Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1999 Proceedings

Americas Conference on Information Systems (AMCIS)

December 1999

An Intelligent Approach to Semantic Query Processing

William Haseman
University of Wisconsin - Milwaukee

Tung-Ching Lin
University of Wisconsin - Milwaukee

Derek Nazareth
University of Wisconsin - Milwaukee

An Intelligent Approach to Semantic Query Processing

W. David Haseman, University of Wisconsin-Milwaukee, daveh@uwm.edu
Tung-Ching Lin, National Sun Yat-Sen University, Taiwan, tclin@mis.nsysu.edu.tw
Derek L. Nazareth, University of Wisconsin-Milwaukee, derek@uwm.edu

Abstract

Semantic knowledge has often been employed to enforce relational database integrity. It also offers the opportunity to transform a query into a semantically equivalent query that is potentially more efficiently processed than the original query. This paper describes a knowledge-based transformation approach that utilizes semantic integrity constraints and transformation heuristics to reduce query processing costs. Implementation in the form of an intelligent system, QUOTA, is also presented.

Introduction

As relational database management systems (DBMS) become more widely used, as evidenced by the current interest in data warehousing, the effective access of data becomes an important issue. Database administrators are faced with the constant threat of runaway queries tying up system resources. In most cases, query processing is typically delegated to the query optimizer provided with the DBMS software. However, conventional query optimizers perform poorly for multiple joins involving several large tables, as well as for correlated or qualified queries. Syntactic query optimization concentrates upon improving the efficiency of query operation while preserving the semantic content of the query. This is done through alternative sequencing of operations, algebraic transformation of query conditions, and alternative access strategies, to create an equivalent query that could be satisfied more efficiently. While many of these approaches are effective, in that they can effectively identify the "best" syntactically equivalent query, the overall savings that is achieved is strictly bounded.

Considerably greater savings can be achieved if some operations (particularly expensive join operations) can be eliminated, new access methods can be employed, and significant fractions of tuples removed from consideration. These forms of savings can only be achieved if the query processor is aware of prevailing semantic information about the database. Semantic knowledge of this nature has been expressed in the form of integrity constraints, and can be traced to the work of semantic data modeling [4], and has been extended to include concepts as semantic integrity constraint processing, and deductive databases [3].

Semantic Query Processing

Several approaches to semantic query processing have been proposed. These include the use of domain-specific semantic knowledge coupled with a resolution theorem prover to improve the search of a database [2], [11]; semantically equivalent query transformation [5], [7], [13]; transformation of relational algebra queries [12]; integrity constraint use in a Prolog database [6], [8] a knowledge-based approach coupled with a Prolog database [10]; use of extensional and intensional schema and integrity constraints in deductive database [9]; logic-based semantic query optimization [1]. The effectiveness of these approaches varies widely, but they are all limited in the query complexity they can handle. This paper presents an alternative approach that handles more complex queries, and generated considerable savings in processing costs. A partial database schema is provided in Table 1 to illustrate various semantic integrity constraints (SIC) and their use in semantic query transformation.

Relations
SHIP(shipname, owner, registry, type, capacity, deadwt)
CARGO(cargo#, ship, cargotype, quantity)
OWNER(ownername, industrytype, assets, headquarters)

Join paths
SHIP.shipname = CARGO.ship
OWNER.ownername = SHIP.owner

Access methods
SHIP: cluster index on SHIP.owner
non-cluster index on SHIP.shiptype
CARGO: cluster index on CARGO.cargotype
OWNER: cluster index on OWNER.industrytype

Table 1. Partial database schema

Several forms of semantic integrity constraints are available. These include:

- Conditional reference constraints
  All cargos use available ships
  CARGO.ship $\subseteq$ SHIP.shipname
- Bounding constraints
  Cargo quantities are bounded by ship capacity
  CARGO.quantity $\leq$ SHIP.capacity
- Implication constraints
  All liquefied natural gas (LNG) tankers have a

capacity of at most 2500 tonnes
SHIP.type = 'LNG $\Rightarrow$ SHIP.capacity $\leq$ 2500

- Equivalence constraint
  All supertankers have deadweights of 100000 or more and all ships with deadweight exceeding 100000 are supertankers
  SHIP.type = 'supertanker' $\Leftrightarrow$ SHIP.deadwt $\geq$ 100000

More complex constraints can be formulated, but they can always be broken down to well-defined integrity constraints presented.

