Subsumption for Semantic Query Optimization in OODB*

Domenico Beneventano°, Sonia Bergamaschi◇, Claudio Sartori°
CIOC-CNR

Abstract

The purpose of semantic query optimization is to use semantic knowledge (e.g. integrity constraints) for transforming a query into an *equivalent* one that may be answered more efficiently than the original version. This paper proposes a general method for semantic query optimization in the framework of OODBs (Object Oriented Database Systems). The method is applicable to the class of conjunctive queries and is based on two ingredients: a description logic able to express both class descriptions and *integrity constraints rules* (IC rules) as types; *subsumption* computation between types to evaluate the logical implications expressed by IC rules.

1 Introduction

In database environment, semantic knowledge is usually expressed in terms of IC rules, that is *if then* rules on the attributes of a *database schema* (i.e., roughly a Tbox of a Terminological Knowledge Representation System (TKRS)). Informally, *semantic equivalence* means that the transformed query has the same answer as the original query on all databases satisfying the IC rules. The notion of semantic query optimization for relational databases was introduced in the early 80's by King [13, 14]; Hammer and Zdonik [10] independently developed very similar optimization methods. The key idea in [13, 14], as well as in [10], is that IC may not only be utilized to enforce consistency of a database but also to optimize user queries. During the last decade, many efforts have been made to improve this technique and to generalize it to deductive databases [17, 18, 19, 8]. More recently, some efforts have been made to perform semantic query optimization in OODBs [9, 11, 16, 6, 1, 2]. The main point is that OODBs provide a very rich type (class) system able to directly represent a subclass of integrity constraints in the database schema. By exploiting schema information

*This research has been partially funded by the LOGIDATA+ project of the National Research Council (CNR).
° Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Viale Risorgimento 2,I-40136 Bologna, Italy.
◇ Facoltà di Ingegneria, Università di Modena, Via G. Campi 213/A, I-41100 Modena, Italy.
e_mail:
{ dbeneventano, sbergamaschi, csartori }@deis.unibo.it

as, for instance, inheritance relations between types (classes), it is possible to perform semantic query optimization.

This work represents an improvement and a generalization of the cited works in the database area for many aspects. Firstly, our method applies to OODBs and, with respect to previous works, we generalize the notion of a database schema as a set of rules expressing *inclusion statement* between general types. Secondly, we provide the theoretical framework (in term of *subsumption*) for developing a theory of semantic query optimization which includes the main query transformation criteria proposed in the database literature. Inclusion statements between concepts, recently proposed in [5] to express a TKRS, constitute a generalization of description logic perspective. This new perspective perfectly fits the usual database viewpoint. In fact, actual database schemata are given in terms of base classes (i.e. primitive concepts); further knowledge is expressed as IC rules. In particular, structural class descriptions are expressed as rules where the antecedent is a name of the class and the consequent is the class description. More generally, rules allow the expression of integrity constraints with an antecedent and a consequent which are types of the formalism. Since query languages for OODBs are more expressive than our formalism we, following [6], ideally introduce a separation of a query into a *clean* part, that can be represented as a type in our formalism, and a *dirty* part that goes beyond the type system expressiveness. Semantic optimization will be performed only over the clean part of a given query. The clean part of a query, in the following referenced as query, corresponds to the so–called conjunctive queries or single operand queries [12] in OODBs and is a *virtual class* (i.e. a defined concept).

