91-KQ

Knowledge-based Query Optimization in an Object-Oriented Database System[*]

Wei Sun[‡], Naphtali Rishe[†], Yuzheng Ding[‡], Weiyi Meng[§] AND Chengwen Liu[§]

Abstract

Object-oriented models have attracted much attention from the database community recently. ***Semantic integrity constraints (SICs)*** and their applications in query optimization in traditional databases have received extensive studies. In this paper, we propose an approach to optimize queries using SICs in an object-oriented database system (OODB). The concepts of SICs are generalized in an OODB environment by incorporating many distinct object-oriented features such as IS-A class hierarchies (subclass assertions), class traversals in specifying selection predicates, and classification concepts (which cluster relevant domain values). Solutions to the problems introduced by incorporating these features are presented. Given a query and a set of SICs, knowledge-based or semantic query optimization (SQO) is performed by revealing contradictions, replacing reference to a class by that to its *most specific subclass*, eliminating unnecessary class traversals, and adding/eliminating useful/useless redundant restrictions. We also show that the time complexity of the proposed strategy is bounded by $O(n^3)$, where n is the larger of the number of classes involved in the query, and the number of SICs involved.

1 Introduction

The object-oriented technique has been widely applied in different disciplines of computer sciences in recent years, including the database community. Object-oriented database systems have been developed in recent years. Some systems such as GemStone [3, 12], Vbase and its successor Ontos [1], ORION [8, 9, 10] and O2 [19, 5] are now commercially available. [6] presented several prototype systems which are among the most representative new-generation database systems.

Issues like generalization/specialization, inheritance, persistence etc. are well addressed in an OODB/OO model. However, issues like specifying restricted associations/relationships among objects, or constraints among states of objects of the same class or different classes have been rarely discussed in current literature. ***Semantic integrity constraints (SICs)*** which were probably first introduced in [7] in the context of relational database systems have been widely used in many current traditional database systems. In [14], SICs in OODBs are proposed, which will capture, to a significantly higher degree, the semantics of real-world objects and their relationships in an OODB. In this paper, SICs are first generalized by incorporating ***(membership/subclass) assertions***[18], which are used to assert the membership of an object in a class to a subclass of this class, ***bilateral class traversals***, and ***classification concepts*** which are used to represent a set of relevant atomic domain values.

It has been shown that semantic integrity constraints can be used in relational or deductive database systems to optimize users' queries (known as ***semantic*** or *knowledge-based* query optimization) so that query processing cost can be reduced [13, 11, 4]. However, there has been little discussions on query optimization in an OODB using SICs, which is the focus of this paper.

Query optimization in an OODB environment is different from that in traditional (say, relational) systems. Bilateral class traversals and *IS-A* hierarchy traversals are involved in a query qualification in an OODB, while *joins* are involved in a query qualification in a relational database. The former may have much more complicated structure. As a result, SICs and the strategies used in this study have generalized those used in traditional models/systems, as in [13, 11, 4], by incorporating new OO features IS-A assertions, classification concepts, and bilateral traversals. We note that a semantically optimized query may be further optimized by a conventional query optimizer to achieve certain machine-dependent effects. The following optimization goals have been identified, presented in the decreasing order of priorities. (1) detect a contradiction in a query qualification; (2) replace all references in a query qualification to a class by its most specific subclass(es); (3) eliminate unnecessary and redundant class traversals; (4) add useful (redundant) restrictions; and (5) eliminate unnecessary restrictions.

Goal (1) is important since a query qualification imply-

---

[*] This research is partially supported by a grant from Florida High Technology and Industrial Council.

[†] School of Computer Science, Florida International University, Miami, FL 33199.

[‡] Computer Science Department, University of California at Los Angeles, Los Angeles, CA 90024.

[§] EECS Dept., University of Illinois, Chicago, IL 60680

ISSN# 0-7803-0233-8/91 $1.00©1991 IEEE

ing a contradiction will yield a null answer; (2) is unique in an OODB environment. Since only subclasses, instead of original (super) classes, will be accessed, and accessing an original class implies accessing all subclasses of the original class, there will be a potential significant cost saving.[1]. (3) is very desired since class traversals are usually costly. Goals (4) and (5) will gain additional benefits by reducing unnecessary evaluation of redundant selection predicates (restrictions) and/or introducing predicates on indexed attributes (so that indices may be used to speed up accessing classes). The proposed semantic query optimizer will realize the above goals in sequence. Furthermore, we will also discuss how constraints can be properly managed and some of the useful properties of constraints under an OODB.