Intelligent Semantic Query Processing

Several transformation heuristics can be used to create semantically equivalent queries (SEQ). These include join elimination, restriction elimination, index introduction, qualified tuple reduction or scan reduction, and join introduction. Each of these heuristics has the potential to generate a SEQ for a given query. If several transformations are possible, they may generate several possible SEQs. Their effects are varied, depending on the number of aspects eliminated from the original query. In addition, relationship cardinality and selectivity among relations will also affect expected savings.

Join elimination
This heuristic involves the use removal of a relation from the original query, and requires the presence of an inter-relational SIC.

```
Query:  SELECT  CARGO.cargo#
        FROM    SHIP, CARGO
        WHERE   SHIP.type = 'LNG tanker'
        AND CARGO.cargotype = 'LNG'
        AND SHIP.shipname = CARGO.ship

SIC:    CARGO.type = 'LNG'  ⇒  SHIP.type = 'LNG'

SEQ:    SELECT  CARGO.cargo#
        FROM    CARGO
        WHERE   CARGO.cargotype = 'LNG'
```

Restriction elimination
This transformation involves the removal of a clause from the original query, and typically involve an intra-relational SIC.

```
Query:  SELECT  SHIP.shipname, SHIP.owner
        FROM    SHIP
        WHERE   SHIP.type = 'supertanker'
        AND SHIP.deadwt > 75000

SIC:    SHIP.type = 'supertanker'  ⇒  SHIP.deadwt > 100000

SEQ:    SELECT  SHIP.shipname, SHIP.owner
        FROM    SHIP
        WHERE   SHIP.type = 'supertanker'
```

Index introduction
This involves the augmentation of a query with an additional clause or value restriction representing an attribute that is indexed. Index introduction will involve an intra-relational SIC.

```
Query:  SELECT  OWNER.headquarters
        FROM    OWNER
        WHERE   OWNER.assets > 1 billion

SIC:    OWNER.assets > 1Billion ⇒ OWNER.industry =
        'petroleum'

SEQ:    SELECT  OWNER.headquarters
        FROM    OWNER
        WHERE   OWNER.assets > 1Billion
        AND OWNER.industry = 'petroleum'
```

Savings result from cheaper index access as compared to sequential scans of the relation.

Qualified tuple reduction
This heuristic seeks to add a clause or restriction to the original query so that it will result in fewer tuples that qualify prior to a join operation, thereby making the join cheaper. Consequently, this heuristic will rely on inter-relational SICs.

```
Query:  SELECT  SHIP.shipname
        FROM    SHIP, CARGO
        WHERE   CARGO.destination = 'UK'
        AND CARGO.type = 'urea'
        AND SHIP.shipname = CARGO.ship

SIC:    CARGO.type = 'urea'  ⇒ SHIP.type = 'dry bulk carrier'

SEQ:    SELECT  SHIP.shipname
        FROM    SHIP, CARGO
        WHERE   CARGO.destination = 'UK'
        AND CARGO.type = 'urea'
        AND SHIP.shipname = CARGO.ship
        AND SHIP.type = 'dry bulk carrier'
```

As a result, the SHIP relation is now joined using only 'dry bulk carrier' tuples, whereas previously the join would be performed using all tuples.

