Query Evaluation of Ontology Data Model in E-commerce Organizations

Simona-Elena Varlan, Assistant at University "Vasile Alecsandri" from Bacau, Calea Marasesti no. 157, Bacau, Romania, varlan_simona@yahoo.com

Abstract
The paper presents a theoretical analyzes of the query complexity of ontology data model in the context of virtual organizations with an experimental model for a computer based e-commerce ontology. The paper represents an important part of a study about the utility and improvements of Semantic Web technologies for knowledge management in the context of virtual organizations. We will make a comparison between two types of querying modalities – the one for relational data base model which is used frequently specially in economical applications and the one for ontology data model for Semantic Web which is becoming increasingly popular among researchers and software developers. To make comparison we will express the queries as conjunctive queries as they can be very general and powerful enough to express a domain.

Keywords: Semantic Web, Ontology, E-commerce, Conjunctive Queries.

Introduction

The development of Semantic Web technologies contributed to representation, integration, processing and utilization of knowledge. The ontology is an important aspect of the actual Web and one way to demonstrate its utility is to compare complexity of querying of an e-commerce web site that use an ontology to define concepts and their relations in the domain, to the complexity of querying of a conventional e-commerce web site that use the relational database to store the content information.

It is well known that conjunctive queries are often used to express queries issued on relational data bases. They are in a form of formulae of first-order logic constructed using conjunction $\wedge$ and existential quantification $\exists$, but not using disjunction $\vee$, negation $\neg$, or universal quantification $\forall$.

Many researches have analyzed the complexity of conjunctional queries and in particular we will use in our analysis the method used by Abrahams (2006) with the complexity measures elaborated by Vardi (1982) and terminology and studies published by Horrocks and Tessaris (2000) to translate conjunctive queries into ontology concepts. We will also use notations to express a knowledge base provided by Baader (2002).

Ontologies have been created over the structure of RDF, which provides only one method of describing Web resources through the creation of metadata (Brian 2005). Ontologies provide also description of the structure of concepts and relationships between concepts, declarations of classes and subclasses, types, and cardinality for the properties of concepts, therefore providing adequate support to achieve rule-based reasoning and inference. Therefore they are based on specific concepts of Description Logic (DL). We know that ontologies are logic schemes about concepts and relations between them, so we can say that it may represent a knowledge base - KB – where we ca apply methods of logical representation specific to Description Logic. The ontology will be then represented by TBox and ABox - which incorporates elements of terminology, vocabulary of modelled domain and respectively conceptual assertions. Vocabulary consists of concepts and roles as binary relations between concepts to be modelled. DL is based on semantic description model therefore the statements in the TBox and ABox can be identified by

formulas of first-order predicate type or rules of inference formulated by Baader (2002). OWL, the ontology language, being based on RDF and RDF Schema offers the possibility of terminology description required to define the TBox.

It is important to mention here that we considered the Abox as a relational database and for the complexity comparison we will use query expression complexity as described by Vardi (1982).

There are three ways to measure the complexity of queries in a specific language (Vardi 1982): first, one can fix a specific query in the language and study the complexity of applying this query to arbitrary databases, the complexity is then given as a function of the size of the databases – it is called data complexity; alternatively, one can fix a specific databases and study the complexity of applying queries represented by arbitrary expressions in the language, the complexity is then given as a function of the length of the expressions – this is referred as expression complexity; finally, one can study the complexity of applying queries represented by arbitrary expressions in the language to arbitrary databases, the complexity is then given as a function of the combined size of the expressions and the databases – this is often called as combined complexity. In our research we considered a knowledge base and queries represented by arbitrary expressions (conjunctive queries expresses for relational databases and for ontology). For that we considered for analysis only the query expression complexity.

In the following section we will present the theoretical aspects in order to complete our analysis.

Premises

For our analyze we need a relational databases and an ontology build on the same knowledge base. We choose a simple example about electronically equipment of a computer e-commerce web site.