The rest of this paper is organized as follows. *Section 2* introduces some related concepts and notations. *Section 3* discusses distinct properties of constraints and restrictions in an OODB. A strategy to compute implied restrictions by a query qualification under a set of SICs is presented in *Section 4*; *Section 5* proposes a framework of semantic query optimization. Future works are briefly discussed in *Section 6* which also concludes this paper. Due to space limit we have to omit all the proofs and most of formal descriptions. Interested readers may consult with [17] and [14].

2 OODBs, SICs, and Queries

In this paper, we basically adopt the constraints proposed in [14]. The proposed constraints incorporate many distinct features of an OO model and OODB.

Although there is no consensus so far in both research and industrial communities on what an OO model or an OODB is, some general characteristics and features that an OO model and OODB system should possess were pointed out in [10], as can be briefly summarized as follows: Any entity is uniformly modeled as an *object*, identified by a system-wide and persistent UID. Every object has a *state* (the set of values for the attributes of the object) and a *behavior* (the set of *methods* or programs operating on the state of the object). Objects with similar properties are clustered into classes. Objects in a class will share the same set of attributes and methods. A class can be a *subclass/superclass* of another one. A subclass inherits all methods/attributes of its superclasses, and additional methods and attributes can be specified for a subclass.

Figure 1 shows the illustrative structure of a sample vehicle database.[2] Attributes ended with a * are *primitive* ones whose objects do not have attributes. Domain classes of non-primitive attributes are explicitly attached (shown by an arrow followed by its domain class), *e.g.* class *Engine* is the domain class of the attribute ***drivetrain*** of class ***Vehicle***. ***IS-A*** relationships are also specified among classes, e.g. class ***Sportscar*** *IS-A* class ***Vehicle***. Objects in class Sportscar inherit all attributes of ***Vehicle***, while possesses additional attribute(s) like ***maxspeed***.

Class Vehicle(V)
manufacturer→C
model*
price*
year*
load*
drivetrain→E

Class Engine(E)
cylinder#*
power*
weight*

Class Sportscar(S)
ISA V
maxspeed*

Class Company(C)
name*
country*
president→P

Class Person(P)
name*
birthdate*
owncar→V

Figure 1: A Sample Vehicle OODB

Classification concepts (*CCs* for brief) are used in this paper. A *CC* is a non-atomic value associated with a domain, representing a set of atomic values. For example, *Asia* could be a CC on the domain of the attribute *country*, and so is *Foreign*, which is one level higher than *Asia*. The CC *Asia* may represent the domain values of countries such as *Japan, China, Korea,* and so on. In general, CCs may form lattice-like hierarchies. The root of such a hierarchy can be viewed as the whole domain, and all the leaves are subsets of the domain values. It is assumed that CCs and their hierarchies can be properly maintained in an OODB system, and are easily accessible by a query optimizer. (Details can be found in [15].)

Attribute X of class A can be specified as X_A. A specification of an attribute can also involve ***forward/backward class traversals*** and/or ***IS-A hierarchy traversal***. A *forward* class traversal, in the form $X_A.Y_B$, specifies attribute Y of class B, while class B is the domain class of attribute X of class A; a ***backward*** class traversal, $X_A\text{-}[Y_B]Z_B$, specifies attribute Z of class B while class A is the domain class of attribute Y of class B. Class traversals represent cross-class associations of objects.

A ***restriction*** on an attribute is of the form *Attr_Spec* op *c*, where *Attr_Spec* specifies the attribute, as described above, **op** $\in \{<, \leq, =, \neq, \geq, >\}$, and c is a domain value, or a classification concept CC if **op** is $=$ or $\neq$ (since CCs may not be partially ordered). If *Attr_Spec* is of the simple form X_A, the restriction is said ***simple***.

