Novel Distributed Query Optimization Model and Hybrid Query Optimization Algorithm

Deepak Sukheja
PITM
Indore

Umesh Kumar Singh
Dept. of Computer Science,
Vikram University,Ujjain.

ABSTRACT

Query optimization is the most critical phase in query processing. Query optimization in distributed databases explicitly needed in many aspects of the optimization process, this is not only increases the cost of optimization, but also changes the trade-offs involved in the optimization process significantly .This paper describes the synthetically evolution of query optimization methods from uniprocessor relational database systems to parallel database systems. We point out a set of parameters to characterize and compare query optimization methods, mainly: (i) type of algorithm (static or dynamic), (ii) working environments (re-optimization or re-scheduling) and (iii) level of modification.

The major contributions of this paper are: (I) Understanding the mechanisms of query optimization methods with respect to the considered environments and their constraints (e.g. parallelism, distribution, heterogeneity, large scale, dynamicity of nodes). (ii) Study the problem of query optimization particular in term of heterogeneously environment and pointing out their main characteristics, which allow comparing them and help to Implement new query optimization algorithm and model. These contributions is led to performance enhancement of query optimization in distributed database system through classify by different QEPs and minimize the response time.

Keywords

Query optimization, distributed query optimization, query optimization algorithm.

1. INTRODUCTION

Practically all heterogeneous databases are used for improving the performance of relational operations for different types of applications. These applications are demanding and involve the handling of large volumes of data. A distributed database management system is a type of Heterogeneous Databases. In [1], Ozsu and Valduriez defined the principle of distributed database system. A distributed database system is a collection of multiple, logically interrelated databases distributed over a computer network. This system is defined as the software system that permits the management of the distributed database and makes the distribution transparent to the users. The distribution (including fragmentation and replication) of data across multiple site/processors is not visible to the users is known as transparency. According to Das and Gupta in [2], the distributed/parallel database technology extends the concept of data independence, which is a central notion of database management, to environments where data are distributed and replicated over a number of machines connected by a network. Data independence is provided by several forms of transparencies like network transparency, replication transparency, and fragmentation transparency. Transparent access means that users are provided with a single logical image of the database even though it may be physically distributed, enabling them to access the distributed database. In this paper, we present the design of a semantic query optimizer for a heterogeneous database management system (HDBMS). It is based on a powerful data model. The motivation for this works comes from two different needs. First is the number of non-expert users are growing to access databases and second the information systems no longer tend to be based on a single centralized architecture. They tend to be constituted of several heterogeneous component systems which cooperate to achieve global tasks.

1.1 Uniprocessor Relational Query Optimization

In the uniprocessor relational systems, the query optimization process consists of two steps: (i) logical optimization which consists in applying the classic transformation rules of the algebraic trees to reduce the manipulated data volume, and (ii) physical optimization which has roles of (a) determining an appropriate join method for each join operator by taking into account the size of the relations, the physical organization of the data, and access paths, and (b) generating the order in which the joins are performed with respect to a cost model.

1.2 Query Optimization for Centralized Databases

Early work in query optimization followed two tracks. One was minimization of expression size [3]: Expression size was measured by metrics, such as the number of joins in a query that are independent of the database state. Another track was the development of heuristics based on models that considered the cost of an operator to depend on the size of its operands as well the data structures in which the operands were stored. For example, the cost of a join was estimated using the sizes of operands as well as whether an index to access an operand was available.

While most modern database systems are designed to execute in a client-server environment, relational and object-oriented systems tend to exploit the resources of such environments in significantly different ways.

Relational systems and their descendants are typically based on query shipping, in which the majority of query processing is performed at servers. The benefits of query shipping include: the reduction of communication costs for high selectivity queries, the ability to utilize server resources when they are overflowing, and the ability to tolerate resource-poor (i.e., low cost) client machines. When a user issues a query in a centralized DBMS the optimizer must produce a detailed Query Execution Plan (QEP) that can be passed to the executer. The executer can then follow this plan to carry out the evaluation of the query and return the results to the user. An optimizer can combine selectivity information from the system catalog with an SQL query in order to produce a join graph representation of the given query.

