF patterns generated of selection of optimum solution.

Genetic Algorithm

Genetic algorithm is used in search and optimization with the natural evolutionary process according to which living organisms adapt themselves to changes in the environment[11]. It can include the following for searching and optimization:- Encoding Schemes, Fitness Function and selection of parent's , genetic operators[12]. It can starts the search on the basis of binary strings, real numbers, permutations of elements, lists of rules, program rules[11]. Genetic algorithm(GA) is used to generate 'close' query plans. The aims of GA is to generate query plans that are optimal with respect to the number of sites involved and the concentration of relations in these sites, for answering the user query.

We can apply this algorithm with same query applied into the greedy algorithm for finding the outcomes , get the following

Query	Q1	Q2	Q3	Q4	Q5	Q6
Result	**5**	**5**	**22**	**25**	**18**	**28**
Execution Time (Sec.)	**142**	**120**	**200**	**181**	**150**	**195**
Total No. of Servers accessed	**26**	**40**	**66**	**71**	**48**	**32**
No. of RDF graph	**275**	**170**	**103**	**470**	**287**	**496**

Table 2.1: Applied Queries result

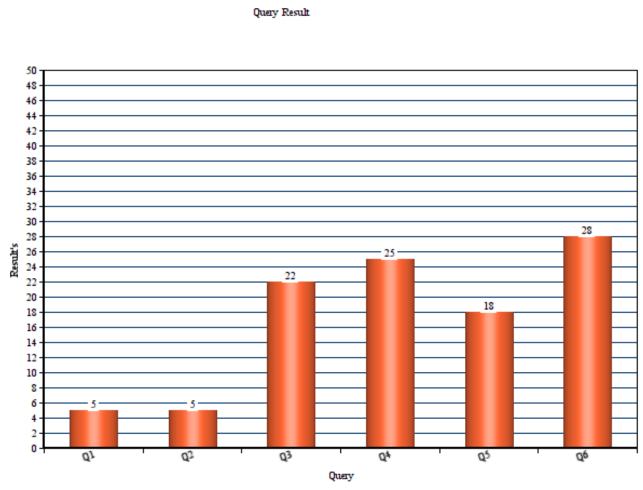


Figure 2.1: Query Result

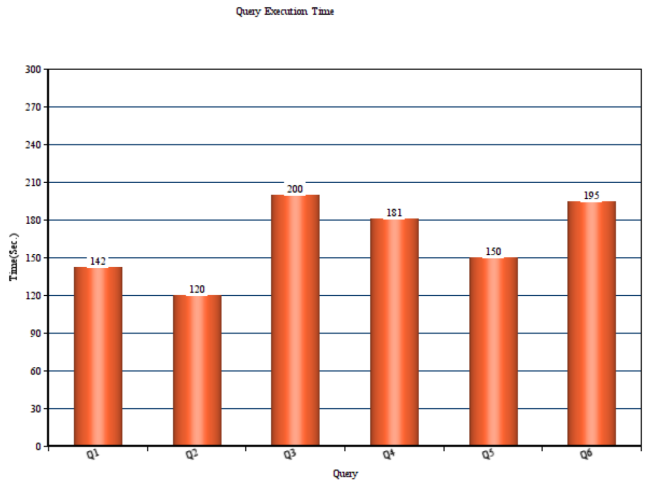


Figure 2.2: Query Execution Time

