International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Performance Analysis of Query Terms during Query Phase in Virtual Databases

Mohanty Anita, Mishra Jyoti Prakash, Mishra Sambit Kumar

Abstract: ***In early days, the queries may be executed by selecting query plans and linking throughout the execution. The main intention in such case may be to treat query execution as a process of streaming the tuples and systematize the query terms on the basis of tuples. The accuracy of routing of tuples may be obtained through semantic properties of the operators and implementation of join predicates. Considering the huge amounts of data in the dynamic environment, it may be required to locate and extract data implementing different techniques. The primary objective in this work may be to implement the virtualization along with conceptualization approaches to improve the throughput as well as turnaround time and to link to associated real databases. In such scenario, the virtual databases may be associated to enhance the computation capabilities of query plans by optimizing the performance and maintaining consistencies in the databases. Accordingly the query latency may be minimized.***

Keywords : Tuples, Virtualization, Query plans, Query latency, Join predicates, Streaming, Join index

I. INTRODUCTION

In case the delay of data sources, the pre-processing of data items may also support to the partial evaluation and outcome of processed data. But in general, it may not join the data and accumulate the join index linked with every query terms. Considering the query phasing, it has been observed that data may be integrated in the transformation process while implementing internal mechanisms. It may be responsible to transform the data items into different phases through scheduling mechanisms. While implementing the initial query phase, the index may be shared and accordingly similarity index may be generated and query terms may be sequenced globally.

Manuscript published on 30 September 2019
* Correspondence Author
Mohanty Anita*, Department of MCA, Ajay Binay Institute of Technology, Cuttack, Odisha, India,
Mishra Jyoti Prakash, Department of Computer sc.&Engg., Gandhi Institute for Education and Technology, Baniatangi, Bhubaneswar, Odisha, India.
Mishra Sambit Kumar, Department of Computer sc.&Engg., Gandhi Institute for Education and Technology, Baniatangi, Bhubaneswar, Odisha,

So, when a request may be sent to the system, response may be generated globally and all the query terms associated with data items may be prioritized to enhance the throughput. Sometimes scrambling may be implemented to enhance the performance as it modifies query execution plans during runtime. In general, it may be linked to simple heuristics to provide better performance. Accordingly, the optimization criteria may be responsible towards making intelligent scrambling options. In addition to that, while considering the approaches towards query scrambling, Sometimes it may be required to optimize the response time and to construct the alternative query plans. In such cases, the optimizers linked to response time may be highly essential towards query scrambling to monitor the throughput and associated query plans. It may be noted that the query optimizers may incline towards join dependencies while validating query plans. Also it may need the information about the cardinalities of the queries along with join and selection predicates. So depending upon the technique, the sub-queries may also be optimized. For general information, the process associated with query scrambling either may dynamically reschedule the query plans or may create new operators to support executing the query plans. Sometimes, uncertain data may be available in the application due to technological advancement and accumulation of data in imprecise manner. In such case, to minimize the query response time in the applications, it may be essential to modify the query processing techniques and prioritizing the query terms. So, while designing the tools the query terms from the linked databases, it may be required to specify the sub-query system along with data items to generate intermediary data sets and to apply the subsequent operations associated with query terms. The technique associated to retrieve the optimal query processing method may be termed as query optimization. In the similar context, the distributed database maybe linked to different nodes in the locations in the distributed networks. The cost of processing as well as transmission may more essential while retrieving optimal performance of query terms and sometimes may be challenging to other similar techniques.

II. REVIEW OF LITERATURE

Fragkiskos Pentaris et al.[1] in their work have experimented on query execution plans and observed that the execution plans are cost effective and maintain similarities throughout the execution process. Vikash et al.[2] in their experimentation focused towards dynamic programming strategies based on search strategies.

