INTRODUCING PERF TO A QUERY OPTIMIZATION ALGORITHM

By

RAEF ABDALLAH

Bachelor of Science

Lebanese American University

Beirut, Lebanon

1997

Submitted to the Faculty of the
Graduate College of the
Oklahoma StateUniversity
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2002

INTRODUCING PERF TO A QUERY OPTIMIZATION ALGORITHM

Thesis Approved:

[illegible]

Thesis Advisor

J Chandler

[illegible]

Timothy J. Pettibone

Dean of the Graduate College

ACKNOWLEDGMENTS

My sincere thanks to Dr. George Hedrick for serving as my advisor on my committee. His support through this thesis is highly appreciated. This work would not have been possible without his continuous guidance and input.

Special thanks to Dr. Dai and Dr. Chandler for serving on my committee and reviewing this thesis.

Finally, I thank the Computer Science Department for giving me the opportunity to pursue my interest in the computer field and helping me succeed in a competitive computer industry.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

A database system is a collection of information that maintains data and enables users to access stored information upon request. This information might exist over a long period of time, and it is managed by a Database Management System (DBMS). A database system is an electronic file cabinet that is significant to the organization or individual who maintains the database [6]. For example, an airline reservation system is a database system that might contain the following information:

- Departure and arrival times or flight number.
- Ticket prices and seat availability.

The user of a database system should have the privileges required to perform a variety of operations on the database. These operations include the following:

- Retrieving data
- Adding new data
- Deleting existing data

These operations are achieved by the DBMS, which has the following functions:

- Allows users to create a new database and assign the logical structure of data.
- Enables users to query the data using language known as a query language.

The evolution of the telecommunications industry and the development of hardware and network structures have paved the way for the domination of distributed databases and have spread the use of decentralized data. However, as businesses expand and continue to decentralize their data, they also continue to demand instant access to the information they need.

In order to achieve faster information access, work has been done on query optimization. Query optimization is the process of estimating the cost of various query plans, and then choosing the best plan to answer expensive queries in a fast, efficient way. The solutions associated with query optimization involve distributed databases as well as centralized databases. The purpose of this thesis is to outline a query optimization algorithm and apply a concept known as Partially Encoded Record Filters (PERF) to it. The problem of query optimization has been researched, and the following two methods have been proven to promote faster information access:

1. Using parallel processing to minimize the response time of the query.
2. Reducing the amount of data transferred from one site to another, thereby minimizing the cost of data transmission over the network.

The effectiveness of the first approach relies on the rapid advancement of the hardware field. High-bandwidth networks and inexpensive processors contributed to the development of the first parallel systems. Although parallel processing improves the speed at which data is transmitted from one site to another, it ignores the amount of data to be transmitted. The ability of a user to access information quickly depends mainly on the structure and communication links within a given network [10] . The notion of 100% reliability within a data transfer system is theoretical; every system is vulnerable to countless variables that may cause failure or malfunction. Therefore, this hardware approach alone does not guarantee a reliable solution.

In this thesis, the second approach to query optimization is considered. This study focuses on minimizing the amount of data transferred from one site to another using semi-joins.

Many algorithms have been proposed. One of the most recent algorithms in this field was proposed by William Bealor in 1995 [4]. He suggested an algorithm that works on optimizing a query in a distributed database. He also claimed that his algorithm is an improvement over the Apers, Hevner and Yao algorithm [2] [3]. The latter was one of the first and most important algorithms implemented for query optimization. This thesis presents Bealor's algorithm in detail, and it presents the algorithm's complexity. It also discusses several other algorithms. A new technique called the Partially Encoded Record Filters (PERF) approach will be applied to Bealor's algorithm and results will be presented.

William Bealor from the University of Windsor, Canada, proposed an algorithm that uses profitability and gain calculation results to assist in choosing reducers. A reducer is the last attribute in a sequence of semi-joins performed under certain conditions that will be discussed later on. Bealor claimed that his algorithm is an enhancement over the AHY algorithm [4].

This study presents an improvement to Bealor's algorithm using PERF joins. The improvement helps in reducing retrieval time and gives better query optimization performance.

This thesis is organized into six chapters. Following this introduction, which is Chapter 1, Chapter 2 contains three main parts: 1) A set of definitions of commonly used technical terms and concepts 2) A literature review containing various query optimization techniques, and 3) A list of basic assumptions and notations to be used throughout this thesis. Chapter 3 describes the W algorithm [4] and the PERF concept. Chapter 4 is a comparative example solved using PERF and the Bealor algorithm. Chapter 5 presents the experimental results, and Chapter 6 summarizes and concludes the thesis.

CHAPTER II

LITERATURE REVIEW

In this chapter, background information is presented to summarize the query optimization problem. The first part includes a set of definitions and key terms. This is followed by presenting different techniques that have been proposed to deal with the problem. The chapter concludes by providing a list of assumptions and notations taken throughout this study.

2.1- Definitions

There are different perspectives to deal with the query optimization problem [9][11][14]:

1. Physical optimization: Consists of normalization of tables, index tuning, database page size and other parametric values to be set correctly.
2. Programmatic/Syntactic optimization: the programmer codes special selection and join modules for the requests in order to perform optimal transmission among different sites when the corresponding programs are used.
3. Systematic/Semantic optimization: Relational database management systems (RDBMS) contain optimization algorithms that can be used internally to optimize each query. Also, semantic information about the database and query can help to improve the efficiency of query evaluation. A coupling of both syntactic and semantic optimization can lead to a better performance.

4. Heuristic optimization: Uses rules to order operations in a query execution strategy.
5. Dynamic optimization: Estimates the cost for executing each plan and chooses the best strategy.

In this work, the main focus is on heuristic and dynamic strategies. However, all strategies aim to minimize certain cost functions such as [6]:

1. Dollar cost function that is represented by the following:
   a- CPU processing
   b- Network usage
   c- Combination of both CPU and network usage.
2. Delay function represented by the following:
   a- Local CPU processing.
   b- Data transmissions across communication links.
   c- Combination of both CPU and network.
3. Volume of data represented as follows:
   a- Processed locally by each CPU.
   b- Transferred across the network between different sites.
4. Total size of partial results.

The processing of a distributed query can be done using two main approaches:

1. Two-Phased approach [6]:

a- Determine a sequence of relational operations to minimize the total size of partial results.

b- Design a polynomial time algorithm to find site locations for executing the relational operations such that the result is optimal.

This approach decomposes the optimization problem into two simple ones, each of which can be solved more efficiently than the undivided problem.

2. Three-Phased approach[7]:

   a- Local processing phase: Involves all relational operations that can be performed on the same site and reduce the amount of data to be shipped.

   b- Reduction phase: A sequence of reducers is applied to minimize the size of partial results.

   c- Final processing phase: Partial results are sent to an assembly site where final query processing is performed.

The most common relational operations performed on a database are as follows [10] [21][22]:

1. Projection: The projection of a relation R on a set of attributes T disregards columns of R that do not belong to T. Projection can also eliminate duplicate rows and is represented by π [21].
2. Selection: The selection of tuples in R with attribute values that are equal to a certain value and is denoted by $R.A_{i} = k$ where A_i is the ith attribute of relation R and k is an attribute value. Selection is also represented by σ [21].

3. Join: The join operation of relations R_1 and R_2 over a common join attribute A_i denoted by $R_1 \bowtie R_{2.}$ is the concatenation of each tuple of R_1 with the corresponding matching tuple from $R_{2.}$ where the common join attribute A_i is equal in both tuples (see Figure 1 on page 9). This operation is fundamental in combining information from different relations. However, joins are very expensive, particularly when relations are large and have a huge number of tuples. Joins between sites send data unfiltered from one site to another, resulting in the transfer of large amounts of data over the network. The cost of executing a join is estimated based on the size of the joining relations and is denoted by $\Theta(n)$ where n is the size of the joining relations $(R_1 \cup R_2)$.

