SEKE'94 June, 21-23

94-SQ1

SEKE'94

The 6th International Conference on Software Engineering and Knowledge Engineering

Printing by Knowledge Systems Institute

94-SQ

Semantic Query Optimization by Class Reference Reduction in Object-Oriented Databases

Sha Guo, Wei Sun, Naphtali Rishe, Yi Deng
School of Computer Science
Florida International University
Miami, Florida 33199, U.S.A.

Abstract

A semantic query optimizer is proposed in an object-oriented database system (OODB) in this study. Although semantic query optimization has been intensively studied in the context of relational and deductive database management systems (DBMSs) [9, 27], certain distinct object-oriented features such as class hierarchy have not been taken into consideration. In this paper, we show that class assertions or constraints *that characterize the common properties of objects in a class and that are widely supported by OODBs can be used to semantically optimize OODB queries by reducing the reference to a class hierarchy in a query to a set of (sub)classes in the hierarchy. This optimization is important, because accessing a subset of classes is always less costly than accessing the whole hierarchy (which implies accessing all its classes) in evaluating an OODB query. The semantic query optimization by making use of class assertions is clearly uniquely pertinent to OODBs, and may yield a substantial gain in evaluating queries in OODBs.*

1 Introduction

The object-oriented technique has been widely applied in various DBMSs. Several object-oriented database systems (OODBs) have been developed in recent years. Some systems such as **GemStone** [6, 22], **Vbase** and its successor **Ontos** [1], **Orion** [20, 21] and **O2** [13] are now commercially available. [16] presented several prototype systems which are among the most representative new-generation DBMSs.

Although there has been no consensus so far on what an OODB is, some general characteristics and features that an OODB should possess were pointed out in [3, 21]. Each object is uniquely identified by a system-wide object identifier (oid). Objects, also called *instances*, are grouped into *classes*; instances in a class will share the same set of attributes and methods; and a class can be a *subclass/superclass* of another one. Thus, a class hierarchy exists in an OODB. This class hierarchy reveals the IS-A relationship, or the generalization/specialization among classes. Supporting IS-A relationship and its associated inheritance is one of the most important properties in an OODB and OO programming language.

Semantic query optimization (SQO) was proposed in the early 80s in relational DBMS [9, 25, 27]. The basic idea of SQO is to optimize queries by transforming a query qualification into another semantically equivalent one using *semantic integrity constraints* (SICs) such that evaluating the transformed query becomes less costly. SICs, which were probably first introduced in [15] in relational DBMSs, have been widely used in many commercial DBMSs. SICs assert permissible or consistent database states (a database state is the stored data at a given time instance) for a database application. For example, in a payroll database, the minimum hourly pay rate has to be $4.35 by law; in a university, the regulation may require that all graduate students maintain a minimum GPA of 3.0. More examples can be found in [9, 27]. Although SQO has been extensively studied in a relational DBMS and deductive DBMS, there has been little discussion in the literature on SQO in an OODB environment.

There are a few general goals to be achieved by SQO. In relational DBMSs [25, 27], SQO will normally involve (in decreasing order of priority of importance):

1. Detecting whether there is a contradiction in a user's query qualification under the set of SICs;

2. Eliminating unnecessary joins (*class traversals* [21] or *functional joins* in an OODB [7, 33]);

3. Eliminating *non-cost-beneficial* and semantically

redundant selection predicates (also called *restrictions*);

4. Adding *cost-beneficial* and semantically redundant restrictions.

Motivations, examples and analyses pertinent to SQO in a relational DBMS can be found in [9, 25, 27]. It seems that these goals and the proposed techniques are also generally applicable to SQO in OODBs, with a possibly slight modification.