Join introduction
This heuristic adds another relation to the query. In general, this would appear to be counter to generating a cheaper query. However, if the attribute that is being scanned is not indexed and the cardinality of the relation is large, then it may be beneficial to add a new relation where the attribute is indexed and the cardinality is considerably smaller.

```
Query:  SELECT  SHIP.shipname
        FROM    SHIP
        WHERE   SHIP.deadwt > 150000

SIC:    SHIP.deadwt > 100000 ⇒ OWNER.industry = 'petroleum'

SEQ:    SELECT  SHIP.shipname
        FROM    SHIP, OWNER
        WHERE   SHIP.deadwt > 150000
        AND SHIP.owner = OWNER.ownername
        AND OWNER.industry = 'petroleum'
```

Cost reduction is based upon the cardinality of the relations in question, and the selectivity among them based on the SIC employed.

Implementation

The semantic query transformer was implemented as a knowledge-based system termed Query Optimization and Transformation Analysis (QUOTA). QUOTA is implemented in a forward-chaining knowledge engineering environment. It comprises two interfaces, three knowledge components, and three processing components. The architecture of QUOTA is depicted in Figure 1. The primary function of the database interface is to format an SEQ into standard SQL. The user interface is considerably more involved. It performs three major functions – translation of the original query into the representation employed by QUOTA, error handling, and relaying of results and explanation to the user. Knowledge in QUOTA is structured into three distinct areas – domain knowledge, control knowledge, and heuristic transformation knowledge. Each of these is relatively independent of the other and can be swapped with other components if the design or its application domain is to be altered. QUOTA uses a forward-chaining inference mechanism to fire three separate procedures blocks – error analysis, SEQ analysis/conflict resolution, and SEQ transformation. If multiple transformations are applicable, there may be potential conflict. QUOTA addresses this by examining all SEQs. A digraph representation allows QUOTA to identify conflicting SEQs, and select promising ones using a greedy heuristic. A cost estimation model that estimates the number of pages to be scanned is also included.

Application and Discussion

A shipping database was employed to evaluate the effectiveness of query transformation by QUOTA. The database comprised 9 relations, and 75 SICs for query transformation purposes. A total of 45 different query types were formulated for evaluation of QUOTA's ability to improve upon query performance. The query mix included single and multiple table queries, and spanned restricted and unrestricted queries. The queries ranged considerably in complexity and size, with potential costs spanning 50 to 3.3 million pages.

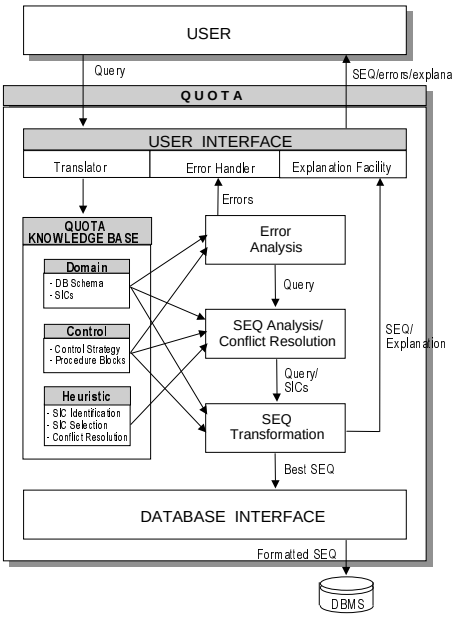

Figure 1. QUOTA Architecture

In all cases, QUOTA was able to return appropriate SEQs. The cost savings ranged from 0% (for cases where no SICs were applicable) to 99.97% in cases where several transformation heuristics were applied. Though QUOTA was implemented as a standalone system, the approach clearly has potential. More complex queries can be dealt with using multiple transformation heuristics, thereby generating greater savings over traditional semantic query transformation. The use of multiple transformation heuristics required an intelligent strategy for sequencing multiple transformations to a query. The approach adopted in QUOTA provided explicit control over the selection of transformation heuristics and the resolution of conflict among them. Though it cannot claim to generate the "optimal" SEQ, QUOTA has performed satisfactorily, consistently generating the SEQ with the lowest cost.

References are available from the last author or at http://www.uwm.edu/~derek/research/T04-014.pdf