4. Semi-Join: A semi-join operation from R_1 to R_2 denoted by $R1 \rtimes R2$ is a projection of R_1 over a join attribute A_i. Next, a shipment of the partial result to the R_2 site is performed where a join is done [23]. In particular, a semi-join is useful for computing joins in distributed systems. In other words, a semi-join operation proceeds as follows:

- Project R_1 over a common attribute $A_{i.}$
- Send the Projection result $R_1[A_i]$ from site 1 to site 2.
- Reduce R_2 by eliminating the tuples whose attributes A_i do not match any value of $R_1[A_i]$ (see Figure 2 on page 10).

R_1

X1	Y2
X2	Y2
X3	Y3
X4	Y3
X5	Y5

R_2

Y1	Z1
Y2	Z2
Y2	Z3
Y3	Z4
Y5	Z5

$R_1 \bowtie R_2$

X1	Y2	Z2
X1	Y2	Z3
X2	Y2	Z2
X2	Y2	Z3
X3	Y3	Z4
X4	Y3	Z4
X5	Y5	Z5

Figure 2.1- Join Operation Example

R_1

X1	Y2
X2	Y2
X3	Y3
X4	Y3
X5	Y5

R_2

Y1	Z1
Y2	Z2
Y2	Z3
Y3	Z4
Y5	Z5

$R_1 \rtimes R_2$

Y2	Z2
Y2	Z3
Y3	Z4
Y5	Z5

$R_2 \rtimes R_1$

X1	Y2
X2	Y2
X3	Y3
X4	Y3
X5	Y5

Figure 2.2 - Semi-Join Operation.

The goal of a semi-join is to reduce the number of tuples in a relation, thereby reducing the byte size before transferring the result to another site. Therefore, semi-joins are desirable in a distributed database system where transmission costs supersede the processing costs. Although semi-joins minimize the amount of data transmitted, they add an overhead cost for local processing [10].

In order for a semi-join to be desirable, its benefits must exceed its cost. The cost of a semi-join is expressed as the amount of inter-site data transfers needed to compute the operation while the benefit is the amount of data eliminated [7]. In the previous example, the cost and benefit of the semi-join $R_1 \rtimes R_2$ are as follows:

Cost = $S(R_1[Y])$, where S is the size of attribute Y in relation R_1 and

Benefit = $S(R_2) - S(R'_2)$ where R'_2 represents the subset of matching tuples of R_2 [24].

In general, the cost and benefit functions are defined as follows:

Cost: $C(R_i\text{-}a_{ik} \rightarrow R_j) = C_0 + C_1 * b_{ik}$

Where:

$(R_i\text{-}a_{ik} \rightarrow R_j)$: Another representation of a semi-join over attribute a_{ik}.

C_0: Start-up cost for a transmission measured in time unit.

C_1: Fixed cost per byte transmitted measured in time unit.

b_{ik}: Size (in bytes) of the data item in attribute a_{ik}.

Benefit: $(R_i\text{-}a_{ik} \rightarrow R_j) = S_j - S'_j$

Where:

S_j: Size (in bytes) of relation R_j

S'_j: Size of relation R_j after reduction

Thus, the benefit of a semi-join is the amount of data eliminated by reduction. The profitability of a semi-join is the difference between benefit and cost and is defined as follows:

$$P(a_{ij} \bowtie a_{kj}) = B(a_{ij} \bowtie a_{kj}) - C (a_{ij} \bowtie a_{kj})$$

Let $\rho(a_{ij})$ be the selectivity of attribute a_{ij}; that is, the number of distinct values of a_{ij} ($|R_i[a_{ij}]|$) divided by all possible domain attribute values ($|D[a_{ij}]|$).Hence, $\rho(a_{ij}) = |Ri[a_{ij}]| / |D[a_{ij}]|$.

In terms of $\rho(a_{ij})$,

$C(a_{ij} \bowtie a_{kj}) = C_0 + C_1 \text{ x } S(R_i[a_{ij}])$ where x is the multiplication operator.

$$\begin{aligned} B(a_{ij} \bowtie a_{kj}) &= S_j - S'_j \\ &= S_j - (S_j * \rho(a_{ij})) \\ &= S_j * (1 - \rho(a_{ij})) \end{aligned}$$

The sequence of multiple semi-joins is very important because the size of partial results will vary. The last attribute in the sequence is called the reducer because it is reduced by all others and it also reduces the cost of joining two relations (Chapter III has more details on that). A reducer is denoted by d^*_{ab} where b is an attribute in relation A. If we have two joining attributes d_{xj} and d_{yj}, we define the marginal profit to be the extra profit we can achieve by using one reducer instead of the other, as in the following formula:

$MP_{Ri}(d^*_{xj} \bowtie d_{yj}) = P(d^*_{yj} \bowtie Ri) - P(d^*_{xj} \bowtie R_i)$ where d^*_{xj} is the reducer for attribute j in relation x. Same explanation applies to d^*_{yj}.

The formula shows the marginal profit gained by using y as a reducer minus the profit gained by using x as a reducer. This shows the critical nature of the choice and the order of reducers.

The gain of semi-join is the sum of the profit and marginal profit.

$$G(d^*_{xj} \bowtie d_{yj}) = P(d^*_{xj} \bowtie d_{yj}) + MP(d^*_{xj} \bowtie d_{yj})$$

A semi-join is said to be cost-effective when its gain > 0; hence, its benefit exceeds its cost and it has a marginal profit [16].

5. Two-Way Semi-Join: A two-way semi-join is an extended version of the semi-join. A two-way semi-join of relation R_j to R_i over the attribute A_k, is obtained by performing two semi-joins. The first one is a semi-join from R_j to R_i and the second one is from R_i to R_j.

Zhe and Ross proved [25]:

Lemma 1: Given a semi-join S, if the two-way semi-join T is performed instead of S on relations R_i and R_j :

1. If S reduces R_j to $R_{j'}$, then T also reduces R_j to $R_{j'}$.

2. The cost and benefit for R_j reduction are the same for S and T.

2.R_i is always reduced by T in a cost-effective way.

This lemma leads to the following corollaries [25]:

Corollary 1: For semi-join $R_i - A \longrightarrow R_j$, if it is cost effective, then so is the two-way semi-join $R_i \longleftarrow A \longrightarrow R_j$, if it is not cost-effective, then the semi-join $R_i - A \longrightarrow R_j$ is not also cost-effective.

Corollary 2: For a two-way semi-join $R_i \longleftarrow A \longrightarrow R_j$, if it is not cost effective, then the semi-join $R_i - A \longrightarrow R_j$ is not also cost effective.

2.2- Previous Work

Many algorithms and strategies were proposed aiming to find the most nearly optimal solutions. In this section, we show some of the most important and widely used techniques developed.

2.2.1-SDD-1 Query Optimization Algorithm

This algorithm was developed to optimize queries for the SDD-1 distributed database system by minimizing the quantity of inter-site data because network transmission was the slowest component in query processing [4][7]. In SDD-1, queries are processed as follows:

1. Query Mapping: The query is mapped to an envelope E which is a superset of the database required to answer the query.
2. Envelope Evaluation: Data retrieval is accomplished by translating E into a program P with a relational operation (reducer) and a set of commands to move results of P to the assembly site.
3. Query Execution: The query is executed at the assembly site.

The goal is to construct a reducing program P and choose the most appropriate assembly site to minimize costs. The proposed solution is a hill-climbing algorithm that uses a combination of joins and semi-joins:

```
1. Initialize the program P to contain local operations.
2. Repeat
            Add to P profitable non-local semi-joins.
   Until no more profitable semi-joins are found.
3. Select assembly site and append to P the necessary commands to move reduced
   data to assembly site.
```

The disadvantage of this algorithm is its inability to backtrack and consider other alternatives.