1.3 Query Optimization for Distributed Databases

Distributed and Parallel databases are fundamentally similar, distributed query optimization process was implemented around 1990s, a time at which communication over a network was prohibitively expensive and computer equipment was not cheap enough to be thrown at parallel processing. Techniques for exploiting parallelism were largely ignored. Apers et al. [5] discuss the independent parallelism but don't define either pipelined or partitioned parallelism. Thus, for historical reasons, the concept of distributed execution differs from parallel execution. Since the space of possible executions for a query is different, the optimization problems are different. In [6], research work considered minimizing response time as an optimization objective, at other hand most project work, such as in SDD-1, A* and R* Optimizer, focused on minimizing resource consumption. Techniques for distributing data using horizontal and vertical partitioning schemes were developed for distributed data that also find a use in exploit parallelism. The main motivation of the distributed databases is to present data which are distributed on networks of type LAN (Local Area Network) or of type WAN (Wide Area Network) in an integrated way to a user.

The optimization process of a distributed query is composed of three steps: (i) the global optimization consists of determining the best execution site for each local sub-query considering data replication, (ii) finding the best inter-site operator scheduling, and (iii) placing these last ones. As for local optimization, it optimizes the local sub-queries on each site which are involved to the query evaluation. The inter-site operator scheduling and their placement are very important in a distributed environment because they allow reducing the data volumes exchanged on the network and consequently to reduce the communication costs. Hence, the estimation accuracy of the temporary relation sizes that must be transferred from a site to another one is important. In the rest of this section, we present global optimization methods of distributed queries. They differ by the objective function used by the optimization process and by the type of approach: static or dynamic.

1.4 Query Optimization for Parallel Databases

In [8, 9 & 10] several research projects such as Bubba, Gamma, DBS3 and Volcano devised techniques for placement of base tables and explored a variety of parallel execution techniques. This has yielded a well understood notion of parallel execution. Considerable research has also been done on measuring the parallelism available in different classes of shapes for join trees. Hong and Stonebraker [11] proposed the two-phase approach to parallel query optimization. They used a conventional query optimizer as the first phase. For parallelization, they considered exploiting partitioned and independent parallelism but not pipelined parallelism. Hong develops a parallelization algorithm to maximize machine utilization under restrictive assumptions. The parallel machine is assumed to consist of a single disk (RAID) and multiple processors and each operator is assumed to have CPU and IO requirements. Assuming that two operators, one CPU-bound and the other IO-bound to always be available for simultaneous execution, the algorithm computes the degree of partitioned parallelism for each operator so as to fully utilize the disk and all CPUs.

2. Query Optimization Algorithm

A query optimization algorithm can be evaluated on the basis of its operation mode or the timing of its optimization. The different query optimization algorithms are as follows:

2.1 System R algorithm:

In [12] the System R algorithm was examined as it produces optimal execution plans. However, it is not a viable solution in the large autonomous and distributed environments that we consider for two reasons. The first one is that it cannot cope with the complexity of the optimization search space. The second one is that it requires cost estimations from remote nodes. Not only this makes the notes not autonomous, but for a query with n joins, it will take n rounds of message exchanges to find the required information. In each round, the remote nodes will have to find the cost of every feasible k-way join (k=1…N), which quickly leads to a network bottleneck for even very small numbers of n. In other word this algorithm may the optimization algorithm consists of two steps

1. Predict the best access method to each individual relation (mono-relation query) Consider using index, file scan, etc.

2. For each relation R, estimate the best join ordering.

R is first accessed using its best single-relation access method.

Efficient access to inner relation is crucial

2.1.1 System R* Algorithm:

The System R* query optimization algorithm is an extension of the System R query optimization algorithm with the following main characteristics:

Only the whole relations can be distributed, i.e., fragmentation and replication is not considered.

Query compilation is a distributed task, coordinated by a master site, where the query is initiated

Master site makes all inter-site decisions, e.g., selection of the execution sites, join ordering, method of data transfer.

System R* algorithm used in distributed database system as well as in relational database system.

2.2 Dynamic Programming Algorithm:

The basic dynamic programming for query optimization as presented in [17]. It works in bottom up way by building more complex sub-plans from simpler sub-plans until the complete plan is constructed. In the first phase, the algorithm builds access plan for every table in the query. Typically, there are several different access plans for a relation (table). If relation A, for instance, is replicated at sites S1 and S2, the algorithm would enumerate table-scan (A, S1) and table-scan (A, S2) as alternative access plans for table In the second phase, the algorithm enumerates all two - way join plans using the access plans as building blocks. Again, the algorithm would enumerate alternative join plans for all relevant sites; i.e. consider carrying out joins with A at S1 and S2. Next the algorithm builds three-way join plans, using access-plans and two-way join plans as building blocks. The algorithm continues in this way until it has enumerated all n-way join plans. The strength of the dynamic programming is that inferior plans are discarded as early as possible. Dynamic programming algorithm performs well with small number of relations, but this situation is inverted when the query has more relations

2.2.1 IDP – M:

A heuristic extension of the SystemR algorithm, in [13],the Iterative Dynamic Programming IDP-M(k,m) , was propose for use in distributed environments. Given an n-way join query, it works like: First, it enumerates all feasible k-way joins, i.e., all feasible joins that contain less than or equal to k base tables and finds their costs, just like SystemR does. Then, it chooses the best m subplans out of all the subplans for these k-way joins and purges all others. Finally, it continues the optimization procedure by examining the rest $n - k$ joins in a similar to SystemR way. The IDP algorithm is not suitable for autonomous environment as it shares the problems of SystemR mentioned above.

