Introduction & Objectives

1.1 Introduction

Distributed Database System is the synergic combination of two different technologies used for data processing: Database Systems and Computer Networks. The main component of a Database System is the data and is defined in literature as the collection of facts about something. This ‘something’ may be the business data in case of a business corporation, strategic data in case of a military’s database and experimental data in a scientific experiment etc. Data Base is the systematic collection of these data files. Data that constitutes the database has to be correlated and stored on different sites of a computer network to be a part of distributed database. Distributed Database is a very complex and costly technology, so mostly employed by big businesses, or governmental organizations [Ozsu et al. 2006].

Deciding a distributed database design and query processing strategies are the most challenging part of distributed query technology. Some of the sub tasks involved in these are Data Fragmentation, Data Allocation, Operation Allocation, Operation Ordering, Load Balancing etc. It is a proven fact that finding an optimal execution strategy for a query is computationally intractable [Ibarki and Kameda, 1984]. The research in this area dates back to more than three decades, but due to NP-Hard nature of the problem the hunt for better solutions is still prevalent [March, 83], [Navathe et al.84], [Cornell & Yu,90], [Ghaemi et al.2008], [Li & Luo, 2008], [Karimi & Rankoohi,2009], [Kumar et al.2011]. Most of the research work involves looking for better mathematical programming models and finding better heuristics. All this is done in order to minimize the combined cost of storing the database, processing transactions against it and minimizing communication amongst network sites.
Query is defined as a request for an information retrieval, which may require a scan of single or a group of tables (Database), further these databases might be stored at different network sites. Distributed Query Processing deals with designing the data distribution and query optimizing algorithms with a view to either minimize the total cost of a query or minimize the response time of the query [Ozsu et al. 2006].

Figure 1.1: A Distributed Database

Query optimization’s main two goals are: First minimize globally, the data communication amongst various network sites and secondly, minimize the disk accesses locally at a particular site. Most of the existing solutions are based on the use of heuristics on enumerative techniques. They make use of advanced algorithm design techniques like Dynamic Programming, Backtracking, Branch and Bound, Simulated Annealing etc [Bing&Jiang(2000)]. The major problem with all deterministic solutions is that when the problem size scales up to large number of sites and query joins, these tend to go computationally intractable. Evolutionary techniques for optimization of distributed database queries has been successfully implemented in last two decades, by many in database research community: [March, 1983][Navathe et al., 1984], [Cornell & Yu, 1990], [Drenick & Smith, 1993], [March & Rho, 1995], [Kwok et al., 1996], [Li & Jiang, 2000], [Ahmad et al., 2002], [Cui & Lin, 2004], [Owais et al. 2005], [Barker et al., 2006], [Zehai
Zou, 2007], [Ghaemi et al., 2008], [Li & Luo, 2008], [Karimi & Rankoohi, 2009], [Kumar et al., 2011].

In this thesis, a randomized solution based on Genetic Algorithms is proposed. It is implemented using a stochastic simulator & aims to optimize the Access Strategies for a Distributed Database Query. From a global viewpoint, *Access Strategies* refer to the procedures applied on sub query allocation scheme to minimize the *Communication Costs* by minimizing the movement of data across the network sites. From a local viewpoint, it refers to *Access Path Selection* by minimizing the movement of data from secondary memory to main memory while doing local processing operations.

An abstract representation of the problem formulation is as given below

**Part I**-

Given a Relation ‘R’ and a query ‘Q’ in a simulated distributed database environment, GA (Genetic Algorithm) based stochastic simulator for a distributed database query optimizer, fragments ‘R’ into a set of vertical fragments so as to reduce disk access costs, which consist of local CPU (Central Processing Unit) and I/O (Input / Output) costs.

Proposed Genetic Algorithm for fragmentation and Access Strategies, (GA_FA)\(^1\) has an objective to produce a set of fragments \(F = \{f_1, f_2, f_i\}\), such that local access costs are minimized.

**Part II**-

Fragments are to be placed at sites of a network \(S = \{s_1, s_2, \ldots, s_m\}\) and

Query ‘Q’ is decomposed into a set of subqueries \(Q = \{q_1, q_2, \ldots, q_q\}\);

Objective Function of another proposed genetic algorithm, Subquery Operation Allocation (GA_SA)\(^2\) is to find a sub-query allocation plan for sites given by set \(S\), in such a way so as to minimize the cost of communication amongst various sites.

