ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 11(18), DOI: 10.17485/ijst/2018/v11i18/122267, May 2018*

Analysis of Query Optimization Components in Distributed Database

Imran Ali Rana[1]*, Shafiq Aslam[1], Muhammad Shahzad Sarfraz[2] and Umar Shoaib[2]

[1]Department of Information Technology, Faculty of Computing and Information Technology, University of Gujrat, Gujrat, Punjab, Pakistan; ranaimran.it@gmail.com, shafeakaslam@gmail.com, [2]Department of Computer Science, Faculty of Computing and Information Technology, University of Gujrat, Gujrat, Punjab, Pakistan; shahzad.sarfraz@uog.edu.pk, umar.shoaib@uog.edu.pk ranaimran.it@gmail.com, shafeakaslam@gmail.com, shahzad.sarfraz@uog.edu.pk, umar.shoaib@uog.edu.pk

Abstract

Objectives: This paper brings to light different query optimization components and their optimizing functionalities which are helpful to improve the response time of query and the efficiency of distributed database. A cache based optimization is also analyzed to highlight the query optimization process. **Methods:** As data is the most valuable asset for any organization due to this they want to get access and use it efficiently and in a timely manner. To evaluate the efficiency of query optimization its different components e.g. search space, search strategy and cost model are evaluated with the help of examples, tables and diagrams. By comparing the different results, a cache based optimization technique is also evaluated. **Findings:** It is observed that in search space generated plans are equivalent in the sense they provide same results but their operation, implementation and performance is different. Different algorithms of search strategy are also examined to get the quicker and accurate results and notice that movement of search strategy is greatly depend upon join ordering and cost model. It is also observed that the cost model is helpful to select the best query execution plan but it depends upon the different parameters for example queue length, sever distance, server capacity and load. The latest cache based query optimization technique is also examined and noted that it is key to improve the response time of query as its computational cost is very low. It will be more helpful if it is placed at each site. **Applications and Future Improvements:** Currently cache based query optimization is applicable only for homogeneous distributed databases. In future this technique can also be implemented for heterogeneous type of databases.

Keywords: Distributed Database, Query Processing, Query Optimization, Search Space, Search Strategy, Cost Model, Centralized Database, Cache

1. Introduction

Databases allow users to efficiently store, retrieve and analyze data. Databases are very vital for business, research organizations and other fields where data has important role to play. Globally in business environment, organizations are to process more data than ever before. Data is most important asset for any organization[1]. All the crucial business decisions are made on the basis on that available data. Only a well-designed data management policy will make data more reliable to help make better decisions.

The volume of data is continuously increasing, the centralized databases are becoming bottleneck for organizations that are physically dispersed and have to access data remotely[2,3]. Data management is very easy in centralized databases because only a single database administrator can handle it[4]. Despite the fact that centralized databases have high communication cost and also most importantly these have high response time. In *Principles for Distributed Databases in Telecom Environment* it is suggested that there is an attractive option for such organization to switch on the concept of distributing the data over multiple sites, because it has benefits over centralized databases such as availability, reliability, reduced communication overhead, data localization, improved performance and an easier system expansion[2,5].

The working and performance of a Distributed Database[6] heavily relies on the capability of the query

optimizer[7] to implement suitable query processing plans. Implementation of a query processing plans by using query optimizer is very complicated in Distributed Database as compared to Centralized Database[8]. The only reason for the same is an enormous number of parameters which affect on working of queries in distributed database. In distributed databases relations are fragmented[9] and/or replicated to the different sites[10] and all sites want to get data from each other. Therefore, the response time of query is very high. There are different query execution plans for distributed databases[11]. The responsibility of the optimizer is to find out the indices and sequences in the operation of query performed. The role of optimizer is to indicate the alternatives plans and costs associated with them by using cost model, and select that plan which is cost effective[12].

Joins is another important factor that affects the determination of suitable query execution plans, is to fetch the data from multiple sites[1]. In distributed databases the data comes from multiple sites, so optimization of join query in distributed databases is more complex than centralized databases. Simply a lot of effort is required on query joins in centralized databases and optimization in distributed database[13]. To solve the issue different algorithms are used to provide the results of user's query as soon as possible based on time and cost factor.

To improve the response time of the user's query a cache based query optimization model is used recently. In case, if the query of the user entertained from the cache it saves the huge computational cost and time. Of course, accessing the data placed in cache is faster than accessing the database[14]. Cache based query optimization model is also based on the main four factors i.e. server distance, server capacity, server load and current queue length[15].