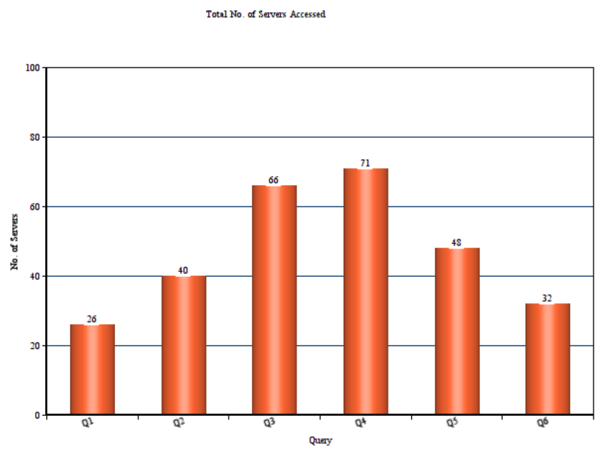


Figure 2.3 Total No. of Servers Accessed

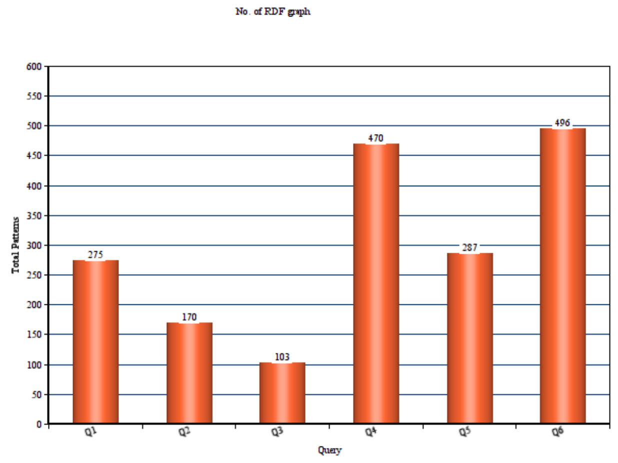


Figure 2.4: Total No. of RDF graph

Here Figure 2.1 represents the total result generated by the SPARQL with the help of genetic algorithm. Figure 2.2 represents the total time taken in the process of query generation. Figure 2.3 represents the total number of server accessed in the web for generation of the optimum result. Figure 2.4 represents the total number of RDF patterns generated of selection of optimum solution.

Hybrid Algorithm

As we know that Greedy algorithm is a good algorithm for entire query processing and evaluation but its sub-steps like accessPlan() and joinPlan() are not optimized at all. So replace all such plans with GA with appropriate operator change. After that every iteration for optimum use of site on network. Also modification in JOINPlan. Only two modification reduce the unnecessary join operation and amplifies less communication with less complexity.

We can apply this algorithm with same query applied into the greedy algorithm for finding the outcomes, get the following

Table 3.1: Applied Queries result.

Query	Q1	Q2	Q3	Q4	Q5	Q6
Result	9	7	25	32	21	30
Execution Time (Sec.)	132	103	195	180	145	193
Total No. of Servers accessed	27	41	66	75	50	39
No. of RDF graph	305	180	107	535	237	320