A combined objective of both the solutions is to minimize the query processing costs.

\[
QPC_i = LPC_i + TC_i
\]

\(^1\) Genetic Algorithm for Fragmentation and Access Strategies

\(^2\) Genetic Algorithm for SubQuery Allocation
Where \( QPC_i \): Query Processing Costs, \\
\( LPC_i \): Local Processing Cost \\
\( TC_i \): Communication Cost.

1.2 Objectives of Distributed Query Processing

Distributed Query Processors map a high-level query (Relational Calculus/SQL) on a distributed database into a sequence of database operations (Relational Algebra) on relation fragments. While decomposing a calculus query into an algebraic query, data accessed by the query has to be localized so as to translate the operations to bear on local data fragments. Finally the algebraic query on fragments must be reorganised to minimize the use of computing resources such as I/Os, CPUs, and communication network [Ozsu et al. 2006].

There are two popular measures of resource consumption, the Total Cost and Response Time of the query [Saco & Yao, 1982]. Total cost is the sum of all times incurred in processing the various operations of the query and inter-site communication. Response time is the time elapsed from origin to completion of the query [Ozsu et al. 2006]. In this thesis the approach adopted for query optimization is Total Cost Minimization for OLTP(Online Transaction Processing) queries.

Queries are categorized mainly in two groups, OLTP and DSS (Decision Support System) queries. The former are generally repetitive in nature and hence optimizer should seek good throughput by concentrating on reducing the total cost of the query. DSS queries more often seek response time optimization and use of parallel processing [Virk & Singh, 2011].

1.3 Types of Query Optimization

Query Optimization can be broadly categorised into two, Static and Dynamic Optimization. Static Optimization is done at query compilation time, whereas Dynamic is done at execution time. A Distributed Query is usually executed as sum of many sub operations and the size of intermediate relations is unknown till run time. Therefore static methods have to estimate them based on database statistics and various estimation theories & techniques. A few of the famous works on size estimation are parametric
Dynamic methods have accurate knowledge of previous operations & intermediate result sizes at run time, and don’t rely much on database statistics and estimation procedures. These have a disadvantage in the form of being expensive as they have to be repeated for each execution of the query [Ozsu et al. (2006)].

In optimizing a query one has to choose from a solution space of all possible execution plans. A simple strategy can be formulated as to first exhaustively scan the entire solution space, predict the cost of each plan and choose the minimum cost plan. This can be a foolproof method to find the best plan but for the size of solution space. These exhaustive strategies may work fine till a query involves six to seven relation and sites, when number of relations is increased from seven, and numbers of joins or sites are increased, the size of solution space becomes enormous and exhaustive procedures go intractable. Heuristics are used to prune the solution space, by exploiting the equivalence relations for relational algebra operations. A few of popular heuristics are: minimizing the size of intermediate relations, performing selections and projections at an earliest possible stage, avoiding computationally costly operations like Cartesian products, and optimal ordering of operations by the increasing sizes of their intermediate relations.

Other techniques like, Randomized Strategies: Iterative Improvement, Simulated Annealing, Genetic Algorithms etc, are used for data and operation allocation for big size query problems [March,1983], [Ioannidis and Wong,1987],[March&Rho1995], [Bing & Jiang, 2000].

1.4 Layers of Distributed Query Processing

The problem of distributed query processing has been described in database literature by a layered model[Ozsu et al.2006]. It decomposes it into various layers as shown next in a diagram Proposed algorithms in this thesis GA_FA and GA_SA are shown to be working on different layers of this model

1.4.1 Query Decomposition

This layer decomposes the distributed calculus query into an algebraic query on global relations. The information needed for this transformation is found in the global
conceptual schema describing the global relations. But the information about data
distribution is not used here. Query decomposition is the first phase of query processing
that transforms a relational calculus query into a relational algebra query. The successive
steps of query decomposition are given below:

1. Normalization: makes query representation suitable for later steps.


4. Restructring: Algebraic equivalence applied to avoid worse executions.

Figure 1.2: Generic Layers of Query Processing
1.4.2 Data Localization

This layer determines which fragments are involved in the query and transforms the distributed query into a fragment query. In this layer, Fragmentation schema is applied to transform or materialize the global relation query into a query which acts on fragments of those global relations.