2. Distributed Database

"A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network"[6]. The distributed database is widely used with the extension of computer networks and database tools. Due to global business strategy, distributed database has become very trendy and they are mostly used worldwide[15]. Data stored in distributed environment on remote areas which are interconnected using different network topologies or it may be on separate computers which are located in the same geographical location[16].

Distributed database is physically dispersed on multiple sites by fragmentation and replication of data[9,10]. Fragmentation can further be divided into two types horizontal (row wise) used for select operation and vertical (column wise) used for projection operation[17]. Each fragment can be placed on multiple locations so it can be easily used where necessary. This is the best when the same information is accessed from applications that keep running at various destinations[18].

A distributed database is beneficial with the following advantages:

- **Improved Performance:** Since the information is stored on different destinations, so the overhead on one machine diminishes which enhances the execution power.
- **Localization:** Localization means the information is available nearby the site where it is required, in this manner information can be accessed in less time and data transfer time also decreases as well.
- **Availability and Reliability:** The accessibility of the information increments because the multiple copies of the information is stored at various sites. Reliability of data is also increased in case of one site fizzles, because data can be obtained from the other nearby site where the backup of the data is available. In this way, in distributed environment, failure of one site does not cause any distractions or inconvenience.
- **Reduced Communication Overhead:** Communication cost decreases in distributed environment because a relation is accessible at each site locally that contains the replicas of the data.
- **Easier System Expansion:** The capacity of distributed database can be expanded effectively by adding the computers to the network.

No doubt Distributed database systems have so many advantages. However, there are some basic issues, which must be considered in distributed databases and researchers are working on these e.g. management of data, concurrency control, security issue, database administration and response time.

In this paper, major issue that is analyzed is 'response time' because distributed databases are geographically dispersed and users cannot wait for results after initiating the query. Users always assume that computer provides correct data, so their main concern is always efficiency

of the system mean how quickly their queries are entertained. But in designers point of view the main concern is correctness than efficiency, because if the efficiency of the system is high and do not provide accurate result it will be useless for any user or organization. Therefore designer must design the database in a way that provides accurate result without too much delay and also focuses that query consume minimum resources of database. Designer can only overcome these issues if they consider the important aspects of query e.g. query processing and optimization.

3. Query Optimization

The term 'Query' means to search or to find. But in the context of database it is code that transfer to the database to get the information from the database. Queries are usually created using SQL (structured query language), which is similar to high-level programming language. In early stages of database, Codd created the relations. Initially query optimization was not in proper form. Later on the researchers made contributions to produce a query in proper form when networking came into use the distributed query optimizers were introduced[19,20].

Query optimization is one of the key functions of DBMS in which different plans are examined and amongst them identify the suitable plan. Best query plan can be produced by applying different strategies, so the plan that comes after the query optimization may be accepted or rejected[21]. For selecting the best query plan from the given set is dependent upon the cost based query optimizer. In next process the code of selected query execution plan is generated andthe results of query are obtained. This code may be compiled or interpreted[22]. The different steps of Query Optimization Process are shows in Figure 1.

In distributed systems, Query Execution Plan cost is comprised of transmission cost and the local processing cost[6]. While query optimizer generates the best query plan by considering the 3 W's:

- What are the number of alternative plans of query
- What is the cost of every plan of query
- Which plan is the cheapest of query

3.1 Search Space

The search space is the set of equivalent operator trees that can be produced by transformation rules[17], such as

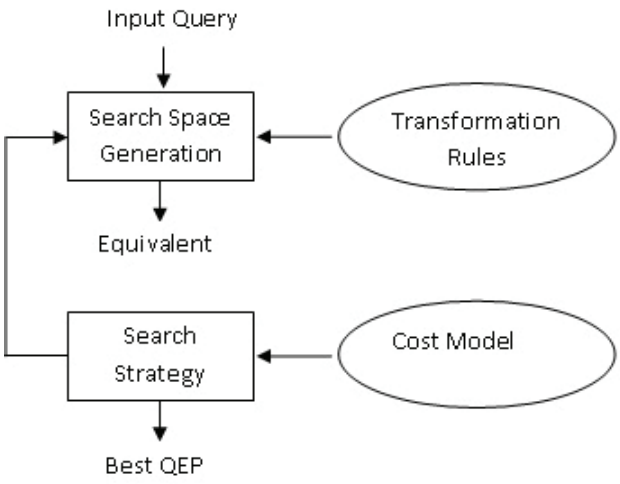

Figure 1. Query optimization process.

those for relational algebra. These generated plans are equivalent, in the sense that they yield the same result, but their operation, implementation and performance is different. Concluding, they have different time and cost consumption. When we apply transformation rule on query we obtain a search space for given query. Then the cost model checks the cost of each execution plan and finds out the suitable one. Cost model can only provide accurate results if it has knowledge of distributed execution environment.