The chosen strategy for optimization is the following. Prior to the evaluation of any query, we *compile*, once at all, the given schema (classes + IC rules), giving rise to an enriched schema obtained by adding (*all the new*) *isa relationships* which are logically implied by the original schema. The compilation process is based on the generation of the *semantic expansion* in *canonical form* (i.e. a form which permits to abstract from different syntactical representation of semantically equivalent types) of the schema types. Following the approach of [17, 19] for semantic query expansion, the semantic expansion of a type, say $EXP(S)$ permits to incorporate any possible restriction which is not present in the original type but is *logically implied* by the type and by the schema. $EXP(S)$ is based on the iteration of this simple transformation: if a type implies the antecedent of an IC rule then the consequent of that rule can be added. Logical implications between these types (the type to be expanded and the antecedent of a rule) are evaluated by means of *subsumption computation* [4, 3, 2].[1]

At run time, we add to the compiled schema the query Q and activate the process again for Q: $EXP(Q)$, possibly including new *isa* relationships is obtained. If new *isa* relationships are found, it is possible *to move the query down* in the schema hierarchy: this is the main optimization achievement of the proposed method. The main points of our optimization strategy are: (1)

[1]The subsumption is similar to the *refinement* or *subtyping* adopted in OODBs [7, 15].

$EXP(Q)$ is the *most specialized query* among the equivalent queries. Moreover, during the transformation, we compute and substitute in the query, at each step, the *most specialized classes* satisfying the query. (2) A filtering activity (*constraint removal*) is performed by detecting the *eliminable* factors of a query, that is, the factors logically implied by the query.

2 Complex Objects and IC Rules

The main extension of the description logic proposed in [2] is the generalization of the notion of a database schema as a set of rules which express *inclusion statement* between general types.

Let $\mathbf{D}$ be the countably infinite set of base-values $(d_1, d_2, \ldots)$ and $\mathbf{A}$ be a countable set of *attributes* $(a_1, a_2, \ldots)$. The set $\mathcal{V}$ of all *values* are built by the following rule: $v \rightarrow d|\{v_1, \ldots, v_q\}|[a_1\colon v_1, \ldots, a_p\colon v_p]$, with $p, q \geq 0$ and $a_1, \ldots, a_p$ distinct in $\mathbf{A}$. Let $\mathbf{B}$ be a countable set of base-type designators $(B, B', \ldots)$ and $\mathbf{C}$ be a countable set of names for *base class* $(C, C' \ldots)$, such that $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are pairwise disjoint. A *path* p is a sequence of elements $p = e_1 . e_2 . \ldots . e_n$, with $e_i \in \mathbf{A} \cup \{\forall\} \cup \{\exists\}$. We denote with ϵ the *empty path*. $\mathbf{S}$ denotes the set of all *type* (S, S', ...) over given $\mathbf{A}, \mathbf{B}, \mathbf{C}$, obtained according to the following syntax rule:

$$\begin{array}{rcl} S & \rightarrow & \top \mid B \mid C \mid [a_1 : S_1, \ldots, a_k : S_k] \mid \\ & & \forall\{S\} \mid \exists\{S\} \mid S \sqcap S' \mid (p : S) \end{array}$$

[] denote the usual type constructor of tuple. Two different set constructors $\forall\{\}$, $\exists\{\}$ are introduced. $\forall\{S\}$ corresponds to the usual set constructor and denotes sets whose elements are *all* of type S; $\exists\{S\}$ denotes sets where *at least* one element is of type S. The construct $\sqcap$ holds for *conjunction*. The *path type* $(p\colon S)$ represents a short notation.

A *database schema* $\mathbf{R}$ is a subset of the cartesian product of a given type system $\mathbf{S}$. An element $R = (S^a, S^c)$ of $\mathbf{R}$ espresses a rule where S^a is the antecedent and S^c the consequent: $S^a \rightarrow S^c$. A rule R is universally quantified over all values $\mathcal{V}$: for all v, if v is of type S^a then v must be of type S^c. Given $\mathbf{R}$, we denote with $\mathbf{R}_\sigma$ the subset of $\mathbf{R}$: $\mathbf{R}_\sigma = \{R \in \mathbf{R} \mid \exists C \in \mathbf{C}, S^a = C\}$ and with σ a total function from $\mathbf{C}$ to $\mathbf{S}$:

$$\sigma(C) = \left\{ \begin{array}{ll} \top & \text{if } \nexists R \in \mathbf{R} : S^a = C \\ \bigsqcap_i S_i^c & \forall R_i \in \mathbf{R} : S_i^a = C \end{array} \right.$$

The subschema $\mathbf{R}_\sigma$ represent the classes and their structures and the function σ associates class names to their descriptions.