1.4.3 Global Query Optimization

A query working on fragments rather than global relations is input to this layer. All the previous layers of optimization don’t take into account the actual placements of fragments and ordering operations at various sites of network, minimizing communication cost differences between various sites and this layer takes care of all these optimization goals [Abolfazel et al.2009].

An Evolutionary Genetic Algorithm solution for subquery allocation to various networks sites (GA_SA), proposed in this thesis, operates at this layer and dynamically takes care of some of the global layer’s goals and procedures.

1.4.4 Local Optimization

Finally all the sites chosen dynamically at penultimate layer have to perform subqueries on local fragments at their respective sites. These may further be optimized using local schema and classic central database techniques.

Query optimization process broadly involves three main steps

First step involves producing and evaluating all alternative access plans that produce the same result to the query. Set of all these possible access plans is called a State Space, which is usually represented by a Processing Tree.

Second step is to associate with Each Access Plan a query Cost Function, which predicts the cost of this query execution plan.

Third step is exploring the vast search space generated in first step and comparing cost of each plan of search space to find the best plan (One with the minimum cost function value).
Proposed genetic algorithm for Fragmentation and Access Strategies (GA_FA) partly works at this layer by choosing an efficient combination of fragments and access strategies.

1.5 Motivation & Key Work Objectives

Two critical components of any relational distributed query optimizer are first to fragment the distributed database tables, to reduce disk access costs and then secondly determine the subquery allocation plan over various sites of a wide area computer network, to reduce the overall communication costs. In this thesis a Genetic Algorithm solution has been proposed to optimize above said two components of the query optimization process. It first optimizes the Access Strategies for a set of local disk operations and secondly optimizes the Sub Query Operation Allocation Strategies for a global operation.

- Proposed Genetic Solution (GA_FA)$^3$ aims to find a near optimal fragmentation solution that minimizes the disk accesses. Access Strategies included are Sequential, Clustered and Unclustered Index Scan for local disk accession at various sites. A cost model is proposed and effect of fragmentation schemes by varying access strategies is studied through this genetic solution.

- Proposed Genetic Solution (GA_SA)$^4$ aims to find Sub Query Operation Allocation plan for various sites. It involves breaking a complex distributed query into a set of smaller sub-queries, further objective is to find a sequence of network sites, on which these subqueries will be processed. It has to find a sequence out of a very large number of different permutations of network sites. For example, all possible enumerative plans for ‘10’ number of joins on ‘10’ number of sites produce a search space of order of $O(10^{10})$, i.e. 10 billion query trees. This exponential growth is what makes it a NP Hard problem. Experiments were conducted to verify whether randomized solutions are a better option for bigger size problems.

- A mathematical model is designed for an evolutionary Genetic Algorithm, for optimizing sub query allocation (GA_SA). It simulates the allocation of various Sub Queries (operations) to different nodes of a given network, so that total communication cost of the query is minimized.

---

$^3$ Genetic Solution for Fragmentation and Access Strategies
$^4$ Genetic Solution for Sub-Query Allocation to Various Sites
An experimental distributed database test environment is designed and simulated. Further GA_FA and GA_SA are implemented using a set of Wisconsin Benchmark Distributed Database Queries. Experiments are performed on various sets of test data and queries to compare it with other popular algorithms. Analytical analysis is performed to validate the results according to the proposed cost model.

Effect of varying various Genetic Operators and parameters, is to be observed and analyzed for GA_SA.

1.6 Methodology

GA’s (Genetic Algorithms) are designed to mimic the working and robustness of natural biological systems. The main focus of designers of artificial systems is to somehow mimic the great robustness, efficiency and flexibility exhibited by natural biological systems, like features of self repair, self guidance, instinct to select a good mate and reproduce a fit child. These are naturally there for biological systems (By God’s Grace!), whereas they barely exist even in the most sophisticated artificial (man made) systems [Goldberg (1999)]. In thesis an effort has been made to design and implement two GA’s for a part of Distributed Database Query Optimization process and highlight their superiority over deterministic strategies in case of large scale query systems.

Distributed Databases Design Process is a very complex process due to the huge size of the problem and interdependency of many complex sub processes. Research Objective is to propose and implement Genetic Algorithm Based randomized solutions for reducing the total cost of distributed database queries and further compare it with other prevalent solutions. It is achieved by minimizing the resource consumption from distributed database system. System resources mainly comprise of CPU Time, System Memory, I/O Devices, and Communication Channels etc. A transaction based approach on multiple relations is followed to design cost models and solutions for Fragmentation, Access Strategies, and Sub Query Allocation & Optimization.