The main consideration of query optimizer is the join ordering. Query optimizer tries to utilize the join in best way due to the following reasons:

- It is easy to understand the problem.
- Joins cost is most effective.
- Most frequent type of queries having joins, selection and projection operator.

Example 1: *Assumed that user initiates this query:*

```
SELECT   SNAME, DEPT
FROM     STU, ENR, CRS
WHERE    STU.SNO = ENR.SNO
AND      ENR.CID = CRS.CID
```

The equivalent join trees of the above query are given below by using the commutative and associative properties of binary operators.

All three strategies produce the accurate, same and exact results, but the cost occurs in their execution is different. The job of query optimizer is to select the best one from the given strategies having low cost. In Figure 2, operator tree (iii) cannot be considered the best plan of search space, because in this plan base table has no join and use Cartesian product which has high cost as compared to rest of the join trees.

Table 1 shows that when numbers of relations in a query are increased then the cost is also increased, because every plan in the search space has its own cost[23]. The basic purpose of the query optimizer is to trace out the best joining method from the given set of relations[24].

Heuristics for Search Space

Sometimes calculating the cost of each execution plan is prohibitive in query optimization because it might be more expensive than actual execution of query. Therefore query optimizer control the size of search space by applying some restrictions[6].

- The first check is that it applies to control the size of relation is heuristics.
  - When accessing the base relation, perform selection and projection first.
  - Avoid from the relation (having Cartesian product) which has no join relation.
- Shapes of the tree may also vary in search space. An important restriction must be followed is the shape of tree.

3.1.1 Linear Tree

In this type of tree minimum one of the operand must be base relation. The volume of search space is reduced up to $O(2^N)$ in it. In the Figure 3 (i) given linear tree, there are four relations and at each stage at least one relation is base relation.

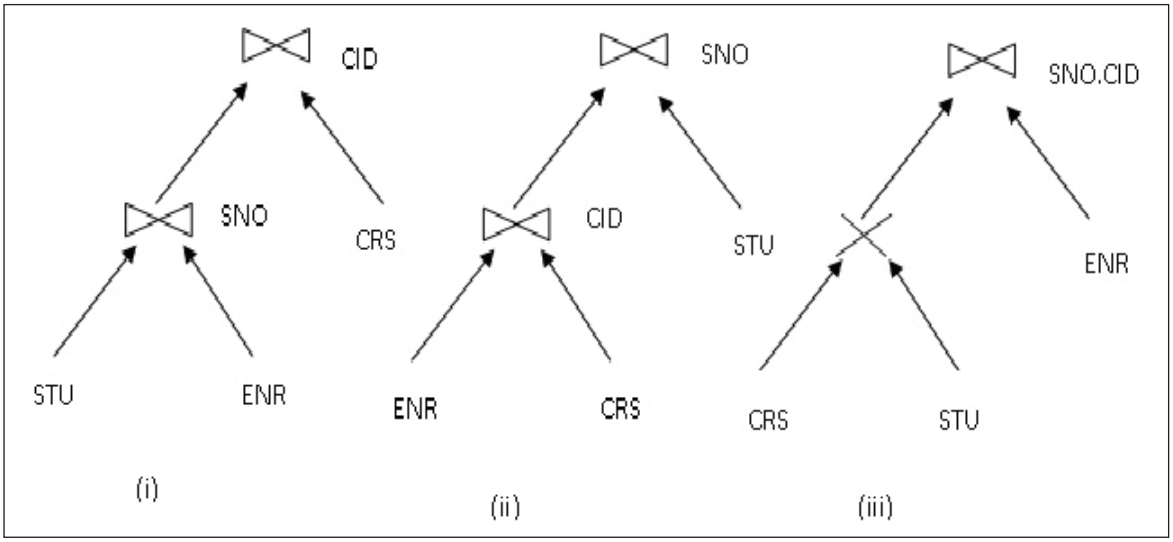

Figure 2. Query equivalent trees.