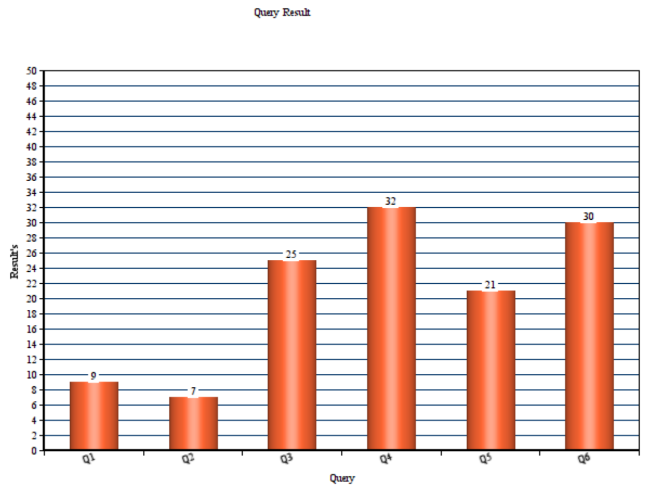


Figure 3.1 : Query Result

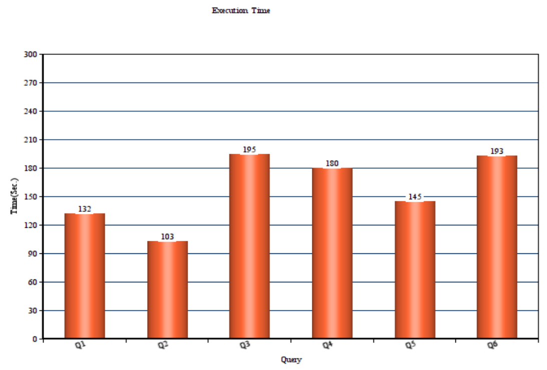


Figure 3.2: Query Execution Time

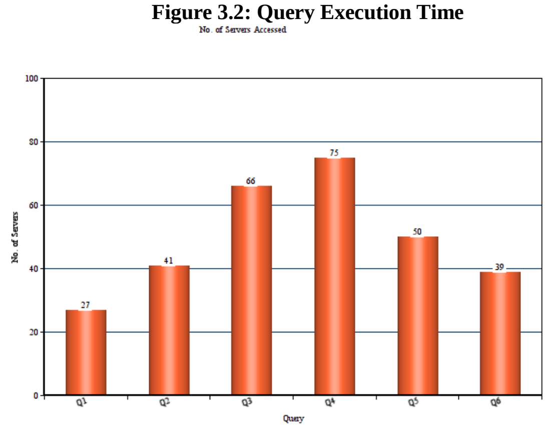


Figure 3.3: Total No. of Servers Accessed

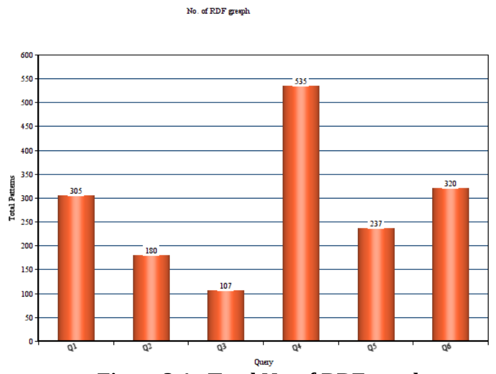


Figure 3.4: Total No. of RDF graph

Here Figure 3.1 represents the total result generated by the SPARQL with the help of hybrid algorithm. Figure 3.2 represents the total time taken in the process of query generation. Figure 3.3 represents the total number of server accessed in the web for generation of the optimum result. Figure 3.4 represents the total number of RDF patterns generated of selection of optimum solution.

V. Evaluation Results

As the above findings if we compare the query result and the time consumed by the query execution process, we can prepare graphs as.

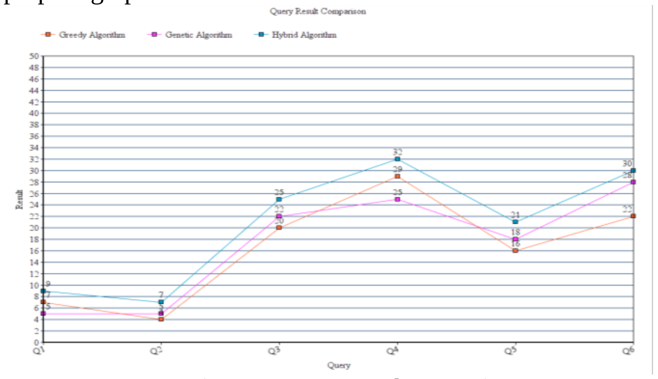


Figure 5.1 Query result comparison

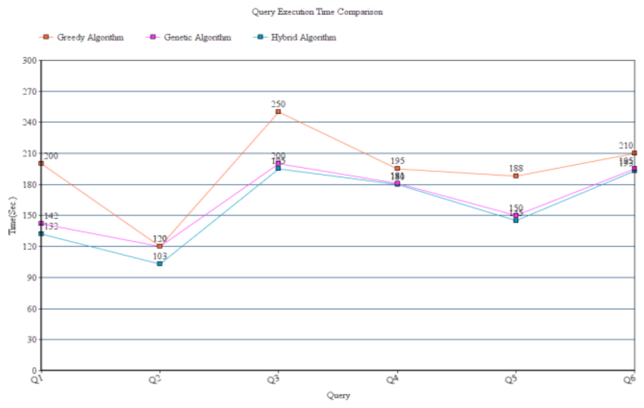


Figure 5.2 Query Execution Time

According to this comparative study of optimum query execution, we can find the better optimum solution given by the hybrid algorithm based query on the SPARQL.