In [20] a query model in an OODB is provided. Queries in an OODB have many distinct characteristics. One of the major features in an OODB is that a query may be posed against a single class, or a class hierarchy rooted at a class (including all classes in the hierarchy). In the latter case, when a query is evaluated, each instance in each class in the class hierarchy will be accessed and evaluated against the query qualification. In this paper, we show that it is possible that (1) all instances in a class may not contribute to the answer to the query, and (2) all instances in a class will be in the answer to the query. Let the class of the first type be called a C_Class (a Contradictory Class), and the class of the second type be called an I_Class (an Implied Class) (also see *Examples 1 & 2* below). For a C_Class, accessing instances in the class becomes unnecessary for evaluating the query, because none of the instances will satisfy the query qualification; For an I_Class, evaluating the query qualification against the instances in the class becomes unnecessary, because all the instances of the class satisfy the query qualification, and thus will be in the answer to the query. In fact, in many OODBs, objects are indexed by their oids. Thus, all oids in the oid index file constitute the evaluation if we ignore how objects are presented to users (in graphs, voices, motion pictures, or normal texts). Given a query against a class hierarchy, we want to identify as many C_Classes and I_Classes as possible in the hierarchy for an OODB. We call this task the *class reference reduction*, that is, to reduce a reference of a class hierarchy (including all classes in the hierarchy) in a query to the reference of a subset of classes in the class hierarchy. This optimization goal is uniquely pertinent to an OODB, and is clearly distinct from the traditional SQO goals. This subject has not been addressed in the literature. It is clear that in order to identify these C_Classes and I_Classes with respect to a query qualification, the properties of objects in a class should be characterized or asserted. It is popular that certain *constraints* can be associated with objects in a class, for example, the value for the attribute *age* in class *Employee* must be greater than 18 by the federal law, and so on. Exactly due to the reason that *similar* objects are grouped into a class, these common features can be captured by using assertions. This type of mechanism by using constraints are supported by almost all OODBs and OO languages such as Gemstone, Smalltalk, C++ and O2. In this study, we only focus on conjunctive queries and conjunctive constraints of a class, since conjunctive queries and constraints are believed to be the most popular ones used in a practical setting. More precisely, queries and constraints are conjunctions of the inequalities of the types $(Attr_x \textbf{ op } C)$ or $(Attr_y \textbf{ op } Attr_z)$, where C is a constant of the domain $Attr_x$, $Attr_y$ and $Attr_z$ are class attributes such as *age* [1], and $\textbf{op} \in \{<, \leq, >, \geq, =, \neq\}$. This assumption is reasonable, since $(Attr_y \textbf{ op } Attr_z)$ represents a θ*-join*, and $(Attr_x \textbf{ op } C)$ represents a *selection* in a relational database system (similar operations are also supported and widely used in an OODB). Let the set of constraints defined for a class be called the *class assertion* of the class. In order to identify the C_Classes with respect to (wrt) a query, an efficient solution to decide whether the query qualification, together with a class assertion, is *satisfiable* or not is essential. In order to identify the I_Classes, an efficient solution to decide whether the query qualification is *implied* by the class assertion of a class is essential. These implication and satisfiability problems are also encountered in many other database problems such as identifying the sites that may contain a relation fragments that may contribute to a query in a distributed fragmented database system [10, 31], semantic query optimization [9, 30], global query optimization [8, 17, 24], efficient updates via views [2, 4, 5]. Thus, the proposed implication and satisfiability algorithms will be of use in solving these problems.

In this paper, we assume that all classes are independent files in the underlying physical system. In [29] several physical storage models for object-oriented databases are provided. The physical storage scheme we assumed has been adopted by OODBs such as Orion[19]. In fact, it is not difficult to observe that OODB queries which have been semantically optimized as proposed will always be executed more efficiently under any OODB physical model (or in the worst case, the equivalently efficiently).

Figure 1 shows a simplified object-oriented data model concerning one class hierarchy using a sample

[1] We note that attributes may be rather complex, which may involve class traversals [21]. Since this is not quite related to our discussion of SQO, we ignore it only for the simplification of presentation.

university database. This hierarchy is represented as a rooted, directed (from superclasses to subclasses), acyclic graph (a DAG for short), where each node is a class. The hierarchy says that class **Student** has two subclasses **Graduate** and **Undergraduate**; and in turn class **Graduate** has two subclasses **RA** and **TA**. We assume each class has its own instances that physically belong to the class. For example, students include some non-degree track students (who are neither graduates nor undergraduate students) in addition to undergraduate students and graduate students. Graduate students include RAs, TAs, and those who are neither RAs nor TAs. The following simplified examples based on the sample database illustrates the motivation of the proposed SQO.