Table 1. Number of Relation with possible order and sequence with cost

No of Relation (N)	Formula	Name of Relation	Number of possible order	Sequence of possible order	Cost Factor
2	O(N!)	A, B	2	AB, BA	Low
3	O(N!)	X,Y,Z	6	XYZ,XZY,YXZ, YZX, ZXY,ZYX	High

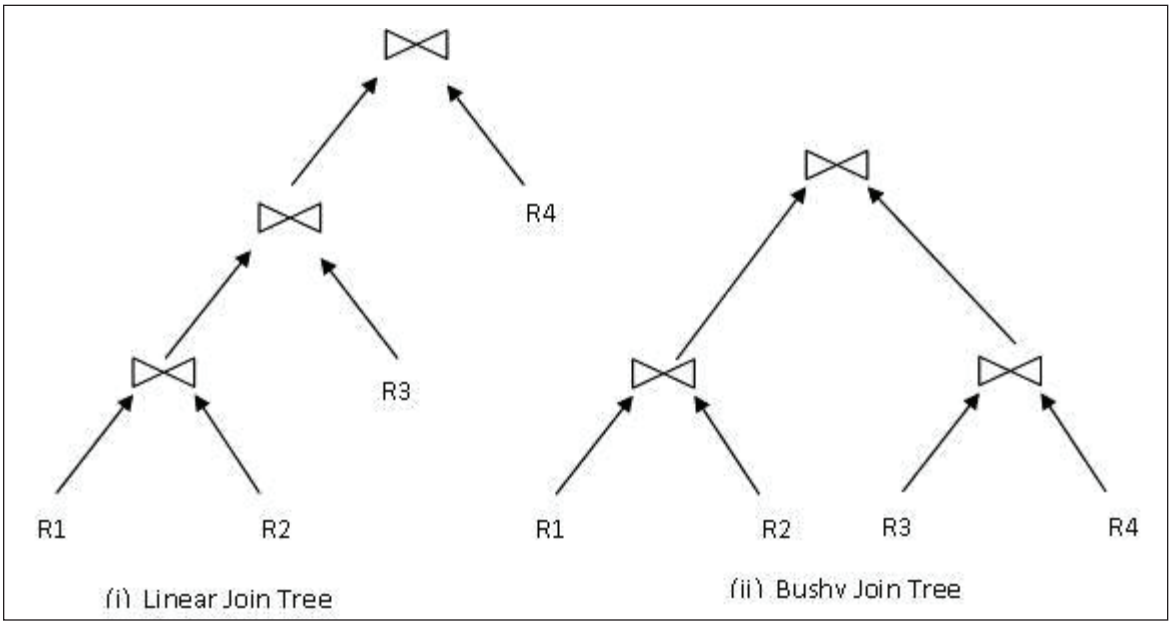

Figure 3. Linear vs. Bushy Join Tree.

3.1.2 Bushy Tree

In this type of tree the operand may or may not be the base relation, it is possibility in this scenario that both operands are intermediate operator[25]. In Figure 3 (ii) bushy tree working is shown; this approach is useful to execute the queries in parallel way or relations which are partitioned on disjoint homes[26].

3.2 Search Strategy

It can be described as how we "move" in the search space. In this query optimization phase we scan or explore the search space and from which chose the most suitable Query Execution Plan by using cost model. Deterministic, Randomized and Genetic are three most common types of algorithms for join-ordering optimization[27].

There are basically three different algorithms to solve the query optimization problem.

- Deterministic
- Randomized
- Genetic

3.2.1 Deterministic

It is work on building plan. It takes start from the base relation and keeps adding more relation at each step unless it can get the complete plan. The pruned plans are ignored in the initial stage and the low cost plans are kept to construct the complete plan. Table 2 show the comparison of Deterministic strategies.

- **Dynamic Programming**

In this approach all bad plans are ignored at the very initial stage of query optimization[28]. It is also known as pruning strategy, because the bad plans are removed as early as possible and consider theplans which provide the same result with low cost[7].

Table 2. Comparison of different strategies of Deterministic Approach

Deterministic Strategies	Plan Build Using	Query Planning	Speed	Works at	Better For
Dynamic	Breath-First	Inferior Plans are Discarded earlier	Slow	Lower Cost	Less Relations or Fragments
Greedy	Depth-First	Produce Worse Plans	Fast	High Cost	More Relations or Fragments

- **Greedy Strategy**

It makes plans by using depth-first searching technique[7]. It generates worse plans, but these plans are faster than dynamic programming[23].