The database is presented in Figure 1 and the ontology is presented in Figure 2:

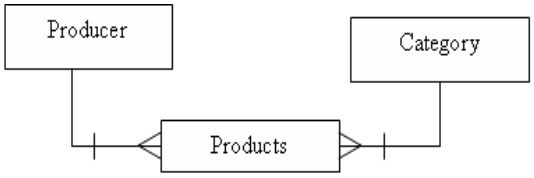


Fig 1. ERD for relational database

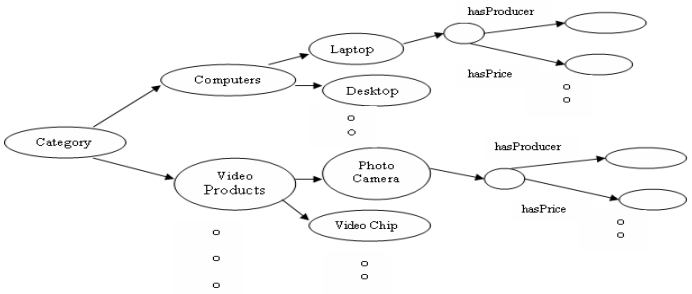


Fig 2. The Ontology

In the figures we have two ways of representing knowledge about computer equipment each with differences of implementation.

First we establish our query for the both types of knowledge representation. We want to select the products from the category laptops produced by HP Company and with a price no higher than 1000 euros. For a relational database the corresponding query in SQL would be:

Select name from products, category, and producer

Where category.idCategory = products.idCategory and

products.idProducer = producer.idProducer and

producerName like "HP" and price <= 1000

We transform the query in a conjunctive query:

Q1(x): Products(X) $\wedge$ hasCategory(x, y) $\wedge$ Category(y, Laptop) $\wedge$ hasProducer(x, z) $\wedge$ Producer (z, HP) $\wedge$ hasPrice(x, <=1000)

The same query for ontology would the following:

Q2(x): Laptop and hasProducer HP and hasPrice <= 1000

or

Laptop $\prod$ ($\exists$ hasProducer.{HP}) $\prod$ ($\exists$ hasPrice.{<=1000})

As we can see, there are two expressions of the query for the same knowledge base. In the first one we have role terms (x,y):R and variables and in the second one we have only concept terms. So we need to transform the conjunctive query into a concept expression and then to analyze the expressions.

To transform the first query into a concept expression we will fallow the method described in Abrahams (2006) and use the theorems and definitions from Horrocks (2000) and Vardi (1982) respectively.

Theoretical aspects

Conform to Vardi we have:

KB – knowledge base,

With KB = T+A, where T - set of terminologies TBox (terminological axioms) and A - set of assertions or ABox (individual axioms);

C and D description of concepts C, D $\in$T,

C$\subseteq$D – inclusion,

C$\equiv$D – equivalent,

R – role,

a,b – assertions, a,b$\in$ A of the form $C_{(a)}$ and $D_{(b)}$ with (a,b) : R or R(a,b).

Definition 1 (Vardi 1982)

A conjunctive query is of form:

Q : $q_1 \wedge q_2 \wedge \ldots \wedge q_n$ are query terms, where q_i is of the form x:C or (x, y):R and C is a concept, R a role, and x, y are either individual names or variables.

If we have variables in out query then we will assume the existence of a set of variables V [5] that is disjoint from the set of individuals names: V Π IN = $\varnothing$.

Having this definition we can create a conjunctive query over the KB from the relational model view. In the following we will use (Vardi 1982) to transform the conjunctive query into a concept expression in relation with the ontology model view of the KB.

Because a conjunctive query may contain concept terms a_i:C but also role terms (a_i, b_i):R, a conversion of every role term into a concept term must be done. This procedure it is called rolling up a query.

Theorem 1:

Let $\Sigma = \langle T, A \rangle$ be a DL knowledge base, with C_1, C_n concepts in T, R a role, and a,b individual names in A. Given a new concept name P_b not appearing in Σ:

$\langle T , A \rangle \models \langle a, b \rangle$:R $\wedge$ b:$C_1 \wedge \ldots \wedge$ b:C_n

if and only if

$\langle T , A \cup \{b:P_b\} \rangle \models$ a:$\exists$ R.($P_b \Pi C_1 \Pi C_2 \Pi \ldots \Pi C_n$).

Basically we will transform the binary relation (x,y):R into an existential restriction x: $\exists$R.(y).

An even more complex rolling up procedure is needed in order to deal with variables in the query, because variables can be interpretive as any element of the domain. In order to eliminate any variable from the query terms, Horrocks and Tessaris (2000) had introduced the notion of a directed graph structure induced by the query, in which there is a node x for each variable or individual name in the query and a direct edge R from node x to node y for every role (x,y):R in the query.

Definition 2:

The transformation of a conjunctive query seen as a direct graph G into a concept expression can be done be applying one of the following steps (Horrocks 2000):

1) If the graph G contains a leaf node y then the role term (x,y):R is rolled up, and the edge (x,y) is removed from the G;
2) If G contains a confluent node y then all role terms (x,y):R are rolled up, and all edges (x,y) are removed from G;
3) If G contains edges but no leaf nodes and no confluent nodes then it must contain a cycle. A nod y in a cycle is chosen and rolled up.