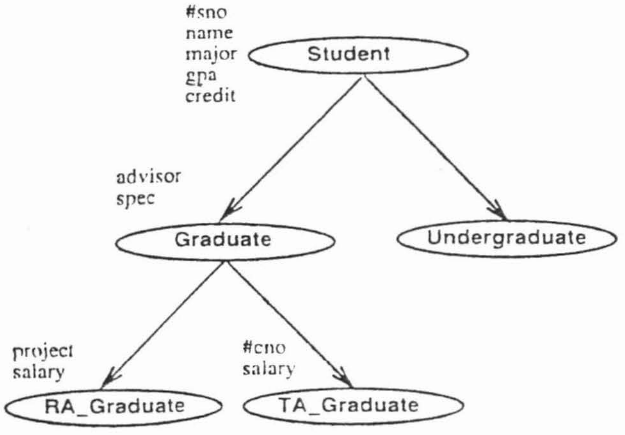


Figure 1: An example database schema

Example 1 *Suppose a query is ("list all graduate students who take courses of 3 credits"). The evaluation of the query will access all graduate students in class* Graduate *as well as all graduate students in its subclasses* RA *and* TA, *apply the query qualification to each of them, and return those qualified graduate students (instances). If the following constraint is given ("RA and TA graduate students must maintain full-time status by registering courses of at least 6 credits in order to maintain their assistantships"), then clearly the answer to this query does not include any TA and RA graduate students without accessing/evaluating classes* RA *and* TA. *Therefore, it is sufficient to only access graduate students (instances) in the class* **Graduate** *in order to evaluate the query. Classes* RA *and* TA *are the C_Classes with respect to the query. If only 1/10 graduates are not TAs or RAs, then the semantically optimized query evaluation plan only needs to access 1/10 of instances — a significant gain.* ■

This example shows that when the qualification of a query conflicts with the class assertion of a class, it can be directly concluded that none of the instances in the class (a C_Class) will contribute to the answer of the query. As a result, there is no need to access any instances of this class for the evaluation. The following indicates another scenario that a query can be semantically optimized (the I_Classes).

Example 2 *Suppose a query is ("list all graduate students who take courses of at least 3 credits"). The conventional strategy to evaluate this query will apply the query qualification/selections to all instances (graduate students) in classes* Graduate, RA, *and* TA. *Assume that the same class assertion for class* Graduate *as shown in* Example 1 *is given, the answer to this query includes all RA and TA graduate students without physically accessing/evaluating them on an individual instance basis.* ■

Example 2 shows that if the class assertion of a class implies a query qualification, then all instances of this class are qualified to the query. In many OODBs, objects in a class are indexed by their oids. Thus, all the oids in the object index file are the desired result.

The rest of the paper is organized as follows: In *Section 2*, we show how to identify all the C_Classes and I_Classes when given a query. In *Section 3*, we discuss how to solve the implication and satisfiability problems efficiently and effectively, which is the essential part of the proposed SQO in an OODB. Finally, *Section 4* concludes this paper.

2 Semantic query optimization

In this section, we introduce some basic concepts, and provide an overall strategy.

Definition 1 *A* **class assertion** *of a class X, denoted as A_X, consists of conjunctive* **constraints**. *A constraint is of the form ($Attr_x$* **op** *C) or ($Attr_y$* **op** *$Attr_z$), where $Attr_x$, $Attr_y$, and $Attr_z$ are attributes of a class, C is a constant of the domain of $Attr_x$, and* **op** *$\in \{<, =, >, \leq, \neq, \geq\}$. An instance I belongs to class X only if I satisfies A_X.* ■

The objective of this paper is to identify as many I_Classes and C_Classes as possible when given a query.

This is achieved by testing the relationships (implication or satisfiability/contradiction) between the qualification of the query and class assertions, as precisely defined below.

Definition 2
$\{C_1/Attr_{x_1}, C_2/Attr_{x_2}, ..., C_n/Attr_{x_n}\}$ is said to be an **assignment** *for a class assertion A_X or a query qualification q if every occurrence of $Attr_{x_i}$ in all inequalities in q or A_X is simultaneously replaced by C_i, $1 \leq i \leq n$. An assignment* **satisfies** *q or A_X* **if and only if** *the q or A_X evaluates true under the assigned values. There exists a* **contradiction** *in q or A_X* **if and only if** *there does not exist an assignment which will satisfy q or A_X. In the latter case, we also say that q or A_X is* **unsatisfiable**.

Definition 3

Implication: *q is* **implied** *by A_X (or A_X implies q), denoted as $A_X \Rightarrow q$,* **if and only if** *every assignment that satisfies A_X also satisfies q.*

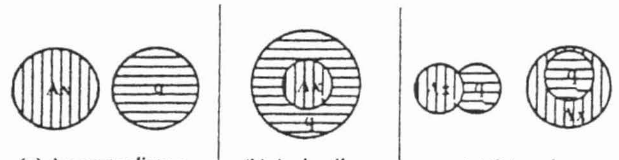
Satisfiability: *A_X and q is* **satisfiable if and only if** *there exists an assignment for A_X and q that will satisfy A_X and q (i.e., A_X and q is true under the assignment).* ■

Given a query $Q[X : q]$ that will retrieve all objects in the classes in the class hierarchy rooted at X, and the class assertion A_X of class X, there are three possible relationships between q and A_X as shown in the *Figure 2* below:

A_X contradicts q: In this case, all instances in class X will not contribute to the answer of the query.

A_X implies q: In this case, all instances in class X will be in the answer of the query.

Otherwise: Instances in class X need to be accessed and evaluated on an individual basis as usual.



