Using Ontologies for Semantic Query Optimization of XML Database

Wei Sun and Da-Xin Liu

College of Computer Science and Technology, Harbin Engineering University,
Harbin Heilongjiang Province, China
sunwei78@hrbeu.edu.cn

Abstract. As XML has gained prevalence in recent years, the management of XML compliant structured-document database has become a very interesting and compelling research area. Effective query optimization is crucial to obtaining good performance from an XML database given a declarative query specification because of the much enlarged optimization space. Query rewriting techniques based on semantic knowledge have been used in database management systems, namely for query optimization。 The main goal of query optimization is to rewrite a user query into another one that uses less time and/or less resources during the execution. When using those query optimization strategies the transformed queries are equivalent to the submitted ones. This paper presents a new approach of query optimization using ontology semantics for query processing within XML database. In fact, our approach shows how ontologies can effectively be exploited to rewrite a user query into another one such that the new query provides equally meaningful results that satisfy the intention of the user. Based on practical examples and their usefulness we develop a set of rewriting rules. In addition, we prove that the results of the query rewriting are semantically correct by using a logical model.

1 Introduction

Recently, XML has emerged as the *de-facto* standard for *publishing* and *exchanging* data on the Web. Many data sources export XML data, and publish their contents using DTD's or XML schemas. Thus, independently of whether the data is actually stored in XML native mode or in a relational store, the view presented to the users is XML-based. The use of XML as a data representation and exchange standard raises new issues for data management.

A large number of research approaches have used semantic knowledge for supporting data management to overcome problems caused by the increasing growth of data in local databases, and the variety of its format and model in distributed databases. The use of semantic knowledge in its various forms including meta-models, semantic rules, and integrity constraints can improve query processing capabilities by transforming user queries into other semantically equivalent ones, which can be answered in less time and/or with less resources. Known as semantic query optimization (SQO), has generated promising results in deductive, relational and object databases. Naturally, it is also expected to be an optimization direction for XML query processing.

Among the three major functionalities of an XML query language, namely, pattern retrieval, filtering and restructuring, only pattern retrieval is specific to the XML data model. Therefore, recent work on XML SQO techniques [1,2,3] focuses on pattern retrieval optimization. Most of them fall into one of the following two categories:

1. Query minimization: For example, Query tree minimization [1,3] would simplify a query asking for “all auctions with an initial price” to one asking for “all auctions”, if it is known from the schema that each auction must have an initial price. The pruned query is typically more efficient to evaluate than the original one, regardless of the nature of the data source.

2. Query rewriting: For example, “query rewriting using state extents” [2] assumes that indices are built on element types. In persistent XML applications, it is practical to preprocess the data to build indices. However, this is not the case for the XML stream scenario since data arrives on the fly and usually no indices are provided in the data.

Currently, research work on the Semantic Web and data integration are focusing on using ontologies as semantic support for data processing. Ontologies have proven to be useful to capture the semantic content of data sources and to unify the semantic relationships between heterogenous structures. Thus, users should not care about where and how the data are organized in the sources. For this reason, systems like OBSERVER and TAMBIS allow users to formulate their queries over an ontology without directly accessing the data sources. In this paper, we present a new approach on how to improve the answers of queries based on semantic knowledge expressed in ontologies. Given an XML database, we assume the existence of an ontology which is associated with the database and which provides the context of its objects. We show how ontologies can be exploited effectively to rewrite a user query such that the new query can provide more "meaningful" results meeting the intention of the user.

2 Related Works

Work related to rewrite user query using semantic knowledge has emerged in two different research areas: Semantic query optimization and global information processing area.

Semantic query optimization. The basic idea of semantic query optimization (SQO) is to rewrite a query to another more efficient query, which is semantically equivalent, i.e. provides the same answer. Here, SQO approaches use semantic knowledge in various forms including semantic rules and range rules. Range rules states facts about the range of values of a given attribute, whereas semantic rules define the regularity of data for a given database. Therefore, these rules can be driven from the non-uniform distribution of values in a database. Expressing semantics in the form of horn clause sets allows the optimizer to make possible reformulations on an input query involving the insertion of new literals, or the deletion of literals, or the refuting the entire query. Several approaches on SQO have been developed to address different aspects of query processing: In [11] semantic rules have been used to derive useful information, which can reduce the cost of query plans. In [12, 13] algorithms have been developed for optimizing conjunctive sub-queries. To this end, learning techniques have been applied to generate semantic (operational) rules from a database automatically [14]. While the previous approaches are based on extracting semantic knowledge from the underlying database, current research approaches use knowledge from additional source [15, 16].