An ***(subclass) assertion*** is of the form *class*1(*class*2), a boolean function on objects of class *class*2, where *class*1 is a subclass of *class*2. It is *true* if the evaluated object of *class*2 in fact belongs to the subclass *class*1 and is *false*

---

[1] We assume that all classes are independent files in the underlying physical storage system. Systems such as Orion[2, 8] adopted this scheme.

[2] Examples are for illustration only and do not necessarily represent real-world situations.

otherwise. Assertions may also be referred to as a restriction if no ambiguity occurs.

A *Semantic Integrity Constraint (SIC)* is of the form LHS $\Rightarrow$ RHS, where LHS (may be empty) and RHS are restrictions/assertion functions in conjunction, representing that whenever LHS is *true* on an object, RHS should also be *true* on the object.

The following example shows some SICs, together with restrictions and assertions, that may be applicable to the sample OODB in *Figure 1.*

Example 1.

domain assertion: The load of any vehicle is at least 500*lbs*. ($\Rightarrow load_V \geq 500$).

in-class SIC: Toyota is a Japanese Company. ($name_C =$ "Toyota" $\Rightarrow country_C = Japan$).

cross-class SICs: "Corvette is a kind of American sportscar" ($model_V =$ "Corvette" $\Rightarrow manufacturer_V.country_C =$ America$\wedge Sportscar(V)$);
"A car of 200hp or higher must be a sportscar" ($drivetrain_V.power_B \geq 200 \Rightarrow Sportscar(V)$);
"Any '91 sportscar has maximum speed above 150*mph* and costs more than \$10,000."
($year_S = 1991 \Rightarrow maxspeed_S > 150 \wedge price_S > 10,000$);
Only Japan makes engines of weight less than 500*lbs*.
($weight_B < 500 \Rightarrow weight_B$-$[drivetrain_V]$ $manufacturer_V.country_C = Japan$). □

SICs in a database can be explicitly specified by application users and/or DBAs. [20, 16] showed many situations that SICs can be automatically or semi-automatically acquired.

A *query* is assumed to be of the form $Q[c : q]$, where c, the *classname*, is called the *target*, and q is the query's *qualification* consisting of restrictions in conjunction. The evaluation of the query will return all uid's of qualified objects in class c[10]. How the qualified objects are presented to an user/application is a different and separate issue. For simplicity, we further assume that the qualification of a query must be relevant to the target, i.e., any class involved in the qualification must be either the target class itself or reachable from the target via class traversals and/or *IS-A* traversals, in which case the traversal path(s) should also be specified in the qualifications. (in relational terms, cross-products are avoided.) The following is a sample query.

Example 2. The query "find all engines of weight < 500*lbs* that are made in Japan" can be expressed as $Q[E : weight_B < 500 \wedge weight_B$-$[drivetrain_V]manufacturer_V.country_C =$ Japan$]$. □

Queries $Q_1[c : q_1]$ and $Q_2[c : q_2]$ are said *(semantically) equivalent* under a set of SICs S (denoted as $Q_1 \cong_S Q_2$) if they return the same set of objects. In the case S is obvious or irrelevant, we just say $Q_1 \cong Q_2$.

Example 3. The query in *Example 2* is equivalent to the query $Q[E : weight_B < 500]$ under the SICs in *Example 1.* □

3 The Knowledge Base

The knowledge base consists of all SICs and an *inference engine.* An *inference engine* will take a set of known facts (such as restrictions from a query qualification), deduce those implied restrictions/assertions under the SICs. The purpose of semantic query optimization is to transform a query qualification into another equivalent one under the SICs such that it costs less to evaluate the transformed qualification. The first step towards this is to find out as many restrictions as possible that are implied by the original query qualification under the SICs by using an inference engine. This is essential to achieve all the 5 optimization goals listed in Section 2. In a relational database system, this issue have been discussed in [20]. However, SICs proposed here are more complicated and different from those proposed in relational database systems, the strategies applicable to relational systems may not be directly applicable. The following show several cases that have not been addressed in relational database systems, which are unique to an OODB system.

Example 4.

