Database Mining in the Architecture of a Semantic Pre-processor for State-Aware Query Optimization

Sarabjot S. Anand David A. Bell John G. Hughes
Department of Information Systems
Faculty of Informatics
University of Ulster (Jordanstown)
Northern Ireland
E-Mail : ssanand,dbell,jgh@uk.ac.ulster.ujvax

Abstract

Most applications of in database mining at present seem to be concentrated around the discovery of knowledge for application-oriented decision support. In this paper we describe a tool for enhancing the performance of the database management system itself using semantics captured by database mining techniques. We investigate the use of database mining in supporting *state-aware* query optimization where improved performance is sought by dynamically increasing the utilization of available resources.

We consider, in particular, the use of semantic knowledge in enhancing the system performance and the utilization of specialized hardware such as ICL's SCAFS by semantic query reformulation. The idea is to discover semantic information about the data stored in the database using database mining techniques and use it to reformulate queries to make better use of *available resources* at the time of execution of the query. Thus the execution strategy of the query is based on the current state of the system (and so 'state-aware'). To achieve this we create a *rewrite system*, consisting of rules and equations mined from the database, that rewrites queries made to the database for *state-aware* execution.

We provide an architecture for this query pre-processor and discuss performance considerations involved in its implementation.

Keywords : Database Mining, Semantic Query Reformulation, Rewrite Systems, Associative Memories, Specialized Database Hardware, Resource Utilization.

1. Introduction

Developers of large database applications are often faced with performance bottlenecks and are required to develop new, innovative methods for dealing with them. In this paper we present the architecture of a query pre-processor that employs database mining techniques to help distribute the query load over the available resources ('state-aware' query optimization) reducing problems of resource boundedness.

Database mining techniques can be employed for achieving *state-aware* Query Optimization for three separate purposes

- Detection of resources being under-utilised
- Formulation of heuristics affecting the utilisation of the resource
- Discovery of rules from data required to reformulate queries to increase utilisation of the resource

For the first two purposes, i.e. the detection of under-utilised resources and the formulation of heuristics about the operation of the resource, database mining algorithms need to be used for detecting *association patterns* in data about queries executed, their associated statistics and their execution plans.

We concentrate in this paper on the third purpose - we investigate the possible role of database mining to discover rules from data required to reformulate queries. This is done by using the STRIP algorithm [ANAN93b]. The STRIP algorithm can be employed in two different ways

- Discovery of all rules implicitly present in the database, independently of what is required for query reformulation
- Goal directed discovery of rules

In section 5 we describe how the STRIP mining algorithm can be employed for the discovery of rules independently of the query reformulation while section 6 describes the role of STRIP in the goal-directed discovery of rules.

The motivation for the work presented in this paper came from performance related work undertaken by the authors at the Northern Ireland Housing Executive (NIHE). The NIHE property management database system is one of the largest relational database systems under development in Europe. The database is approximately 20 GB in size, distributed over five regions of the housing executive. It has an expected work load of 3700 on-line transactions per region per day and 400 on-line users. It is implemented on 11 ICL's DRS6000 machines using INGRES/Star.

The size of the NIHE database application presented the developers with a number of performance problems and the need for innovative solutions was realised. This lead to the authors undertaking an investigation of the Ingres Search Accelerator developed by ICL in collaboration with Ingres Corp. [INGR]. The heart of the Ingres Search Accelerator is SCAFS, the current generation of ICL's Content Addressable File Store (CAFS) [BABB79, BABB85,COUL72,LEUN85]. SCAFS is a backend filter database machine that can perform associative searches[1] on relational database tables. The advantages of the Ingres Search Accelerator are :

- it off-loads the CPU which would otherwise be using most of its time in trivial comparisons
- it reduces the I/O traffic as only those tuples that satisfy the query are passed into main memory

Initial studies performed on SCAFS [ANAN93a] for the NIHE application showed that the Ingres Query Optimizer (IQO) utilises SCAFS to a bare minimum. The conclusion was that in its present state SCAFS would not be of much help to this application. This led to a search for methods that could make SCAFS 'usable' [ANAN93] for the NIHE application in particular, and large database applications with high transaction rates in general.

The study of SCAFS [ANAN93a] resulted in a set of heuristics that could predict when the IQO would decide on using SCAFS and when it would decide against it. This suggested that the way forward was to design a 'Query Pre-processor' that could transform the query input into a form that would make the IQO decide to use SCAFS.