Ontology. The term "Ontology" or "Ontologies" is becoming frequently used in many contexts of database and artificial intelligence researches. However, there is not a unique definition of what an ontology is [7-10]. An initial definition was given by Tom Gruber: "*an ontology is an explicit specification of a conceptualization*" [7]. However, this definition is general and remains still unsatisfied for many researchers. In [8] Nicola Guarino argues that the notion of "conceptualization" is badly used in the definition. We note that many real-world ontologies already combine data instances and concepts [9]. Our definition differ from this point of view as we show later . Informally, we define an ontology as an intentional description of what is known about the essence of the entities in a particular domain of interest using abstractions, also called *concepts* and the *relationships* among them.

Semantic query optimization of XML data. The diversity of the XML queries (referred to in this paper as *structural queries*) results from the diversity of possible XML schemas (also called *structural schemas*) for a single conceptual model. In comparison, the schema languages that operate on the conceptual level (called *conceptual schemas*) are *structurally flat* so that the user can formulate a determined query (called *conceptual query*) without considering the structure of the source. There are currently many attempts to use *conceptual schemas* [4, 5] or *conceptual queries* [6] to overcome the problem of structural heterogeneities among XML sources.

Contributions. In brief, we make the following contributions in this paper: We propose an approach for using ontologies based graph model to represent semantic information of heterogeneous XML sources. This model integrates semantic information both the XML nesting structure and the domain content. These ontologies are processed lossless with respect to the nesting structure of the XML document. Finally, we describe an id-concept rule for rewriting XML query based on semantic information. The optimization is based on a mapping model based on ontology and the rules of rewriting the query on the XML sources.

3 The Problem Representation

Using semantic knowledge to optimize query has generated promising results in deductive, relational and object databases. Naturally, it is also expected to be an optimization direction for XML database query processing. Therefore, recent work focuses on XML optimization techniques based on semantic. It is becoming a crucial problem, how to represent the semantic information of XML database. The result is a set of semantically constrained axioms and semantically constrained relations between axioms. When a query is given to the system, the semantic transformation phase uses these stored semantic constrained sets to generate semantically equivalent queries that may be processed faster than the original query. In Fig.1, there is one DTD of XML data, which will be used as follow.

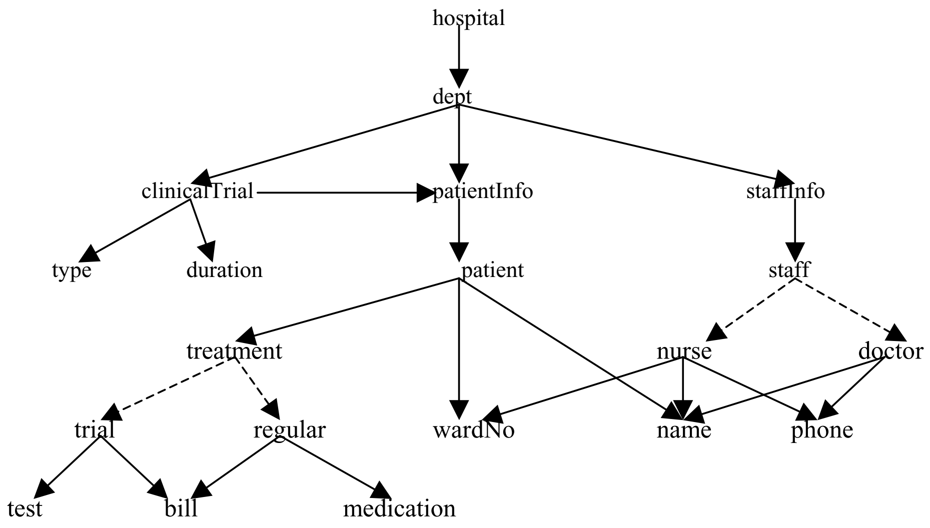

Fig. 1. The DTD of XML document

4 XML Semantic Model (XSM)

In this section, we propose the model of XML semantic, which is represented by ontologies about the content of XML document and schema. The model XSM can transform a normal query to a a semantically equivalent query, and the equivalent query has less time than the origin one to be processed.