As you can se our small ontology is a direct tree with the root Category, and only edges of type (x,y). We choose the first case to transform a conjunctive query into a concept expression.

To compare query complexity we will use the expression complexity method.

Definition 3 (Vardi 1982):

Let φ be a sentence in this language of size s (a sentence represents a query). φ has at most s variables. In order to evaluate φ on a database of size n is suffices to cycle through at most $\mathbf{n^s}$ possible assignments of values from database to the variables, and this can be done in space O(log n).

Conform to definition 3 we have:

φ - Conjunctive query of size s

$\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$, with φi of the form x_i:C or (x.,y):R and the maximum number of the variables is s. That means for a query we search through n rows (database size) to assign values from the domain to the query variables (s).

That means the n^s cycles of possible assignments of values from database to the variables represent the Cartesian product of possible assignments for every variable in the query. We can declare that the complexity is of form:

$C_{(Q)} = n_1 * \ldots * n_{v_i}, v - query\text{var}\, iable, i \leq s$

Comparing Complexity

Our analyze is based on the maximum number of possible values assignment to the variables in the query, so we establish for query complexity the n is the number of individuals and s is the number of the variables in the query.

Starting from the conjunctive query presented above we will apply the definitions from the section 2.1 – the rolling up technique.

Q1(x): Products(X) ∧ hasCategory(x, y) ∧ Category(y, Laptop) ∧ hasProducer(x, z) ∧ Producer (z, HP) ∧ hasPrice(x, <=1000)

Applying definition 2 the query will become:

Products(x) ∧ hasCategory.(x, Laptop}) ∧ hasProducer(x, HP) ∧ hasPrice(x, <=1000)

Applying theorem 1 the query will become:

Products $\prod$ ($\exists$ hasCategory.{Laptop}) $\prod$ ($\exists$ hasProducer.{HP}) $\prod$ ($\exists$ hasPrice.{<=1000})

As we can see from the ontology example the products are instances of the concept Category which means that the query can be shortened reducing predicate hasCategory and transforming the class Laptop into the root class:

Q2(x): Laptop $\sqcap$ ($\exists$ hasProducer.{HP}) $\sqcap$ ($\exists$ hasPrice.{<=1000})

This query can be edited in Protégé for example in this maner:

Q2(x): Laptop and hasProducer HP and hasPrice <= 1000

We will write the Q2(x) as conjunctive queries to make better comparison between them.

Q2(x): Laptop(x) $\wedge$ hasProducer(x,HP) $\wedge$ hasPrice(x, <=1000)

We can see in Q1(x) that the join between the Product and Category is made through the variable y and the join between the Product and Producer is made through the variable z, so we have 2 joins modelled in the query, 6 query terms and 3 variables. In the query Q2(x) written as conjunctive queries we have 0 joins, 1 variable and only 3 query terms. To calculate the complexity we suppose that we have in the database **a** number of products with price no higher than 1000 euros, **b** number of laptops, **c** number of products made by HP Company, and **d** number of HP laptops with the price lower than 1000 euros.

We analyze query Q1(x).

We have 3 variables: x, y and z. x represent the number of products with the property hasPrice no higher than 1000, that means the value assignments for variable x is a; y represents the number of products that are laptops, that means the value assignments for variable y is b; and z represents the number of HP products that means the variable assignments for z is c.

The query complexity is therefore $C_{(Q1(x))} = a \times b \times c$, where $a, b, c \in N$.

We analyze query Q2(x):

We have 1 variable x that represents the HP laptops with price <= 1000. That means the value assignments for x is d.

The query complexity is $C_{(Q2(x))} = d$, where $d \in N$.

It is obvious that the Q2(x) has a lower expression complexity than the query Q1(x).

Practical tests

In the fallowing we will make a series of 2 practical examples to test the hypothesis that a specific query over ontology has an expression complexity lower than over a relational database.

We have to mention that the Ontology and the Databases together with the properties are made in Romanian, and we translated only the formulas and queries we used.

For our tests we create the ontology as described in section 2 with the help of Protégé editor, and for relational databases we used Access. In the ontology the class *Category* has more subclasses, types of categories (e.g. class Computer, Video Components etc.) which in turn may have different subclasses that will bind to Product's class instances through specific object properties. In the database there is a single table Category that is linked with the table Products through a 1: N relation. The ideea is that we can model the ontlogy with how many clases we want and bind the instances of the classes together through object properties, but in a relational databases this means to create a lot of tables and strict relations between them, which can be very difficult to handle.