1. Suppose we know that the restriction ($weight_B > 500$) is given. Then, ($drivetrain_V.weight_B > 450$) is also true. In other words, if ($drivetrain_V.weight_B > 450$) is a restriction in the LHS of a constraint, then it is satisfied.
2. Assertions and IS-A relationships among classes have to be addressed. For example, a constraint which holds on a class also holds on its subclasses. References to a class in a query qualification may also be semantically equivalently replaced by references to its subclasses under certain circumstances.
3. Suppose we have a constraint $weight_B < 500 \Rightarrow weight_B$-$[drivetrain_V]manufacturer_V.country_C =$ Japan. Then $drivetrain_V.weight_B < 500 \Rightarrow manufacturer_V.country_C =$ Japan can also be asserted. This requires analyzing semantic relationships among *Attr_Specs*. □

The above examples also show that there is a need to properly manage these constraints in the knowledge base in order to support deductions and inferences by the inference engine. As discussed above, we are interested in computing the restrictions implied by a query qualification under a set of constraints. This requires solutions for the following problems.

First of all, we consider the implication of restrictions. A restriction r is ***arithmetically deducible*** from a set of restrictions R if $r \in R$ or r is deducible from R by certain simple computations on the domain values and/or CC values. For example, $x = 3 \overset{d}{\leftarrow} x > 2 \wedge x < 4$ if the domain values for x are integers, where $\overset{d}{\leftarrow}$ represents the deduction. Arithmetic deduction of this kind or the like has been studied (for algorithms see [20]). However, in our case, classification concepts and attribute specification involving class traversals are involved. We believe that CC can be easily incorporated into the formula. Details for CC manipulations can be found in [15]. The following proposition may be useful in dealing with the latter issue. Again, we state all propositions without proofs due to space limit.

Proposition 1. A restriction ($\beta.\alpha$ **op** c_1) is *implied* by (α **op** c_2) if α is a *Attr_Spec*, β is a class traversal sequence (may be empty) that can make $\beta.\alpha$ a proper *Attr_Spec*, and (x **op** c_1) is arithmetically deduced from (x **op** c_2) for any x in the domain class of α. □

More complicated cases involving a set of restrictions can be handled in a similar manner.

Secondly, we turn to the implication problem of assertions. It is important to deduce the implied (sub)class assertions because more specific class assertions can be used to replace occurrences of the references to their superclasses, and enable the inference engine to apply constraints whose LHSs have assertions. We say an assertion $c_1(c_3)$ is *transitively deducible* from a set of assertions AS if (1) $c_1(C_3) \in AS$, or (2) $c_1(c_2)$ and $c_2(c_3)$ are *transitively reducible* in AS.

Proposition 2. An assertion $\alpha : c_1(c_2)$ is *implied* by a set of assertions AS if and only if $\alpha \in AS$ or α is transitively deducible from AS. □

Let $r_{[c_2|c_1]}$ denote the restriction formed by substituting all occurrences of classname c_1 by classname c_2 in the restriction r. The following can be directly observed.

Proposition 3. Restriction r implies $r_{[c_2|c_1]}$ if c_2 ISA c_1. □

Proposition 4. If c_2 ISA c_1, then assertion $c_2(c_1)$ and restriction r implies $r[c_1|c_2]$. □

Now we start to identify several cases where constraints can be transformed into other forms so that certain drawbacks in earlier approaches can be overcome, and distinct OO features can be incorporated. The first one is trivial.

Proposition 5. If $r_1 \Rightarrow r_2$ is a constraint in the knowledge base, where r_1 and r_2 are restrictions, then $\neg r_2 \Rightarrow \neg r_1$ is also true. □

The following four statements involve class traversals in specifying *Attr_Spec*:

Proposition 6. Let $X_A\alpha_1 \Rightarrow Y_A\alpha_2$ be a constraint that holds in a database, where X and Y are attributes of the same calss A, α_1 and α_2 are proper ending sequences of the corresponding restrictions. For any traversal sequence β, if the domain class of β is A, then $\beta \cdot X_A\alpha_1 \Rightarrow \beta \cdot Y_A\alpha_2$ is also a constraint that holds in the database. □

It is said that a *referential integrity constraint (RIC)* exists from class A to class B on attribute X if X_B references every object in X_A. This definition, as well as the following corollary, is adopted from [16] with minor change.

Proposition 7. if $Z_B\beta \cdot X_A\alpha_1 \Rightarrow Z_B\beta \cdot Y_A\alpha_2$ is a constraint, and there exists a RIC for every pairs of attributes of adjacent classes along the traversal path from A to B, then $X_A\alpha_1 \Rightarrow Y_A\alpha_2$ is also a constraint. □

The following results show the transformations between forward and backward traversals.