Reformulation of queries using knowledge has been studied in great detail in the topic of Semantic Query Optimization (SQO) [KING81,HAMM80,BERT92,SIEG92,AN91]. The pre-processor proposed here is different from SQO because it has a different goal. While SQO is aimed at reducing the cost of query processing, the semantic query pre-processor transforms queries to make better use of the available resources. For example, if the CPU is very busy, reformulating a query to make it use SCAFS may be more beneficial for the system as a whole even if the response time for this query would be greater than the response time of the original query. In this sense the optimization is 'state-aware'. So the overall workload execution is enhanced although the response time of the reformulated query may be greater than that for the original query. Thus, the goal of our query pre-processor is to optimise the utilization and throughput performance indices [FERR81] rather than the response time which is often the metric of interest in performance studies. For a transaction intensive application like that of the housing executive, optimising the throughput and utilization of resources are of primary importance.

The rules discovered from databases using database mining [ANAN93b,PIAT91,PIAT93,AGRA93a, AGRA93b] can be considered to form rules of a rewrite system [DERS90]. In our case the terms of the rewrite system would be the search conditions in the query (see section 3 for a detailed discussion on the rewrite system used in our query pre-processor). Representing transformation heuristics in Semantic Query Optimization as rewrite rules has been

[1]The basic idea of an associative search is to apply some hardware search logic directly to the data as it is read from disk , with the intention of reducing the amount of data being to be transmitted across the channel to the host computer , thereby not only reducing communication costs but also reducing the amount of processing to be carried out by the processor .

studied by Chakravarthy et. al. [CHAK84]. Rewrite systems have also been used in the context of representing algebraic transformations[2] in Rule Based Query Optimization [GRAE87]. The Semantic Query Pre-processor (SQP) regards the rules discovered from the database as rules in a rewrite system.

The rest of the paper is in the following format. First we have two review sections for our system application. In the next section we give a review of work being done in semantic query optimization. In particular we focus on heuristics suggested by researchers in SQO. Section 3 reviews rewrite systems and introduces the rewrite system required for the Semantic Query Pre-processor introduced in this paper. Then we focus on how the database mining itself works. First in section 4 we present a set of heuristics that are used by the Semantic Query Pre-processor as domain knowledge to guide the search for knowledge by database mining algorithms for query reformulation. These heuristics are the result of studies performed earlier on SCAFS by the authors [ANAN93a]. These heuristics are cross-referenced to the well known heuristics in section 2. Section 5 describes the architecture of the semantic query pre-processor while section 6 discusses some of the performance related issues that need to be considered when implementing the query pre-processor. This includes a discussion on how goal-directed discovery of rules using database mining techniques can be used in the query pre-processor.

2. Semantic Query Optimization

King [KING79,KING81] and, independently, Hammer et. al. [HAMM80] introduced a technique of query optimization that used knowledge about the data stored in the database, i.e. domain dependent strategies, for query processing. This technique came to be known as Semantic Query Optimization (SQO). SQO uses knowledge about the data stored in the database to transform queries into other queries that are semantically equivalent i.e. that produce the same answer as the original queries but have a lower execution cost. The knowledge on which the transformations are based are the semantic integrity constraints of the database. Apart from integrity constraints, Yu et. al [YU89] suggested the use of *Dynamic Constraints*. These are rules that are frequently satisfied by the database but need not always be true. These constraints would need to be updated when updates are made to the database as an update may affect the 'truth-value' of the constraint. These Semantic Query Optimizers have the advantage of being relatively easy to change. New domain facts can be easily added as can new optimization techniques.

Semantic Query Optimization uses two types of knowledge :

- Semantic Integrity and Dynamic Constraints or *Transformation Knowledge*, used to transform given queries into semantically equivalent queries.
- Heuristics from Conventional Query Optimization or *Domain Knowledge* that help to guide the transformations.

King [KING81] provides a set of heuristics for controlling the transformations of queries. The heuristics are sub-divided into two groups :

- Heuristics for considering a transformation and
- Heuristics for rejecting a transformation

2.1 Heuristics for considering transformations

For a given query there are possibly a large number of candidate semantic transformations available in the rewrite system. Thus, a method of constraining the number of candidate semantic transformations is required. The heuristics provided in this section define a useful semantic transformation i.e. a transformation that would reduce the cost of execution of the query.

2.1.1 Index Introduction : If a table involved in the query is indexed on an attribute that is not constrained in the query, a constraint involving the indexed attribute should be introduced, if possible i.e. if it can keep semantic equivalence.

[2] An algebraic transformation transforms a query to an algebraically equivalent query. Algebraically equivalent queries return the same answer for any database state

For example, consider the following query

```
select surname,address
from employees              ..... Q1
where income > 10000
```

Suppose the table *employees* is indexed on the field *emp_no* and we have the following *Transformation Knowledge* :

if income > 9000 then emp_no > 300.

The query may be transformed to

```
select surname,address
from employees          ..... Q2
where income > 10000
and emp_no > 300
```

This query can now utilize the index on *emp_no* and therefore has a lower execution cost.

2.1.2 Join Introduction : If a relation not involved in the query has a clustering link[3] into a much larger file that is involved in the query, the relation should be added to the query even though this introduces an extra join.