4.1 Ontology Definition

Informally, we define an ontology as an intentional description of what is known about the essence of the entities in a particular domain of interest using abstractions, also called *concepts* and the *relationships* among them. Basically, the hierarchical organization of concepts through the inheritance ("ISA") relationship constitutes the backbone of an ontology. Other kinds of relationship like part-whole ("PartOf") or synonym ("SynOf") or application specific relationships might exist. To the best of our knowledge, there is no work until now addressing the issue of using ontology relationships at the database instance level. Despite the disagreement upon a common meaning of an "ontology", the role of ontologies that must play is clear: Ontologies should provide a concise and unambiguous description of concepts and their relationships for a domain of interest. Ontologies are shared and reused by different agents i.e. human or/and machines.

Formally, we define an ontology as a set $\aleph$ and a set $\Re$ as follows: $\aleph = \{c_1, ..., c_n\}$ and $\Re$ = {"ISA"; "SynOf"; "PartOf"}, where $c_i \in \aleph$ is a concept name, and $r_i \in \Re$ is the type of the binary relation relating two concepts (c_i and r_i are non-null strings). Other domain-specific types may also exist. At the top of the concept hierarchy we assume the existence of a universal concept, called "Anything", which represents the most general concept of an ontology. In the literature, the word "concept" is frequently used as a synonym for the word "concept name". Hence, for the design of an ontology only one term is chosen as a name for a particular concept. Further, we assume that the terms "concept" and "concept name" have the same meaning.

4.2 Ontology Formal Representation

This section presents a graph-based representation of an ontology. We introduce its basic formal settings, and some related operations relevant to further discussions.

Graph oriented model. We represent an ontology as a directed graph *G*(*V;E*), where *V* is a finite set of vertices and *E* is a finite set of edges: Each vertex of *V* is labelled with a concept and each edge of *E* represents the inter-concept relationship between two concepts. Formally, the label of a node $n \in V$ is defined by a function *N*(*n*) = c_i that maps *n* to a string c_i from $\aleph$. The label of an edge $e \in E$ is given by a function *T*(*e*) = r_i that maps *e* to a string r_i from $\Re$.

In summary, an ontology is the set $O = \{G(V,E), \aleph, \Re, N, T\}$.

Graph operations. In order to navigate the ontology graph, we define the following sets of concepts: *Rparent*, *DESC*, *SUBT*, *SY Ns*, *PARTs* and *WHOLEs*. We need these operations to identify nodes in the graph, which hold concepts that are of interest for our query reformulations.

Let $P_{ths}(n_1 - n_2)$ be a set of directed paths between two nodes n_1 and n_2. We denote by node(c) the node labelled by a concept c, and by child(n) and parent(n) the child-node and parent-node of a node n, respectively. Given two nodes $n_1 = node(c_1)$ and $n_2 = node(c_2)$ the operation are formulated as follows:

Rparent(r, c_1)=c_2 iff n_2=parent(n_1) and T[(n_2, n_1)]=r

DESC(r,c)=$\{ s \in \aleph \mid \exists p \in P_{ths}(node(c) - node(s)) : \forall e \in p, T(e) = r \}$

SYNs(c)=$\{ s \in \aleph \mid \exists p \in P_{ths}(node(c) - node(s)) : \forall e \in p, T(e) = \text{"}SynOf\text{"} \}$

SUBT(c)=$\{ s \in \aleph \mid \exists p \in P_{ths}(node(c) - node(s)) \}$

Informally, *Rparent*(*r; c*) returns the label of the parent node of a concept c by following an edge of type r. *DESC*(*r; c*) returns the set of all concepts in *O* whose nodes are children of the node of c by means of descending edges of type r. Similarly, *SUBT*(*c*) returns all descendants of c for any edge-type and *SY Ns*(*c*) returns the set of all synonyms of c in *O*. In addition, we define an *Outgoings*(*n*) as a set of edge-types going out from a node n and *PARTs*(*c*) as the set of concepts whose nodes are related to the node(c) through the edges of type "part of". Here, two cases must be distinguished:

Case 1: If Outgoings(node(c)) $\ni$ "Part of" then $PARTs(c) = A \cup B \cup C$, Where

$A = DESC$("Part of"*;* c)
$B = DESC$("ISA"*;* a), $a \in A$
$C = SYNs(h) \cup SYNs(l)$, $h \in A$ and $l \in B$

Informally, *PARTs*(*c*) is the set of concepts obtained by retrieving the labels of all nodes that are PartOf-children of the node(c) together with their ISA-descendants and synonyms.