2.2.2- A Hash Partition Strategy for Distributed Query Processing

The concept behind this strategy is to partition relations to many fragments and distribute those fragments to a number of sites for parallel execution. This strategy creates a hash function to partition a relation in two disjoint fragments. An example of such function would be one to partition a relation R_1 into two fragments F_{11} and F_{12} where F_{11} contains all tuples whose join attribute values are odd number (supposing the join attribute domain is integer with uniform distribution) and F_{12} contains all tuples whose join attributes are even numbers. Similarly, the second relation R_2 can be partitioned into F_{21} and F_{22} such that the following is true:

$$F_{11} \bowtie F_{22} = \emptyset \text{ and } F_{12} \bowtie F_{21} = \emptyset$$

Where $\emptyset$ represents the empty set { }.

The hash partition strategy is an enhancement over the traditional partition strategy because of the following:

1. Local processing (join cost) at each site is small due to the fact that only a fragment of each relation is involved.
2. Communication cost is reduced because only a fragment of the relation is transmitted rather than the whole relation.

Several variables are used to compute the size of partitions, destination sites and other parameters that influence the performance of the algorithm and its resulting cost.

2.2.3- GroupWise Processing of Relational Queries

By definition, a query Q is a group query with respect to certain partitioning attributes S if it is possible to answer Q by [8]:

1. Partitioning the data according to values of attributes.
2. Evaluating another query Q_1 on each partition of the database.
3. Taking the union of the results.

This strategy aims at dividing a complex query into simpler ones, which are easier to optimize and to evaluate. Every base relation is partitioned according to a certain partitioning set. Each partition is likely to have a small size, fits in main memory, and reduces CPU cost. Executions of simple queries can be done in parallel and the final step is the union of partial results.

For example, consider the following complex query that can be divided into simple ones: find the average checking account balance for all customers who have greater balance than their average saving account balance.

This query can be divided into:

Query1: Find average checking balance for each customer.

Query2: Find average saving balance for each customer.

Query3: Find tuples that have average checking balance greater than their corresponding average saving balance.

Query4: Group over the user's identifier.

This algorithm was proved to enhance the complexity time for finding optimal results because simple queries can be solved with less effort and more parallelism [8].

2.2.4- The AHY Algorithm

Apers, Henver and Yao developed an algorithm for a special class of queries called simple queries using semi-joins. The first proposed two algorithms PARALLEL and SERIAL find strategies with respectively minimum response time and total time. Then, a new extended version called GENERAL was developed to process general distributed queries [1].

2.2.5- The W Algorithm

This algorithm was developed by Todd Bealor from Windsor University in 1995. It uses "reducers" to optimize data to be transferred across the network [4]. This algorithm showed a considerable enhancement over the AHY algorithm. It will be also discussed in detail in the next chapter.

2.2.6- Others

Special algorithms were designed also to optimize queries over global information systems such as the Internet that involves a large number of information sources distributed over a network. Due to the large number of sites, the query optimizer uses descriptions that relate contents of the site relations to a uniformed view of the information space called world-view. Optimization is done by using constraints in site descriptions and query to prune irrelevant and redundant information.

2.2.7- Benchmarks

Proposing algorithms is not sufficient, but what is needed is the tool to measure and evaluate the performance of algorithms on test databases. Such tools are called benchmarks. Benchmark results depend on the workload, application requirements and system design and implementation. The workload is the amount of work assigned to be performed by the database system within a certain period of time. Using benchmarks, one can compare the estimated cost of the algorithm with the actual cost of its running on test data. Many benchmarks were proposed, some of which are [19]:

1. Wisconsin Benchmark for relational databases.
2. TPI benchmark for OLTP.
3. TPC-D benchmark for decision support systems.
4. Client / Server benchmark.

However, a detailed study of those benchmarks is beyond the scope of this thesis.

2.3 Assumptions and Notations

2.3.1- Assumptions

In order to complete this study, many assumptions were made:

1. In the calculations, local cost is negligible compared to transmission cost. Local processing is considered to be ϵ (nearly 0).
2. The network is considered to be homogeneous in all the sites.
3. No PERF calculation method is used but PERF concept is applied to W algorithm and calculation is done according to W algorithm.
4. The selectivity of a relation is considered to be the same as the selectivity of the joining attribute in the relation.
5. The cost calculated through the cost function and along the schedules is the time needed to transfer the given amount of data from one location to another. The cost is given in units of time (ms, s, ...).

2.3.2- Notations

The following symbolic relations are used throughout this study:

$R_1, R_2, .., R_m$	: Collections of relations in the database.
α_i	: Number of attributes in relation R_i.
$d_{ij}, j=1, 2, \alpha_i$	: Attributes in the relation R_i.
d_{ij}	: jth attribute in relation R_i
$\lvert R_i \rvert$	: Cardinality of relation R_i (number of distinct tuples).

N_i	: Number of tuples in relation R_i.
$S(R_i)$	: Size in bytes of relation R_i
$\lvert d_{ij}\rvert$	: Cardinality of distinct attribute values of the jth attribute in relation R_i
$S(d_{ij})$	: Size in bytes of data item in attribute d_{ij}.
$R_i - d_{ik} - R_j$	: Join of relation R_i and relation R_j over join attribute d_{ik}.
$R_i - d_{ik} \longrightarrow R_j$	: Semi-join of relation of relation R_j by relation R_i over join attribute d_{ik}.
$R_i \leftarrow d_{ik} \rightarrow R_j$	: Two-way semi-join of relation R_i and relation R_j over the common join attribute d_{ik}.
$D(d_{ij})$	: Domain of possible values for attribute d_{ij}.
$\lvert D(d_{ij})\rvert$	: Cardinality of $D(d_{ij})$ which is the number of distinct values that make up the domain for d_{ij}.
$\rho(d_{ij})$	: Selectivity of attribute d_{ij} where selectivity is defined as $\lvert d_{ij}\rvert / \lvert D(d_{ij})\rvert$.
d^*_{ab}	: Reducer.

CHAPTER III

ALGORITHMS

This chapter presents the details of an existing heuristic algorithm. algorithm W. A proposed new version of this algorithm is presented using PERF joins instead of semi-joins. The PERF concept is applied to the W algorithm. A short complexity analysis is also presented (Table 1). The proposed WPERF algorithm is an enhancement to W algorithm and constitutes the main contribution of this thesis.

3.1- The W Algorithm

The primary aim of this algorithm is to minimize total time by using reducers to eliminate unnecessary data. This algorithm is characterized by two distinct phases [4][5]:

1. Semi-join schedules for constructing each reducer are formed using a cost-benefit analysis based on estimated attribute selectivities and sizes of partial results.
2. Schedule is executed.

Before giving the details of the heuristic. some assumptions and definitions are needed. As shown in the previous chapter. calculations are done using the cost. benefit, profit, marginal profit and gain equations. The following criteria are needed:

Criteria 1: Profitability is not a sufficient condition for performing a semi-join during the construction of the reducer.

Criteria 2: If there is no marginal profit in a semi-join then it should not be performed.

Criteria 3: If the marginal profit is greater than the cost, then the semi-join should be performed. Consider the semi-join $d^*_{xj} \ltimes d_{yj}$, which is part of a sequence to construct a reducer. If there exists a relation R_i, $i <> y$ such that,

$$[|R_i|(\rho(d^*_{xj}) - \rho(d^*_{yj}))] - [|d^*_{yj}|] > 0$$

then,

$G\ (d^*_{xj} \ltimes d_{yj}) > 0$ and the semi-join should be performed.