Let $\mathcal{I}_\mathbf{B}$ be the (fixed) standard interpretation function from $\mathbf{B}$ to $2^\mathbf{D}$. For a given $\mathcal{V}$, each type S is given a subset of values as its interpretation by the *interpretation function* $\mathcal{I}$ such that: $\mathcal{I}[\top] = \mathcal{V}$, $\mathcal{I}[B] = \mathcal{I}_\mathbf{B}[B]$, $\mathcal{I}[C] \subseteq \mathcal{V}$,

$$
\left.
\begin{array}{l}
\mathbf{R}_\sigma \left\{
\begin{array}{lcl}
\sigma(\texttt{Manager}) & = & [\texttt{name}: \texttt{String}, \texttt{salary}: 40 \div \infty, \texttt{level}: 5 \div 15] \\
\sigma(\texttt{TManager}) & = & \texttt{Manager} \sqcap [\texttt{level}: 8 \div 12] \\
\sigma(\texttt{Material}) & = & [\texttt{name}: \texttt{String}, \texttt{risk}: \texttt{Int}, \texttt{feature}: \forall\{\texttt{String}\}] \\
\sigma(\texttt{SMaterial}) & = & \texttt{Material} \\
\sigma(\texttt{Storage}) & = & [\texttt{managed-by}: \texttt{Manager}, \texttt{stock}: \forall\{\texttt{Material}\}] \\
\sigma(\texttt{SStorage}) & = & \texttt{Storage}
\end{array}
\right. \\
\\
\begin{array}{ll}
(R_1) & \texttt{Material} \sqcap [\texttt{risk}: 10 \div \infty] \rightarrow \texttt{SMaterial} \\
(R_2) & \texttt{Storage} \sqcap (\texttt{stock}.\exists.\texttt{feature}.\exists: \texttt{''F1''}) \rightarrow [\texttt{managed-by}: \texttt{TManager}] \\
(R_3) & \texttt{Storage} \sqcap [\texttt{stock}: \exists\{\texttt{SMaterial}\}] \rightarrow \texttt{SStorage}
\end{array}
\end{array}
\right\} \mathbf{R}
$$

Table 1: The Company domain schema

$\mathcal{I}[S \sqcap S'] = \mathcal{I}[S] \cap \mathcal{I}[S']$ and

$$
\begin{aligned}
\mathcal{I}[\forall\{S\}] &= \Big\{\{v_1, \ldots, v_p\} | v_i \in \mathcal{I}[S], 1 \leq i \leq p\Big\} \\
\mathcal{I}[\exists\{S\}] &= \Big\{\{v_1, \ldots, v_p\} | \exists i, 1 \leq i \leq p, v_i \in \mathcal{I}[S]\Big\} \\
\mathcal{I}[[a_1: S_1, \ldots, a_p: S_p]] &= \Big\{ \ [a_1: v_1, \ldots, a_q: v_q] | \\
& \qquad\quad p \leq q, v_i \in \mathcal{I}[S_i], 1 \leq i \leq p\Big\}
\end{aligned}
$$

For the path types we have
$\mathcal{I}[(p: S)] = \mathcal{I}[(e: (p': S))]$ if $p = e.p'$ where
$\mathcal{I}[(\epsilon: S)] = \mathcal{I}[S]$, $\mathcal{I}[(a: S)] = \mathcal{I}[[a: S]]$, $\mathcal{I}[(\forall: S)] = \mathcal{I}[\forall\{S\}]$, $\mathcal{I}[(\exists: S)] = \mathcal{I}[\exists\{S\}]$

An interpretation $\mathcal{I}$ is a *instance* of $\mathbf{R}$ iff for all $R \in \mathbf{R}$, $\mathcal{I}[S^a] \subseteq \mathcal{I}[S^c]$. Note that, in a instance $\mathcal{I}$: $\mathcal{I}[C] \subseteq \mathcal{I}[\sigma(C)]$, that is the objects of a class satisfy the class description.