(a) Ax contradicts q (b) Ax implies q (c) Otherwise

Figure 2: **Relationships between query qualification q and class assertion A_X**

We note that class X may represent a single class or the class hierarchy rooted at X. In the latter case, when A_X contradicts q, all instances in X and in its subclasses will not contribute to the answer of the query; and when A_X implies q, all instances in class X and its subclasses will be in the answer of the query. This is obvious, because an object must satisfy all class assertions of its superclasses (or in other words, class assertions are also inherited).

We want to identify the maximal set of I_Classes and C_Classes, or equivalently, minimize the classes to be accessed/evaluated. The following recursive algorithm is direct. A query qualification q and a class name X (representing the class hierarchy rooted at X to be accessed by the query) are the input to the algorithm. When the algorithm terminates, **ISet**, initialized to be empty when the algorithm is invoked, contains all the I_Classes; and **CSet**, initialized to be empty when the algorithm is invoked, contains all the C_Classes;

```
Algorithm SQO-CRR(X, q, ISet, CSet)
  begin if X = ∅ then return; /* the exit rule */
    if conflict($A_X \wedge q$) then
      SQO-CRR(∅, q, ISet, CSet ∪ $X^{all}$);
      else if imply($A_X$, q) then
        SQO-CRR(∅, q, ISet ∪ $X^{all}$, CSet);
        else for $X_1, X_2, ..., X_k$ of $X$ do
          SQO-CRR($X_S$, q, ISet, CSet);
  end;
```

where the function conflict($A_R \wedge q$) returns *true* if A_X conflicts q, *false* otherwise; the function imply(A_X, q) returns *true* if A_X implies q, *false* otherwise; X^{all} denotes the set of all classes in the hierarchy rooted at X. *Example 3* below shows how the algorithm works.

Example 3 *Assume that a student must take courses of at least one credit to maintain his/her student status. Each student is assigned a unique student number ranging from "S00000" to "S50000". The following is the class assertion for class* Student*:*

$$A_{Student} = \{ \mathit{Student.credit} \geq 1, \mathit{Student.\#sno} \leq \text{"S50000"}, \mathit{Student.\#sno} \geq \text{"S00000"} \}$$

A graduate student must maintain his/her GPA at least 3.0. Graduate students have student number from "S01000" to "S09999".

$$A_{Graduate} = \{ \mathit{Graduate.gpa} \geq 3.0 \quad \mathit{Graduate.\#sno} \leq \text{"S09999"} \quad \mathit{Graduate.\#sno} \geq \text{"S01000"} \}$$

A undergraduate student must maintain his/her GPA at least 2.0. Undergraduate students have student number from "S10000" to "S50000".

$$A_{Undergraduate} = \{ \begin{array}{l} Undergraduate.gpa \geq 2.0 \\ Undergraduate.\#sno \leq \text{"S50000"} \\ Undergraduate.\#sno \geq \text{"S10000"} \} \end{array}$$

RAs and TAs must take courses of at least 6 credits.

$A_{RA} = \{ RA.credit \geq 6 \}$
$A_{TA} = \{ TA.credit \geq 6 \}$

Suppose that a query is Q[Student : ((*#sno* ≥ "*S00500*") ∧ (*#sno* ≤ "*S09999*"))], *which says "list the students whose id numbers are within S00500 and S09999". The query target is the class hierarchy rooted at* Student, *or equivalently, it refers to the set of classes* {Student, Graduate, Undergraduate, RA, TA}. *The above algorithm can thus be invoked as* SQO-CRR(Student, ((*#sno* ≥ "*S00500*") ∧ (*#sno* ≤ "*S09999*")), ISet, CSet), *where* ISet *and* CSet *have been initialized to be* ∅ *(empty). We know that the class assertion* $A_{Graduate}$ *of class* Graduate *implies the query qualification, the class assertions of class* Graduate*'s subclasses* RA *and* TA *imply the query qualification, and the class assertion of class* Undergraduate *contradicts the query qualification due to (Undergraduate.#sno ≥ "S10000"). When the algorithm terminates,* CSet = {Undergraduate} *and* ISet = {Graduate, RA, TA }. *Consequently, it is sufficient to access/evaluate objects in the class* Student. ■

As discussed above, efficiently and effectively solving the implication and satisfiability problems (the conflict() and imply() procedures) is central to the proposed strategy. In the next section, we discuss how to solve these problems.

3 Solving the satisfaction and implication problems

We first discuss the implication problem in *Section 3.1*. In *Section 3.2*, the satisfiability problem is discussed.

3.1 Solving the implication problem

The implication problem (whether A_X implies q) in the integer domains has been shown to be NP-hard[23]. In the following, we provide an efficient algorithm to determine whether A_X implies q in the real domains.