Proposition 8. If $X_A\alpha_1 \Rightarrow X'_{A\text{-}}[Y_B]Z_B\alpha_2$ is a constraint, where A is the domain class of Y_B, α_1 and α_2 are proper ending sequences of the corresponding restrictions, then $Y_B \cdot X_A\alpha_1 \Rightarrow Z_B\alpha_2$ is also a constraint. □

Proposition 9. If $Y_B \cdot X_A\alpha_1 \Rightarrow Z_B\alpha_2$ is a constraint, then $X_A\alpha_1 \Rightarrow X_{A\text{-}}[Y_B]Z_B\alpha_2$ is also a constraint. □

The above rules can be used to organize the knowledge base and used by the inference engine in deduction.

4 Restriction Extension

Given a set of restrictions (in conjunction) R, a restriction r is *deducible* from R, denoted as $r \overset{d}{\leftarrow} R$, if $r \in R$ or r is implied by R according to ***Propositions 1 - 4***. We denote the resulting set by removing r from R (if $r \in R$; or R itself if $r \notin R$) as $R \setminus r$.

Given a set of restrictions R (in conjunction) and a set of SICs S, a *restriction extension* of R under S, denoted as R^*_S, is a set of restrictions/assertions that satisfies the following:

1. $R \overset{d}{\leftarrow} R^*_S$;
2. if $s : L \Rightarrow r \in S$, and $L \overset{d}{\leftarrow} R^*_S$, then $r \overset{d}{\leftarrow} R^*_S$;
3. R^*_S contains only those the above specified.

Intuitively, R^*_S contains restrictions that are implied R under S. It is semantically equivalent to R.

Proposition 10. $Q[c : q] \cong_S Q[c : q^*_S]$. □

An algorithm that follows the above definition to compute R^*_S can be easily constructed, similar to that in [20], except that the applicability of constraints is checked according to ***Propositions 1 - 4***, and the application of SICs may result in certain implied restrictions as discussed in last section. The key step for computing this restriction extension is to repeat the application of SICs whose LHS have been satisfied to deduce new restrictions (the RHSs). Practically, it is reasonable to assume that the number of attributes involved in a single class can be bounded by a constant, so does the number of levels of each involved CC hierarchy, and the number of restrictions in the LHS of each SIC. Under such assumptions, if both the number of classes involved in the query and the number of relevant SICs are bounded by $O(n)$, the restriction extension can be computed in $O(n^3)$ time. Furthermore, it should be noted that the algorithm can easily detect contradiction if any.

Example 5. Given a restriction set $R : \{year_V = 1991, model_V =$"Corvette"$\}$, and S consists of the SICs in *Example 1*. R_S^* will include $\{load_{V/S} \geq 500$, $manufacturer_{V/S}.country_C =$America, $Sportscar(V)$, $price_{V/S} \geq 10{,}000$, $maxspeed_S \geq 150$, $year_V = 1991, model_V =$"Corvette"$\}$. If the query is $Q[V : R \wedge price_V < 9{,}000]$, a contradiction will be detected. □

5 Semantic Query Optimization

In this section, we discuss how to achieve the five optimization goals proposed above.

In computing restriction extension, a contradiction, if existed, will be detected by the algorithm. If it is not the case, the restriction extension provides all restrictions implied/deduced from the query qualification under the constraints. We then try to restrict the access or references to classes needed by the query to those references to the necessary (sub)classes based on the assertions deduced/implied. More precisely, we want to substitute all occurrences of classname A in restrictions by classname B, where B is a subclass of A. Since the size of a subclass is monotonically smaller than that of its superclass, there is a clear gain by doing so.