In general there may be two techniques to be implemented to solve the problems related to search strategies. Yannis E et al.[3] in their work have discussed the basic approaches of randomized algorithm towards optimizing the large join queries. They have also compared the performance of simulated annealing and iterative improvement by conducting number of experiments on a variety of queries and database. They have also tried to obtain the optimal query plan solution at some specific points. Donald kossmann et al.[4] in their work have discussed about the deterministic behavior of randomized algorithms along with estimated running time. They have also made the case studies on randomized algorithms linked to queries and observed better performance. Yadav et al.[5] in their work have focused towards the distributed query optimization problems. They observed that the result of the query optimization may be more effective when the required data may be saved at the same location. Otherwise, if the data required to answer user query may be on some other locations, in that situation the query processing efficiency may not be so efficient. Giri et al.[6] in their work have focused on the two-phase optimization algorithm linked towards iterative improvement and simulated annealing. In such situation, the techniques associated with iterative improvement may be associated with simulated annealing. Gultekin Ozsoyoglu et al.[7] in their work have observed that the solution mechanisms associated with simulated annealing may yield better result in the fixed amount of time. Kumar S. et al [8] in their research have focused on heuristic query optimization. They have implemented heuristic function approach to evaluate query search efficiency in database operations. In such cases, the experimentation may enhance an increase in query search compared to traditional query searches. Gupta Er. Shobit et al. [9] in their work have focused on data execution linked to the databases. They focused on inline query techniques as well as query optimization methods implementing the remote machines. Vishal Hatmode et al.[10] in their study have focused on heuristic query optimization. They have implemented the technique during selection and optimization of queries. They observed that the data access time may be minimized as per the reduction of number of tuples and the number of columns. Vineet M et al.[11] in their work have focused towards android application, in which it may be required to calculate data access time to maintain data access performance of the android application. Accessing the data as well as users at the same time may slow down data access time. In such cases, this application may require efforts to maintain the performance of data access time. Accordingly, the query optimization method may transform the query with a number of rules to speed up the performance of data execution in database and may try to minimize the number of accesses by reducing the number of tuples and number of columns to obtain optimal execution time. Sunita M et al.[12] during their studies have projected towards query join which may be divided into two parts, i.e. tree and cyclic. The result Conversion of cyclic queries into the query tree using different approaches may increase data execution time. Dokeroglu T et al.[13] in their work have focused regarding complex queries including subexpressions. They have also focused towards join operations as well as simultaneous execution of complex queries. Jean H et al.[14] in their work have focused regarding efficiency of queries as well as accessing data. It is understood that database administrators may have high capabilities in query optimization techniques to maintain data access performance. So it may be essential to determine the best join operation execution time in a relational database towards the query optimization problem.

III. PROBLEM FORMULATION

Algorithm 1 : Regeneration of the query terms with heterogeneity

Step 1. Generate the query terms linked to database, Qt
Step 2. Rearrange the query terms in ascending order, qt
Step 3. Retrieve the query plans associated with the query terms linked to the database, qi
Step 4. Insert the query plans to the set and validate the query terms
Step 5. while(min(cost(query plans)!=0)) do
Step 6. for (each query term qt in database, Qt) do
Step 7. for (each query plan, qi in the dataset) do
Step 8. if (qi.value/2 > qt.value) then
Step. 9. TQ = Qt ; (TQ may be defined as no. of heterogeneous databases in the system), end if
Step 10. if(TQ==0) then break
Step 11. else update and link TQ to Qt The query terms associated with the databases with heterogeneity may be evaluated from the dataset associated with the system and the query plans may require considerable quantum time towards retrieval during executing the databases. In this regard, techniques may be adopted to optimize the query performance implementing the algorithm.

Algorithm 2 : Linking index terms of query plans to the databases

Step 1. for (each qi in dataset) do
Step 2. for (each qt in TQ) do
Step 3. if (qi > qt) then
Step 4. Add qi list in Qt and qt list in TQ
Step 5. else if (qt > qi) then
Step 6. if (Qt!=n) then (n denotes total no. of query terms)
Step 7. Add qi to TQ

Algorithm 3 : Optimizing query terms

Step 1. Check the cardinality of the databases. If any tuple has null value, synchronize may not be possible with no solution.
Step 2. Select the size of population and initialize in random.
Step 3. Check the query plans linked to the population and compare with next query term. If any violation, mutation may be done randomly with the query plans. It may be repeated until each query term may be associated with query plans.
Step 4. Fitness parameter of individual query term may be evaluated.
Step 5. If the fitness parameters of individual query terms may be 0, then no partition solution may be obtained.