2.3 Mariposa:

The Mariposa query optimization algorithm [14] is a two-step algorithm that considers conventional optimization factors (such as join orders) separately from distributed system factors (such as data layout and execution location).

First, it uses information that it keeps locally about various aspects of the data to construct a locally optimal plan by running a local optimizer over the query, disregarding the physical distribution of the base relations and fixing such items as join order and the application of join and restriction operators. It then uses network yellow pages information to parallelize the query operators, and a bidding protocol to select the execution sites, all in a single interaction with the remote nodes. The degree of parallelism is statistically determined by the system administrator before query execution and is independent of the available distributed resources.

2.4 A* algorithm:

One major heuristic algorithm proposed for query optimization is A star (A*) algorithm. This algorithm is useful for queries with few relations [15]. It normally gets stuck with some local minima if the numbers of relations are substantially increased, producing an output sub standard to the exhaustive search. Heuristic algorithms have helped in reducing the time of optimization process at the cost of quality of output. The A* algorithm can be explained as follows. Each state in the query optimization can be considered to be a node in the strategy tree. Each node contains, in addition to a description of the problem state it represents, an indication of the cost it takes to reach from its parent to the node. It is very helpful to implement distributed query optimization process.

3. PROBLEM STATEMENTS

According to different query optimization algorithm, query processing is an important concern in the field of distributed databases. The main problem is query optimization in distributed database are: if a query can be decomposed into subqueries that require operations at geographically separated databases, determine the sequence and the sites for performing this set of operations such that the operating cost (communication cost and processing cost) for processing this query is minimized.

The problem is complicated by the fact that query processing not only depends on the operations of the query, but also on the parameter values associated with the query. Distributed query processing is an important factor in the overall performance of a distributed database system. Query optimization is a difficult task in a distributed client/server environment as data location becomes a major factor.

4. PROPOSED QUERY OPTIMIZATION MODEL AND HYBRID ALGORITHM

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.

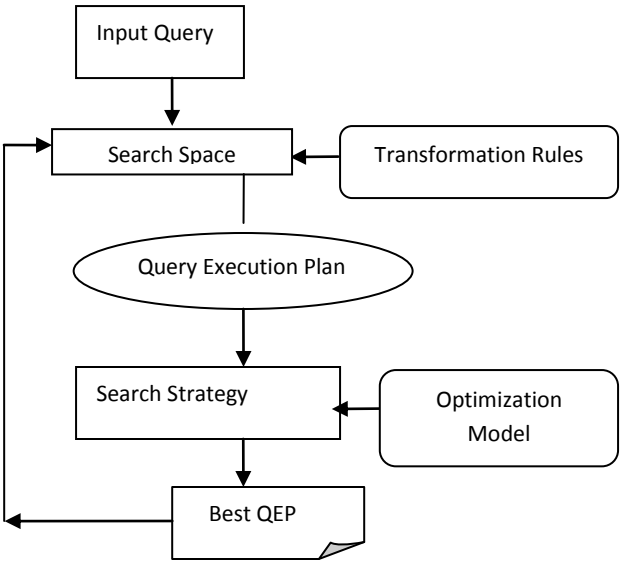


Figure-1.1 (query optimization process).

4.1 Novel query Optimization model

The Query optimization process shown in figure-1.1 consists of getting a query on n relations and generating the best Query Execution Plan (QEP) and the Layers of distributed query processing and optimization process shown in proposed optimization model figure-1.2.

Query optimization refers to the process by which the "best" execution strategy for a given query is found from among a set of alternatives.