2.1 Subsumption and Semantic Expansion

Subsumption w.r.t. $\mathbf{R}$: $S \sqsubseteq_{\mathrm{R}} S'$ iff $\mathcal{I}[S] \subseteq \mathcal{I}[S']$, for all instance $\mathcal{I}$ of $\mathbf{R}$.
Subsumption w.r.t. $\mathbf{R}_\sigma$: $S \sqsubseteq_\sigma S'$ iff $\mathcal{I}[S] \subseteq \mathcal{I}[S']$, for all instance $\mathcal{I}$ of $\mathbf{R}_\sigma$.

The *semantic expansion* $EXP(S)$ of a type S is a type such that $EXP(S) \simeq_{\mathrm{R}} S$ and for all $S' \in \mathbf{S}$ such that $S' \simeq_{\mathrm{R}} S$ we have $EXP(S) \sqsubseteq_\sigma S'$; i.e., $EXP(S)$ is the most refined type among the types $\simeq_{\mathrm{R}}$–equivalent to the type S as it includes all the possible restrictions implied by the schema $\mathbf{R}$. Thus, it is the lowest with respect to the $\sqsubseteq_\sigma$ relation among all the types $\simeq_{\mathrm{R}}$–equivalent to S. Note that, the recursive steps generating $EXP(S)$ individuate a class of $\simeq$–equivalent types, where each element is a type $\simeq_{\mathrm{R}}$–equivalent to the original type S. An element of this class of equivalence can be evaluated by means of the following function Γ from $\mathbf{S}$ to $\mathbf{S}$:

$$
\Gamma(S) \quad = \quad \left\{
\begin{array}{ll}
S \sqcap \bigsqcap_k (p_k : S_k^c) & \forall R_k, p_k\text{: } S \sqsubseteq_\sigma (p_k: S_k^a), \\
 & S \not\sqsubseteq_\sigma (p_k: S_k^c) \\
S & \text{otherwise}
\end{array}
\right.
$$

and we set $\tilde{\Gamma} = \Gamma^{\bar{i}}$, where $\bar{i}$ is the least integer such that $\Gamma^{\bar{i}} = \Gamma^{\bar{i}+1}$. [2]

THEOREM 1 *For all $S \in \mathbf{S}$, $EXP(S)$ can be computed by means of $\tilde{\Gamma}(S)$.*

COROLLARY 1 *For all $S, S' \in \mathbf{S}$ we have $S \sqsubseteq_{\mathrm{R}} S'$ if and only if $\tilde{\Gamma}(S) \sqsubseteq_\sigma S'$.*

Therefore, computing subsumption in a schema $\mathbf{R}$ can be performed by firstly determining the semantic expansion of a type and, secondly, by computing subsumption in the subschema $\mathbf{R}_\sigma$ [2].

Let us give two simple query optimization examples related to schema of table 1.

Q: "Select storages storing *at least* a material having a risk ≥ 15"

$$Q = \texttt{Storage} \sqcap [\texttt{stock}\colon \exists\{\texttt{Material} \sqcap [\texttt{risk}\colon 15 \div \infty]\}]$$

By rules R_1, R_3:

$$\tilde{\Gamma}(Q) = \texttt{SStorage} \sqcap [\texttt{stock}\colon \exists\{\texttt{SMaterial} \sqcap [\texttt{risk}\colon 15 \div \infty]\}]$$

In this way, we can optimize the query by obtaining the *most specialized generalization* of the classes involved in the query.