Proposition 11. Given a query $Q[c : q]$, if an assertion $c_2(c_1) \in q$, then if $c_1 \neq c$, $Q[c : q] \cong_S Q[c : q[c_2|c_1]]$; if $c_1 = c$, $q[c : q] \cong_S q[c_2 : q[c_2|c]]$. □

First, all implied assertions are grouped according to *IS-A* hierarchies. For assertions in an *IS-A* hierarchy, it is possible to partially order all assertions involved.[3] Let $c_1, c_2, ..., c_m$ be such an order, where $c_{i+1}(c_i) \in R_S^*, i = 1, ..., (m-1)$, and no assertion of the form $c(c_m)$ in R_S^*. Class c_m is said to be the ***most specific class*** for $c_i, i = 1, ..., m$. Restrictions in R_S^* are then examined and all occurrences of references to $c_i, i = 1, ..., (m-1)$ will be replaced by the most specific class c_m. Furthermore, if c_1 is the target class of the query, then the target is changed into c_m too.

Example 6. Suppose the query is "Report all vehicles whose engine power are above 400 hp." ($Q[V : drivetrain_V.power_E > 400]$). In order to evaluate this query, class *Vehicle* including its subclass *Sportscar* will be accessed. However, since we know that $drivetrain_V.power_E > 200 \Rightarrow Sportscar(V)$. Therefore, $Sportscar(V)$ is in the restriction extension. By applying the above strategy, the original query can be semantically equivalently transformed into: $Q[S : drivetrain_S.power_E > 400]$. In the latter case, only a much smaller class *Sportscar* is accessed. □

Class traversals are rather costly operations. Therefore, it is very desirable that certain unnecessary traversals can be eliminated. In our case, eliminating class traversals is a special form of eliminating redundant restrictions. A restriction r is said ***redundant*** with respect to q under S if $Q[c : q] \cong_S Q[c : q \setminus r]$.

Proposition 12. If $L \Rightarrow r \in S$, $L \overset{d}{\leftarrow} q$, then $Q[c : q] \cong_S Q[c : q \setminus r]$. □

If eliminating r would yield fewer classes to be traversed in evaluating the query, then class traversals are eliminated; otherwise, redundant restrictions are eliminated. The priority goal at this stage is to eliminate as many unnecessary class traversals as possible. A similar problem has been shown to be NP-hard [13]. We therefore propose to employ the following heuristics: frist sort restrictions in the restriction extension by the number of classes involved in the restriction in decreasing order. We then randomly consider one restriction from all the restrictions involving the same number of traversals in the above decreasing order and test whether it is redundant with respect to the rest of the extension under SICs. If yes, it is eliminated. In this way, it is likely that the number of classes to be traversed is decreased, since restrictions involving large number of traversals are likely to be eliminated first; We repeat this process until the last restriction that involves traversal is tested.

Now we apply the similar strategy to test the redundancies of simple restrictions, but only eliminate ***useless*** ones. A restriction is said ***useful*** if it is a ***simple*** restriction on an indexed attribute; otherwise it is said ***useless.*** Intuitively, ***useful*** redundant restrictions may help reduce query evaluation cost by using the fast-access pathes. Since all implied restriction are included in the restriction extension, as a by product, all useful ones are also there already.

It is easy to see that under the same assumptions as used in last section, the time complexity of the above transformations are also bounded by $O(n^3)$.

Example 7. Assume we have the following constraints "Ferrari is a French Sportscar" ($model_V =$"Ferrari"$\Rightarrow$

---

[3] Although it is possible that a class may have more than one subclass, it is a contradiction that there exist more than one assertion that has the same parent class assertion in the restriction extension under a common assumption that classes represent objects exclusively. This type of contradiction is due to conflicting assertions due to IS-A hierarchy, which is clearly unique to an OODB.

$manufacturer_V.country_C =$"France"$\wedge Sportscar(V)$), "the price of any new Ferrari car is at or above \$40,000" ($year_V \geq 1991 \wedge model_V =$"Ferrari" $\Rightarrow price_V \geq \$40,000$), "the engine weight of any Ferrari car is above 600*lbs*"($model_V =$"Ferrari"$\Rightarrow drivetrain_V.weight_B > 600$), and "any engine of weight 500*lbs* or higher must have 6 or more cylinders" ($weight_B \geq 500 \Rightarrow cylinder\#_B \geq 6$). The query is "find all new Ferrari cars that have more than 4 cylinders" ($Q[V : year_V \geq 1991 \wedge model_V =$"Ferrari"$\wedge drivetrain_V \cdot cylinder\#_B > 4]$). We also assume the class V (as well as S) is clustered (indexed) by the attribute *price* in the underlying system. Using the above scheme, it can be followed that the resulting query will be $Q[S : model_S =$"Ferrari"$\wedge year_S \geq 1991 \wedge price_S \geq 40,000]$. The benefit is obvious: the access to class V is reduced to the access to S, redundant traversal is removed, and a restriction on the indexed attribute is added. If only 5% of the vehicles in the system are sportscars, and 80% of sportscars cost less than \$40,000, the query cost then may be reduced to about 1% of that of the original one. □