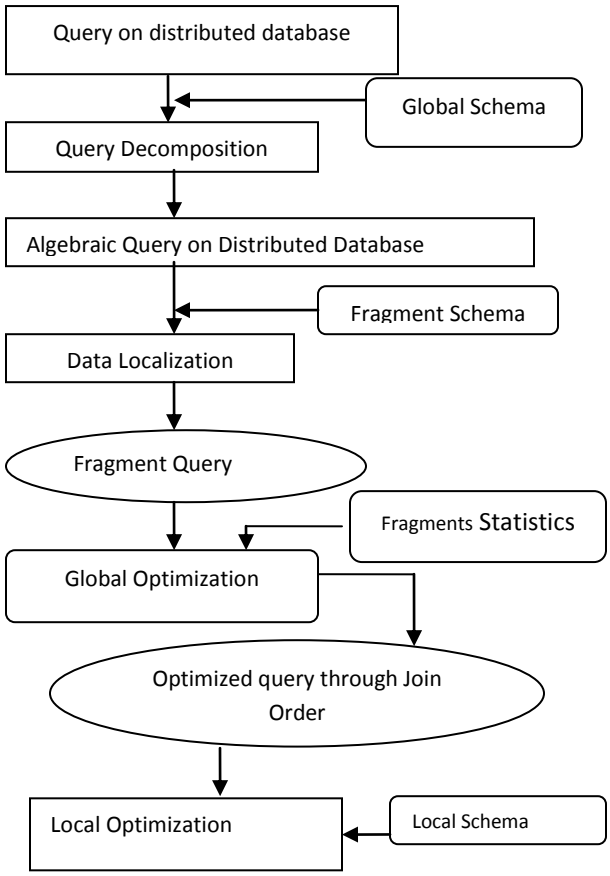


Figure-1.2 distributed query optimization model.

4.2 Proposed hybrid Query Optimization Algorithm:

To design a new hybrid query optimization algorithm, we attempt to resolve the weaknesses of distributed query optimization which is mentioned in previous section. This query optimization algorithm attempts to find the best execution plan for a join query which accesses data on two remote sites by considering the impact of data size, transmission speed, and server process speed. This algorithm calculates the response time for the possible execution plans in both sequential and parallel way. Finally, this algorithm executes the plan which has the minimum estimated response time. The algorithm is implemented in 4 stages, these stages are:

Generation of logical plans

Generation of physical plans

Conversion and distribution.

Calculate response time

Choose best execution plan.

Execute plan

4.2.1 Generation of logical plans:

The given query tree is initially represented in the directed acyclic graphs formulation. For example query trees of A×B×C initial represented in figure 1.3. The equivalence nodes are shown as boxes, while the operation nodes are shown as circles.

After initialization of given query applying all possible transformations on every node of the initial LQDAG (logical Query directed acyclic graphs) to represents all logical plans.

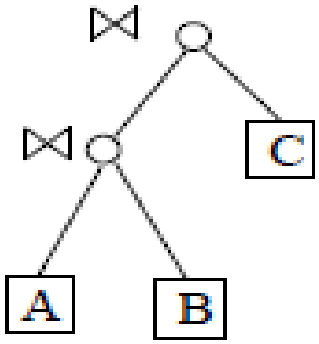


Figure1.3: Initial query

Then the plans (A×(B×C)) and ((A×C)×B) as well as several plans equivalent to these modulo commutatively can be obtained shown in figure 1.4.

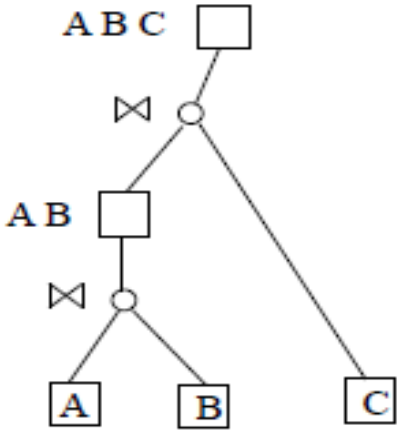


Figure 1.4: Plan 2

Expanded logical plan is shown in figure 1.5, which include all logical plans of given query. In [16], define the procedure for generation of expanded logical Plan.

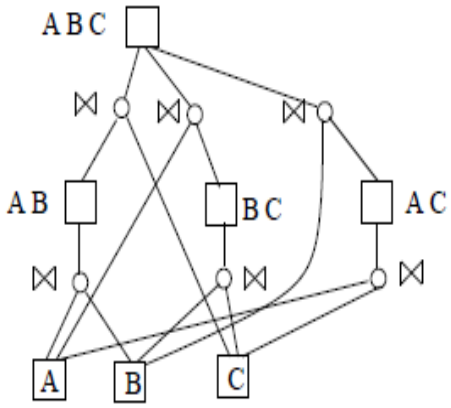


Figure 1.5 Expanded logical plan

4.2..2 Generation of Physical Plans:

The plans represented in the Logical Query DAG are only at an abstract, semantic level and, in a sense, provide "templates" that assurance of semantic accuracy for the physical plans. The logical plan does not consider the physical properties of the results, like sort order on some attribute, into account since results with different physical properties are logically equivalent.