There might be semi-joins that are not profitable but are still gainful because their marginal profit is large enough. This type of semi-joins is called gainful non-profitable semi-joins [25]. In this case, it is advisable to execute this type of semi-joins because they ensure a great reduction effect due to the increased selectivity propagated to later semi-joins. This leads to the following corollary:

Criteria 4: A sufficient, but not necessary condition for adding a semi-join, to the schedule for reducer construction is that the marginal profit is greater than the cost of the semi-join.

Algorithm W works as follows:

Establish schedules for the construction of reducers. For each join attribute *j*, construct schedule for the reducer d^*_{mj}. At each level each schedule is considered independently. Hence, no semi-joins are executed yet. This is achieved in two phases:

I) Sort attributes by increasing size such that:

$$S(d_{aj}) \leq S(d_{bj}) \leq - - - \leq S(d_{mj})$$

II) Evaluate semi-joins in order beginning with $d_{aj} \rtimes d_{bj}$. Append semi-join to schedule if:

a- It is profitable and marginally profitable. Hence,

$$P(d_{aj} \rtimes d_{bj}) > 0 \text{ and } MP(d_{aj} \rtimes d_{bj}) > 0 \text{ or,}$$

b- It is gainful but not profitable. Hence,

$$P(d_{aj} \rtimes d_{bj}) < 0 \text{ but } G(d_{aj} \rtimes d_{bj}) > 0.$$

If semi-join is appended then $d^*_{bj} \rtimes d_{cj}$ is evaluated next; otherwise, $d^*_{aj} \rtimes d_{cj}$ is considered.

Repeat this process until all semi-joins in the sequence are evaluated. The last attribute in the sequence will be called the reducer.

Examine the effects of reducers. Consider the reduction effects of the reducers on all applicable relations by:

I) Sorting the reducers from smallest to largest.

II) Estimating the cost and benefit of a semi-join with each admissible relation and for each reducer. Profitable semi-joins are appended to the schedule.

III) Review of unused semi-joins. For non-profitable reducers, reexamine the possibility of having profitable semi-joins for that particular join attribute. This step is done using the following sub-steps:

a- Sort attributes by increasing size.

b- Evaluate each semi-join and append profitable semi-joins to the final schedule. Note that marginal profit is not considered in this step.

IV) Execute the schedule. During this phase, reducers are constructed and shipped to designated sites to reduce the corresponding relations. Then, reduced relations are shipped to the assembly site.

This heuristic is simple and efficient. It aims to construct in the cheapest possible way, reducers that are highly selective. Those reducers are then used to eliminate tuples from participating relations prior to shipment to the query site (assembly site).

It should be noted that Algorithm W ameliorates the choice of join attributes and their order but does not eliminate redundant transmissions because schedules are treated separately.

3.3.1- Complexity Analysis of Algorithm W

This algorithm is cost-effective in the sense that it constructs reducers with a minimum cost overhead associated with their construction. In step 1 of the algorithm, phase 1 sorts attributes with complexity of $O(m\log m)$ where m is the number of attributes. This step is repeated for *n* common-join attributes and hence takes $O(nm\log m)$. Phase 2 calculates profit and marginal profit for m-1 semi joins of n attributes leading to a complexity of $O(nm^2)$. In step 2, phase 1 takes $O(n\log n)$ to sort n attributes and step 2 calculates the cost and benefit in $O(nm)$. Step 3 reconsiders α non profitable reducers in $O(\alpha m)$ with $\alpha \leq n - 1$ where n is the number of common join attributes. Hence, the complexity of Algorithm W in the worst case has a complexity of $O(nm^2)$.

3.2- The PERF Concept

PERF is a new two-way semi-join that works as follows:

Considering two relations R_1 and R_2,

- Project relation R_1 on a common attribute "A" and get $R_1[A]$
- Ship $R_1[A]$ to R_2
- Reduce the size of R_2 by a semi-join with $R_1[A]$
- Send back to R_1 a bit vector that contains one bit for every tuple in $R_1[A]$. In case the $R_1[A]$ tuple matches a tuple in R_2, a 1 is returned to relation R_1; otherwise, 0 is returned.

$R_1(X,Y)$

X1	**Y2**
X2	**Y2**
X3	**Y3**
X4	**Y3**
X5	**Y5**

$R_2(Y,Z)$

Y1	**Z1**
Y2	**Z2**
Y2	**Z3**
Y3	**Z4**
Y5	**Z5**

PERF(R_1)

0
1
1
1
1

PERF(R_2)

1
1
1
1
1

Figure 3.1: PERF for R_1 and R_2

The following two corollaries are derived [25]:

Corollary 1: Consider two relations R and S, with R physically ordered in some fashion. S contains the join reduce information for R. The PERF relation R with respect to S denoted as PERF(R) is a bit vector of |R| bits. The join of PERF(R) is set if and only if the jth tuple of R appears in the join result R $\bowtie$ S.

Corollary 2: Let R and S be relations at distinct sites in a distributed database. The PERF join R with S consists of:

1. Sending R^x to the site of S where R^x is the result of the projection of R over the attribute x.
2. Performing S $\rtimes R^x$
3. Sending PERF(R) back to the site R to reduce R.

3.3- The WPERF Algorithm

As it can be concluded from its name, WPERF algorithm is a version of Algorithm W using PERF joins instead of semi-joins. Both the PERF concept and the W algorithm appear earlier in this thesis. Applying the PERF joins concept to the W algorithm will result in these steps:

1. Establish schedule for the construction of reducers using the following two phases:

I) Sort attributes by increasing size such that:

$$S(d_{aj}) \leq S(d_{bj}) \leq - - - \leq S(d_{mj})$$

II) Evaluate PERF joins in order beginning with daj and dbj. The cost is calculated by adding to the transmission cost (forward cost) the backward overhead that consists of a bit vector representing the matching tuples from the target site. Define profitability, marginal profitability, and gain in terms of this cost. It is important to mention that in order to get the bacward cost, it is necessary to check the PERF list item concerning the two relations involved. If the value is 0, the PERF has not been executed before, then calculate the forward cost as explained previously. Else, the total cost = 0 for this PERF.

Repeat this step until all PERF joins are evaluated.

Mark with 1 in the PERF list all items corresponding to PERF joins that have been accepted in the selected schedule.

2. Examine the effect of reducers. It is to be noted here that all cost, benefit and profitability calculations are performed by adding the backward overhead when necessary and eliminating the cost when appropriate.
3. Review unused PERF joins using same logic as above.
4. Schedule execution.

3.3.1- Complexity Analysis of WPERF

The PERF concept does not increase time of the algorithm. As far as algorithm W is concerned, the worst case complexity was $O(nm^2)$ where:

n is the number of common-join attributes.

m is the number of attributes.

Finally, the PERF join was introduced by K. Ross as an enhancement over two-way semi-join [25]. Bloom joins suffered from loss of information due to hash collisions. PERF proved itself as a promising technique capable of filling many gaps created by semi-joins. However, PERF relies heavily on the scan order of the relations involved in the query. That is, after performing one PERF join and until the query is executed, the order of tuples in the participating relations should remain the same because a PERF bit vector has been stored. However, in real time online systems this scan order might not be preserved. This can be solved by writing a special routine to update the PERF vectors whenever tuples are updated in relations that are currently participating in a query.

Algorithm	Complexity
The W Algorithm	$O(nm^2)$ n: number of common-join attributes m: number of attributes
The WPERF Algorithm	$O(nm^2)$ n: number of common-join attributes m: number of attributes

Table 3.1: Summary of Complexity Analysis.

CHAPTER IV

A COMPARATIVE EXAMPLE

This chapter presents a comparative example solved using the two algorithms: W and WPERF. It aims at illustrating the use of those algorithms and comparing the results. A sample query follows and solutions to this query are presented using the discussed methods in earlier chapters.