A. Klug and J. Ullman proposed an algorithm to solve the implication problems involving the inequalities of the form ($Attr_x$ **op** $Attr_y$), a special case of our situation. We will also have a brief discussion of the Klug-Ullman's algorithm for a comparison.

Klug and Ullman's approach [28] uses an idea similar to the way that functional dependencies in a relational database system are handled, where a collection of axioms is used [28]. The Klug-Ullman axioms, as shown below, for inequalities are then shown to be sound (only inferring correct inequalities) and complete (inferring every correct inequalities) [28].

A1: ($Attr_x \leq Attr_x$)
A2: ($Attr_x < Attr_y$) implies ($Attr_x \leq Attr_y$)
A3: ($Attr_x < Attr_y$) implies ($Attr_x \neq Attr_y$)
A4: ($Attr_x \leq Attr_y$) and ($Attr_x \neq Attr_y$)
imply ($Attr_x < Attr_y$)
A5: ($Attr_x \neq Attr_y$) implies ($Attr_y \neq Attr_x$)
A6: ($Attr_x < Attr_y$) and ($Attr_y < Attr_z$)
imply ($Attr_x < Attr_z$)
A7: ($Attr_x \leq Attr_y$) and ($Attr_y \leq Attr_z$)
imply ($Attr_x \leq Attr_z$)
A8: ($Attr_x \leq Attr_z$), ($Attr_z \leq Attr_y$),
($Attr_x \leq Attr_w$), ($Attr_w \leq Attr_y$)
and ($Attr_w \neq Attr_z$) imply ($Attr_x \neq Attr_y$)

Then, Klug-Ullman's algorithm first computes the closure of A_X, denoted as A_X^+, by applying axioms A1 to A8 on A_X until there is no any more new inequalities to be generated. The closure computation procedure is as follows:

1. Convert each < relationship, say ($Attr_x < Attr_y$), into ($Attr_x \leq Attr_y$) and ($Attr_x \neq Attr_y$).
2. Compute the transitive closure of the ≤ relationships.
3. Apply axioms A8 to infer additional ≠ relationships.
4. Reconstruct the < relationships using axiom A4; that is, ($Attr_x < Attr_y$) if ($Attr_x \leq Attr_y$) and ($Attr_x \neq Attr_y$).

The total time complexity of Klug-Ullman's algorithm to test whether A_X implies q is $O(|A_X|^3 + |q|)$ due to Step(3).

In [26] a more efficient algorithm with the complexity $O(|A_X|^{2.376} + |q|)$ is proposed to compute A'^+_X.

For our case, we construct an inequality set A'_X from A_X: let $(C_1, C_2, ..., C_k)$ be all the distinct constants in ascending order of their values, which are used in all inequalities of the form $(Attr_x \text{ op } C_i)$ in A_X, we introduce k dummy attributes $(Attr_{w_1}, Attr_{w_2}, ..., Attr_{w_k})$ to represent these k distinct constants, and $2k-2$ inequalities, $A_{ADDED} = \{(Attr_{w_1} \leq Attr_{w_2}), (Attr_{w_2} \leq Attr_{w_3}), \ldots, (Attr_{w_{k-1}} \leq Attr_{w_k}), (Attr_{w_1} \neq Attr_{w_2}), (Attr_{w_2} \neq Attr_{w_3}), \ldots, (Attr_{w_{k-1}} \neq Attr_{w_k})\}$ to represent the relationships among the newly introduced attributes. For each $(Attr_x \text{ op } C_i) \in A_X$, we transform it into $(Attr_x \text{ op } Attr_{w_i})$, where $Attr_{w_i}$ represents the dummy attribute for C_i. Let A_{REP} be the inequality set after the above transformation, and $A'_X = A_{REP} \cup A_{ADDED}$. $G_{A'_X}$ is constructed as the way of constructing G_{A_X}. Then the closure of A'_X, denoted as A'^{+}_X, is computed [26, 28]. After A'^{+}_X is computed, dummy attributes are replaced back with corresponding constants. For each attribute $Attr_x$, let $C^x_{up} = \min(C_i)$ for all $(Attr_x \leq C_i) \in A'^{+}_X$, and $C^x_{low} = \max(C_i)$ for all $(Attr_x \geq C_i) \in A'^{+}_X$. It is noted that after the above transformation, the size of A'_X is still bounded by $O(|A_X|)$.

The following lemma basically follows the soundness and completeness of the Klug-Ullman axioms [28]. A proof can be found in [26].