For example, consider the following SQL query

```
select surname,address
from employees    ..... Q3
where income > 10000
```

If there exists a clustering link from table *emperks* into table *employees* and *emperks* is much smaller than *employees*, the following query will have a lower execution cost even though a join has been introduced.

```
select employees.surname,employees.address
from employees,emperks
where employees.income > 10000            .... Q4
and employees.emp_no = emperks.emp_no
```

2.1.3 Scan Reduction : If one of the tables involved in a join is unrestricted while the other table is highly restricted, restrictions should be added, if possible, on the unrestricted table.

For example, consider the SQL query

```
select e.surname,e1.car
from employees e,emperks e1     ..... Q5
where e.income > 10000
and e.emp_no = e1.emp_no
```

If we have the following *Transformation Knowledge*

if income > 10000 then house_allowance = 10

The following query is semantically equivalent to the query above

```
select e.surname,e1.car
from employees e,emperks e1
where e.income > 10000       ..... Q6
and e1.house_allowance = 10
and e.emp_no = e1.emp_no
```

2.1.4 Join elimination : If a table is involved in a join but none of it's attributes are in the output list of the query, transformation to eliminate the table from the query should be applied.

For example, consider the SQL query below :

```
select e.surname,e.address
from employees e,emperks e1
where e.income > 10000    ..... Q7
```

[3] A clustering link / index from table 1 to table 2 is a link from a unique key of table 1 to a non-unique field (foreign key) of table 2. Normally table 2 is larger than table 1.

```
            and e.emp_no = e1.emp_no
```
The following SQL query is semantically equivalent and has no join and therefore has a lower execution cost :

```
            select surname,address
            from employees          ..... Q8
            where e.income > 10000
```
(Note : The query Q7 would have a lower execution cost if there was a clustering link between the tables employees and emperks)

2.2 Heuristics for rejecting a transformation

Given a particular query, the heuristics provided in this section define semantic transformations that would not be useful in the attainment of a lower cost of execution of the query. Such transformations should not be applied to the query.

2.2.1 Do not introduce unlinked joins

The example given in 2.1.2 would not be a cost reducing transformation if there was no clustering link from *emperks* into *employees.*

2.2.2 Do not remove tables with highly restricted attributes which have clustering links on them. In the example in section 2.1.4 if there existed a clustering link from *emperks* and *employees*, the transformation from Q7 to Q8 would increase the execution cost and should therefore not be performed.

2.2.3 A relation highly restricted on an attribute with a clustered index should not be transformed.

```
            select surname,address
            from employees          .... Q8
            where income > 10000
            and emp_no > 300
```
If we have the following *Transformation Knowledge*

if income > 10000 then emp_no > 300

A transformation of Q9 to the query below should be rejected if there is a clustering index on *emp_no.*

```
            select surname,address
            from employees          ..... Q9
            where income > 10000
```

The Semantic Query Pre-processor proposed in this paper differs from query optimizers (both conventional and semantic) in that it takes into account the *present state* of the running database system. Consider a large database application with a large workload. Use of Conventional and Semantic Query Optimizers can reduce the cost of processing a query, but what happens when the CPU cannot handle any more queries or the I/O channels are too busy to support any new queries ? The proposed SQP takes these environmental limitations into account and using techniques borrowed from SQO reformulates the query to be more suitable for processing using specialized hardware. In the environment of our application the specialized hardware is SCAFS. Using it increasing the load on the I/O channels and CPU only marginally allowing a larger number of queries to be processed concurrently. The *Transformation Knowledge* in the SQP is the same as that in the SQO but the SQP differs in its *Domain Knowledge*. The *Domain Knowledge* has the same role, i.e. to guide the query transformation, but it differs due to the change in goal. The goal of the SQP is not reducing the cost of processing or reducing response time but to make the query more suitable for execution using SCAFS i.e. making more efficient use of available resources.

The *Domain Knowledge* for the SQP consists of a set of heuristics arrived at by a previous study of SCAFS undertaken by the authors [ANAN93a]. At present work is being undertaken in the authors' laboratory to find ways of discovering these heuristics themselves automatically using database mining techniques.

3. Rewrite Systems

In this section we first give some general definitions of rewrite systems along with a brief review of research work in progress in this field. We then define the rewrite system used by us for our semantic pre-processor.

Rewrite Systems [DERS90] are a set of directed equations used to compute the simplest form possible for a formula by repeatedly replacing sub-terms of the given formula with equal terms. The final result of an unextendible sequence of rule applications is called a *normal form.*

In rewrite systems we have a set of terms called *constructor terms* built using *constructors.* Constructors are the set of irreducible symbols. The objective of a rewrite system is to replace sub-terms consisting of non-constructor or *selector* terms with constructor sub-terms. This process is repeated until a constructor term is obtained. A system is said to be *sufficiently complete* if according to the semantics, every term is equal to (i.e. can be rewritten as) a term built only from constructors.