Query: Give the part number, name and total quantity for all parts that are currently on order from suppliers who supplies that part to jobs 30 or 40 (figure 4).

The database used contains the following relations:

1. PARTS (P#, PNAME): This relation contains part numbers and names.
2. ON_ORDER (P#, S#, QTY): This relation contains part numbers, supplier numbers and quantity on order.
3. P_S_J (P#, S#, J#): This relation relates each job number to a part number and identifies the supplier for those parts.

The database is distributed and each relation resides at a different site. The two joining attributes are P# and S#. The cost function to be used throughout this chapter is: $C(X) = 20 + X$.

The cost function is a linear function in the form of $y = aX + b$ where:

a: cost added per byte transmitted measured in a unit of time (milliseconds for example).

b: fixed cost dependent on the network used. In this example, $a = 1$ and $b = 20$.

X: is the size of data to be transferred in bytes.

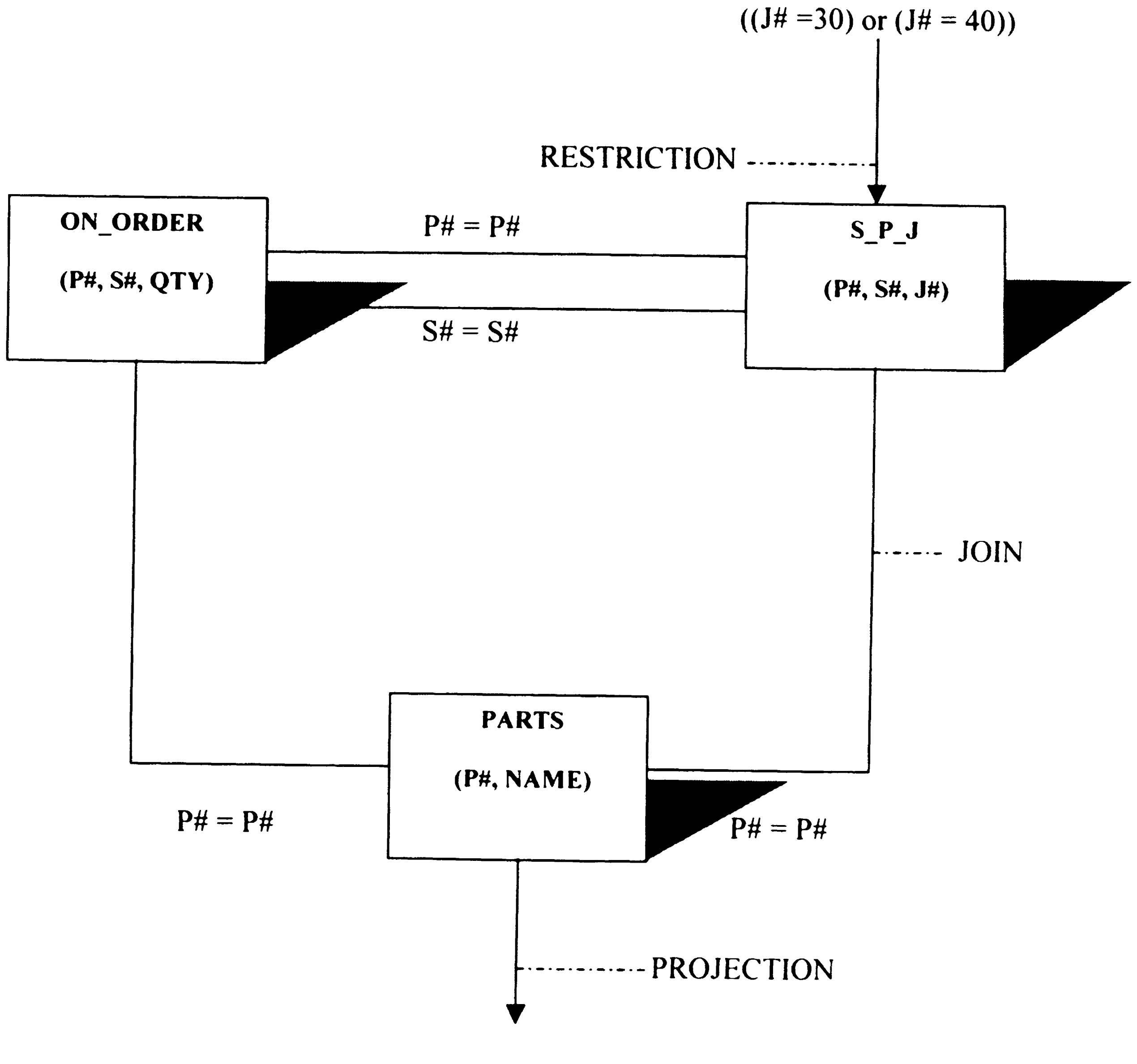

Figure 4.1 - Sample Query

The corresponding size and selectivity relations are given in the following table:

Relation: Ri	**|Ri|**	**Si**	**d_{i1} = P#**		**d_{i2} = S#**	
			b_{i1}	**ρ_{i1}**	**b_{i2}**	**ρ_{i2}**
R_1: ON_ORDER	70	1000	400	0.4	100	0.2
R_2: S_P_J	140	2000	400	0.4	450	0.9
R_3: PARTS	150	3000	900	0.9	-	-

Table 4.1 - Relations Description.

For each relation above:

$|R_i|$: cardinality of the relation (number of tuples).

S_i: size of the relation in bytes.

b_{ii}: for each joining attribute, the size in bytes, of the column in the corresponding relation.

ρ_{ii}: for each joining attribute, the corresponding selectivity.

4.1- The W Algorithm

Start by establishing schedules for the construction of the two reducers:

Reducer for d_{11}: The first semi-join is considered $d_{11} \rtimes d_{21}$ with cost $S(d_{11}) = 420$ ms and benefit $S(R_2) - (S(R_2) * \rho(d_{11})) = 1220$ bytes.

After transferring data from site 1 to site 2, with a cost of 420 ms, relation 2 is reduced. At this stage, a transmission from site 2 to any other site will not require the transfer of all the tuples of relation 2 but only the matching tuples that were the result of the semi-join which occurred between site 1 and site 2. The size of data to be transmitted is equal to the size of the corresponding column multiplied by the selectivity of the same attribute from relation 1. This is the concept of data reduction that semi-joins introduced.

Hence, the marginal profit for R_3:

$$\begin{aligned} MP_{R3} &= S(R_3) \text{ x } (\rho(d_{11}) - \rho(d_{21}) + S(d_{11}) - S(d_{21})) \\ &= 3000 \text{ x } (0.4 - 0.4 * 0.4) + 420 - 180 \\ &= 960 \end{aligned}$$

where x is the multiplication operator.

Since both profit and marginal profit are positive, this semi-join is added to the schedule.

Next, the semi-join $d^*_{21} \bowtie d_{31}$ is examined with cost 180 and benefit 2540 (d^* is a reducer) . The marginal profit of this semi-join with respect to R1 is:

$$MP_{R1} = 1000 \times (0.4 - 0.4 \times 0.9) + 180 - 164 = 56$$

Again, both profit and marginal profit are positive; therefore, the semi-join is added and the reducer is d^*_{31}. It is constructed by the following schedule:

$$\begin{tikzcd} d_{11} \arrow[r, "400"] & d_{21} \arrow[r, "160"] & d_{31} \end{tikzcd}$$

Reducer for d_{i2}: The only semi-join to be considered is $d_{12} \bowtie d_{22}$ where the cost 120 and the benefit is 1620. The marginal profit with respect to R_1 is:

$$MP_{R1} = 1000 \times (1 - 0.9) - 90$$

$$= 10 \text{ bytes}$$

Therefore, the schedule for constructing d^*_{22} is:

$$\begin{tikzcd} d_{12} \arrow[r, "100"] & d_{22} \end{tikzcd}$$

The second phase considers the reduction effect of the reducers starting with the smallest. Calculations are based on the fact that certain relations are reduced by the construction of the reducer. The reduction $d^*_{22} \longrightarrow R_1$ is considered first with cost 110 and benefit 120. This semi-join is appended to the schedule. Next, the reducer d^*_{31} is considered keeping in mind we had the following:

-$S(R_1) = 900$ since R_1 was reduced by d^*_{22}.