Lemma 1 *A_X implies q* if and only if *A_X is unsatisfiable, or*

- *for any $(Attr_x \leq Attr_y) \in q$, $(Attr_x \leq Attr_y) \in A'^{+}_X$; and*
- *for any $(Attr_x \neq Attr_y) \in q$, $(Attr_x \neq Attr_y) \in A'^{+}_X$; and*
- *for any $(Attr_x \leq C) \in q$, $C \geq C^x_{up}$; and*
- *for any $(Attr_x \geq C) \in q$, $C \leq C^x_{low}$; and*
- *for any $(Attr_x \neq C) \in q$, $(Attr_x \neq C) \in A'^{+}_X$ or $C < C^x_{low}$ or $C > C^x_{up}$.* ■

3.2 Solving the satisfaction problem

The satisfiability problem in the integer domains (i.e., all the domains of attributes contain integers) has been shown to be NP-hard[23]. The known NP-hard problem of determining whether an undirected graph is *three colorable* [12, 14] is reduced to an instance of this satisfiability problem as follows: Let S denote $(A_X \wedge q)$, which is a conjunctive formula containing only the inequalities of the form $(Attr_{x_i} \geq 1)$ and $(Attr_{x_i} \leq 3)$ for all attributes $Attr_{x_i}$ in S, and some unequalities $(Attr_{x_i} \neq Attr_{x_j})$. It can be seen that each represents an area (thus, only three values – three colors are allowed for "coloring" each area), and the set of unequalities specifies the adjacency of areas, i.e., $(Attr_{x_i} \neq Attr_{x_j})$ indicates that $Attr_{x_i}$ and $Attr_{x_j}$ are adjacent areas.

In the following, we provide a linear $O(|S|) = O(|A_X| + |q|)$ algorithm to determine whether $S = (A_X \wedge q)$ is satisfiable in the real domains.

It is sufficient to consider **op** $\in \{\leq, \neq\}$ for inequalities of the type $(Attr_x \text{ op } Attr_y) \in S$ and **op** $\in \{\leq, \geq, \neq\}$ for inequalities of the type $(Attr_x \text{ op } C) \in S$. For all $(Attr_x \{<, \leq\} Attr_y) \in S$, we construct a *labeled directed graph* $G_S = (V_S, E_S)$, where V_S is the set of distinct attributes in S, and a labeled directed edge from $Attr_x$ to $Attr_y$ in E_S, $(Attr_x, Attr_y, \otimes) \in E_S$, if and only if $(Attr_x \otimes Attr_y) \in S$, where the label $\otimes \in \{<, \leq\}$.

For any two attributes $Attr_x$ and $Attr_y$ in G_S, if they are reachable from each other, $(Attr_x = Attr_y)$ is implied by transitivity. All such attributes as well as the edges among them form a **strongly connected component** (SCC). We may "collapse" each SCC into a single node. This collapsing is in fact to use a single attribute in the SCC, called representative attribute/node of the SCC, to represent all attributes in the SCC. After this step, there is no cycle in the graph.

Now consider all the inequalities $\{(Attr_x \leq C_1), (Attr_x \leq C_2), ..., (Attr_x \leq C_k)\}$ for attribute $Attr_x$: Let $C^x_{up} = \min(C_i)$ for attribute $Attr_x$. Similarly, $C^x_{low} = \max(C_i)$ can be obtained for all the inequalities of the form $(Attr_x \geq C_i) \in S$.

Now we construct $\dot{A}_{min}$ by assigning each attribute of G_S, say $Attr_x$, with the value $A^x_{low} = \max(C_i, C^x_{low})$ according to the topological ordering of all nodes of G_S. C_i is equal to $A^{x_i}_{low}$, the assigned value for $Attr_x$'s *parent* $Attr_{x_i}$ (equivalently, an edge exists from $Attr_{x_i}$ to $Attr_x$, and $Attr_x$ is called a *child* of $Attr_{x_i}$). A^x_{low} is an "open" bound if one C_i is the maximum and the edge which contributes the C_i is labeled with "<"; Otherwise A^x_{low} is a "closed" bound.

Similarly, we construct another assignment, denoted as $\dot{A}_{max}$, of G_S as follows: we assign attributes with $A^x_{up} = \min(C_i, C^x_{up})$ one by one according to the inverse topological ordering, where C_i is equal to $A^{x_i}_{up}$, the assigned value for *$Attr'_x$s child*. The way to determine the closeness or openness of A^x_{up} is similar to that of A^x_{low}.

Lemma 2 *$S = (A_X \wedge q)$ is satisfiable* if and only if *for any attribute $Attr_x$ of $S = (A_X \wedge q)$,*

- *(1) $A^x_{low} < A^x_{up}$, or*
- *(2) $A^x_{low} = A^x_{up}$ with both bounds as "closed", and $(Attr_x \neq A^x_{low}) \notin (S = A_X \wedge q)$.* ■

A proof can be found in [26].