Step 6. Selection of individual query terms may be done randomly as per the fitness parameters and regeneration of query terms may be done until satisfaction of condition.
Step 7. Evaluate the cost of the query terms and find the optimality and feasible solution.
Step 8. While(cost(query terms)!=min(cost(query terms))) choose query plan in random and compare with the current state.
Step 9. If (cost(query plan)!=0) repeat the steps until achieving the condition. The main objective of linking the index terms of query plans as well as optimizing the query terms may be to enhance the performance during phasing. It may gain towards iterative improvement of query terms during the initial stage and may generate better query plans with average query processing cost.

Table i. Analysis of query term accessibility with size of data servers

Sl.No.	Size of data servers	Query term accessibility (%)
1	5	29
2	6	37
3	7	41
4	10	55
5	15	57
6	16	51
7	20	39

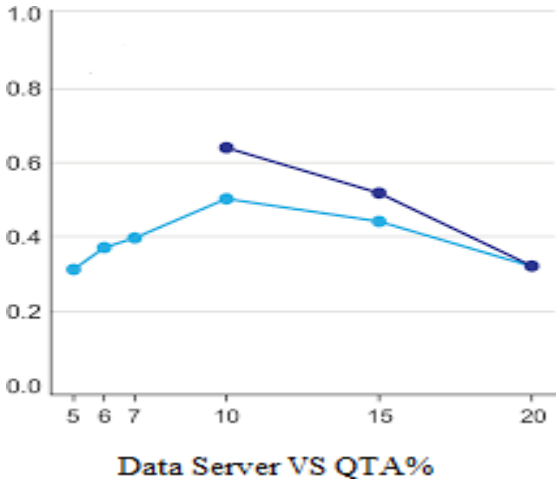

Figure 1. Data server VS Query term accessibility

It has been observed that the query term accessibility in general may depend on the size of data servers. At specific location, depending on the size of the data servers, the optimality of the query execution may be measured.

Table II. Analysis of query term accessibility with query phased locations

Sl.No.	Size of data servers	Query phased locations	Query term accessibility(%)
1	20	9	4.7
2	20	19	9.2
3	20	27	12.7
4	20	33	13.9
5	20	37	14.5
6	20	45	15.4
7	20	47	15.7

Considering the query phased locations along with the fixed size data servers, it has been observed that the query term accessibility may be directly proportional to the query phased locations.

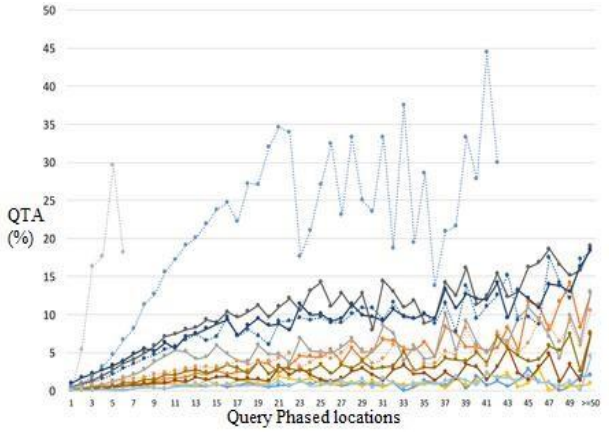

Figure 2. Query phased locations VS Query term accessibility

IV. DISCUSSION AND FUTURE DIRECTION

In general, the optimizer in query processing may be associated with the search strategy along with dynamic programming approach. Being the deterministic approach, it may adopt specific search mechanism to obtain the better search space. It is understood that the peripheral like CPU, I/O may be directly proportional to the execution time of the system. In such scenario, the total cost may be accumulated considering the CPU cost and I/O cost. The procedure and method adopted in this case eradicates the flaws associated with server mechanisms as well as query phased locations. It may be treated as better approach towards obtaining near optimal solution. The main intention may be to obtain approachable feasible solutions to avoid high cost. By implementing the approach it may be easier to produce better optimal query plans with the large size of databases. So it may not only generate the better query plans but also may focus to optimality with respect to time and average query processing cost.