Case 2: If Outgoings(node(c)) $\ni$ "ISA" then $PARTs(c) = PARTs(s_i)$, where

$s_i \in A$ and $\forall (s_1, s_2) \in A^2$ $PARTs(s_1) = PARTs(s_2)$, $A = DESC$("ISA"*;* c). Informally, *PARTs* of a concept c is defined recursively in terms of its sub-concepts. It is equal to the *PARTs* of one of its sub-concepts (if they have the same *PARTs*).

Inversely, we define *WHOLEs* of a given concept c as the set of concepts c_i such that $c \in PARTs(c_i)$.

4.3 XML Semantic Model

The XML semantic model is stated as an extension of the given ontology, denoted by O^* , which includes new concepts and additional relationship-types. The new concepts represent relation names, entity names, attribute names and values of the database unless they already exist. We denote these concepts by NC_R NC_E NC_A NC_V , respectively. Furthermore, we call *id-concepts* the concepts that represent id-values of the database. The additional relationships have to relating these concepts to the existing ones or to each other. Their types are defined as follows:

"ValueOf" is the type of relationship that relates each value-concept to its associated attribute-concept or entity-concept.
"HasA" is the type of relationship between entity-concepts and attribute-concepts.
"InstanceOf" is the type of relationship that relates an Id-concept to its associated entity-concept.
"Tupleof" is the type of relationship that relates entity-concepts to each other, which are associated with a particular tuple.
"RelateTo" is the type of relationship that relates relation-concepts to entity-concepts, one relation-concept with one or more entity-concepts.
"OnlyA" is the type of relationship that relates entity-concepts to each other, which are associated with an entity-concept only.

In summary, O^* is defined as a set $O^* = \{G^*, \aleph^*, \Re^*, N, T\}$, where $\aleph^* = NC_E \cup NC_A \cup NC_V \cup NC_R$, and $\Re^* = \Re \cup$ {"ValueOf"," HasA"," InstanceOf"," Tupleof", "RelateTo"}. Such as Fig.2.

Table 1. XPath expressions and Concepts

XPath expressions	Concept expressions
hospital	hospital
hospital\dept	dept
Hospital\dept\clinicalTrial\patientInfo	clinical-patientInfo
hospital\dept\patientInfo	Non-clinical-patientInfo
…	…
hospital\ dept\patientInfo\patient\treatment\regular\medication	medication

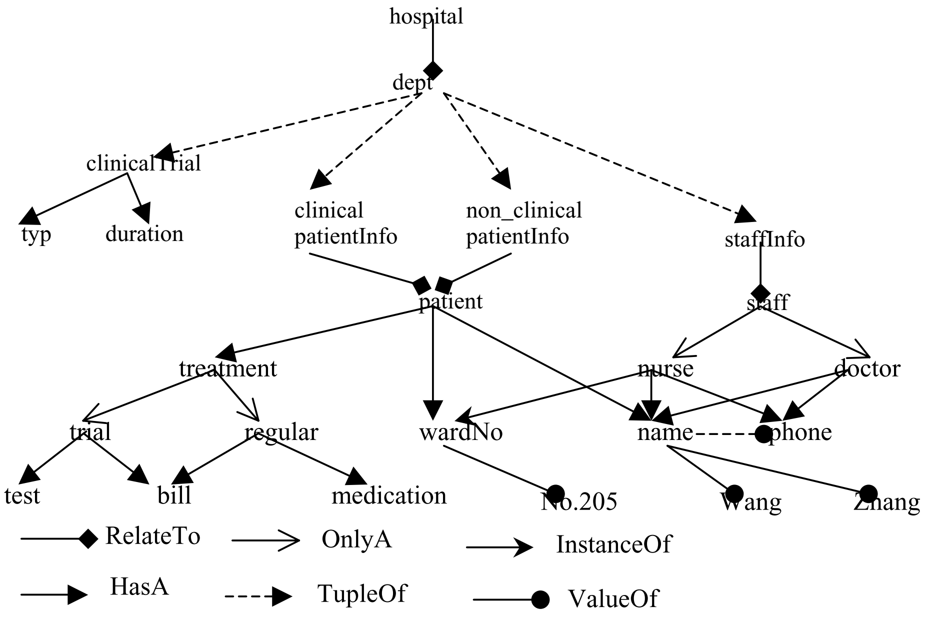

Fig. 2. Shows a portion of the semantic model related to DTD shown in Fig.1

Logical Interpretation. By using the First Order Language (FOL) the semantic model O^* is defined as a theory Γ which consists of an Interpretation I and a set of well formed formulas [12]. I is specified by the set of individuals $\aleph^*$ and an interpretation function $\bullet^I$. In the following, we describe the interpretation of O^*.