An *equation* is an unordered pair of terms $\{s,t\}$ in $\mathcal{T}$ which is denoted by $s \leftrightarrow t$. A term 'l' in $\mathcal{T}$ rewrites to 'r' in $\mathcal{T}$, $l \leftrightarrow_E r$, if l has a sub-term that is an instance of one side of an equation in E and r is the result of replacing that sub-term with the corresponding instance of the other side of the equation.

Rewrite rules impose directionality on the use of equations in proofs. A *rule* over a set of terms, $\mathcal{T}$, is an ordered pair $<l,r>$ of terms, which we write as $l \rightarrow r$. Like equations, rules are used to replace instances of 'l' by corresponding instances of the right-hand side 'r'. A set of such rules form a term-rewriting system. For completeness we restate the formal definition of a rewrite given in [DERS90] :

Defn : For a given term-rewrite system, R, a term s in $\mathcal{T}$ rewrites to a term t in $\mathcal{T}$, written $s \rightarrow_R t$ if
$s|_p = l\sigma$ and $t = s[r\sigma]_p$ for some rule $l \rightarrow r$ in R, position p in s and substitution σ.

A *substitution,* σ, is a special kind of replacement operation uniquely defined by a mapping from variables to terms written as $\{x_1 \rightarrow s_1, x_2 \rightarrow s_2, \ldots, x_m \rightarrow s_m\}$. A term t in $\mathcal{T}$ matches a term s in $\mathcal{T}$ if $s\sigma = t$ for some substitution σ. t is said to be an *instance* of s or we say that s *subsumes* t.

The sub-term, $s|_p$, at which the rewrite can take place is called the *redex.* If no redex exists the term s, is said to be *irreducible* or in normal form.

Let us now consider how we might use this technique in our query pre-processor. Consider a relation $\mathcal{R}$ with attribute set $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$. In accordance with a set of heuristics for pre-processing which will be given in the next section, the attribute set $\mathcal{A}$ can be partitioned into two sets $\mathcal{C}$ and $\mathcal{S}$, where $\mathcal{C}$ is the set of attributes in $\mathcal{A}$ that can be searched using SCAFS and $\mathcal{S}$ is the set of attributes in $\mathcal{A}$ that cannot be searched using SCAFS (i.e. the Ingres query optimizer would prefer not to use SCAFS for searching on these attributes). Therefore, for a query to be processed using SCAFS the search condition of the query must only consist of terms involving attributes in $\mathcal{C}$. Terms involving attributes in $\mathcal{C}$ are called *constructor* terms while those involving attributes in $\mathcal{A}$ are called *selector* terms. Thus, the goal of the rewrite system is to rewrite query search conditions that include terms involving attributes in $\mathcal{S}$ into query search conditions with terms involving only attributes in $\mathcal{C}$.

The rewrite system above consists of equations and rules. In the case of relational databases, equations would represent *partial identity dependencies* while rules would define *partial functional dependencies.* Bell [BELL93] defines a *functional dependency* as a statement denoted $X \rightarrow Y$, where both X and Y are sets of attributes in relation $\Re$, which holds in $\Re$ if and only if, for every pair of tuples, u and v, if $u[X] = u[Y]$ then $v[X] = v[Y]$. Also, an *identity dependency,* denoted $X \leftrightarrow Y$, holds when $X \rightarrow Y$ and $Y \rightarrow X$. We define a *partial functional dependency* as a functional dependency that is not true for the whole domain of X i.e. for some values of X there is a unique value for Y. Associated with a partial functional dependency is a *dependency measure*. The dependency measure, d_m, is calculated as :

$$d_m = \frac{\text{(no. of unique values of X that have a unique value of Y associated with it)}}{\text{(total number of unique values of X).}}$$

We say that a *partial identity dependency* exists between X and Y if for certain values of X there is a unique value of Y and for those values of Y there are unique values of X. As in the case of a partial functional dependency, a partial identity dependency has a dependency measure associated with it, which is measured as :

$$d_m = \frac{\text{(no. of unique values of X that have a unique value of Y associated with it)}}{\text{(total number of unique values of X).}}$$

$$= \frac{\text{(no. of unique values of Y that have a unique value of X associated with it)}}{\text{(total number of unique values of Y).}}$$

Clearly, for a functional dependency or an identity dependency, the value of d_m is 1. Thus the dependency measure indicates as to how close the partial functional or identity dependency between X and Y is to a functional or identity dependency between the two fields.

The equations and rules required for the rewrite system can be automatically generated using algorithms developed in the field of database mining like STRIP [ANAN93b] and KID3 [PIAT91]. We use the STRIP algorithm.