Example 8. If the query in last example is "find all 4-cylinder Ferrari cars" ($Q[V : model_V =$"Ferrari"$\wedge drivetrain_V.cylinder\# = 4]$), then during the construction of q^*_S, the restriction $drivetrain_V.cylinder\# \geq 6$ will be added, and a contradiction be detected. The query result is empty without evaluating the query at all. □

6 Conclusion

In this paper, an approach to optimize queries in an OODB environment using semantic integrity constraints is proposed. The concepts of restrictions and SICs are generalized in an OODB environment by incorporating certain OO features such as classification concepts, bilateral class traversals, and subclass assertions. Solutions to problems introduced by incorporating these new features in deciding the applicability of constraints and semantically optimizing queries are discussed.

There are several related issues that deserve further investigations. First, how to efficient maintain and manage the knowledge base; Secondly, how effective the optimization system is; and thirdly, how a semantic query optimizer and a conventional optimizer can be properly integrated together such that better performance can be achieved.

We are very grateful to Dr. C. Yu for many helpful discussions.

References

[1] Andrews, T., and Harris, C., " Combining Language and Database Advances in an Object-Oriented Development Environment", *Proc. of ACM OOPSLA*, Orlando, Florida, Oct. 1987.

[2] Banejee, J., *et al.*, "Data model issues for object-oriented applications", ACM Trans. on Office Information Systems, 5(1):3-26, Jan. 1987.

[3] Bretl, R., *et al.*, "The GemStone data management system", in *Object-Oriented Concepts, Applications and Databases"*, Won Kim and F. Lochovsky, Eds., Reading, MA: Addision-Wesley, 1989.

[4] Chakravarthy, U., Grant, J., and Minker, J.: "Logic-based approach to semantic query optimization", *ACM TODS*, June 1990, pp. 162-207

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

[6] *IEEE Transaction on Knowledge and Data Engineering*, Special Edition on Next-generation Database System, M. Stonebraker, Ed., Vol. 2, No. 1, 1990.

[7] Hammer, M. and McLeod, D., "Semantic integrity in relational database systems", *in Proc. 1st Very Large Data Bases*, pp. 25-47, Sept. 1975.

[8] Kim, W., *et al.*, "Integrating an object-oriented programming system with a database system", *Proc 2nd Int'l Conf. OOPSLA*, San Diego, Sept., 1988.

[9] Kim, W., "A model of queries for object-oriented databases", in *Proc. 15th Int'l Conf. Very Large Databases*, Amsterdam, The Netherlands, Aug. 1989.

[10] Kim, W., "Object-oriented databases: definition and research directions", *IEEE Trans. on Knowledge and Data Eng.*, pp. 327-341, Vol. 2, No. 3, Sept., 1990.

[11] King, J., *Query Optimization by Semantic Reasoning*, Ann Arbor, UMI Research Press, MI, 1984.

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

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

[14] Sun, W., "Semantic Constraints in Object-Oriented Database Systems", *Proc. of 3rd Int'l Conf. on Software Engineering and Knowledge Eng.*, Skokie, IL, June 1991.

[15] Sun, W., et.al. "Using Classification Concepts to Represent Semantics in OODB systems", manuscript, 1991.

[16] Sun, W., et.al. "Automatic Identification of Semantic Integrity Constraints in Object-Oriented Databases", *this proceeding*.

[17] Sun, W., et.al. "Semantic Query Optimization in Object-Oriented Database Systems", *Technical Report*, Florida International University, 1991.

[18] Sun, W. and Yu, C., "IS-A Relationship Revisited", manuscript.

[19] Velez, F., Bernard, G., and Darnis, V., "The O2 object manager: An overview", in *Proc. 15th VLDB*, Amsterdam, The Netherlands, Aug. 1989.