Let n_1 and n_2 be the nodes of two concepts *a* and *b*, respectively. Formally, Γ:

$$I = (\aleph^*; \bullet^I)$$

$$ISA^I = \{(a,b) \in \aleph^{*2} \mid T(n_1,n_2) = "ISA"\}$$

$$SYN^I = \{(a,b) \in \aleph^{*2} \mid T(n_1,n_2) = "SynOf"\}$$

$PARTOF^I = \{(a,b) \in \aleph^{*2} \mid T(n_1,n_2) = "PartOf"\}$
$HASA^I = \{(a,b) \in \aleph^{*2} \mid T(n_1,n_2) = "HasA"\}$
$VALUEOF^I = \{(a,b) \in \aleph^{*2} \mid T(n_1,n_2) = "ValueOf"\}$
$INSOF^I = \{(a,b) \in \aleph^{*2} \mid T(n_1,n_2) = "Ins\tan ceOf"\}$
$Key^I = \{a \in \aleph^{*} \mid \exists b.T(a,b) = "Ins\tan ceOf"\}$
$TUPOF^I = \{(a,b) \in \aleph^{*2} \mid T(n_1,n_2) = "TupleOf"\}$
$RELATETO^I = \{(a,b) \in \aleph^{*2} \mid T(a,b) = "\mathrm{Re}lateTo"\}$
$WHOLE^I = \{a \in \aleph^{*} \mid \forall b_1 b_2 c.ISA(a,b_1) \wedge ISA(a,b_2) \wedge PARTOF(b_1,c) \rightarrow PARTOF(b_2,c)\}$
$\forall x.ISA(x,x)$
$\forall x.SYN(x,x)$
$\forall x.PARTOF(x,x)$
$\forall xyz.ISA(x,y) \wedge ISA(x,z) \rightarrow ISA(x,z)$
$\forall xy.SYN(x,y) \leftrightarrow SYN(y,x)$
$\forall xyz.SYN(x,y) \wedge SYN(x,z) \rightarrow SYN(x,z)$
$\forall xyz.ISA(x,y) \wedge SYN(y,z) \leftrightarrow ISA(x,z)$
$\forall xy\exists z.VALUEOF(x,y) \rightarrow HASA(z,y)$
$\forall xy\exists z.TUPVAL(x,y) \rightarrow INSOF(y,z)$
$\forall xyz.PARTOF(x,y) \wedge SYN(y,z) \leftrightarrow PARTOF(x,z)$
$\forall xyz.PARTOF(x,y) \wedge ISA(y,z) \leftrightarrow PARTOF(x,z)$
$\forall xyz.PARTOF(x,y) \wedge PARTOF(x,z) \leftrightarrow PARTOF(x,z)$
$\forall xyz.VALUEOF(y,z) \wedge ISA(x,y) \rightarrow VALUEOF(x,z)$
$\forall xyz.VALUEOF(y,z) \wedge SYN(x,y) \rightarrow VALUEOF(x,z)$
$\forall xyz\exists w.INSOF(x,y) \wedge HASA(y,z) \rightarrow TUPVAL(x,w) \wedge VALUEOF(w,z)$
$\forall xyz.WHOLE(x) \wedge ISA(x,y) \wedge PARTOF(y,z) \leftrightarrow PARTOF(x,z)$ $x;\ y;\ z;$
w are variables.

5 Id-Concept Rule and Validations

We note that a common feature of the rules is that after applying a rule to a query Q, the results of the reformulated query might increase. We denote by S_Q and $S_{Q'}$ the result set of Q and Q’, respectively. This augmentation is not arbitrary but it is proved by the semantic model O^*. According to O^*, each tuple-identifier in S_Q is represented by an id-concept, which is related to value-concepts through the ValueOf-relationship and a relation-concept through the TupleOf and InstanceOf-relationship, respectively. O^* interprets the reformulation results of a given rule as the existence of additional value-concepts, which are semantically related to those representing terms in the condition of *Q*. For brevity, we describe only an example of validation of the proposed rules using the available logical expressions from Γ.