4. Domain Knowledge for the SQP

Tests carried out on SCAFS [ANAN93a] for the NIHE application revealed that the Ingres Query Optimizer was not using SCAFS to its full potential. The use of primary and secondary access methods is preferred most of the time to using SCAFS. The Ingres Search Accelerator only employs SCAFS after it has decided to scan the whole table being queried without taking into account the advantages to be gained by using SCAFS. Thus, we came to the conclusion that the Ingres Search Accelerator and the Ingres Query Optimizer are very loosely coupled and some method needed to be developed to bridge this gap between them.

We, therefore, sought a set of heuristics which would indicate situations under which SCAFS would be used for a particular query. These heuristics are used to guide the database mining algorithm in its search for knowledge which could be used for query reformulation. The study of SCAFS resulted in the following heuristics :

H.1 If the search condition in the query contains a restriction on a primary attribute of the relation SCAFS is no used.

H.2 If the search condition contains a restriction on an attribute with a secondary access method, SCAFS is not used unless the expected number of tuples satisfying the query is greater than approx. s% of the relation size where,

$s = 13.83$	for hash file organization
$s = 9.75$	for ISAM file organization
$s = 8.74$	for BTree file organization

H.3 If the search condition contains an attribute with an associated access method but the access method is not usable due to the nature of the predicate, SCAFS is used (e.g. a condition of the type account_no < 100 where the relation is hashed on account_no cannot use the hash function for the search).

H.4 If the search condition has a disjunction in it, SCAFS is always used.

Keeping the above heuristics in mind we shall now discuss the applicability of the heuristics for semantic query optimization given by King [KING81] as listed in section 2.1.

A. Index Introduction : Clearly from the heuristics H.1 - H.4 introducing indexes in the query is not recommended as this would ensure the use of the index rather than SCAFS. Infact, rather than index introduction, for the SQP *Index Elimination* is recommended.

B. Join Introduction : The reason for introducing new joins in SQO is to take advantage of indexes (see 2.1.2). Therefore, Join Introduction is not useful for the SQP.

C. Scan Reduction : Scan Reduction is useful for the SQP as the greater the restrictions on the tables the greater is the advantage of using SCAFS. The join operation has to be performed in the host machine itself and so by restricting relations more we are reducing the number of tuples being sent to the host machine, by SCAFS, for the join operation.

D. Join Elimination : Join elimination is useful for the SQP as a lower number of join operation implies less work for the host machine. Therefore if joins can be eliminated they should be.

5. The Semantic Query Pre-processor and its Usage

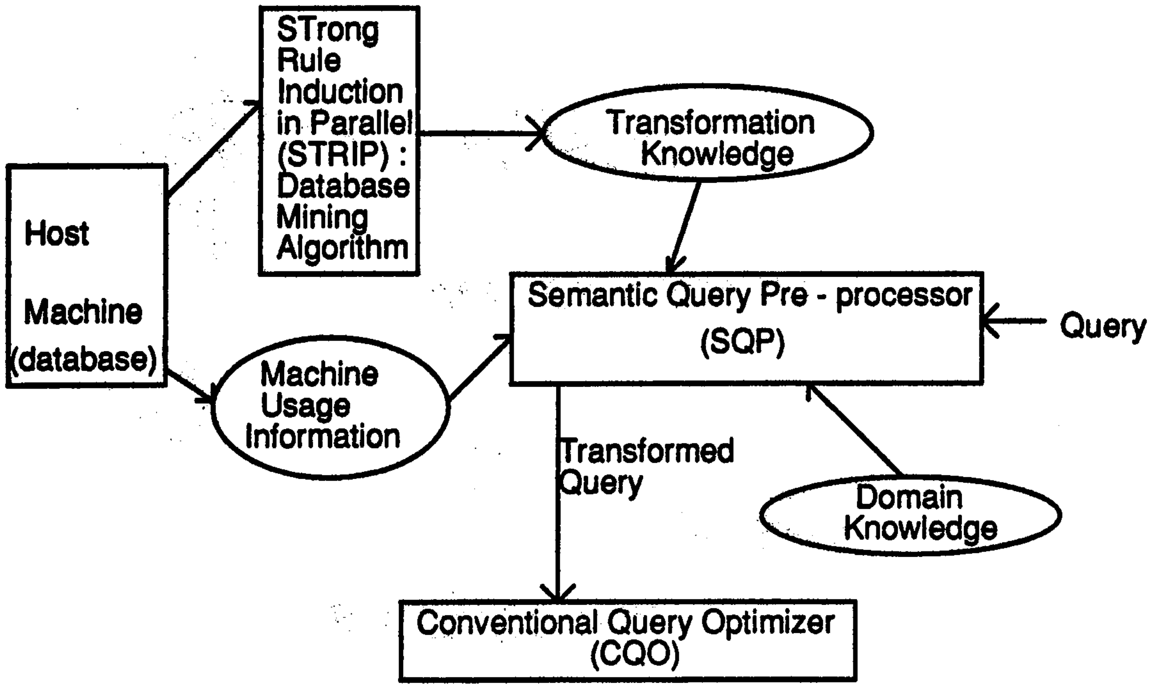
In section 4, we enumerated the Domain Knowledge that would be used by the Semantic Query Pre-processor. As can be seen from the heuristics above, the SQP uses some of the heuristics used by Semantic Query Optimizers. Thus, the SQP performs a certain amount of query optimization though this is more of a by-product rather than the main goal. The main objective of the SQP is to use SCAFS where required. The basic idea behind the SQP is given in Table. 1.

```
if (SCAFS is not busy)
{
    if (CPU is busy)
      or (I/O traffic is heavy)
      or(cost_of_query(SCAFS) - cost_of_query(¬SCAFS) < threshold)
      {
            Reformulate Query using Domain Knowledge
            and Transformation Knowledge to use SCAFS
      }
    else
      {
            Reformulate Query using Domain Knowledge
            and Transformation Knowledge to reduce the
            cost of execution of Query
      }
}
```