The generation of physical query plan from logical query directed acyclic graphs is smoothly defined prasan roy in his research work [16]. Before generation of physical plan search algorithm has been applied to find optimal logical query execution plan those are generated in section 3.1.

4.2.3 Conversion and Distribution:

The first stage is to convert what the user asks for from his data model to a query that accounts for where the data is located in the entire database and then try to find subparts of the query that can be processed at a single site or minimal data transfer that could achieve greater local processing.

4.2.4 Calculate Response Time:

Response time is calculated through "Elapsed time between the initiation and the completion of a query".

Response time = (CPU time + I/O time + Communication time)

CPU time = (unit instruction time ∗ no. of sequential instructions).

I/O time = (unit I/O time ∗

no. of sequential I/Os).

Communication time = (unit msg initiation time ∗ no. of sequential msg + unit transmission time ∗ no. of sequential bytes).

4.2.5 Chose Best Execution Plan:

The algorithm compares the estimated response times of the three execution plans, chooses the one with the minimum estimated execution time, and then executes the plan. The goal of query optimization is to find an execution strategy for the query that is close optimal. An execution strategy for a distributed query can be described with relational algebra operations and communication primitives (send/ receive operations) for transferring data between sites. The query optimizer that follows this approach is seen as three components: A search space, a search strategy and a cost model. The search space is the set of alternative execution to represent the input query. These strategies are equivalent, in the sense that they yield the same result but they differ on the execution order of operations and the way these operations are implemented. The search strategy explores the search space and selects the best plan. It defines which plans are examined and in which order. The cost model predicts the cost of a given execution plan which may consist of the following components.

Secondary storage cost: This is the cost of searching for reading and writing data blocks on secondary storage.

Memory storage cost: This is the cost pertaining to the number of memory buffers needed during query execution.

Computation cost: This the cost of performing in memory operations on the data buffers during query optimization.

Communication cost: This is the cost of shipping the query and its results from the database site to the site or terminal where the query originated.

4.2.6 Execution Plan:

Execute plan with minimal cost and generate the result of input query.

4.3 New Hybrid Algorithm

Step 1: Do initial processing

Step 2: Select initial feasible solution P1

2.1 Determine the candidate result sites, sites where a relation referenced in the query exist

2.2 Compute the cost of transferring all the other referenced relations to each candidate site

2.3 P1 = candidate site with minimum cost

Step 3: Determine candidate plans of P1 into {P2,..Pn}

Step 4: Replace P1 with the split schedule which gives *cost*(p2) + *cost*(local join) + *cost*(P3) +cost (local join+….+ *cost*(Pn) < cost(P1).

Step 5: Recursively apply steps 3 and 4 on Next Query Plan until no such plans can be

Found

Step 6: Check for redundant transmissions in the final plan and eliminate them.

Step 7: Execute Plan.

The mentioned hybrid algorithm attempts to find the best execution plan for a join query which accesses data on two remote sites by considering the impact of data size, transmission speed, and server process speed. This algorithm calculates the response time for the possible execution plans in both sequential and parallel way. Finally, this algorithm executes the plan which has the minimum estimated response time.

5. IMPLEMENTATION AND ANALYSIS OF PROPOSED ALGORITHM

Distributed database system, provides data distribution transparency by hiding the data distribution details from the users. Whenever a distributed query is generated at any site of a distributed system, it follows a sequence of phases namely query decomposition, query fragmentation, global query optimization and local query optimization. The allocation of data considers a set of fragments, a set of locations in a network, and a set of applications placed at Location. These applications need to access the fragments which should be allocated in the locations of a network. The allocation problem consists on finding an optimal distribution of fragment over location. Thus, distributed cost model includes cost functions to predict the cost of operators, database statistics, base data, and formulas to calculate the sizes of intermediate results. In a distributed system, the cost of processing a query is expressed in terms of the total cost measure or the response time measures. The total cost measure is the sum of all cost components. If no relation is fragmented in the distributed system and the given query includes selection and projection operations, then the total cost measure involves the local processing cost only. However, when join and semi join operations are

executed, communication costs between different sites may be incurred in addition to the local processing cost.