Constructing G_S, G'_S, $\dot{A}_{min}$, and $\dot{A}_{max}$ only takes $O(|S|)$ time; Finding those $Attr_x$ with $A^x_{low} = A^x_{up}$ (both bounds are closed) takes $O(|S|)$ time; To test whether any unequality of S is violated by implied equations also takes $O(|S|)$ time. As a result, the total complexity is $O(|S|)$, or $O(|A_X| + |q|)$.

Example 4 *Consider the set of inequalities S:*

$$\begin{array}{rlll} S = & \{ & Attr_{x_1} \leq Attr_{x_2}, & Attr_{x_1} \leq Attr_{x_3} \\ & & Attr_{x_1} < Attr_{x_5}, & Attr_{x_1} \geq -3.0, \\ & & Attr_{x_1} \leq 1.5, & Attr_{x_1} \neq -3.0, \\ & & Attr_{x_2} < Attr_{x_6}, & Attr_{x_2} > 3.5, \\ & & Attr_{x_2} \leq 5.5, & Attr_{x_2} \neq Attr_{x_5}, \\ & & Attr_{x_2} \neq 4.0, & Attr_{x_2} \neq 5.0, \\ & & Attr_{x_3} \leq Attr_{x_4}, & Attr_{x_3} \geq 1.0, \\ & & Attr_{x_3} \neq X_6, & Attr_{x_3} \neq 1.2, \\ & & Attr_{x_3} \neq 1.4, & Attr_{x_3} \neq 4.0, \\ & & Attr_{x_4} \leq Attr_{x_3}, & Attr_{x_4} < 1.5, \\ & & Attr_{x_4} \neq X_5, & Attr_{x_5} < Attr_{x_6}, \\ & & Attr_{x_5} \leq 4.0, & Attr_{x_5} \geq 4.0, \\ & & Attr_{x_6} \leq 4.5, & Attr_{x_6} > 2.0, \\ & & Attr_{x_6} \neq 4.0 \; \} & \end{array}$$

$Attr_{x_3}$ and $Attr_{x_4}$ constitute an SCC, thus they are collapsed into a single node, denoted as $Attr_{x_{34}}$ (see Figure 3*). The pair inside each node denote C^x_{low} and C^x_{up} for that node. The first element of the pair outside a node $Attr_x$ is A^x_{low} of the node $Attr_x$ in $\dot{A}_{min}$, the second element is A^x_{up} of the node $Attr_x$ in $\dot{A}_{max}$.*

4 Conclusions

A novel strategy is proposed in this paper to semantically optimize queries by class reference reduction in an OODB. Class assertions of classes are utilized. Efficient and effective algorithms for solving implication and satisfiability problems are presented which are also needed in many other database areas. A potential significant gain in evaluating a query may be achieved by the proposed SQO.

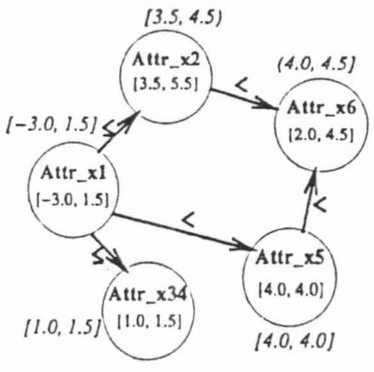


Figure 3: The construction of the assignment which satisfies G_S

References

[1] Andrews, T., and Harris, C., "Combining language and database advances in an object-oriented development environment", *Proc. of ACM OOPSLA*, Orlando, Florida, October 1987.

[2] Astrahan, M.M. *et al.* "System R: Relational Approach to Database Management", *ACM Trans. on Database Systems*, Vol.1, No.2, June 1976, pp.97-137.

[3] Bertino, E. and Martino, L., "Object oriented database management systems: concepts and issues", *IEEE Computer*, Vol. 24, No. 4, 1991, pp.33-47.

[4] Blakeley, J.A., Coburn, N., and Larson, P.A. "Updating Derived Relations: Detecting Irrelevant and Autonomously Computable Updates", *Proc. 11th Intl. Conf. on Very Large Data Bases*, 1986, pp.457-466.

[5] Blakeley, J.A., Larson, P.A., and Tompa, F.W. "Efficiently Updating Materialized Views", *Proc. ACM SIGMOD*, 1986, pp.61-71.

[6] Bretl, R., *et al.*, "The GemStone Data Management System", *Object-Oriented Concepts, Applications and Databases*, Won Kim and F. Lochovsky, Eds., Reading, MA, Addision-Wesley, 1989.

[7] Carey, M. J., D. J. DeWitt and S. L. Vandenberg, "A Data Model and Query Language of Exodus", *ACM-SIGMOD*, June, 1988, pp 413-423

[8] Chakravarthy, U., and Minker, J. "Processing Multiple Queries in Database Systems", *Database Engineering*, Vol.1, 1983.