One of the major design objective is to reduce the movement of irrelevant data across the sites. Moreover data that is accessed together frequently, should be placed together or nearby. Emphasis is put on reducing the Total Time or the Total Cost of the OLTP (Online Transaction Processing) Queries, by improving the overall utilisation of system resources. OLTP queries are chosen for test environment as they are repetitive in
nature and query systems focus more on reducing their total query cost rather than optimizing the response time. This research work can be broadly divided into following two parts.

Part 1 : GA_FA (Genetic Solution for Fragmentation and Access Strategies):
- The process of Binary Vertical fragmentation splits the relation into two fragments, Primary Fragment and the Secondary Fragment. Primary Fragment is the one which contains the scan attribute. All the attribute details required by a query result may reside in Primary Fragment or in both Primary as well as Secondary Fragment. Therefore total number of Disk Accesses required to retrieve tuples is sum of disk accesses required from Primary and Secondary Fragments. A cost model for estimating disk accesses by various disk access path strategies: Clustered Index Scan, Unclustered Scan and Sequential Scan are designed. It is based on analytical theories and disk access cost estimation formulae by various pioneer researchers in the field by March & Rho, Cornell & Yu, Cardenas and Yao. A cost based query optimization requires the least possible references to secondary memory. A synergic combination of Vertical Fragmentation and Access Strategies can help achieve this goal.
- Experiments were conducted after coding the Genetic Algorithm for fragmentation and access strategies (GA_FA) in PASCAL programming language. Genetic Operators formulations are based on patterns of the pioneer work of Goldberg for a simple Genetic Algorithm (SGA). Experimentation involved varying the number of attributes and the number of transactions on a given relations in sets of ten transaction profiles. It is done by varying problem complexity from small problems to medium and finally to very large problems.

Part 2 : GA_SA (Genetic Solution for SubQuery Allocation):
Second major component of a distributed database query optimization processor is the allocation of various sub operations involved in the query to the sites of a computer network. This part of research work focuses on Sub Query Operation Allocation Problem. A distributed query is first broken into various relational algebra operations like Selections /Projection or Joins/Semi-joins etc, then these
sub-operations may be performed at different possible permutations of network sites.

Experiments were conducted after coding and implementing and running the GA_SA on an experimental set of Wisconsin Benchmark Database queries. The complexity of queries is increased in terms of nodes of the query tree. It is varied from a simple query with 7 nodes to a moderately complex query with 39 nodes and further very complex queries with nodes greater than 50. The query parameters ‘Number of Joins’ & ‘Number of Sites’ are varied from 1 to 20. Execution Time analysis of various algorithms to find an optimal solution is performed.

1.7 Research Contribution

The stochastic simulator was run a number of times on various test databases and benchmark queries by varying the number of transaction attributes, transaction frequencies and scaling the size of the problem for Genetic Algorithm for fragmentation and access strategies (GA_FA).

In case of Genetic Solution for SubQuery Allocation(GA_SA) the parameters varied were number of joins & number of sites, for a set of Wisconsin Benchmark Database queries. Various existing techniques like deterministic Exhaustive procedures, Dynamic Programming, Branch & Bound, Simulated Annealing technique’s run time and costs are compared with GA_SA.

In a nut shell a summary of the research contribution is as following,

- GA_FA gives an innovative heuristical approach for integrating access strategies with fragmentation scheme at a local site. It contributes significantly in reducing overall total cost of a distributed query.

- The number of disk accesses and hence query costs showed a reduction of 61 percent as compared to an un-partitioned scheme. Reduction of disk access was to the tune of 33 percent in comparison to that of popular deterministic techniques of Cornell & Yu, and by 23% of the genetic solution by March. GA_FA’s disk access cost showed a 10% decrease in access and communication costs as compared to Barker & Jun’s approach [Barker et al.2006].
GA_SA’s findings are that when number of joins and sites individually grow more than seven, deterministic and heuristic procedures tend to go exponential and computationally intractable quickly. Stochastic Solutions like GA_SA still take time around a minute or less, because they are independent from query tree search space size. Exhaustive Enumeration Procedures choke very quickly for very large problem sizes and computing time to find a solution goes into units of hours, which is not permissible in any interactive query system. Whereas, Genetic Algorithm GA_SA’s run time rises very slowly and is virtually independent of the problem complexity. Though, GA’s does not guarantee the best or optimal solution, but give a reasonably optimal solution in a very small computing time.