3.2.2 Randomized

For complex and complicated queries Randomized strategy is used[29]. Its main focus is to search the optimal solution. It does not usually generate an optimal access plans. But is save the cost of optimization in the context of storage and time used 30. Table 3 shows the comparison of Randomized Strategies with respect to their movement strategy and other factors.

3.2.3 Genetic

Genetic strategy[30] is also not usually used to generate an optimal access plan. However in it is observed that genetic and randomized at the end provide similar results on the given parameters, but doing some experiment it is observed that in some cases genetic strategy produced better result than random[6].

3.3 Distributed Cost Model

This cost model includes the cost function to forecast the cost incurred in statistics, formulas, operators and the size of intermediate results. It can be measured in the context of time taken by the query during execution. Storage is also another factor that costs of the query execution plan can be calculated, but it is not recently considered a main factor due to low prices of storage devices. So in distributed cost model the cost functions can be described with respect to the total time, or the response time and some important parameters[25]. Time can further be calculated or translated into other units e.g. dollars.

3.3.1 Total Time

Total Time includes and calculates all the time taken during processes. It includes processing time of instruction, read/write operation of an instruction, size of message and transfer time of message from one site to another. In cost model these resources are utilized in a way that these take minimum possible time. The optimal use of resources helps to minimize the cost of every component and improve the throughput of the system.

The TOTAL TIME (TT) is calculated on the basis of following parameters[31], where

- T_{CPU} = Time taken by the CPU for processing instruction
- $T_{I/O}$ = Time taken by a disk I/O
- T_{MSG} = Fixed time of initiating and receiving a message
- T_{TR} = Time takes to transmit a data unit from one site to another. One typical assumption regarding transmission time is constant. In WAN where some sites are farther away from other it may be not true. However the only reason to take it constant is that it simplifies the query optimization.

Total Time = T_{CPU}* # of instructions + $T_{I/O}$* # of I/Os+ T_{MSG}* No. of Messages+ T_{TR}* # of bytes

The topology of the network enormously impacts the proportion between communication cost and cost of T_{CPU}+ $T_{I/O}$. In WAN, for example, the Internet, the communication time is by and large the predominant variable. In LAN, however, there is more of balance in components. Earlier studies refer to proportions of communication time to I/O time for one page to be on the request of 20:1 for WAN while it is 1:1.6 for a run of early typical era Ethernet (10Mbps) shows in Table 4. Thus most of the researchers of early Distributed DBMSs (design for WAN) did not consider the cost of local processing. Their focus is to minimize the communication cost. On the other hand Distributed DBMSs (design for LAN) consider every one of the three cost factors. Newly introduced faster networks both WAN & LAN have improved the ratios. However, communication cost is still key factors in terms of WAN.

Table 3. Comparison of different strategies of Randomized Approach

Randomized Strategies	Guarantee of Best Solution	Cost Of Optimization	Movement of Strategy	Inner Loop Called As
Iterative Improvement	NO	Low	Down-Hill	Local Optimization
Simulated Annealing	NO	Low	Up-Hill	Stage

Table 4. Comparison of Communication and CPU+I/O Cost in WAN & LAN

Type of Network	Communication Cost : CPU+I/O
Wide Area Network	20 : 1
Typical Local Area Network	1 : 1.6

3.3.2 Response Time

The time taken by the query after its initiation to completion is known as response time. In case when the response time is the main consideration point then parallel local processing and communication are also considered. This time can be calculated by using given formula.

RT = T_{CPU}* # of instructions+ $T_{I/O}$*# of I/Os + T_{MSG}* # of messages + T_{TR}*# of bytes

Thus any processing and communication that are performed on the parallel basis is ignored here.

3.3.3 Difference between Total Time and Response Time by Example

With the help of Figure 4 it is shown that two different relations stored on two different sites e.g. Site 1 and Site 2 and then result of the given query is going to be calculated at Site 3. (Only communication cost is considered here).