[9] Chakravarthy, U., Grant, J., and Minker, J., "Logic-Based Approach to Semantic Query Optimization", *ACM TODS*, June 1990, pp. 162-207.

[10] Chen, A., Brill, D., Templeton, M., and Yu, C. "Distributed Query Processing in a Multiple Database Systems", *IEEE Journal on Selected Areas in Communications*, 1989, pp. 390-398.

[11] Coppersmith, D. and Winograd, S., "Matrix Multiplication via Arithmetic Progressions", *Proc. of the 19th Annual ACM Symposium on the Theory of Computing*, 1987, pp 1-6.

[12] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[13] Deux, O., *et al*, "The Story of O_2", *IEEE Transaction on Knowledge & Data Engineering*, March 1990.

[14] Garey, M., Johnson, D., and Stockmeyer, L. "Some Simplified NP-Complete Problems", *Theor. Comput. Sci.*, Vol.1, 1976, pp.237-267.

[15] Hammer, M. and McLeod, D., "Semantic Integrity in Relational Database Systems", *Proc. 1st Very Large Data Bases*, September 1975, pp.25-47.

[16] *IEEE Transaction on Knowledge & Data Engineering*, Special Edition on Next Generation Database System, M. Stonebraker, Ed., Vol. 2, No. 1, 1990.

[17] Jarke, M. "Common subexpression isolation in multiple query optimization", *Query Processing in Database Systems*, W. Kim, D. Reiner, and D. Batory, Eds., Springer, New York, 1984.

[18] Klug, A., "On conjunctive queries containing inequalities," *J. ACM*, Vol.35, No.1, January 1988, pp.146-160.

[19] Kim, W., *et al.*, "Integrating an Object-Oriented Programming System with a Database System", *Proc 2nd Intl Conf. OOPSLA*, San Diego, September 1988.

[20] Kim, W., "A model of queries for object-oriented databases", *Proc. 15th Intl Conf. Very Large Data Bases*, Amsterdam, The Netherlands, August 1989.

[21] Kim, W., "Object-Oriented Databases: Definition and Research Directions", *IEEE Trans. on Knowledge & Data Engineering*, Vol. 2, No. 3, September 1990, pp. 327-341.

[22] Maier, D., Stein, J., Otis, A., and Purdy, A., "Development of an Object-Oriented DBMS", *Proc. of ACM OOPSLA*, Portland, Oregon, October 1986.

[23] Rosenkrantz, D.J., and Hunt III, H.B., "Processing Conjunctive Predicates and Queries," *Proc. 6th Intl. Conf. on Very Large Data Bases*, 1980, pp.64-72.

[24] Sellis, T. "Global Query Optimization", *Proc. ACM SIGMOD*, 1986, pp.191-205.

[25] Shenoy, S. T. and Ozsoyoglu, Z. M., "Design and Implementation of a Semantic Query Optimizer", *IEEE TKDE*, Sept, 1989, pp 344–361.

[26] Sun, W., and Weiss, M.A. "An Efficient Algorithm for Testing Implication Involving Arithmetic Inequalities", to appear in *IEEE Trans. on Knowledge and Data Eng.*.

[27] Sun, W., and Yu, C., "Semantic Query Optimization for Tree and Chain Queries", to appear in *IEEE Trans. on Knowledge & Data Engineering*.

[28] Ullman, J.D. *Principles of database and knowledge-base systems Vol. I & II*, Computer Science Press, 1989, pp.885-892.

[29] Willshire, M.J., "How spacey can they get? space overhead for storage and indexing with object-oriented databases", *IEEE Int'l Conf on Data Engineering*, Kobe, Japan, 1991, pp.14-22.

[30] Wong, E., and Youssef, K., "Decomposition – A Strategy for Query Processing", *ACM Trans. on Database Systems*, Vol. 1, No. 3, Sept. 1976

[31] Yu, C., Guh, K., Brill, D., and Chen, A. "Partition strategy for distributed query processing in fast local networks", *IEEE Trans. on Software Engineering*, 1989, pp. 780-793.

[32] Yu, C., and Sun, W., "Automatic Knowledge Acquisition for Semantic Query Optimization", *IEEE Trans. on Knowledge & Data Engineering*, September 1989, pp.362-375.

[33] Zaniolo, C. "The database language Gem", *ACM-SIGMOD*, San Jose, California, May 1983, pp.207-217.

ISBN 0-9641699-0-8 (paper)

Additional copies can be ordered from

Knowledge Systems Institute
3420 Main Street
Skokie, IL 60076
USA

Printed in the United States of America