Concerning this rule the S_Q -identifiers are formally expressed by the following set of individuals Ω_1, we obtain the set of individuals from Q which represents all id-concepts of the tuples in $S_{Q'}$. Formally,

$$\Omega_1 = \{x \mid \exists z \forall a VALUEOF(z,a) \wedge TUPOF(z,x) \wedge INSOF(x, NC_E) \rightarrow TUPOF(z, NC_E) \wedge [ISA(NC_V, a) \vee SYN(NC_V, a)]\}.$$

6 Conclusions

Recently, there is a growing interest in ontologies for managing data in database and information systems. In fact, ontologies provide good supports for understanding the meaning of data. They are broadly used to optimize query processing among the distributed sources. In this paper, we use ontologies within XML database and present a new approach of query optimization using semantic knowledge from a given ontology to rewrite a user query in such way that the query answer is more meaningful to the user. To this end, we propose a set of query rewriting rules and illustrate their effectiveness by some running examples. Although these rules might not be ideal, we hope that they can bring more insight into the nature of query answers. Our approach is appropriate for database applications, where some attributes are enumerated from a list of terms. In the future, we will develop additional rewriting rules and intend to address the problem of how to establish mapping information between the database objects and ontological concepts present in an ontology associated with a specific database.

References

1. S. Amer-Yahia, S. Cho, L. V. Lakshmanan, and D. Srivastava. Minimization of Tree Pattern Queries. *In Proc. of SIGMOD*(2001) 497–508
2. M. F. Fernandez, D. Suciu. Optimizing Regular Path Expressions Using Graph Schemas. *In Proc. of ICDE* (1998) 14–23
3. Z. Chen, H. Jagadish and L.V.S. Lakshmanan et al. From Tree Patterns to Generalized Tree Patterns; On Efficient Evaluation of XQuery. *In Proc. of 29th VLDB* (2003) 237-248
4. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-Based Integration of XML Web Resources. *In Proceedings of the 1st International Semantic Web Conference (ISWC 2002)* 117–131
5. B. Amann, I. Fundulaki, M. Scholl, C. Beeri, and A. Vercoustre. Mapping XML Fragments to Community Web Ontologies. *In Proceedings of the 4th International Workshop on the Web and Databases (WebDB 2001)* 97–102

6. S. D. Camillo, C. A. Heuser, and R. S. Mello. Querying Heterogeneous XML Sources through a Conceptual Schema. *In Proceedings of the 22nd International Conference on Conceptual Modeling (ER2003* 186–199
7. Gruber, T.: A translation approach to portable ontology specifications. *In: Knowledge Acquisition,* 5(2) (1993) 199-220
8. Guarino, N., Giaretta, P.: Ontologies and knowledge bases: towards a terminological clarification. *In: Knowledge Building Knowledge Sharing,ION Press.* (1995) 25-32
9. Noy, N., Hafner, C.D.: The state of the art in ontology design. *AI Magazine* 3(1997) 53-74
10. Chandrasekaran, B., Josephson, J., Benjamins, V.: What are ontologies, and why do we need them? *In: IEEE Intelligent Systems*, (1999) 20-26
11. Hsu, C., Knoblock, C.A.: Semantic query optimization for query plans of heterogeneous multidatabase systems. *Knowledge and Data Engineering,* 12 (2000) 959-978
12. Yu, C.T., Sun, W.: Automatic knowledge acquisition and maintenance for semantic query optimization. IEEE Trans. *Knowledge and Data Engineering,* 1 (1989) 362-375
13. Sun, W., Yu, C.: Semantic query optimization for tree and chain queries. *IEEE Trans. on Data and Knowledge Engineering* 1 (1994) 136-151
14. Hsu, C.: Learning effective and robust knowledge for semantic query optimization (1996)
15. Peim, M., Franconi, E., Paton, N., Goble, C.: Query processing with description logic ontologies over object-wrapped databases. technical report, University of Manchester (2001)
16. Bergamaschi, S., Sartori, C., Beneventano, D., Vincini, M.: ODB-tools: A description logics based tool for schema validation and semantic query optimization in object oriented databases. *Advances in Artificial Intelligence, 5th Congress of the Italian Association for Artificial Intelligence*, Rome, Italy (1997) 435-438