Total Cost Measured

$T_{cost} = T_{cpu} * Insts + T_{i/o} * C_0 + C_1 * x$

where, T_{cpu} is the CPU processing cost per instructions, insts represents the total number of CPU instructions, + $T_{i/o}$ is the I/O processing cost per I/O operation, C_0 is the start-up cost of initiating transmission, C_1 is a proportionality constant, and X is the amount of data to be transmitted.

To verify the feasibility of "hybrid query optimization algorithm" we developed simulation models in a dynamic environment, experiments were conducted in a distributed database environment using the functionality of multithreading, networking and JDBC concept. To develop the simulation model, use following software and hardware specifications.

5.1 Software Description and Specifications:

The detail software descriptions shown in table 1

Software	**Description**	**Detail**
Operating System	WINDOWS SERVER 2003	Enterprise Edition (32-bit x86)
Database	MySql	*Bytes* SGA : 1, 174, 405, 120
Java	Version 1.2	Java 2 Platform, Enterprise Edition

Tbale 1: Software description

5.2 Hardware Description and its specifications:

We have carried out extensive experiments to evaluate the effectiveness of "hybrid query optimization algorithm", use 3 terminals which are interconnected through LAN and the configuration of each terminal is 2.3 GHz AMD Athlon™XP 2600 PC with 2Gb of RAM and SATA disk.

Before start the execution of experiment, it will make the following assumptions: Accurate Statistics**:** We assume that statistics regarding the cardinalities and the selectivity are available. This information can be collected through standard protocols that allow querying the host database about statistics, or by caching statistics from before query executions. Communication Costs: We assume that communication costs remain roughly constant for the duration of optimization and execution of the query, and that the optimizer can estimate the communication costs incurred in data transfer between any two sites involved in the query. No Pipelining across all terminals: We assume that there is no pipelining of data among query operators across the terminals.

5.3 Experimental Setup:

In this experiment we use MySql database and we create the banking relation schema which holds the following relational tables with size and its location (terminal) as mentioned in table 2. Although these relational schema are not too large, the experimental results clearly demonstrate the relative merits of using restructured according to "new optimization model" and "hybrid query optimization algorithm".

Branch: (branch_name,branch_city,assets)

Customer:(customer_name,customer_street, customer_city)

loan: (loan_number,branch_name,amount)

borrower: (customer_name,loan_number)

account : (account_number, branch_name, balance)

depositor:(customer_name, account_number)

Table size is calculated through following commands;

SELECT DATA_LENGTH FROM TABLES

WHERE TABLE_SCHEMA = 'banking'

AND TABLE_NAME = 'customer'

Relation	Terminals	Table Size Bytes
customer	Termianl1	2058451
Branch	Terminal 2	301653
Loan	Terminal 3	236554
Borrower	Terminal 1	1166529
Account	Terminal 3	2245256
Deposit	Terminal 1	325182

Table 2: Schema description

In this simulation model we are executing some different plane of single query through different thread. All threads having same priority and execute simultaneously. Same simulation model run on all connected terminals. The simulator is implemented in Java because it's directly support to multithreading, networking and JDBC.

5.4 Experimental Results:

This section details the results that were obtained in the execution of the more than 5 queries against the database to retrieve the data from the different terminal. Theses queries are combination of single table query (simple query), multi table query (query with join operation, complex query) where tables are not fragmented and multi table query (query with join operation, complex query) where tables are fragmented on different terminal, because of the different type of queries, we can perform a more thorough analysis and gain some insights.

In this experiment we consider replication fragmentation. Execute all queries and compared the execution time of each plan in recommended mode.

Execution time of each plan of query, respectively its terminal (performance metrix) is shown in table 4. For example consider the query Q1:

Q1: select customer_name, borrower.loan_ number, amount from borrower,loan where

borrower.loan_number = loan.loan_number and

branch_name = 'Annpurna' and amount > 2000000;

Both relations are stored on a different terminal and generate different access plan according to mentioned process in section 5 and generate the different logical and physical execution plan of query and then apply hybrid query optimization algorithm in conversion and distribution phase of distributed query processing and optimization process to reduce the total cost and response time of database query. The different execution plan and execution time as shown in table 3