-$S(R_2) = 160$ after the construction of reducers.

-$S(R_3) = 480$ which has been reduced during the construction of reducers.

Therefore, the reduction d^*_{31} has a cost of 164 and a benefit of 596. It is also appended to the schedule.

Therefore, the final schedule for execution is

$$\begin{tikzcd}
d_{11} \arrow[r, "420"] & d_{21} \arrow[r, "180"] & d_{31} & d^*_{31} \arrow[r, "164"] & R_1 & R_1 \arrow[r, "344"] & QS \\
 & & & & & R_2 \arrow[r, "180"] & QS \\
d_{12} \arrow[r, "120"] & d_{22} & & d^*_{22} \arrow[r, "110"] & R_1 & R_3 \arrow[r, "500"] & QS
\end{tikzcd}$$

where "$\longrightarrow$" is another representation of a semi-join.

The total cost is 2018 ms.

4.1 The WPERF Algorithm

The same logic is applied but the semi-joins are replaced with PERF joins. Considering again the schedule created for the second reducer which is:

$d_{12} \xrightarrow{120} d_{22}$ $d^*_{22} \xrightarrow{110} R_1$ $R_3 \xrightarrow{500} QS$

Using the PERF joins, the transfer from d_{12} to d_{22} costs nearly 128 (because of the backward cost added), and it eliminates the necessity to send $d^*_{22} \longrightarrow R_1$ as the information needed from this transmission is stored in the bit vector. Therefore, the cost of the previous transmission becomes 0, and the W algorithm reduced the total cost of the schedule by at least 90. The contribution of the PERF to W algorithm was nearly 4.46 %.

WPERF is the application of the PERF concept to the W algorithm. The construction of reducers is almost the same but semi-joins are substituted by PERF joins. In the case where a semi-join is used twice, the semi-join will have a transmission cost of 0 the second time. The first time an overhead (backward transmission cost) is added.

Chapter V

Experimental Results

This chapter presents several tests done on the two algorithms discussed previously. Different programs were developed to implement the underlying methods and different test scenarios were considered.

The programs used for this thesis were adjusted to Bealor's programs [4].

-Create_Query: This program creates statistical information about the relations and attributes that participates in the query after all processing takes place.

While varying the number of parameters entered for this program, different queries are generated with the following characteristics:

- Each query consists of between 1 and 6 relations and the number of attributes varies between 2 and 4.
- The cardinality of each join attribute domain varies between 500 and 1500.
- Each relation has between 800-6000 tuples.
- Each relation has at least one other non-joining attribute that is required at the query site.

The query is generated in four steps:

a- Given the number of relations and the maximum number of join attributes, the cardinality of the domain for each join attribute is chosen randomly.

b- The occurrence of join attributes within each relation is determined randomly such that the desired connectivity is satisfied.

c- The cardinality of each join attribute is chosen randomly such that it does not exceed the cardinality for its associated domain.

d- The cardinality of each relation is chosen randomly such that the cardinality of the relation exceeds the cardinality of any of its join attributes.

The programs are as follows:

- PERF: This program implements the PERF algorithm using the statistical files and generating the related schedule.
- W: This program implements the W algorithm using the statistical files and generating the related schedule.
- PERFW: This program applies PERF concept to the W algorithm.

5.1 Experimental Scenarios

Different scenarios can be used in order to evaluate the performance of different algorithms.

Basic Module: Using different ***relation-attribute*** combinations, all algorithms are applied to the following cases:

2-2	*2-3*	*2-4*
3-2	*3-3*	*3-4*
4-2	*4-3*	*4-4*
5-2	*5-3*	*5-4*

Scenario 1: Consider the attribute width to be 1 byte for all attributes. then run the basic module for 1000-1500 times.

Scenario 2: Consider the attribute width to be 5 bytes for all attributes. then run the basic module for 1000-1500 times.

Scenario 3: Consider the attribute width to be 50 bytes for all attributes. then run the basic module for 1000-1500 times.

The same scenarios were repeated for different values of the constant network cost (the b variable in the linear cost function in the form of $y = ax + b$).

5.2- Data Simulation

The following data simulation are made based on the test run with b = 0.

5.2.1- Scenario 1:

TYPE	W (%)	WPERF (%)	WPERF – W (%)
2-2	29.79	33.24	3.45
2-3	43.88	47.98	4.11
2-4	56.18	60.63	4.45
3-2	30.63	32.64	2.04
3-3	41.67	44.35	2.69
3-4	52.36	55.32	2.96
4-2	41.45	42.31	0.86
4-3	47.14	48.64	1.50
4-4	55.35	57.12	1.77
5-2	51.74	51.99	0.25
5-3	54.63	55.37	0.74
5-4	60.08	61.14	1.05
TOT:	47.07	49.23	2.16

Table 5.1:Scenario 1- W Versus WPERF

The values in the second and third columns are percentages resulting from subtracting the corresponding algorithm cost from the unoptimized method cost and dividing the result by unoptimized method cost.

The first column in the table above represents a relation-attribute combination. The results in the first and second column show the performance of each algorithm with respect to an unoptimized method. For example, the second column is a result of subtracting W algorithm cost from the unoptimized method cost and dividing the result by unoptimized method cost. The whole result is multiplied by 100 to get the percentage. That is, ***(unoptimized method cost – W algorithm cost)100 / unoptimized method cost*** represents the results in the table above. The same applies for the third column with respect to WPERF algorithm. The fourth column is the difference between the third and second columns.

The WPERF algorithm outperformed the W algorithm by 2.16%. PERF concept applied to W algorithm always increased the performance; hence, reduced the time and number of bytes transferred across the network. This is due to the backward reduction feature that was introduced to the PERF concept. It is true that the PERF joins add an overhead to the forward phase but this overhead is negligible compared to the reduction gained for redundant transmissions.

Series1 = W AND Series2 = WPERF

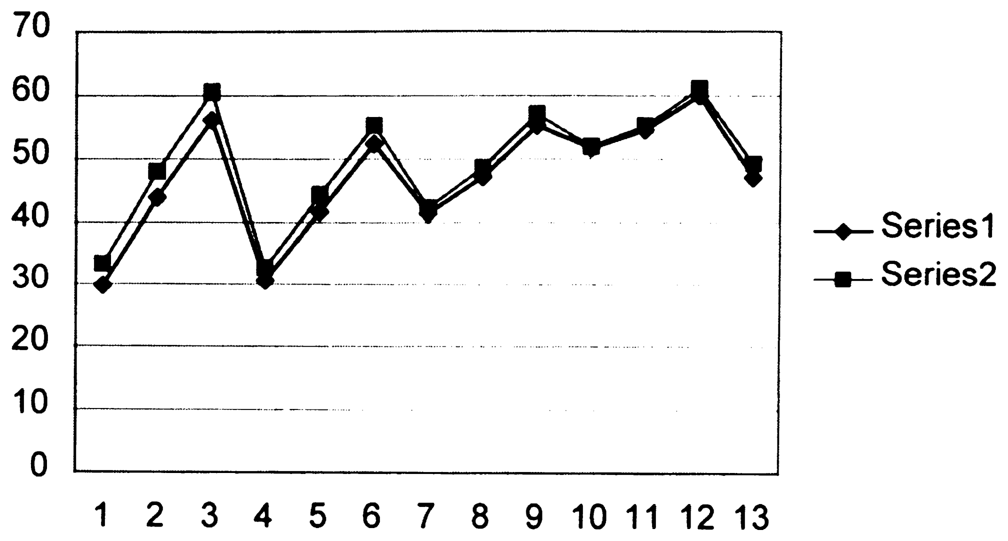

Figure 5.1: Scenario 1- W / WPERF Results

The figure above is a graphical representation of the data generated in the second and third column of table 3.