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

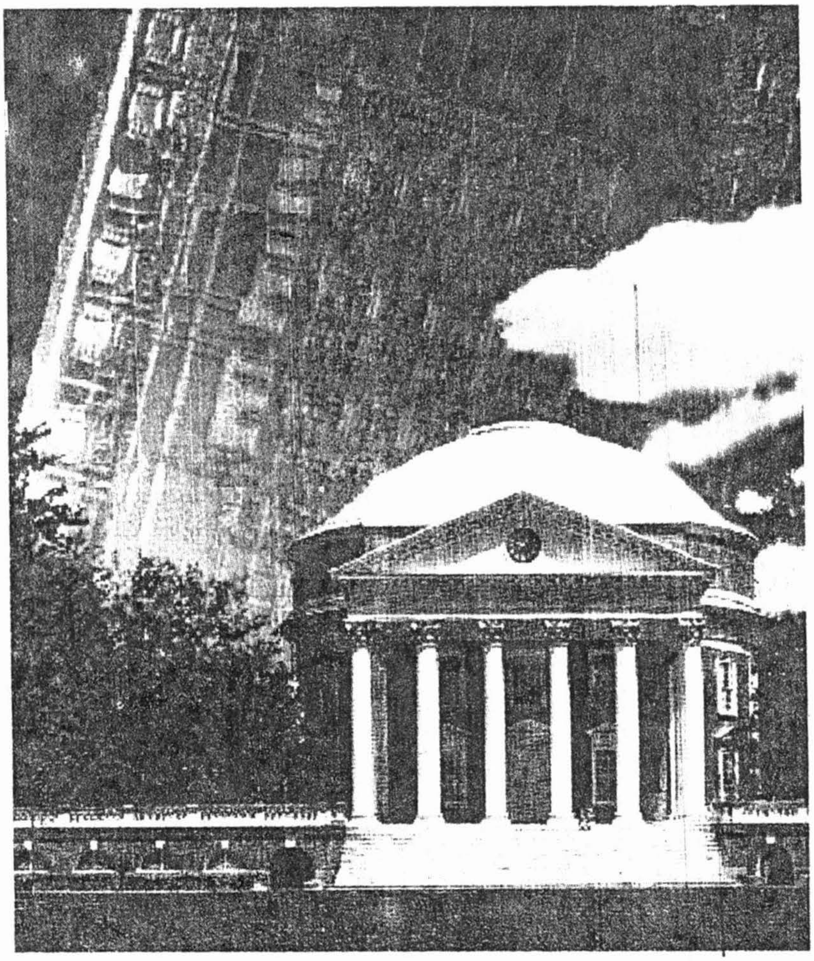
Naph

Conference Proceedings

1991 IEEE / SMC

International Conference
on
Systems, Man, and Cybernetics

October 13-16, 1991



Omni Charlottesville Hotel
and the
University of Virginia
Charlottesville, Virginia

Volume 3

IEEE

91CH3067-6

Conference Proceedings

1991 IEEE International Conference on Systems, Man, and Cybernetics

"Decision Aiding for Complex Systems"

October 13-16, 1991

Omni Charlottesville Hotel
and the
University of Virginia
Charlottesville, Virginia

IEEE

Volume 3

91CH3067 - 6

SCHOOL OF
ENGINEERING
& APPLIED SCIENCE

1991 IEEE International Conference on Systems, Man, and Cybernetics

IEEE Catalog Number: 91CH3067-6
Softbound: 0-7803-0233-8
Casebound: 0-7803-0234-6
Michofiche: 0-7803-0235-4
Library of Congress Number: 91-58127

Additional copies of this publication are available from

IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854-4150

1-800-678-IEEE

1991 IEEE
International Conference
on
Systems, Man, and Cybernetics

Organizing Committee

General Chair:	Chelsea C. White, III University of Michigan
Program Chair:	Donald E. Brown University of Virginia
Invited Sessions and Tutorials:	Donald E. Brown University of Virginia
Contributed Sessions:	Julia Pet-Edwards University of Virginia
Publications:	K. Preston White, Jr. University of Virginia
Finance:	Edward A. Sykes University of Virginia
Publicity & Public Relations:	Stephen G. Strickland University of Virginia
Local Arrangements:	William T. Scherer University of Virginia