Test 1

The tested query is the one from the previous section.

The Database

Conjunctive Query:

Q1(x): $Produse(x) \wedge hasCategory(x,y) \wedge Category(y, Laptop) \wedge hasProducers(x,z) \wedge Produser(z, HP) \wedge hasPrice(x, >=1000).$

Query Q1(x) in Access using SQL is of form:

SELECT p.name

FROM Products AS p, Category AS c, Produser AS pr

WHERE c.idCategory = p.idCategory and p.idProducer=pr.idProducer and

c.name like "Laptop" and pr.name like "HP" and price>= 1000;

The results are:

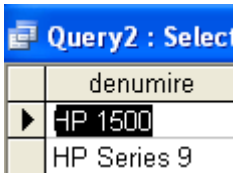


Fig 3: Results Test 1 in Access

The Ontology

Concept query:

Q1'(x): $Laptop \prod (\exists hasProducer.\{HP\}) \prod (\exists hasPrice.\{>=1000\}).$

Cnjunctive Query:

Q1'(x): $Laptop(x) \wedge hasProducer(x,HP) \wedge hasPrice(x, >=1000).$

Results iîn Protégé:

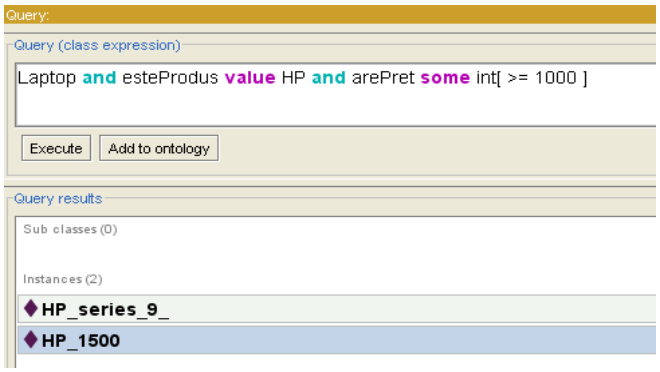


Fig 4: Test 1 in Protégé

Table 1: Query Comparison for Test 1

	Databases	Ontology
Terms	6	3
Joins	2	0
C_n	a x b x c	d

Conform to the table, the number of terms is reduced to 3 in the ontology case and the number of joins at 0. The expression complexity has a lawer value also comparing to the one for databases.

Test 2:

The query is to find all the laptops produced in USA. We have to mention that a new concept was added to the ontology and in databases – Country. In the databases the table Country is connected with Product through table Producer (we don't refer to the country where the product is made in), but in the ontology, we can infer that if a product has a producer and that producer is located in a country, than the product is produced in that country.

The Database

Conjunctive Query:

Q2(x): *Products(x)* $\wedge$ *hasCategory(x,y)* $\wedge$ *Category(y,z)* $\wedge$ *hasProducer(x,z)* $\wedge$ *fromCountry(z,t)* $\wedge$ *Country(t, USA)*

Query in Access is:

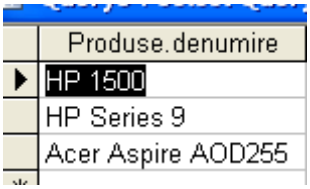
SELECT Products.name, Category.name

FROM Country INNER JOIN (Producer INNER JOIN (Category INNER JOIN Products ON Category.idCategory = Products.idCategory)

ON Produser.idProducer = Products.idProducer) ON Country.idCountry = Producer.idCountry

WHERE (((Country.name)="USA") AND ((Category.name)="Laptop"));

Results in Access:



	Produse.denumire
▶	HP 1500
	HP Series 9
	Acer Aspire AOD255

Fig 5: Results Access Test 2

The Ontology

Concept Query:

Q2'(x): $Laptop(x) \wedge hasProducer(x,z) \wedge fromCountry(z,t) \wedge Country(t, USA)$

We apply Definition 2 and Theorem 1 and the query become:

Q2'(x): $Laptop(x) \wedge hasProducer(x,z) \wedge fromCountry(z,USA)$

Q2'(x): $Laptop \prod (\exists hasProducer.(\exists fromCountry.\{USA\}))$

Because we can infer that a product is produced in a country where the producer is located thorugh an object property that we will call *producedIN*, then the query become:

Q2'(x): $Laptop \prod (\exists producedIN.\{USA\})$

Conjunctiv query:

Q2'(x): $Laptop(x) \wedge producedIN(x,USA)$

Results in Protégé:

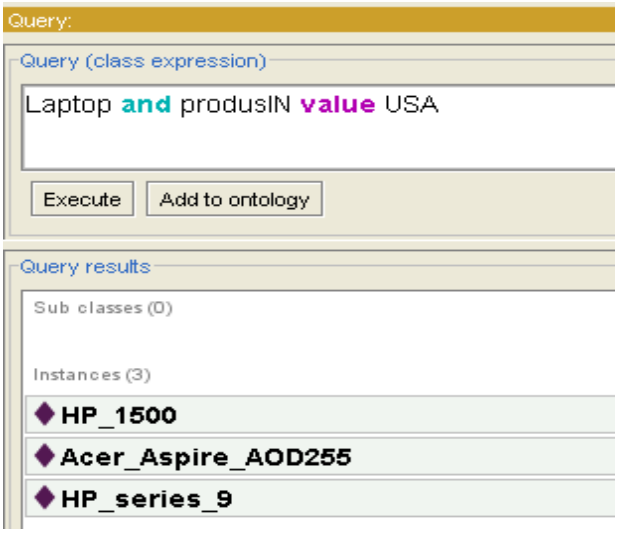


Fig 6: Test 2 in Protégé

We presum that we have a number of laptops and b products with producers from USA. In Q2(x) are 4 variables, x,y,z şi t, but because for the same number of laptops we have the same number of producers we will take into acount only variable y and t. Query Q2'(x) has a single variable x, that represents laptops produced in USA.

Table 2: Query Comparison for Test 2

	databases	Ontology
terms	6	2
Joins	3	0
C_n	a x b	b

We have 6 terms in first query, 3 joins, and expression complexity is the product between the number of variable assigments for y and number of variable assigments for t. The second query (for ontology) we have 2 terms, 0 joins and the expression complexity is the number of laptop produces in USA.

Conclusions

The paper presents analyze between two types of query expressions for the same knowledge base. First query expression was created with conjunctive queries specific for a database model and the second one with concept expressions specific for the ontology data model. The database model represents a conventional e-commerce web site and the ontology data model represents the semantic e-commerce web site.

The analyze result show a difference between the complexity of the queries. There are some advantages that Semantic Web environment can bring to knowledge management in an virtual organization, as it is proved we can obtain a reduced complexity query, a reduced number of queries and a better processing of knowledge though semantics (we create a direct relation for products, categories and producer so we query starting from the Laptop class). In the future we need to create a system based on ontology data model and to effectively experiment our presumptions.

Acknowledgment

This work was supported by CNCSIS-UEFISCSU, project number PN II-RU code 188/2010.

References

[1] Abrahams, B., Dai, W., (2005) 'Architecture for automated annotation and ontology based querying of Semantic Web resources', IEEE/WIC/ACM International Conference on Web Intelligence, ISBN: 0-7695-2415-X, 19-22 September, Campaign, France, 413 - 417;

[2] Abrahams, B. (2006) Tourism Information Systems Integration and Utilization Within The Semantic Web, PhD Thesis, Victoria University, School of Information Systems, Faculty of Business and Law, Australia;

[3] Baader, F., Nutt, W., (2002), 'Basic Description Logics', [Online], [Retrieved 30 Octomber], http://www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-02.pdf;

[4] Brian M. (2005) Semantic Web Technologies, CCLRC Rutherford Appleton Laboratory;

[5] Horrocks, I., Tessaris, S., (2000), 'A conjunctive query language for description logic ABoxes', Conference on Artificial Intelligence AAAAI/IAAI, 399-404;

[6] Horrocks, I., (2008) 'Ontologies and the Semantic Web', *Communications of the ACM*, vol 51 (issue no. 12), pp. 58-67;

[7] Ruckhaus, E., Ruiz, E., Vidal, M.E., (2008), 'Query evaluation and optimization in the semantic web', *Journal Theory and Practice of Logic Programming*, vol. 8 (issue no. 3), pp. 393-409;

[8] Tim Berners-Lee, Hendler, J., Lassila, O. (2001), 'The Semantic Web', Scientific American, 29-37;

[9] Vardi, M., (1982), 'The complexity of relational queries', *ACM SIGACT*, vol. 143(issue no.2), pp. 137–146;

[10] Zheleznyakov, D., Calvanese, D., Kharlamov, E., Nutt, W., (2010), 'Updating TBoxes in DL-Lite', Proc. Of DL 2010, CEUR, vol. 573 (issue no.)1, pp.102-113;

[11] www.semanticweb.com

[12] http://protege.stanford.edu/