V.CONCLUSION

In general the specific query optimization may make the selected tuples in the databases much less before join index operations. Accordingly, at the time of join operation, it may produce a minimum tuples and may require a relatively short execution time. It has been observed that data execution speed sometimes may use query optimization, 29% faster than original query.

REFERENCES

1. FRAGKISKOS PENTARIS and YANNIS IOANNIDIS, “Query Optimization in Distributed Networks of Autonomous Database Systems”, ACM Transactions on Database Systems, Vol. 31, No. 2, June 2006, Pages 537–583.

2. Vikash Mishra and Vikram Singh "Generating Optimal Query Plans for Distributed Query Processing using Teacher-Learner Based Optimization", Elsevier Procedia Computer Science 54 (2015) 281 – 290.
3. Yannis E. Ioannidis ,Younkyung Cha Kang ,"Randomized Algorithms for optimizing large join queries" ACM 1990, pp 312-321.
4. Donald kossmann and konrad stocker, "Iterative Dynamic Programming: A New Class of Query Optimization Algorithms", ACM Transactions on Database Systems, Vol. 25, No. 1, March 2000, Pages 43–82.
5. Yadav, Pramod Kumar, Rizvi, S.A.M," Efficient Query Optimization: A novel approach for generating optimal query plans using Iterative Improvement and Simulated Annealing", in the International Journal of Control Theory and Applications Vol No: 09, Issue No: 21, October, 2016, pg No: 243-249. ISBN: 09745572, SJR=0.53.
6. Giri, A. K., and R. Kumar. "Distributed query processing plan generation using iterative improvement and simulated annealing", 2013,3rd IEEE International Advance Computing Conference (IACC), 2013.
7. Gultekin Ozsoyoglu, Ismail Sengor Altingovde, AbdullahAalhamdani, Selma Ayse Ozel and Ozgur Ulusoy, Zehra Meral Ozsoyoglu, "Querying Web Metadata: Native Score Management and Text Support in Databases", ACM Transactions on Database Systems, Vol. 29, No. 4, December 2004, Pages 581–634.
8. Kumar S., Khandelwal G, Varshney A., Arora M. 2011 Cost-Based Query Optimization with Heuristics, International Journal of Scientific & Engineering Research Volume 2, Issue 9.
9. Gupta Er. Shobit 2015 Database Management – Inline Queries vc Stored Procedures – an Extended Analysis, International Journal Of Emerging Technologies in Computational and Applied Science (IJETCAS), Page 234-238.
10. Vishal Hatmode, Sonali Rangdale 2014 Heuristic Based Query Optimization, International Journal Of Advanced Research In Computer and Communication Engineering, Vol 3.
11. Vineet M, Kaushik A, Amandeep S.B. 2014 Issues in Query Processing and Optimization, International Journal of Modern Trends in Engineering and Research, International Journal of Modern Trends in Engineering and Research (IJMTER), Volume 01, Issue 05.
12. Sunita M. Mahajan, Vaishali P. Jadhav 2012 General Frame Work For Optimization Of Distributed Queries, International Journal of Databse Management System (IJDMS), Vol 4 No 3.
13. Dokeroglu T., Bayir M. A., Cosar A. 2015 Robust Heuristic Algorithms For Exploiting The Common Tasks Of Relational Cloud Database Queries Applied Soft Computing Journal, Elsivier Vol 30 pages 72-82.
14. Jean H, 2015, Query Optimization Techniques – Tip For Writing Efficient And Faster SQL Queries, International Journal Of Scientific & Technology Research, Vol 4 Issue 10.

AUTHORS PROFILE

Mrs. Anita Mohanty is having more than 10 years of experience in different AICTE approved institutions**.** She is having good number of publications in different indexed Journals.

Prof. Jyoti Prakash Mishra is having more than 20 years of experience in industries as well as education in India and abroad. He is having his research contributions in many referred indexed Journals.

Dr.Sambit Kumar Mishra is having more than 22 years of experience in different AICTE approved institutions. He has more than 29 publications in different peer reviewed International Journals and editorial board member of different peer reviewed indexed Journals.