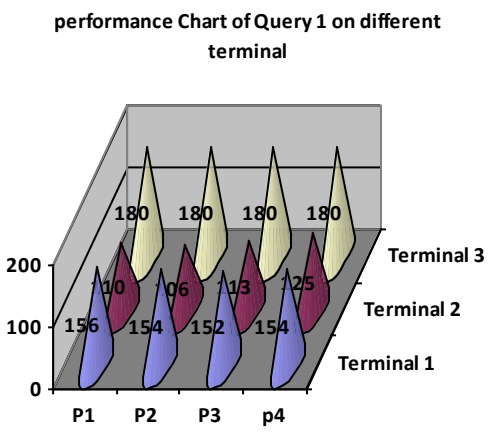
Alternative Plan	Execution strategy	Location	Execution time in m. sec.
Plan 1	Exec (borrower, loan) at terminal 1	Terminal 1	156
Plan 2	Exec (borrower, loan) at terminal 3	Terminal 3	154
Plan 3	Exec ((sub query (borrower)) at terminal 1, (sub query (borrower)) at terminal 3,Exec(Join operation query at terminal 1))	Terminal 1,3	152
Plan 4	Exec ((sub query (borrower)) at terminal 1, (sub query (borrower)) at terminal 3,Exec(Join operation query at terminal 1))	Terminal 1,3	154

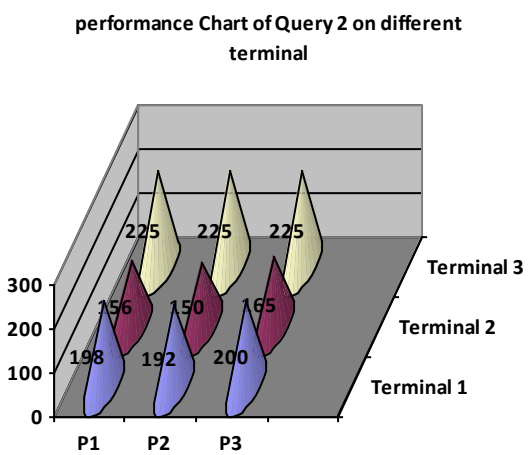
Table 3: Query execution plan table

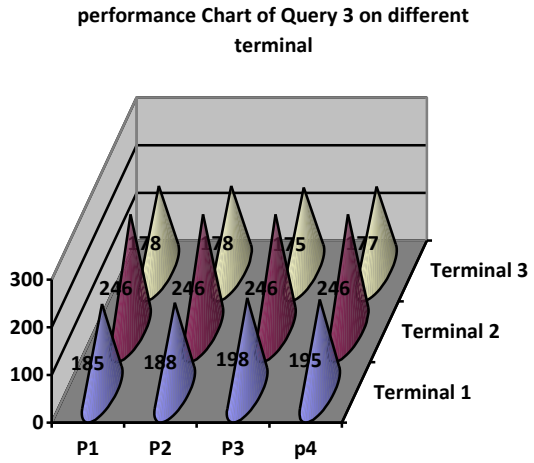
Same process applied on Q2, Q3, Q4 and Q5 to get all possible plans of all respective queries and execute in same manner. Execution time of different execution plan of every query with respective its terminal (site) is shown in table 4.

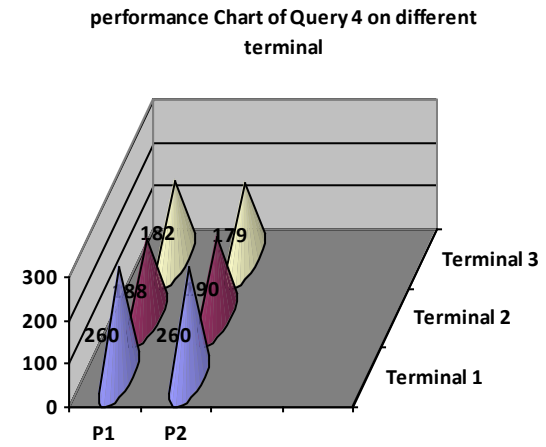
Queries / Terminals	Q1	Q2	Q3	Q4	Q5
Terminal 1	Plan / Time M.Sec P1 156 P2 154 P3 152 P4 154	Plan / Time M.Sec P1 198 P2 192 P3 200	Plan / Time M.Sec P1 185 P2 188 P3 198 P4 195	Plan / Time M.Sec P1 260 P2 260	Plan / Time M.Sec P1 192 P2 188 P3 188 P4 193
Terminal 2	Plan / Time M.Sec P1 110 P2 106 P3 113 P4 125	Plan / Time M.Sec P1 156 P2 150 P3 165	Plan / Time M.Sec P1 246 P2 246 P3 246 P4 246	Plan / Time M.Sec P1 188 P2 190	Plan / Time M.Sec P1 203 P2 203 P3 203 P4 203
Terminal 3	Plan / Time M.Sec P1 180 P2 180 P3 180 P4 180	Plan / Time M.Sec P1 225 P2 225 P3 225	Plan / Time M.Sec P1 178 P2 178 P3 175 P4 177	Plan / Time M.Sec P1 182 P2 179	Plan / Time M.Sec P1 188 P2 185 P3 187 P4 185

Table 4: Execution Detail









This performance is calculated on replication fragmentation distributed scheme. In this experiment database install on are terminals and plans are executed through threads. Time shown in mentioned graph is Thread execution time. It means time will more reduce when the queries will directly apply on actual distributed database. Table 5 showing the best plan of each query with respect to the terminal.