Another rewriting rule proposed in [17, 19] is the *constraint removal*, i.e., removal of implied factors. We formalize constraint removal by subsumption. As an example, consider the query:

Q: "Select storages managed by managers with a level $(6 \div 14)$ and storing at least a material with a feature F1":

$$Q = \underbrace{\texttt{Storage} \sqcap (\texttt{stock.}\exists\texttt{.feature.}\exists\colon \textit{''F1''})}_{S} \sqcap \underbrace{(\texttt{managed-by.level}\colon 6 \div 14)}_{S'}$$

By rule R_2 and by $\sigma(\texttt{TManager})$, we have $S \sqsubseteq_{\mathrm{R}} S'$, as $\tilde{\Gamma}(S) \sqsubseteq_\sigma S'$. Thus, S' can be eliminated from Q avoiding the access to the class `Manager`.

References

[1] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Using subsumption in semantic query optimization. In A. Napoli, editor, *IJCAI Workshop on Object-Based Representation Systems*, Chambery - France, August 1993.

[2] S. Bergamaschi and B. Nebel. Acquisition and validation of complex object database schemata supporting multiple inheritance. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks and Complex Problem Solving Technologies*, 1993. to appear.

[2] The existence of $\bar{i}$ is guaranteed as the number of rules is finite and a rule cannot be applied more than once with the same path $(S \not\sqsubseteq_\sigma (p\colon S^c))$.

[3] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM Transactions on Database Systems*, 17(3):385–422, September 1992.

[4] R.J. Brachman and J.G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[5] M. Buchheit, F.M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *13th. Int. Joint Conf. on Artificial Intelligence*, 1:704–709, 1993.

[6] M. Buchheit, M. A. Jeusfeld, W. Nutt, and M. Staudt. Subsumption between queries to object-oriented database. In *EDBT*, pages 348–353, 1994.

[7] L. Cardelli. A semantics of multiple inheritance. In *Semantics of Data Types - Lecture Notes in Computer Science N. 173*, pages 51–67. Springer-Verlag, 1984.

[8] U. S. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2):162–207, June 1990.

[9] Edward P.F. Chan. Containment and minimization of positive conjunctive queries in oodb's. In *Principles of Database Systems*, pages 202–11. ACM, 1992.

[10] M.M. Hammer and S. B. Zdonik. Knowledge based query processing. In *6th Int. Conf. on Very Large Databases*, pages 137–147, 1980.

[11] M. Jeusfeld and M. Staudt. Query optimization in deductive object bases. In Freytag, Maier, and Vossen, editors, *Query Processing for Advanced Database System*. Morgan Kaufmann Publishers, Inc., June 1993.

[12] W. Kim. A model of queries for object-oriented database systems. In *Int. Conf. on Very Large Databases*, Amsterdam, Holland, August 1989.

[13] J. J. King. *Query optimization by semantic reasoning.* PhD thesis, Dept. of Computer Science, Stanford University, Palo Alto, 1981.

[14] J. J. King. Quist: a system for semantic query optimization in relational databases. In *7th Int. Conf. on Very Large Databases*, pages 510–517, 1981.

[15] C. Lecluse and P. Richard. Modelling complex structures in object-oriented databases. In *Symp. on Principles of Database Systems*, pages 362–369, Philadelphia, PA, 1989.

[16] H. H. Pang, H. Lu, and B.C. Ooi. An efficient semantic query optimization algorithm. In *Int. Conf. on Data Engineering*, pages 326–335, 1991.

[17] S. Shenoy and M. Ozsoyoglu. A system for semantic query optimization. *ACM-SIGMOD*, pages 181–195, May 1987.

[18] S. Shenoy and M. Ozsoyoglu. Design and implementation of a semantic query optimizer. *IEEE Trans. Knowl. and Data Engineering*, 1(3):344–361, September 1989.

[19] M. Siegel, E. Sciore, and S. Salveter. A method for automatic rule derivation to support semantic query optimization. *ACM Transactions on Database Systems*, 17(4):563–600, December 1992.