Some assumptions for given example:

Time Unit = T_{MSG} and T_{TR}
Data Units = x (transferring data from Site 1 to Site 3), y (transferring data from Site 2 to Site 3)
Total time for the given query can be calculated as:
$TT = 2 * T_{MSG} + T_{TR}* (X + Y)$
Response time for the same query can be calculated as:
$RT = \max \{T_{MSG} + T_{TR}* X, T_{MSG} + T_{TR}* Y\}$

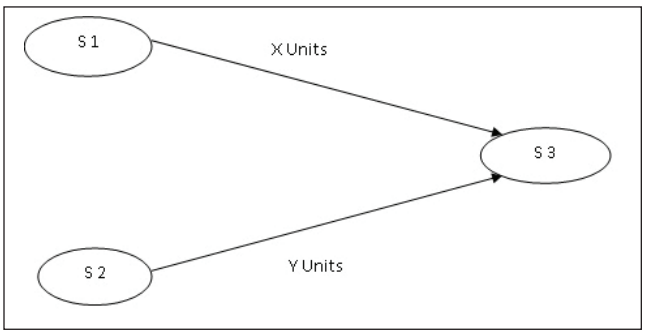

Figure 4. Example of Data Transfer of Query.

RT is reduced by increasing the number of parallel executions. It does not guarantee that due to this TT is also reduced. The minimization of TT can be obtained by the utilization of the resources properly; due to this the system throughput also increases. So it is observed that there is some tradeoff between TT and RT[6]. Beside[32] in some cases it is focused on improve the total time and in some cases, it is focused on to improve the response time.

3.4 Parameters

For query process, cost optimizer is responsible for the suitable selection of node because the different sites are used for the replication of data[10]. It is the responsibility of the cost base optimizer to choose the best processing node as per the following parameters.

- Current Queue Length: It is the amount of requests that are executing or on the way to get processed on the server.
- Server Distance: It shows the remoteness between the requesting client and on the base of their geographical position. The cost of retrieving the data diminishes as the user nigh the server.
- Server Capacity: The capacity of the server depends upon the proper functioning of the server without any hindering
- Load: It can be obtained by dividing the genuine processes executing on the server by server's capacity i.e. Load = Existing Length of queue/Server's Capacity[15].

4. Cache Based Query Optimization

In distributed database data access is faster, but search complexity is rising due to the introduction of new applications for distributed databases. So there is a need of better algorithm that speeds up the queries results as compare to traditional databases. To solve the hurdle of query optimization in distributed database system, a cache based query optimization has been introduced. In this model cache is placed between database server and local optimizer at each site. The data stored in the cache might be the already calculated values or redundant values that stored on elsewhere. If the data is found from the cache then it is recorded as cache hit that is faster than accessing data without cache. If the data is not found from the cache then it is called cache miss. Performance of the

system will drastically improveif most of the queries are entertained from the cache[33,34].

4.1 Algorithm's Working

This method totally stands on the reality that the cache must be deployed nigh the server. A user initiates the query by using the application interface. This query is broken down with the help of query optimizer in arranging the decision regarding the best node where these sub-inquires are analyzed for further handling [35, 36]. It is the responsibility of the query divisor to distribute the sub-inquiry to the suitable node. At each node the local optimizer examines that the output of the query gets from cache or not. If the output of the query is same then it is repeated query else it is new one. If information is retrieved through database the cache is upgraded as needed and outputs are shown on application or user interface. But this technique is suitable for homogeneous databases.

4.2 Comparison of Cache Models

Comparison of two papers is shown with the help of table and diagram. In which Wen-Syan's model work on a network based cache[34] whereas Mantu's Model proposes the cache at each site[15]. Usually the result of query optimization by using cache is based upon two factors e.g. number of queries and their types. But here comparison is made on just one factor i.e. number of queries. The only reason to ignore query type is that cache has very little significance when queries receive from the users are continuously repeated.

The relationship of number of queries and cache hit ratio percentage is shown in Figure 5 on the base of Table 5. Cache hit ratio is drastically increased when the number of queries are 20 and the curves are nearly balanced when the number of queries reach to 100. So it is concluded that Manu model works sharply when the number of queries are in the range of 1 to 60. But as the numbers of queries increase there is very less difference in their models.