5.2.2-Scenario 2:

TYPE	W	WPERF	WPERF / W
2-2	27.56	31.12	3.56
2-3	42.31	46.41	4.10
2-4	55.25	59.74	4.49
3-2	28.62	30.74	2.12
3-3	40.63	43.27	2.64
3-4	52.15	55.10	2.94
4-2	40.35	41.17	0.81
4-3	45.54	47.13	1.59
4-4	54.95	56.80	1.85
5-2	50.85	51.16	0.31
5-3	55.11	55.87	0.76
5-4	61.46	62.48	1.02
TOT:	**46.23**	**48.41**	**2.18**

Table 5.2:Scenario 2- W Versus WPERF

The values in the second and third columns are percentages resulting from subtracting the corresponding algorithm cost from the unoptimized method cost and dividing the result by unoptimized method cost.

Series1 = W AND Series2 = WPERF

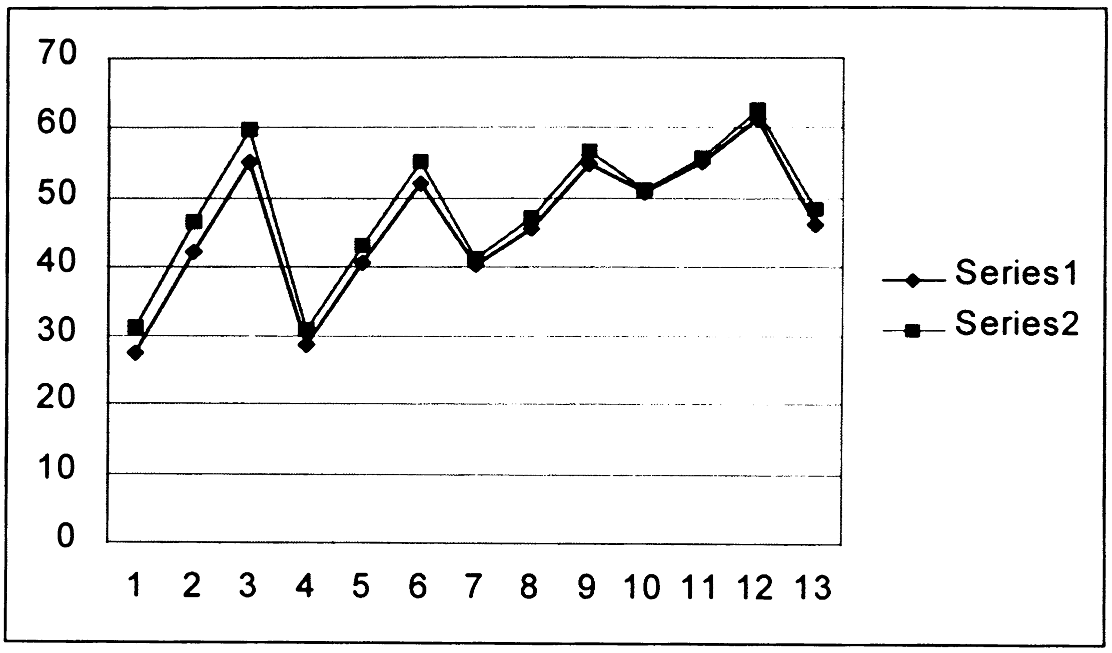

Figure 5.2: Scenario 2- W / WPERF Results

The figure above is a graphical representation of the data generated in the second and third column of table 4.

5.2.3- Scenario 3:

TYPE	W	WPERF	WPERF / W
2-2	28.24	31.81	3.57
2-3	42.73	46.75	4.02
2-4	57.23	61.60	4.37
3-2	28.78	30.85	2.07
3-3	41.67	44.42	2.75
3-4	52.03	54.94	2.92
4-2	40.87	41.68	0.82
4-3	46.10	47.56	1.45
4-4	54.76	56.61	1.85
5-2	51.48	51.76	0.28
5-3	54.42	55.23	0.81
5-4	60.96	61.96	1.00
TOT:	**46.60**	**48.76**	**2.16**

Table 5.3: Scenario 3- W Versus WPERF

The values in the second and third columns are percentages resulting from subtracting the corresponding algorithm cost from the unoptimized method cost and dividing the result by unoptimized method cost.

Series1 = W AND Series2 = WPERF

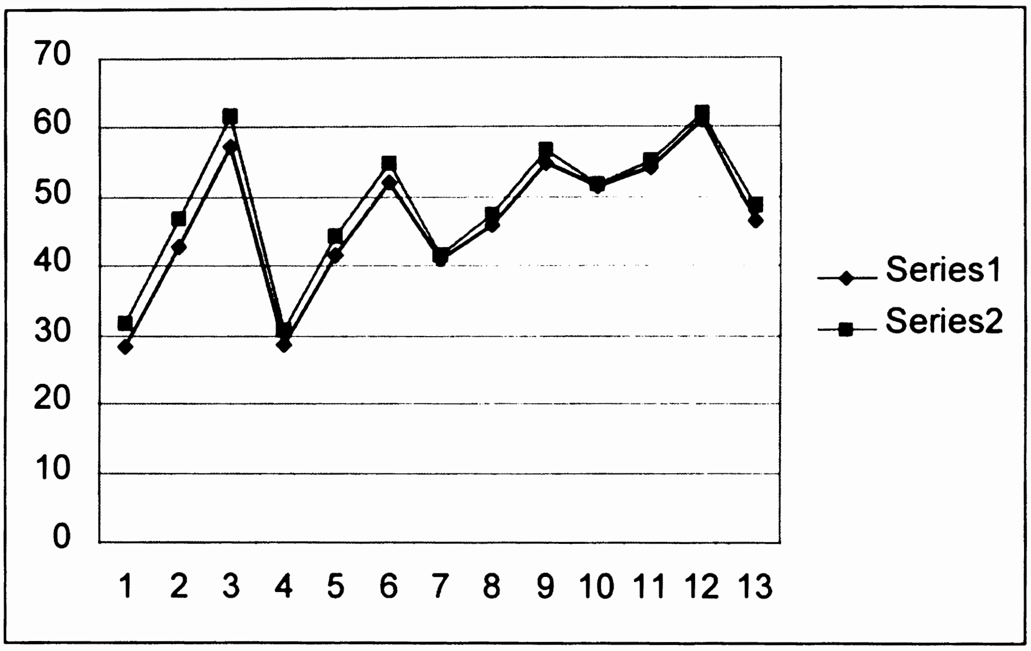

Figure 5.3: Scenario 3- W / WPERF Results

The figure above is a graphical representation of the data generated in the second and third column of table 5.

Three different scenarios were used in order to compare the performance of the W and WPERF algorithms. The WPERF algorithm has the best performance for a field of width 50 bytes. This is the expected result because of the overhead added by PERF to the backward phase. The PERF concept consists of returning a bit vector representing the matching tuples to the original site. The backward cost is more considerable when the original field is less than or equal to 1 byte because sometimes it might be more profitable not to send back this data. This backward cost becomes more negligible compared to the forward cost when having 50 byte width attributes.