Large scale Stochastic Testing is done to study effect of different implementations of various Genetic Operators (SELECTION, CROSSOVER, and MUTATION) and their Parameter Value Variations, on the Quality of Genetic Solution. Results are analyzed and verified empirically. Crossover rates are varied from 0.2 to 0.9 and Mutation rates are varied from 0.005 to 0.2 on various test queries. For example (0.7, 0.2) combination of Crossover Percentage & Mutation percentage provided the best quality of solution for medium size sub query allocation problems for SA_GA.

Experiments were performed for determining the optimal parameter values for LAN/WAN to produce a good quality genetic solution.

As the main concern of this thesis is optimizing a distributed retrieval query by the total cost reduction method, it is assumed that other components of DDBMS\(^5\) take care of Updating, Concurrency, Control, Integrity, Security and Recovery issues.

### 1.8 Roadmap

This thesis is organized as follows.

**Chapter 2** gives a brief introduction to architecture of a query processor and use of database profiles for query optimization. Next it reviews previous works and literature related to distributed database query optimization. Distributed query optimization procedures aim for either reducing ‘Total Time Cost’ of a query or ‘Response Time’ of a query. Total Time models aim to reduce overall utilization of the

---

\(^5\) Distributed Database Management
system and network resources, and hence aim to reduce the commercial cost of a query. Response Time models aim to reduce the response time of query by parallelizing independent operations and speed up the operations. Next it introduces evolutionary computing and genetic algorithms (GA). Basic operators and parameters of GA like Selection, Crossover and Mutation are introduced. Basic GA lingo is discussed.

Chapter 3  
It begins with introducing the fragmentation process, its types and advantages. Next sections discuss Access Strategies and describe a formulation of the designed cost model and an objective function. It also describes cost estimation formulation using three major ‘local disk access path’ strategies: Clustered Index Scan, Unclustered Scan and Sequential Scan. Next a general outline of the genetic algorithm for fragmentation and access strategies (GA_FA) is given along with a detailed description of genetic parameter settings and operator’s (Selection, Crossover, Mutation) design. Later part illustrates functioning of access cost methods, by taking an example transaction profile. Finally it discusses experimental results and analysis for Access Strategies chosen.

Chapter 4  
in the beginning it briefly describes various prevalent query allocation techniques and earlier research in the area of sub query allocation and optimization. Further sections illustrate a Cost Model & Objective Function Formulation for sub query allocation. Use of database statistics for the experimental query environment is highlighted. Next sections describe details of the sub query operations and their cost analysis based on the proposed mathematical model. Finally it analytically analyzes and validates the solution by taking a set of Queries on a Wisconsin Benchmark Database. Then a comparison of the GA_SA’s query allocation ‘Costs’ and ‘Execution Times’ is done with the other widely followed methods like Exhaustive Enumeration, Dynamic Programming, Branch & Bound and Simulated Annealing etc.

Chapter 5  
Role of general genetic parameters like population size, crossover rate, and mutation rate on the performance of GA_SA (Genetic Algorithm for SubQuery Allocation) was examined in first few sections of this chapter. Population size is usually kept fixed. Observation shows that if it’s a small number N (between 10 and 40) then minimum number of generations to reach a good quality solution is high. But if it’s a big
number N (between 100 and 200) then time taken by the GA_SA to reach a good solution (Computation Time) increases. Crossover Rate for GA_SA is varied from a low level of 0.2 to high of 1.0. If Crossover Rate is kept very low it takes lot many generations to reach a good solution and probability of finding a good solution reduces considerably. If it is kept high like 1.0 where whole population crosses, quality of solution is poorer than we get with moderate values like 0.6.

Next observation was that Mutation should not occur very often, because then GA will in fact change to a random search. It is should be applied to less than 1% of the population of a generation. Mutation rate of 0.2 was found to be most suitable for genetic Sub Query Allocation Algorithm (GA_SA). Finally total cost breakups into CPU, I/O and Communication costs were studied for LAN/WAN environments by varying the number of joins involved in a query. When ‘number of joins’ were increased more than 12, Communication costs started dominating all other costs for a WAN but surprisingly I/O costs remain dominating costs for a LAN.

*Chapter 6* contains the drawn conclusions and discusses proposed solution’s limitations and future scope of work.