Table 1 : The Semantic Query Pre-processor (SQP)

Figure 1 shows the role of the SQP. It acts as a filter between the query input and the conventional query optimizer.

The *Transformation Knowledge* used by the SQP consists of two types of knowledge : *Static* (Integrity Constraints) and *Dynamic* (knowledge about the data stored in the database that is frequently true but not always true) Knowledge. The *STRIP* database mining algorithm working independently of the query pre-processor discovers equations and rules that form the *Dynamic Transformation Knowledge* of the pre-processor (we discuss the goal-directed discovery role of database mining in the next section). These equations and rules need to be updated along with any updates made to the data in the database. To reduce this overhead of updating the *Dynamic Knowledge*, the size of the knowledge base may be restricted to a certain predefined size using a method similar to that used by Siegel et. al [SIEG92] for discovering rules for SQO. We define our method in the next section. The *Transformation Knowledge* forms the rewrite system that rewrites queries as described in section 3 above.


STrong Rule Induction in Parallel (STRIP) : Database Mining Algorithm
Transformation Knowledge
Host Machine (database)
Semantic Query Pre - processor (SQP)
Query
Machine Usage Information
Transformed Query
Domain Knowledge
Conventional Query Optimizer (CQO)

- fig. 1 : The role of Database Mining in the Semantic Query Pre-processor (SQP)

The *Domain Knowledge* used by the pre-processor has already been described in section 4 as a set of heuristics used to guide the semantic transformations (The role of the STRIP algorithm in goal-directed database mining is discussed in the next section).

The *Machine Usage Information* is provided by the UNIX system activity report (sar) utility which is a standard UNIX performance monitoring utility. The estimation of query costs is carried out using a method similar to that given by King [KING81].

6. Performance Considerations

In this section we will discuss how the performance of the SQP can be improved i.e. how the cost of the SQP can be reduced. The main idea is to restrict the size of the transformation knowledge and use database mining techniques in a goal-directed rule discovery role (see fig. 2).

The only variable in the architecture of the SQP is the size of the transformation knowledge. There is a trade-off between the cost of the SQP and the completeness of the transformation knowledge. The effect of the size of the transformation knowledge on the cost of the SQP is two fold :

- The larger the size of the transformation knowledge base, the greater is the cost of reformulation of queries as there would be a larger set of candidate transformations for any given query.
- The larger the transformation knowledge base, the greater is the cost of updating the dynamic transformation knowledge whenever updates are made to the database.

Thus the cost of the SQP is directly proportional to the size of the knowledge base and restricting the size of the knowledge base can be beneficial from a performance viewpoint.

We concentrate only on the transaction queries here and ignore MIS queries, as they use SCAFS anyway.

The transaction queries are predefined and are known at the design stage of the application. Thus, the static transformation knowledge (Integrity Constraints, also known at the design stage) can be applied to these queries at the implementation stage of the application and the two versions of the query stored. As the static transformation knowledge never changes, these copies of the queries will always be semantically equivalent. Thus, we are left to deal only with the dynamic transformation knowledge and a set of partially transformed queries.

Now suppose we have to limit the size of the dynamic transformation knowledge to k rules / equations due to performance considerations. Using the STRIP database mining algorithm [ANAN93b] we can generate from the database, as above, an initial random set of k rules / equations. To each of these rules we associate an *interest measure*. Various formulae for interest measure of rules have been suggested in Database Mining literature [PIAT91,PIAT93] but we reckon that the measure given by Siegel et. al. [SIEG92] is the most appropriate for the SQP. Siegel et. al. suggested a measure based on database usage patterns. During a transformation if a particular rule is not found in the transformation knowledge base, that rule is considered to be a *candidate rule* and gets an interest measure associated with it as well. Whenever the interest measure associated with any of the candidate rules becomes greater than the interest measure of any of the rules in the transformation knowledge base, STRIP is invoked to verify the candidate rule which, if verified, replaces the rule with a lower interest measure in the transformation knowledge base.