Table 5. Comparison of No. of Queries and their % cache hit

Number of Queries	% cache hit in Wen-Syan (2003)	% cache hit in Mantu (2012)
0	0	0
20	30	60
40	50	71
60	64	72
80	75	75
100	80	76
120	81	81
140	82	83

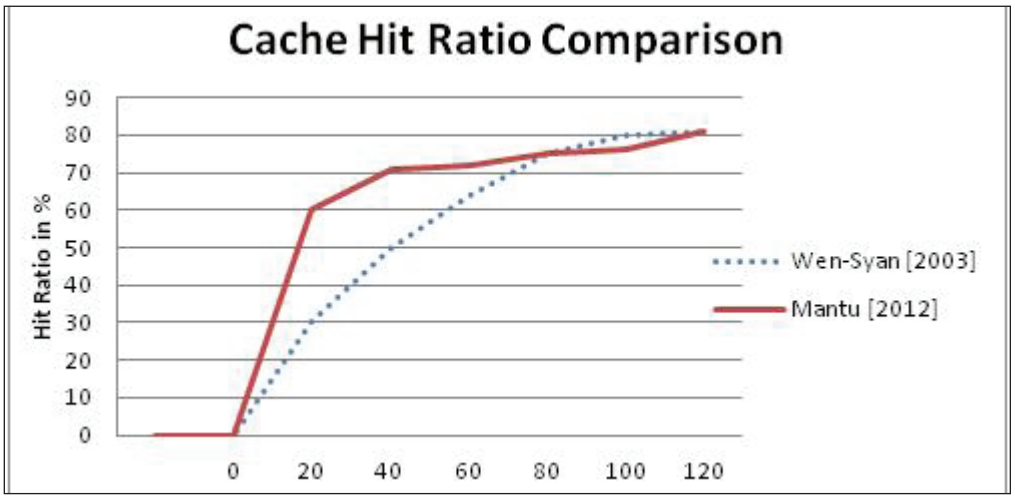

Figure 5. Number of Queries vs. Cache hit ratio %.

5. Conclusion

In this paper the authors have worked to explore one of the concerns in distributed database, Query optimization technique, one of the major issues analyzed is response time. It is explored that how the response time is improved by using query optimization. In this analysis the working of distributed databases in terms of basic components of query optimization has been explored e.g. search space, search strategy and cost model and their effects on response time. To improve the optimization efficiency of the query these components areanalyzed with the help of tables, diagrams and examples. It is also discussed that the cost factors which play a key role in query optimization. At the end it is also examined that the improvement of response time by using cache based query optimization is suitable option because it saves huge computational cost and time, especially introducing cache at each site instead of implementation of one network based cache. It is observed that currently cache is implemented in homogeneous distributed databases. Concerning future work, the proposed model can be executed for heterogeneous distributed databases. Still there is a lot of room for researchers to do in this regard our paper should sever a guide and an initiative to work forward for others also.

6. References

1. Jiang S, Ferner C, Simmonds D, Reinicke B, Clark U. Optimizing Join Query in Distributed Database. Annals of the Master of Science in Computer Science and Information Systems at UNC Wilmington. 2011; 5(1).
2. Principles for Distributed Databases in Telecom Environment. Available from: Crossref. Date accessed: 11/01/2018.
3. Wehrle P, Miquel M, Tchounikine A. A grid services-oriented architecture for efficient operation of distributed data warehouses on globus. Advanced Information Networking and Applications, AINA'07. 21st International Conference. 2007; p. 994-9 Crossref.
4. Difference between Distributed Database and Centralized Database. Available from: Crossref. Date accessed: 29/05/2011.
5. Pawandeep Kaur. Join Query Optimization in Distributed Databases. International Journal of Scientific and Research Publications. 2013; 3(5):1-3.
6. Ghaemi R, Fard AM, Tabatabaee H, Sadeghizadeh M. Evolutionary query optimization for heterogeneous distributed database systems. World Academy of Science. 2008; p.43-9.
7. Aljanaby A, Abuelrub E, Odeh M. A Survey of Distributed Query Optimization. The International Arab Journal of Information Technology. 2005; 2(1):48-57.
8. Distributed Query Optimization: Use of mobile Agents. Available from: Crossref. Date accessed: 15/11/2017.
9. Elleithy K. Innovations and advanced techniques in systems, computing sciences and software engineering. Springer Science & Business Media. 2008; p. 1-1. Crossref.
10. Menon S. Allocating fragments in distributed databases. IEEE transactions on parallel and distributed systems. 2005; 6(7):577-85. Crossref.
11. Kossmann D. The state of the art in distributed query processing. ACM Computing Surveys (CSUR). 2000; 32(4):422-69. Crossref.
12. Kossmann D, Stocker K. Iterative dynamic programming: a new class of query optimization algorithms. ACM Transactions on Database Systems (TODS). 2000; 25(1): 43-82. Crossref.
13. Pawandeep Kaur. Join Query Optimization in Distributed Databases. International Journal of Scientific and Research Publications. 2013; 3(5):1-3.
14. Kossmann D, Franklin MJ, Drasch G, Ag W. Cache investment: integrating query optimization and distributed data placement. ACM Transactions on Database Systems (TODS). 2000; 25(4):517-58. Crossref.
15. Kumar M, Batra N, Aggarwal H. Cache based query optimization approach in distributed database. International Journal of Computer Science Issues. 2012; 9(6):389-95.
16. Gupta S, Kuntal Saroha B. Fundamental research in distributed database. IJCSMS. 2011; 11(2):1-9.
17. Ozsu MT, Valduriez P. Principles of distributed database systems. Springer Science & Business Media. 2011; p. 1-866.
18. Jonker W. Databases in Telecommunications. International Workshop, Co-located with VLDB-9. Proceedings Springer; Scotland. 2006.
19. Rothnie Jr JB, Bernstein PA, Fox S, Goodman N, Hammer M, Landers TA, Reeve C, Shipman DW, Wong E. Introduction to a system for distributed databases (SDD-1). ACM Transactions on Database Systems (TODS). 1980; 5(1):1-7. Crossref.
20. Williams R, Daniels D, Haas L, Lapis G, Lindsay BG, Ng P, Obermarck R, Selinger PG, Walker A, Wilms PF, Yost RA. R*: An overview of the architecture. IBM Thomas J. Watson Research Division. 1981; p. 1-28.
21. Stocker K, Kossmann D, Braumandi R, Kemper A. Integrating semi-join-reducers into state-of-the-art query processors. Data Engineering, 2001. Proceedings of 17th International Conference. 2001; p. 575-84 Crossref.