VI. Conclusion

In this paper, comparative study of algorithms are implemented in SPARQL for finding the optimized query results. These include greedy, genetic and a hybrid algorithm. The evolution result of each algorithm is presented along with the table and graphs. For each algorithm initial settings are same and same queries are passed for finding the optimum solution of query result with less time and searching results from various servers or sites. The hybrid algorithm provides better optimized query results. The genetic algorithm provides second better optimized result and greedy algorithm provides third better optimized solution.

References

[1] Jim Rapoza "SPARQL Will Make the Web Shine" eWeek. 2006
[2] Segaran, Toby at al: Programming the Semantic Web. O'Reilly Media, P-84,2009.
[3] P. Hoefler, Linked Data Interfaces for Non-expert Users. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC vol. 7882, pp. 702–706, 2013.
[4] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V.C. Doshi, J. Sachs, Swoogle: A Search and Metadata Engine for the Semantic Web. In: 13th ACMConference on Information and Knowledge Management, Washington D.C. 2004.
[5] G. Tummarello, R. Delbru, E. Oren,Sindice.com:Weaving the open linked data. The Semantic Web, pp:552-565. Springer Berlin Heidelberg, 2007.
[6] M. d'Aquin, M. Sabou, E. Motta, S. Angeletou, L. Gridinoc, V. Lopez and F. Zablith, "What can be done with the Semantic Web? An Overview of Watson-based Applications," 5th Workshop on Semantic Web Applications and Perspectives, SWAP Rome, Italy, 2008.
[7] Franklin, M.J., Halevy, A.Y., Maier, D.: From databases to dataspaces: A new abstraction for information management. SIGMOD Record 34(4) (December 2005) 27–33.
[8] Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C recommendation (January 2008) Retrieved June 11, 2009, from http://www.w3.org/TR/rdf-sparql-query /.
[9] Steinbrunn M., Moerkotte G., and Kemper A., "Heuristic and Randomized Optimization for the join Ordering Problem" VLDB JOURNAL, vol. 6, no. 3, pp. 191-20, 1997.
[10] Kossmann D. and Stocker K., "Iterative Dynamic Programming: A New Class of Query Optimization Algorithm", ACM TODS, March 2000.
[11] M. Mitchell, "An Introduction to Genetic Algorithms", MIT Press, 1998.
[12] J. H. Holland, "Adaptation in natural and artificial Systems", University of Michigan Press, 1975.
[13] Xiangning Liu, Bharat K. Bhargava, "Data Replication in Distributed Database Systemsover Large Number of Sites",Computer Science Technical Reports. Paper 1229
[14] X. M. Chandy and J. Misra, "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems " in ACM, 1982.
[15] B. M. M. Alom, F. Henskens, and M. Hannaford, "Deadlock Detection Views of Distributed Database," in International

conference on Information Technology & New Generartion (ITNG- 2009) Las Vegas, USA: IEEE Computer Society, 2009.
[16] Parul Tomar, Megha “An Overview of Distributed Databases”, International Journal of Information and Computation Technology. ISSN 0974-2239 Volume 4, Number 2 (2014), pp. 207-214
[17] Maniural B.M et al.,”Query Processing and Optimization in distributed database”,IJCSNS, vol 9,No.9,2009
[18] Bhuyar P.R. “Horizonatal Fragmentation technique in Distributed database”,IJSRP,vol2,issue 5,2012

Authors Profile

Mr. Rakesh Kumar Pandey is currently working as Research Scholar toward the PhD degree at the University Department of Statistics and Computer Applications, T. M. Bhagalpur University, Bhagalpur-812007, India. He received the Master of Computer Applications degree from T. M. Bhagalpur University, Bhagalpur-812007, India in year 2010. His area of research interests includes Distributed Database, Query Processing and Optimization.

Dr. S. K. Azad is an Associate Professor and Head, in University Department of Statistics and Computer Applications at Tilka Manjhi Bhagalpur University, Bhagalpur, India.