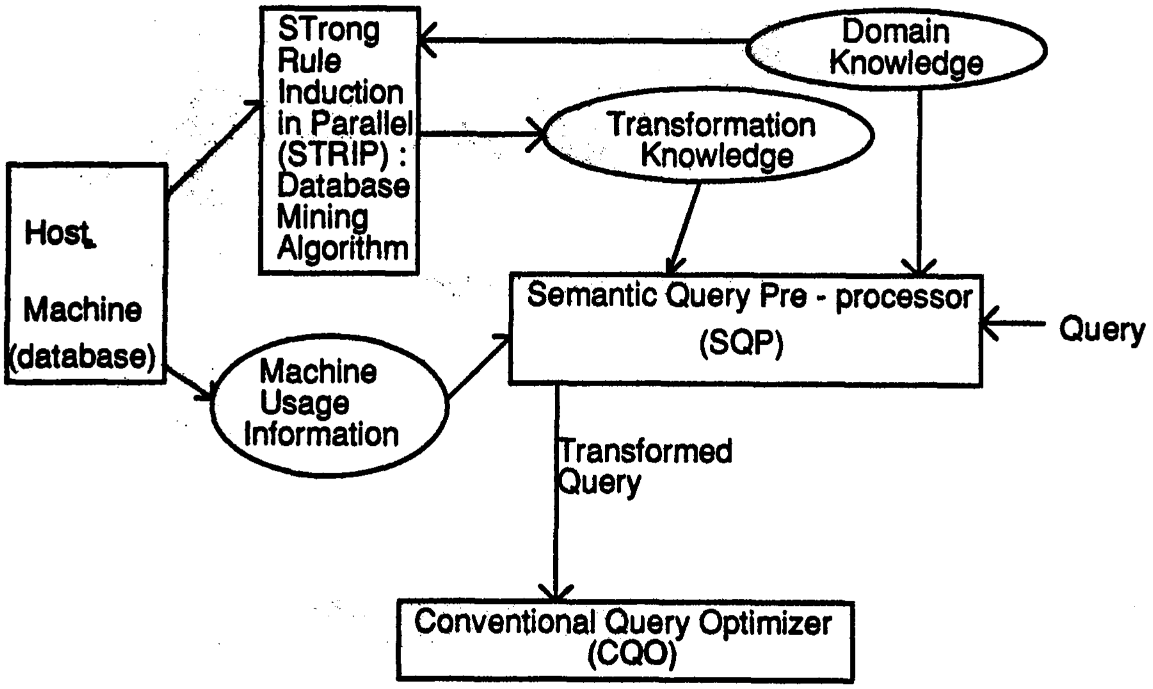

STrong Rule Induction in Parallel (STRIP) : Database Mining Algorithm
Domain Knowledge
Transformation Knowledge
Host Machine (database)
Semantic Query Pre - processor (SQP)
Query
Machine Usage Information
Transformed Query
Conventional Query Optimizer (CQO)

fig. 2 : The role of Database Mining in the SQP as a goal-directed rule discoverer

The formula for interest measure used in the SQP is based on the scheduling method used by SCAFS [ANAN93a]. In this formula, apart from database usage patterns, the *inactivity* of rules present in the transformation knowledge base as well as candidate rules affects the interest measure of the rule. The interest measure of a rule diminishes with the amount of time that it has been inactive. This rule measure would be more accurate for the SQP than the one given by Siegel et. al., as the following example will show :
Consider a rule, R1, in the transformation knowledge base that is used 200 times a day within an hour. Then our

interest measure would replace this rule by another rule, R2, once R1 has been inactive for a while, as the interest measure of R1 will have diminished. This would not be the case if we use the formula provided by Siegel et. al.

The database usage patterns and inactivity of rules information is stored as domain knowledge and forms part of the input to the STRIP database mining algorithm changing its role to a goal-directed rule discovery algorithm (see fig. 2).

7. Conclusion and Future Work

In this paper we have shown how database mining can be used in a system performance problem as opposed to its usual application usage. Optimization of utilization of resources and throughput is carried out by transforming queries into semantically equivalent queries that use resources whose availability varies. We have discussed how database mining techniques form an integral part of the architecture of such a query pre-processor in obtaining rules for query reformulation, either independent of the goal or goal-directed. We believe that the use of the Semantic Query Pre-processor in large, transaction intensive database management systems promises additional performance enhancement, particularly if we have a possibility of under-utilization of some valuable resources.