Query / *Terminal*	Q1	Q2	Q3	Q4	Q5
Terminal 1	P3	P2	Not used	Not used	P2/P3
Terminal 2	P2	P2	Not used	P1	Not used
	Not used	Not used	P 3	P2	P2/ P4

Table 5: Best execution plan

6. CONCLUSION

In this paper, we study the query optimization process in different database environment and also discuss the different query optimization algorithms. After the study we have defineed query optimization model and implement a new hybrid query optimization algorithm. This implemented algorithm is simulated through java program. The result of this algorithm is provied the query execution plan with respect to the terminal and also defined query execution strategy to best use of system resource by minimize network traffic.

7. REFERENCES

[1] M.T. Ozsu and P. Valduriez. "Distributed Database Systems: Where Are We Now?", IEEE Computer, 24(8): 68 - 78, August 1991.

[2] Sanjib Kumar Das, Sayantan Das gupta,"Middle Layer Java Software for the Implementation of a Homogeneous Distributed Database System",DBMS LAB – 2007.

[3] A.V. Aho, Y. Sagiv, and J.D. Ullman. efficient Optimization of a Class of Relational Expressions. Transactions on Database Systems, 4(4):435–454, 1979.

[4] P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price. Access Path Selection in a Relational Database Management System, In Proceedings of ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 1979.

[5] P.M.G. Apers, A.R. Hevner, and S.B. Yao. Optimization Algorithms for Distributed Queries. IEEE Transaction on Software Engineering, 9(1), PP 57-68 1983.

[6] L. F.Mackert andG.M.Lohman. R*Optimizer Validation and Performance Evaluation for Local Queries. Technical report, IBM Research Division, January 1986. IBM Research Report RJ 4989.

[7] M.T.Ozsu and P.Valduriez. Principles of Distributed Database Systems, 2nd edition englewood cliffs, NJ: PrenticeHall, 1991.

[8] H. Lu, M-C. Shan, K-L Tan, "Optimization of Multi-Way Join Queries for Parallel Execution" Proceedings of the 17th International Conference on Very Large Databases, Barcelona, PP: 550-560. September 1991

[9] K-L Tan, H. Lu, "On Resource Scheduling of Multi-Join Queries in Parallel Database Systems", Information Processing Letters 48,1993.

[10] G. Hallmark, "Oracle Parallel Warehouse Server", IEEE 1997.

[11] W.Hong andM. Stonebraker. Optimization of ParallelQuery Execution Plans inXPRS. In Proceedings of the First International Conference on Parallel and Distributed Information Systems, December 1991.

[12] Donald D. Chamberlin, Morton M. Astrahan, Mike W. Blasgen, Jim Gray, W. Frank King III, Bruce G. Lindsay, Raymond A. Lorie, James W. Mehl, Thomas G. Price, Gianfranco R. Putzolu, Patricia G. Selinger, Mario Schkolnick, Donald R. Slutz, Irving L. Traiger, Bradford W. Wade, Robert A. Yost: A History and Evaluation of System R. Commun. ACM 24(10): 632-646 (1981).

[13] STONEBRAKER, M., AOKI, P. M., DEVINE, R., LITWIN, W., AND OLSON, M. A. 1994. Mariposa: A new architecture for distributed data. In *Proceedings of the 10th International Conference on Data Engineering* (Houston, TX, Feb. 14–18). IEEE Computer Society Press, Los Alamitos, CA, 54–65.

[14] STONEBRAKER, M., AOKI, P. M., LITWIN, W., PFELLER, A., SAH, A., SIDELL, J., STAELIN, C., AND YU, A Mariposa, "A wide-area distributed database system", *VLDB journal ,vol.* 1, 48–63, 1996.

[15] Michael Steinbrun, "Heuristic and randomized optimization for the join ordering problem", The VLDB Journal vol: 6, PP: 191–208, 1997 .

[16] Prasan Roy, Multi-Query Optimization and Applications, Ph.D. Thesis, Dept. of Computer Science and Engineering, IIT-Bombay, December 2000.

[17] Kossmann D. "The State of Art in Distributed Query Optimization", ACM Computing Surveys, September 2000