Chapter VI

Summary, Conclusion, and Suggested Future Work

The query optimization problem has attracted much attention. Chapter one presents two methods that have been proven to promote the query optimization problem. Chapter two is a set of definitions, an overview of previous work done on query optimization, and a list of assumptions and notations. Chapter three introduces the W algorithm and the PERF concept. The last part of this chapter combines the W algorithm and the PERF concept to form the WPERF algorithm. It also presents a complexity comparison of both algorithms. In Chapter four, an example is presented and solved using the two algorithms: W and WPERF. Chapter five provides experimental results in the form of tables and graphs done on the W and the WPERF algorithms.

A PERF join algorithm is presented in details. Also, both concepts of semi-joins and PERF joins are explained, and then an optimization algorithm using semi-joins (W algorithm) is enhanced by applying PERF joins to it (WPERF).

Theoretically, the advantages of PERF joins over semi-joins mainly involve the removal of the cost associated with redundant transmissions by adding a relatively negligible cost to the backward phase of each PERF join.

Different series of experiments are conducted showing the efficiency of PERF joins from different perspectives. The study also considers the best case for which PERF joins performs at most. Based on this study, the use of PERF joins is recommended for huge and textual and graphical databases where the width of some join attributes is quite large, as well as for ordinary data.

However, based on the fact that during the query processing, data in the relations should not be updated without updating the PERF vector accordingly, PERF joins are viewed as the best solution for distributed query optimization that can be adapted for huge static warehouses where data is not changed very frequently.

In this thesis, one technique of many possible is studied thoroughly, but there are still many other strategies waiting for further study to prove their unique characteristics and advantages as well as their drawbacks.

We would like to see further work on this problem, mainly, applying the PERF concept to different query optimization algorithms. By doing so, new algorithms arise, hopefully, minimizing the cost of data transmission from one site to another.

BIBLIOGRAPHY

[1] Adiba, M., Chupin, J. C., Demolombe, R., Gardarin, G., and Bihan, J. L., Issues in distributed data base management systems: A technical overview. In Proceedings of the 4th International Conference on Very Large Data Bases, IEEE, (West Berlin, Sept. 1978), pp. 89 – 110.

[2] Apers, P.M.G., Hevner, A.R., Yao, S.B. Optimization Algorithms for Distributed Queries. IEEE Transactions on Software Engineering, Vol 9:1, 1983, pp. 57 - 68.

[3] Bealor, William, "Semi-join Strategies For Total Cost Minimization in Distributed Query Processing", Master's Thesis, University of Windsor Canada, 1995.

[4] Bealor, William, Distributed Query Optimization Using Semi-joins. Research Survey, University of Windsor, Windsor, Ontario, Canada, 1994.

[5] Begg, C. and Connolly, T., "Database Systems: A Practical Approach to Design, Implementation, and Management", Addison Wesley Longman Limited, United States, 1999.

[6] Bernstein, P.A., Goodman, N., Wong, E., Reeve, C.L, Rothnie, J., Query Processing in a System for Distributed Databases (SDD-1), ACM Transactions on Database Systems 6:4 (Dec 1981), pp. 602 - 625.

[7] Chatziantoniou, D. and Ross, K.A., "Groupwise Processing Of Relational Queries", Proceedings Of The 1997 Very Large DataBases Conference, The VLDB Journal, pp. 476 – 485, August 1997.

[8] Chaudhuri, S., An overview of query optimization in relational systems; Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1998, pages 34 – 43.

[9] El-Masri, Navathe, Fundamentals of Database Systems, Second Edition, Addison Wesley, United States, 1994.

[10] Graefe, G., Query Evaluation Techniques for Large Databases. In ACM Computing Surveys: Vol. 25, No 2., June 1993, pp. 73 – 169.

[11] Haraty, R. and Fany, R., Proceedings of the 2000 ACM Symposium on Applied Computing 2000 (Volume 1), 2000, pp. 284 - 288.

[12] Hevner, A. R., Wu, O., Q and Yao, S., B., Query Optimization on Local Area Networks. ACM Transactions on Office Information Vol.3, pages: 35 – 62, January 1985.

[13] Henver, A. R., Data Allocation and Retrieval in Distributed Systems. In Advances in Data Management, Vol. II. Wiley, New York, 1983, pp. 52 – 71.

[14] Karwin, Bill, InterBase Server Configuration and Optimization. Borland Developer's Conference, 1996.

[15] Lei, H. Kenneth Ross, "Faster Joins, Self-Joins and Multi-Way Joins Using Indices", International Workshop On Next Generation Information Technologies And Systems, IEEE, June 1997, pp. 23 – 39.

[16] Morrissey, J.M., Bandhyapadhyay, S. and Bealor, Todd, "A Heuristic For Minimizing Total Cost In Distributed Query Processing", Proceedings Of The 7th International Conference of Computing and Information, Journal of Computing and Information, July 1995, pp. 74 – 80.

[17] Mullins, Graig, Distributed Query Optimization, Technical Support Magazine, July 1996.

[18] Ono, K., Lohman, G.M. Measuring the Complexity of Join Enumeration in Query Optimization. In Proc. of Very Large Databases, Brisbane, Australia 1990.

[19] Seng, Jia-Long, Benchmarking Relational Database Systems – Overview, Taipei – Taiwan.

[20] Simmen, D., Shekita E., Malkemus T., Fundamental Techniques for Order Optimization. In Proc. of ACM Special Interest Group on Management of Data, Montreal, 1996.

[21] Ullman, J. D., "A First Course in Database Systems", Prentice Hall, April 1997.

[22] Ullman, J. D., "Principles of Database & Knowledge-Base Systems", Second Edition, Freeman, Vol. 1, W. H. Company, March 1988.

[23] Yannis, E. Ioannidis, Query Optimization, ACM Comput. Surv. 28, 1 (Mar. 1996), pp. 121 – 123.

[24] Yao, B. S., Optimization of Query Evaluation Algorithms, ACM Trans. Database Syst. 4, 2 (Jun. 1979), Pages 133-155

[25] Zhe Li, K. A. Ross, "PERF Join: An Alternative to Two-Way Semi-Join and Bloomjoin", Proceedings of the 1995 Conference on International Conference on Information and Knowledge Management, 1995, pp. 137 – 144.

[26] Zhe Li, K.A. Ross, Fast Joins Using Join Indices, VLDB Journal, Vol. 8, No. 1, 1999, pp. 1-24,

VITA 2

Raef Abdallah

Candidate for the Degree of

Master of Science

Thesis: INTRODUCING PERF JOINS TO A QUERY OPTIMIZATION ALGORITHM

Major Field: Computer Science

Biographical:

Education:

Received Bachelor of Science degree in Computer Science from the Lebanese American University, Beirut, Lebanon in February 1997.
Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December 2002.

Experience:

Employed as a computer instructor at CompuBase and the American Language Center in Beirut, Lebanon, 1996 – 1997.
Employed as a programmer at Oklahoma Department of Mental Health and Substance Abuse Services, Oklahoma City, Oklahoma, 1999 – 2000.
Employed as a telecommunication engineer; Inet Technologies, Inc., Richardson, Texas, 2000 to present.

VITA 2

Raef Abdallah

Candidate for the Degree of

Master of Science

Thesis: INTRODUCING PERF JOINS TO A QUERY OPTIMIZATION ALGORITHM

Major Field: Computer Science

Biographical:

Education:

Received Bachelor of Science degree in Computer Science from the Lebanese American University, Beirut, Lebanon in February 1997.
Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December 2002.

Experience:

Employed as a computer instructor at CompuBase and the American Language Center in Beirut, Lebanon, 1996 – 1997.
Employed as a programmer at Oklahoma Department of Mental Health and Substance Abuse Services, Oklahoma City, Oklahoma, 1999 – 2000.
Employed as a telecommunication engineer; Inet Technologies, Inc., Richardson, Texas, 2000 to present.