As the heuristics for the utilisation of SCAFS were known to the authors from a previous study of SCAFS [ANAN93a], we have not used database mining techniques for discovering the heuristics in the presented work. However we are currently studying methods of confirming the heuristics induced during the empirical study of SCAFS [ANAN93a] using database mining techniques.

8. References

[AGRA93a] Agrawal, R., Imielinski, T., Swami, A. Database Mining : A Performance Perspective IEEE Transactions on Knowledge and Data Engineering, Special Issue on Learning and Discovery in Knowledge - Based Systems, Vol. 5, No. 6, Dec. 1993.

[AGRA93b] Agrawal, R., Imielinski, T., Swami, A. Mining Associations between Sets of Items in Massive Databases ACM SIGMOD, Washington DC., May 1993.

[AN91] An, H. C., Henschen, L. J. Knowledge - Based Semantic Query Optimization Lecture Notes in AI, 1991, Vol. 542, Pg. 82-91.

[ANAN93] S. S. Anand, D. A. Bell, J. G. Hughes Using Semantic Knowledge to Enhance the Performance of Specialized Database Hardware Workshop of the FSTTCS'93 Conference, IIT Bombay, December, 1993.

[ANAN93a] Anand, S. S., Hughes, J. G., Bell, D. A. An Empirical Study of ICL's SCAFS Internal Report of the Department of Information Systems, University of Ulster at Jordanstown

[ANAN93b] Anand, S. S, Bell, D. A, Hughes, J. G. An Approach to Database Mining Using Evidential Reasoning, Internal Report of the Department of Information Systems, University of Ulster at Jordanstown

[BABB79] Babb,Ed Implementing a Relational Database by means of Specialised Hardware ACM transactions of database systems, Vol. 4, No. 1, March 1979.

[BABB85] Babb,Ed CAFS file - correlation unit ICL Technical Journal, Nov. 1985.

[BELL93] Bell, D.A. From Data Properties to Evidence, IEEE Transactions on Knowledge and Data Engineering, Special Issue on Learning and Discovery in Knowledge - Based Databases, Vol. 5, No. 6, Dec. 1993

[BERT92] Bertino, E., Musto, D. Query Optimization by using Knowledge About Data Semantics Data & Knowledge Engineering 9, 1992, Pg. 121 - 155.

[CHAK84] Chakravarthy, U., Fishman, D., Minker, J. Semantic Query Optimization in expert systems and database systems Proceedings from the 1st International Workshop on Expert Database Systems ed. L. Kerschberg Pg. 659 - 674.

[COUL72] Coularis,G.F., Evans,J.M., Mitchell,R.W. Towards content addressing in databases The Computer Journal, Vol. 15, No. 2, 1972.

[DERS90] Dershowitz, N., Jouannaud, J. P. Rewrite Systems Formal Models And Semantics ed. Jan Van Leeuwen, Elsevier, 1990.

[FERR81] Ferrari,D., Serazzi,G., Zeigner,A. Measurement and Tuning of Computer Systems Prentice-Hall, 1981.

[GRAE87] Graefe, G., DeWitt, D. The EXODUS optimizer generator Proceedings of the 1987 ACM SIGMOD Conference on Management of Data (San Francisco, May 1987) Pg. 160 - 171.

[HAMM80] Hammer, M., Zdonik, S. B. Jr. Knowledge Based Query Processing Proc. of the 5th International Conference on Very Large Databases , Montreal, October 1980, Pg. 137 - 147.

[INGR] The Ingres Search Accelerator Users Guide

[KING79] King, J. J Exploring the use of Domain Knowledge for Query Processing Efficiency Stanford University Computer Science Department Heuristic Programming Project Technical Report HPP-79-30, December 1979.

[KING81] King, J. J QUIST : A System For Semantic Query Optimization In Relational Databases Proc. of the 6th International Conference on Very Large Databases, 1980, Pg. 510 - 517.

[LEUN85] Leung, C.H.C, Wong, K.S. File processing efficiency on the content addressable file store Proc. VLDB conference, 1985.

[PIAT91] Piatetsky-Shapiro, G., Frawley, W. J (ed.) Knowledge Discovery in Databases AAAI Press / MIT Press, 1991.

[PIAT93] Piatetsky-Shapiro, G. (Workshop Chair) Working Notes of Workshop on Knowledge Discovery in Databases, AAAI-93, 1993.

[SIEG92] Siegel, M., Sciore, E., Salveter, S. A Method For Rule Derivation to Support Semantic Query Optimization ACM Transactions on Database Systems Vol. 17, No. 4, 1992, Pg. 563 - 600.

[YU89] Yu, C., Sun, W. Automatic Knowledge Acquisition and Maintenance of Semantic Query Optimization IEEE Trans. Know. Data Eng. 1,3 (Sept. 1989) Pg. 362 - 375.