22. Elmasri R, Navathe SB. Fundamentals of Database Systems. Reading, MA: Addison-Wesley. 2000. PMid:10877424.
23. Steinbrunn M, Moerkotte G, Kemper A. Heuristic and randomized optimization for the join ordering problem. The International Journal on Very Large Data Bases. 1997; 6(3):191-208. Crossref.
24. Ioannidis YE, Kang Y. Randomized algorithms for optimizing large join queries. ACM Sigmod Record. 1990; p. 312-21.
25. Melita L, Ganapathy G, Hailemariam S. Genetic algorithms: An inevitable solution for query optimization in web mining - A review. Computer Science & Education (ICCSE), 2012 7th International Conference. 2012; p. 89-94.
26. Kremer M, Gryz J. A survey of query optimization in parallel databases. Rapport Technique. 1999; p. 1-13.
27. Zelenay K. Query optimization. ETH Zurich, Seminar Algorithm for Data bank system. 2005. PMCid:PMC554795.
28. Kossmann D. The state of the art in distributed query processing. ACM Computing Surveys (CSUR). 2000; 32(4):422-69. Crossref.
29. Steinbrunn M, Moerkotte G, Kemper A. Heuristic and randomized optimization for the join ordering problem. The VLDB Journal - the International Journal on Very Large Data Bases. 1997; 6(3):191-208.
30. Lanzelotte RS, Valduriez P, Zait M, Ziane M. Industrial-strength parallel query optimization: issues and lessons. Information Systems. 1994; 19(4):311-30. Crossref.
31. Mackert LF, Lohman GM. R* optimizer validation and performance evaluation for distributed queries. Readings in database systems. 1988; p. 219-29.
32. Khoshafian S, Valduriez P. Sharing, persistence, and object-orientation: a database perspective. ACM. 1990; p. 221-40.
33. Batra N, Kapil AK. Three tier cache based query optimization model in distributed database. International Journal of Engineering Science and Technology. 2010; 2(7):3206-12.
34. Li WS, Po O, Hsiung WP, Candan KS, Agrawal D. Freshness-driven adaptive caching for dynamic content Web sites. Data & Knowledge Engineering. 2003; 47(2):269-96. Crossref. Crossref.
35. Kossmann D, Stocker K. Iterative dynamic programming: a new class of query optimization algorithms. ACM Transactions on Database Systems (TODS). 2000; 25(1):43-82. Crossref.
36. Lee C, Shih CS, Chen YH. Optimizing large join queries using a graph-based approach. IEEE Transactions on Knowledge and Data Engineering. 2001; 13(2):298-315. Crossref.